

UNIVERSITY OF PISA



School of Engineering

Master of Science in Computer Engineering

Large-Scale and Multi-Structured Databases Course

TASK 2 PROJECT

CINEMASPACE

WORKGROUP:

Marie Giannoni

Shirley Caillere

Francisco Payés Erroa

Diego Casu

ACADEMIC YEAR 2019/2020

INDEX

1	APPLICATION DESCRIPTION	3
2	MAIN ACTORS AND REQUIREMENTS	5
2.1	Main actors	5
2.2	Requirements	5
2.2.1	Functional requirements	5
2.2.2	Non-functional requirements	7
3	USE CASES	8
4	ANALYSIS CLASS DIAGRAM	18
5	INTERFACE MOCKUP	19
6	PLATFORM ARCHITECTURE	23
6.1	Overview	23
6.2	Database structure	24
6.2.1	Collections	24
6.2.2	Main queries and indexes	29
7	IMPLEMENTATION	37
7.1	Class diagram	37
7.2	Replica set	45
7.3	Performances of database indexes	52
7.4	Test manual	62
7.4.1	Client user	63
7.4.2	Administrator	76
	APPENDIX: DATASET CREDITS	80

1 APPLICATION DESCRIPTION

CinemaSpace is an application that allows users to consult cinematographic information inside a vast collection of films and to express personal opinions about them in the form of ratings.

To join the community, a sign up is required: in this phase, the user can specify some basic information about herself, like the date of birth and gender; after performing a login, she will have access to the main page and the possibility to browse the film archive.

The latter can be browsed in different ways, according to the personal user needs; in particular, the alternatives are:

1. to perform a direct search by specifying a set of keywords of interest;
2. to ask the application about the most trending films, either by ratings or by number of page visits.

In any case, a list of the films that match the search is presented. To see the information about a film, the user can click on one of the displayed posters or titles, obtaining the associated description page. In addition to the title and the poster, the description page shows a rich collection of cinematographic details like runtime, release date, overview, original language, crew and cast members, country and production company; moreover, it gives access to the rating system of the community.

The user can rate a film and modify her rating whenever she wants: the ratings of the community are saved and used to present some useful statistics when accessing a film page.

For each film, the application generates:

1. the mean rating, computed taking into account all the user ratings;
2. the recent mean rating, computed as the average of the ratings given to a film in the last two years. The goal is to identify an evaluation that reflects the recent opinion about it;
3. the distribution of the ratings, i.e. how the ratings are distributed in the evaluation scale;
4. the distribution of the ratings by demographic, i.e. divided and classified by age and gender.

Furthermore, the application tracks the number of visits to the film pages: both this information and the overall mean rating are exploited to present the most trending films, in alternative to the search by keywords.

Every user can access a personal profile page, which contains an information part and a statistic part. The information part displays the user's details (username, email, gender and date of birth), while the statistic part shows which film genres are the most recurrent in specific and selectable intervals of the expressed ratings (e.g. in the interval of ratings that goes from 4 to 5).

If the user no longer wants to be part of the community, she can close her account, causing the deletion of all her personal data, except for the ratings: the latter will be kept, in order to preserve the consistency of the statistics.

2 MAIN ACTORS AND REQUIREMENTS

2.1 Main actors

- The client user
- The administrator of the community.

2.2 Requirements

2.2.1 Functional requirements

The system must allow the not registered client user:

- to sign up, specifying at least the username, the email address, the date of birth, the gender and the password.

The system must allow the registered client user:

- to perform a login specifying the email and the password;
- to perform a logout;
- to close her user account, with the deletion of the personal information apart from the rates that she has given. The latter will be kept in the rating archive;
- to search films by a set of keywords;
- to see the most trending films, either by ratings or by number of page visits;
- to view the cinematographic information related to a film, like title, poster, runtime, genres, release date, overview, cast, crew, country, original language, budget, revenue and production companies;
- to rate a film, in a scale from 0 to 5, where the half votes are allowed;
- to modify the rating given to a film;

- to view the mean rating of a film, computed considering all the ratings received over time;
- to view the recent mean rating of a film, computed as the average of the ratings given to a film in the last two years;
- to view the distribution of the ratings of a film, i.e. how the ratings are distributed in the evaluation scale;
- to view the distribution of the ratings of the film by demographic, i.e. divided and classified by age and gender;
- to view her profile page, with her personal details (at least username, email address, date of birth and gender);
- to view on her personal profile page statistics about which film genres are the most recurrent in specific and selectable intervals of her expressed ratings.

The system must allow the administrator:

- to perform a login with the privileges of an administrator;
- to perform a logout;
- to add a film to the archive uploading a JSON file;
- to add a collection of films to the archive uploading a JSON file;
- to search a film by keywords;
- to delete a film and all the related information.

2.2.2 Non-functional requirements

The system needs to manage an archive characterized by a large dimension: the number of films and users are going to grow over time, together with the number of ratings. The actions allowed by the client application are simple, intuitive and commonly adopted by competing services, meaning that the user expects quick responses to her requests and actions. Furthermore, the reactivity of the service is more important than the fact that all the users can view the same up-to-date archive and the view of stale data for a brief period of time is accepted.

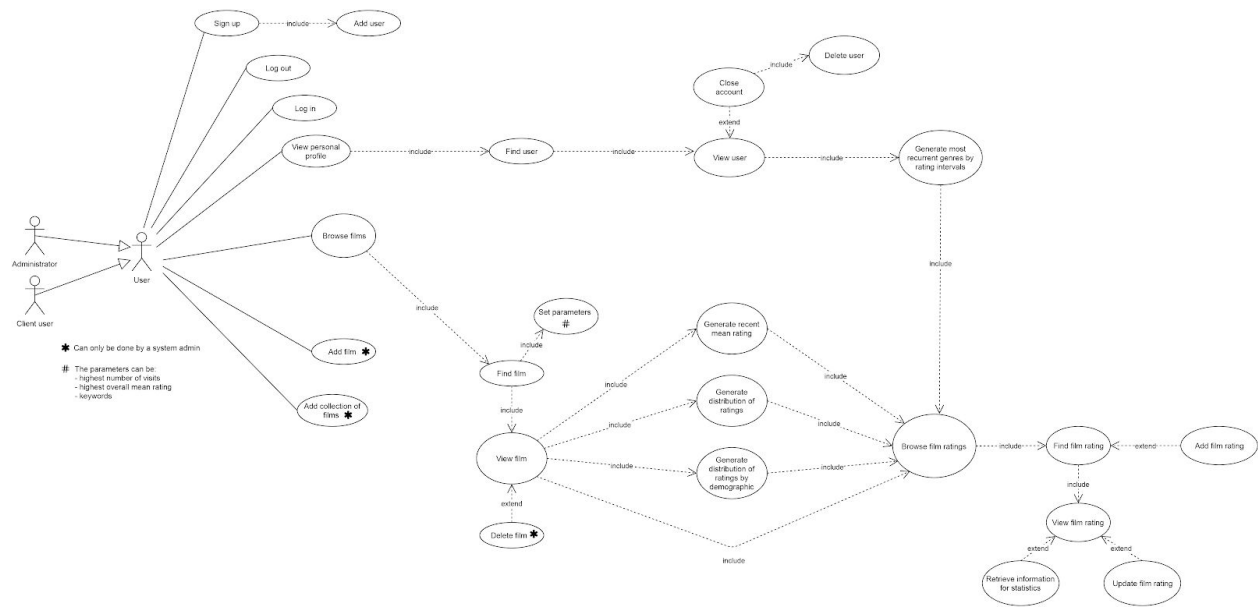
Given these considerations, the system must ensure:

- high availability;
- tolerance to network partitions, i.e. must be available also if a portion of the network is isolated;
- quick response to read operations (a read operation is completed when the data are received from one server);
- quick response to write operations (a write operation is completed when the data are written in one server);
- since the number of read requests is expected to be higher than the number of write requests — the latter are only the creation of accounts and the addition/update of ratings — eventual read optimizations that can cause a degradation of write performances are allowed, if the service time will remain acceptable;
- the preservation of the data using replica sets;
- the increase of at least the responsiveness to read operations when deploying replica sets (the replicas must respond to read operations).

Finally, the system's archive must ensure:

- flexibility of the schema;
- the possibility to import data from JSON files.

3 USE CASES



Sign up
Brief description: A user registers for the first time on the application, by submitting her information.
Preconditions: N/A
Basic flow of events: <ol style="list-style-type: none"> 1. The user opens the application. 2. The user sees a page where she can either sign up or log in. 3. The user fills up the form to create an account with the following information: <ul style="list-style-type: none"> • Username • Password • Email address • Gender • Date of birth. 4. The user submits the information by clicking on the “Confirm” button. 5. The system checks the presence of the required fields. 6. The system logs a message about the current action’s results.
Extensions: <ol style="list-style-type: none"> 5a. Required fields are missing <ul style="list-style-type: none"> • The system highlights the missing fields that should be filled. 6b. The system fails to “Sign up” <ul style="list-style-type: none"> • The system logs information about the system’s error and no modification is made to the current state of the application.
Post-conditions: The user account is created.

Log in
Brief description: A user logs in with her credentials to have access to the system.
Preconditions: The user must have an account. The user cannot be logged in.
Basic flow of events: <ol style="list-style-type: none"> 1. The user opens the application. 2. The user sees a page where she can either sign up or log in. 3. The user logs in by entering email and password and by clicking on the “Confirm” button. 4. The system checks the presence of the required fields. 5. The system logs a message about the current action’s results.
Extensions: <ol style="list-style-type: none"> 4a. Required fields are missing <ul style="list-style-type: none"> • The system highlights the missing fields that should be filled. 5a. The system fails to “Log in” <ul style="list-style-type: none"> • The system logs information about the system’s error and no modification is made to the current state of the application.
Post-conditions: The user is logged into the system and has access to its functions.

Log out
Brief description: A user logs out of the application.
Preconditions: The user must have an account. The user must be logged in.
Basic flow of events: <ol style="list-style-type: none"> 1. The user logs out by clicking on the “Disconnection” button on the side bar. 2. The system logs a message about the current action’s results.
Extensions: <ol style="list-style-type: none"> 2a. The system fails to “Log out” <ul style="list-style-type: none"> • The system logs information about the system’s error and no modification is made to the current state of the application.
Post-conditions: The user is not logged into the system anymore. The user has not access to the functions of the system until she logs in again.

Close a user account
Brief description: A user closes her account.
Preconditions: The user must have an account. The user must be logged in. The user must be in her personal page.
Basic flow of events: <ol style="list-style-type: none"> 1. The user submits the closing action by clicking on the “Close Account” button at the top right of the page. 2. The system logs a message about the current action’s results.
Extensions: <ol style="list-style-type: none"> 2a. The system fails to “Close Account” <ul style="list-style-type: none"> • The system logs information about the system’s error and no modification is made to the current state of the application.
Post-conditions: The user account and the system are no longer accessible.

Browse films by keywords
Brief description: A user browses films by keywords, using the search bar on the home page.
Preconditions: The user must have an account. The user must be logged in.
Basic flow of events: <ol style="list-style-type: none"> 1. The user writes the keywords in the search bar. 2. The user submits the keywords by clicking on the "Search" button of the search bar. 3. The system checks the presence of films with matching keywords. 4. The posters and titles of the films resulting from the search are displayed.
Extensions: <ol style="list-style-type: none"> 3a. There are no keywords in the search bar <ul style="list-style-type: none"> • The system highlights the search bar that should be filled. 4a. The system fails to “Browse films by keywords” <ul style="list-style-type: none"> • The system logs information about the system’s error and no modification is made to the current state of the application.
Post-conditions: The page is updated with the titles and posters of the matching films.

Browse films by visits
Brief description: A user browses films by visits, going on the page “Most Popular”.
Preconditions: The user must have an account. The user must be logged in.
Basic flow of events: <ol style="list-style-type: none"> 1. The user submits the “Browse films by visits” action by clicking on the “Most Popular” button on the side bar. 2. The user sees the page of the most popular films with the posters and titles of the films ordered by number of visits.
Extensions: <ol style="list-style-type: none"> 2a. The system fails to “Browse films by visits” <ul style="list-style-type: none"> • The system logs information about the system’s error and no modification is made to the current state of the application.
Post-conditions: The page is updated with the titles and posters of the matching films.

Browse films by ratings
Brief description: A user browses films by ratings, going on the page “Highest rated”.
Preconditions: The user must have an account. The user must be logged in.
Basic flow of events: <ol style="list-style-type: none"> 1. The user submits the “Browse films by ratings” action by clicking on the “Highest rated” button on the side bar. 2. The user sees the page of the highest rated films with the posters and titles of the films ordered by rating value.
Extensions: <ol style="list-style-type: none"> 2a. The system fails to “Browse films by ratings” <ul style="list-style-type: none"> • The system logs information about the system’s error and no modification is made to the current state of the application.
Post-conditions: The page is updated with the titles and posters of the matching films.

View a film
Brief description: A user wants to see the information about a film in the application.
Preconditions: The user must have an account. The user must be logged in.
Basic flow of events: <ol style="list-style-type: none"> 1. The user submits the “View a film” action by clicking on the poster or the title of the film whose information she wants to see. 2. The user sees the view of the film with cinematographic details about the film and sees the statistics related to the film, which are the mean rating, the distribution of ratings by demographic and the distribution of ratings by scores.
Extensions: <ol style="list-style-type: none"> 2a. The system fails to “View a film” <ul style="list-style-type: none"> • The system logs information about the system’s error and no modification is made to the current state of the application.
Post-conditions: The page is updated with the film details and statistics.

Rate a film
Brief description: A user wants to rate a film.
Preconditions: The user must have an account. The user must be logged in. The user must be on the page of the film she wants to rate.
Basic flow of events: <ol style="list-style-type: none"> 1. The user submits the “Rate a film” action by clicking on the number of stars she wants to give to the film on the ratings section. by clicking on the rating she wants to give to the film on the ratings section. The rating to be given is from 1 to 5. 2. The rating is displayed.
Extensions: <ol style="list-style-type: none"> 2a. The system fails to “Rate a film” <ul style="list-style-type: none"> • The system logs information about the system’s error and no modification is made to the current state of the application.
Post-conditions: The rating is inserted in the list of ratings of the movie.

Update the rating of a film
Brief description: A user wants to modify the rating of a film that he/she has given.
Preconditions: The user must have an account. The user must be logged in. The user must be on the page of the film. The user must have rated the film.
Basic flow of events: <ol style="list-style-type: none"> 1. The user submits the “Update the rating” action by clicking on the rating she wants to give to the film on the ratings section. The rating to be given is from 1 to 5. 2. The updated rating is displayed.
Extensions: <ol style="list-style-type: none"> 2a. The system fails to “Update the rating” <ul style="list-style-type: none"> • The system logs information about the system’s error and no modification is made to the current state of the application.
Post-conditions: The updated rating is inserted in the list of ratings of the movie.

Delete a film
Brief description: An administrator user wants to delete a film from the database.
Preconditions: The user must have an account with the privilege of an administrator. The user must be logged in. The user must be on the page of the film she wants to delete.
Basic flow of events: <ol style="list-style-type: none"> 1. The user submits the “Delete a film” action by clicking on the “Delete film” button. 2. The system logs a message about the current action’s results.
Extensions: <ol style="list-style-type: none"> 1a. The system fails to “Delete a film” <ul style="list-style-type: none"> • The system logs information about the system’s error and no modification is made to the current state of the application.
Post-conditions: The film and all the related ratings are deleted from the database.

Add film
Brief description: An administrator user wants to add a single film in the database.
Preconditions: The user must have an account with the privilege of an administrator. The user must be logged in.
Basic flow of events: <ol style="list-style-type: none"> 1. The user clicks on the “Add film(s)” button on the side bar. 2. The user uploads a JSON file by clicking on the “Upload” button. 3. The user submits the “Add a film” action by clicking on the “Confirm” button. 4. The system logs a message about the current action’s results.
Extensions: <ol style="list-style-type: none"> 2a. The system fails to “Upload a file” <ul style="list-style-type: none"> • The system logs information about the system’s error and no modification is made to the current state of the application. 3a. The system fails to “Add a film” <ul style="list-style-type: none"> • The system logs information about the system’s error and no modification is made to the current state of the application.
Post-conditions: The film is added to the film database and can now be found by proceeding a search.

Add a collection of films
Brief description: An administrator user wants to add multiple films in the database.
Preconditions: The user must have an account with the privilege of an administrator. The user must be logged in.
Basic flow of events: <ol style="list-style-type: none"> 1. The user clicks on the “Add film(s)” button on the side bar. 2. The user uploads a JSON file by clicking on the “Upload” button. 3. The user submits the “Add a film” action by clicking on the “Confirm” button. 4. The system logs a message about the current action’s results.
Extensions: <ol style="list-style-type: none"> 2a. The system fails to “Upload a file” <ul style="list-style-type: none"> • The system logs information about the system’s error and no modification is made to the current state of the application. 3a. The system fails to “Add a film” <ul style="list-style-type: none"> • The system logs information about the system’s error and no modification is made to the current state of the application.
Post-conditions: The films are added to the film database and can now be found by proceeding a search.

View personal profile
Brief description: A user wants to see her personal profile information.
Preconditions: The user must have an account. The user must be logged in.
Basic flow of events: <ol style="list-style-type: none"> 1. The user submits the “View personal profile” action by clicking on the “Personal Profile” button on the side bar. 2. The system displays the user information.
Extensions: <ol style="list-style-type: none"> 2a. The system fails to “View personal profile” <ul style="list-style-type: none"> • The system logs information about the system’s error and no modification is made to the current state of the application. 2b. The system fails to “View user information” <ul style="list-style-type: none"> • The system logs information about the system’s error and no modification is made to the current state of the application.
Post-conditions: The profile page is populated with the personal information.

See the most recurrent film genres in intervals of expressed ratings
Brief description: A user wants to see her personal rating statistics.
Preconditions: The user must have an account. The user must be logged in. The user must be on her personal page.
Basic flow of events: <ol style="list-style-type: none"> 1. The user sets a max value and a min value expressing an interval between 0 and 5. 2. The user confirms her actions by clicking on the “Confirm” button. 3. The system displays the pie chart that represents which film genres are the most recurrent in the specified interval.
Extensions: <ol style="list-style-type: none"> 3a. The system fails to “View personal statistics” <ul style="list-style-type: none"> • The system logs information about the system’s error and no modification is made to the current state of the application
Post-conditions: The profile page is populated with the personal statistics.

4 ANALYSIS CLASS DIAGRAM



5 INTERFACE MOCKUP

Login/Sign up

Sign Up

Username

Email

Password

Gender ☒ Male ☐ Female

Date of Birth (dd/mm/yyyy)

Confirm

Login

Email

Password

Confirm

Home page (user)

Home

Highest rated


Most Popular

Profile


Disconnection

Search films


FILMS



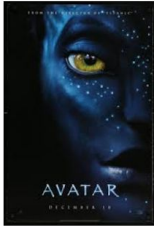
Star Trek




Terminator
Dark Fate





Guardians of the
galaxy vol. 2




Avatar





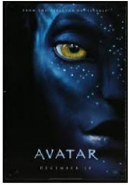




19

Film page (user)

Home
Highest rated
Most Popular
Profile
Disconnection



Avatar (2009)

Certificate : All publics
Runtime : 2h 42min
Genres: Action | Adventures | Fantasy
Release Date : 10 December 2009

A paraplegic Marine dispatched to the moon Pandora on a unique mission becomes torn between following his orders and protecting the world he feels is his home.

Director : James Cameron
Stars : Sam Worthington | Zoe Saldana | Sigourney Weaver ...
Country : USA
Language : English
Budget : \$237,000,000
Production Compagny : Twentieth Century Fox | Dune Entertainment | Lightstorm Entertainment

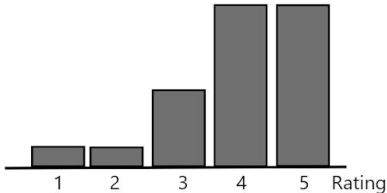
★ **3,9/5** *overall mean rating*

★ **4,3/5** *recent mean rating (of the last two years)*

Rating

	All	<18	18 - 45	45+
All	3,9	3,8	3,8	4
Males	3,9	3,8	3,8	4
Female	3,9	3,8	3,8	4

Rate ☆☆☆☆☆



Personal profile page (user)

Home
Highest rated
Most Popular
Profile
Disconnection

PROFILE

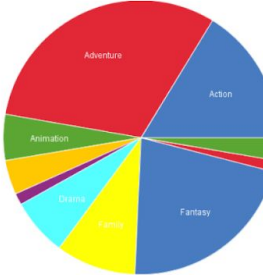
Username : Name
Email : email@email.com
Date of Birth : dd/mm/yyyy
Gender: Gender

Close Account

Film Rated

Min Max

0 5



Home page (administrator)

Home

Highest rated

Most Popular


Add Film(s)

Disconnection


Search films

Search


FILMS




Star Trek




Terminator
Dark Fate





Guardians of the
galaxy vol. 2




Avatar









Film page (administrator)

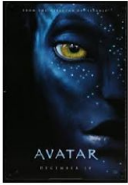
Home

Highest rated

Most Popular

Add Film(s)

Disconnection



Avatar (2009)

Delete Film

★ 3,9/5

overall mean rating

★ 4,3/5

recent mean rating
(of the last two years)

Certificate : All publics
Runtime : 2h 42min
Genres: Action | Adventures | Fantasy
Release Date : 10 December 2009

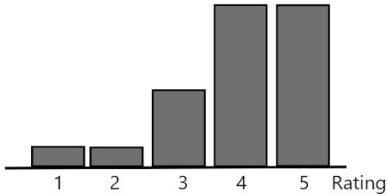
A paraplegic Marine dispatched to the moon Pandora on a unique mission becomes torn between following his orders and protecting the world he feels is his home.

Director : James Cameron
Stars : Sam Worthington | Zoe Saldana | Sigourney Weaver ...
Country : USA
Language : English
Budget : \$237,000,000
Production Compagny : Twentieth Century Fox | Dune Entertainment | Lightstorm Entertainment

Rating

Rate ☆☆☆☆☆

	All	<18	18 - 45	45+
All	3,9	3,8	3,8	4
Males	3,9	3,8	3,8	4
Female	3,9	3,8	3,8	4



Add film (administrator)

Home

Highest rated

Most Popular

Add Film(s)

Disconnection

Add a Film or a Collection of Films

Choose a JSON file ...

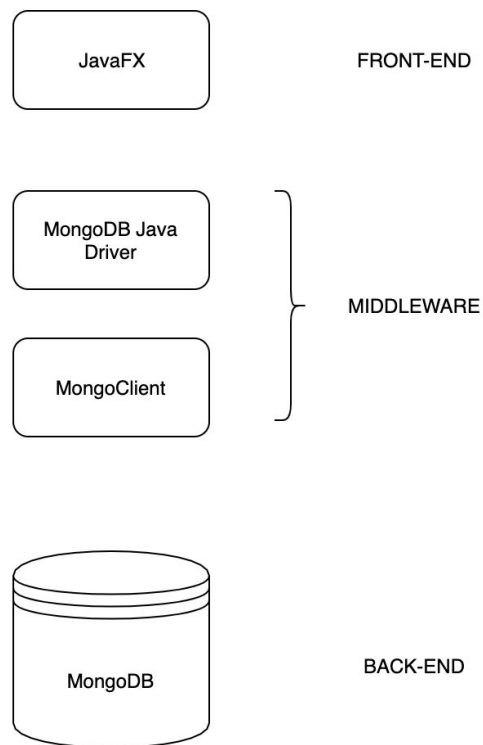
Select the file

Confirm

6 PLATFORM ARCHITECTURE

6.1 Overview

The chosen paradigm is the client-server one, where both the client user and the administrator will access the same application, but with different privileges. The client application will be developed in Java, with the support of JavaFX and FXML for the graphical interface, and it will connect to a remote MongoDB server. The latter suits both the functional and non-functional requirement; moreover, it offers a powerful Java driver library for client-server communications.



6.2 Database structure

6.2.1 Collections

The database is divided into three collections: User, Film and Rating. In the following, the structure of the documents inside each of them is presented.

User

The User collection is responsible for maintaining the account information of the users, both client users and administrators. The structure of a single document is:

```
{
  _id: ObjectId("5de0cc65bab5a6cf31576599"),
  username: "Jeanette Robinson",
  password: "umvawzuf",
  email: "cuzozbot@gik.kh",
  date_of_birth: "19/2/1959",
  gender: "Female",
  administrator: false
}
```

The primary key of the collection is the field `_id` of type `ObjectId` and it is generated automatically during the insertion. The fields representing the personal information are stored as simple strings, while the `administrator` field is a boolean that identifies if the user has the privileges of an administrator or not; an example of an administrator account is:

```
{
  _id: ObjectId("5de29555051f631f6c8227ee") ,
  username: "Admin",
  password: "admin",
  email: "admin@unipi.it",
  date_of_birth: "30/11/2019",
  gender: "Male",
  administrator: true }
}
```


Film

The Film collection is responsible for maintaining the information related to the films, except for the ratings. The structure of a single film document exploits the nesting possibility in order to retrieve in a single operation all the cinematographic details, as follows:

```
{
  _id: ObjectId("5de0ca2109c85f7e66a01c6b"),
  budget: 30000000,
  genres:[
    "Animation",
    "Comedy",
    "Family"
  ],
  homepage: "http://toystory.disney.com/toy-story",
  original_language: "en",
  original_title: "Toy Story",
  overview: "Led by Woody, Andy's toys live happily in his room...",
  poster_path: "/rhIRbceoE9lR4veEXuwCC2wARtG.jpg",
  production_companies:[
    "Pixar Animation Studios",
    ...
  ],
  production_countries:[
    "United States of America",
    ...
  ],
  release_date: "1995-10-30",
  revenue: 373554033,
  runtime: 81,
  spoken_languages:[
    "English",
    ...
  ],
  status: "Released",
  tagline: "",
  title: "Toy Story",
}
```

```

    number_of_visits: 48011,
    keywords:[
      "jealousy",
      "toy",
      "Toy Story",
      ...
    ],
    cast:[
      {
        character: "Woody (voice)",
        name: "Tom Hanks",
        order: 0,
        profile_path: "/pQFoyx7rp09CJTAb932F2g8Nlho.jpg"
      },
      {
        character: "Buzz Lightyear (voice)",
        name: "Tim Allen",
        order: 1,
        profile_path: "/uX2xVf6pMmPepxnvFWyBtjexzgY.jpg"
      },
      ...
    ],
    crew:[
      {
        department: "Directing",
        job: "Director",
        name: "John Lasseter",
        profile_path: "/7EdqiNbr4FRjIhKHYPpDfEEFEG.jpg"
      },
      ...
    ],
    average_rating: 3.5989304812834226,
    number_of_ratings: 374
  }

```

The primary key of the collection is the field `_id` of type `ObjectId` and it is generated automatically during the insertion.

While most of the fields are simple strings or numbers, the *genres*, *production_companies*, *production_countries*, *spoken_languages*, *keywords*, *cast* and *crew* fields are arrays of strings or documents. In particular:

- the *keywords* field contains the set of strings that the user can use to identify a film when performing a search;
- the *cast* field is an array of documents representing the cast members of the film. Each member is characterized by the name, the corresponding character, the order of importance (the lower, the more important) and the path of the personal picture (if given);
- the *crew* field is an array of documents representing the crew members that helped in the realization of the film. The structure is similar to the cast one, with the difference given by the department, the job information and the lack of an order.

Since the cast and crew details are not involved in analytical queries, a strict consistency in their values across different documents is not needed: they can be nested inside the film and be redundant.

Finally, the fields *average_rating* and *number_of_ratings* are used to compute progressively the overall average rating of a film, instead of executing each time an expensive aggregation on the ratings stored in the *Rating* collection. Both these fields can be easily recovered and recalculated starting from the collection of ratings: this means that it is not necessary to ensure their consistency with ACID updates.

This solution cannot be used for keeping track of the recent mean rating: the only possibility, except for the nesting (which is discussed in the next section) would require to compute this average at fixed times (e.g. every day, week or month), making the insertion of a rating not “interactive” from the point of view of the user.

Rating

The Rating collection is responsible for maintaining the film ratings expressed by the users. Despite the strict relationship between a rating and a user/film, which could give the possibility to retrieve all the user/film ratings in a single read operation, this set of documents is separated from the other collections. In general:

- the number of ratings related to a film can grow over time;
- the number of ratings expressed by a user can grow over time.

Using nested arrays in the Film or User collection could cause a continuous growth in the size of the documents, which could force MongoDB to perform frequent reallocations on the disk and, in the case of popular and classic films, to reject an insertion because the maximum document size is reached. In terms of statistics generation, the request frequency for the films is expected to be higher than for the users, not giving a significant advantage with the nesting in the User collection; moreover, the latter would complicate the preservation of the ratings when a user is deleted. Thus, the most convenient solution is to create a separate collection, eventually with the support of indexes, where the structure of a document is the following:

```
{
  _id: {
    userId: ObjectId("5de0cc65bab5a6cf3157658f"),
    filmId: ObjectId("5de0ca2109c85f7e66a01c7c")
  },
  rating: 3,
  timestamp: 867039249,
  age_of_user: 54,
  gender: "Female"
}
```

The primary key of the collection is the composition of a *userId* and a *filmId*, both of type ObjectId: the unique constraint on the primary key guarantees that a user can have at most one rating associated to a film. A rating is characterized by a numerical value, a timestamp in the Unix time format (milliseconds), the age and gender of the user at the moment of the evaluation.

6.2.2 Main queries and indexes

Login

Given username and password, the login operation requires to search for a specific document in the User collection with matching fields. This can be done with the query:

```
db.User.find({  
  email: "soni@ponsufpo.gi",  
  password: "motew"  
})
```

The response time of the search can be improved defining a compound index on the username and password fields:

```
db.User.createIndex({ email: 1, password: 1 }, { name: "login" })
```

Search films by highest number of visits

The operation performs a scan of the Film collection and returns a set of the most popular films by number of visits.

```
db.Film.aggregate([  
  { $sort : { number_of_visits : -1 } },  
  { $limit : 50 }  
])
```

The response time of the search can be improved defining an index on number_of_visits:

```
db.Film.createIndex({ number_of_visits: -1 }, { name: "numberOfVisits" })
```

Search films by keywords

The operation performs a scan of the Film collection based on a set of keywords given by the user. This search exploits the keywords field stored inside each Film document and, in the case of multiple matches, returns a sorted list based on the number of visits:

```
db.Film.aggregate([
  { $match: { keywords: { $in: ["horror", "drama", "action", "rivalry"] } } },
  { $sort: { number_of_visits: -1 } },
  { $limit: 50 }
])
```

The response time of the search can be improved defining an index on the array field keywords as follows:

```
db.Film.createIndex({ keywords: 1 }, { name: "keywords" })
```

The \$sort stage doesn't benefit of the index previously defined on number_of_visits, because it can take advantage of an index only if occurring at the beginning of the pipeline.

Search films by highest ratings

The operation performs a scan of the Film collection exploiting the average_rating field of each document. The query is the following:

```
db.Film.aggregate([
  { $sort : { average_rating: -1 } },
  { $match: { number_of_ratings: { $gte: 1000 } } },
  { $limit : 50 }
])
```

The response time of the search can be improved defining an index on the number_of_ratings field as follows:

```
db.Film.createIndex({ average_rating: -1 }, { name: "averageRating" })
```

Retrieve a film rating given by a user

Given the _id value of both the user and the film, the rating can be identified in the Rating collection as follows:

```
db.Rating.find({
  _id: {
    userId: ObjectId("5de0cc65bab5a6cf3157658f") ,
    filmId: ObjectId("5de0ca2109c85f7e66a01c7c")
  }
})
```

Since the query is on the _id field of the collection, it benefits automatically of the presence of the primary key index.

View distribution of ratings of a film

Given the `_id` value of the film, the distribution of ratings can be obtained scanning the `Rating` collection and grouping on the rating value, as follows:

```
db.Rating.aggregate([
  { $match: { "_id.filmId": ObjectId("5de0ca2209c85f7e66a02c2e") } },
  { $group: { _id: "$rating", count: { $sum: 1 } } },
  { $project: { _id: 0, value: "$_id", count: "$count" } }
])
```

The response time of this aggregation can be improved defining an index on the field `_id.filmId` as follows:

```
db.Rating.createIndex({ "_id.filmId" : 1 }, { name: "filmRating" })
```


View distribution of ratings of a film by demographic

Given the `_id` value of the film, we define the distribution of ratings by demographic as the distribution of ratings by age and gender, where:

- the gender can be "Female" or "Male";
- the age is divided into three intervals < 18 , $[18, 45)$, ≥ 45 .

In addition to the latter, the query calculates the distribution by age only, i.e. the total distribution of female and male users, for a comparison.

```
db.Rating.aggregate([
  { $match: { "_id.filmId": ObjectId("5de0ca2209c85f7e66a02c2e") } },
  { $facet: {
    "groupByGender": [
      { $group: {
        _id: "$gender",
        averageLessThan18: {
          $avg: { $cond: { if: { $lt: ["$age_of_user", 18] }, then: "$rating", else: null } }
        },
        average18_45: {
          $avg: {
            $cond: { if: { $and: [ { $gte: ["$age_of_user", 18] },
              { $lt: ["$age_of_user", 45] } ] },
              then: "$rating", else: null }
          }
        },
        averageMoreThan45: {
          $avg: { $cond: { if: { $gte: ["$age_of_user", 45] }, then: "$rating", else: null } }
        },
        averageOfGender: { $avg: "$rating" }
      }
    ]
  }
}]
```

```

    } },
    { $project: { gender: "$_id", averageLessThan18: 1, average18_45: 1,
                  averageMoreThan45 : 1, averageOfGender: 1, _id: 0 } }
  ],
  "groupTotal": [
    { $group: {
      _id: null,
      averageLessThan18: {
        $avg: { $cond: { if: { $lt: [ "$age_of_user", 18 ] } , then: "$rating", else: null } }
      },
      average18_45: {
        $avg: {
          $cond: { if: { $and: [ { $gte: [ "$age_of_user", 18 ] },
                                { $lt : [ "$age_of_user", 45 ] } ] } ,
                    then: "$rating", else: null }
        }
      },
      averageMoreThan45: {
        $avg: { $cond: { if: { $gte: [ "$age_of_user", 45 ] } , then: "$rating", else: null } }
      }
    }
  ],
  { $project: { averageLessThan18: 1, average18_45: 1, averageMoreThan45 : 1, _id: 0 } }
] } }
])

```

The response time of the aggregation benefits of the index on the `_id.filmId` field previously defined for viewing the distribution of film ratings.

Calculate the recent mean rating of a film

Given the `_id` value of the film, the recent mean rating, which is the mean of the ratings given to a film in the last two years, can be obtained scanning the Rating as follows:

```
db.Rating.aggregate([
  { $match: { "_id.filmId": ObjectId("5de0ca2109c85f7e66a01c7c") } },
  { $group: {
    _id: null,
    recentMeanRating: {
      $avg: {
        $cond: { if: { $gte: [ { $toDate: "$timestamp" },
                              { $subtract: [ new Date(), 63072000000 ] } ] },
                  then: "$rating", else: null } }
      }
    }
  },
  { $project: { _id: 0, recentMeanRating: 1 } }
])
```

The response time of the aggregation benefits of the index on the field `_id.filmId` previously defined for viewing the distribution of film ratings.

Calculate the most recurrent film genres in intervals of expressed ratings

The statistic displays, for a single user and inside a chosen rating interval, how the film genres are distributed (e.g. in the interval [0,2] the distribution is 50% horror, 25% thriller, 25% noir). Given the `_id` value of the user and an interval of ratings [minRating, maxRating], the statistic is generated selecting the film IDs with a user rating inside the given interval and grouping the corresponding genres retrieved from the Film collection. For example, given the interval [3,4]:

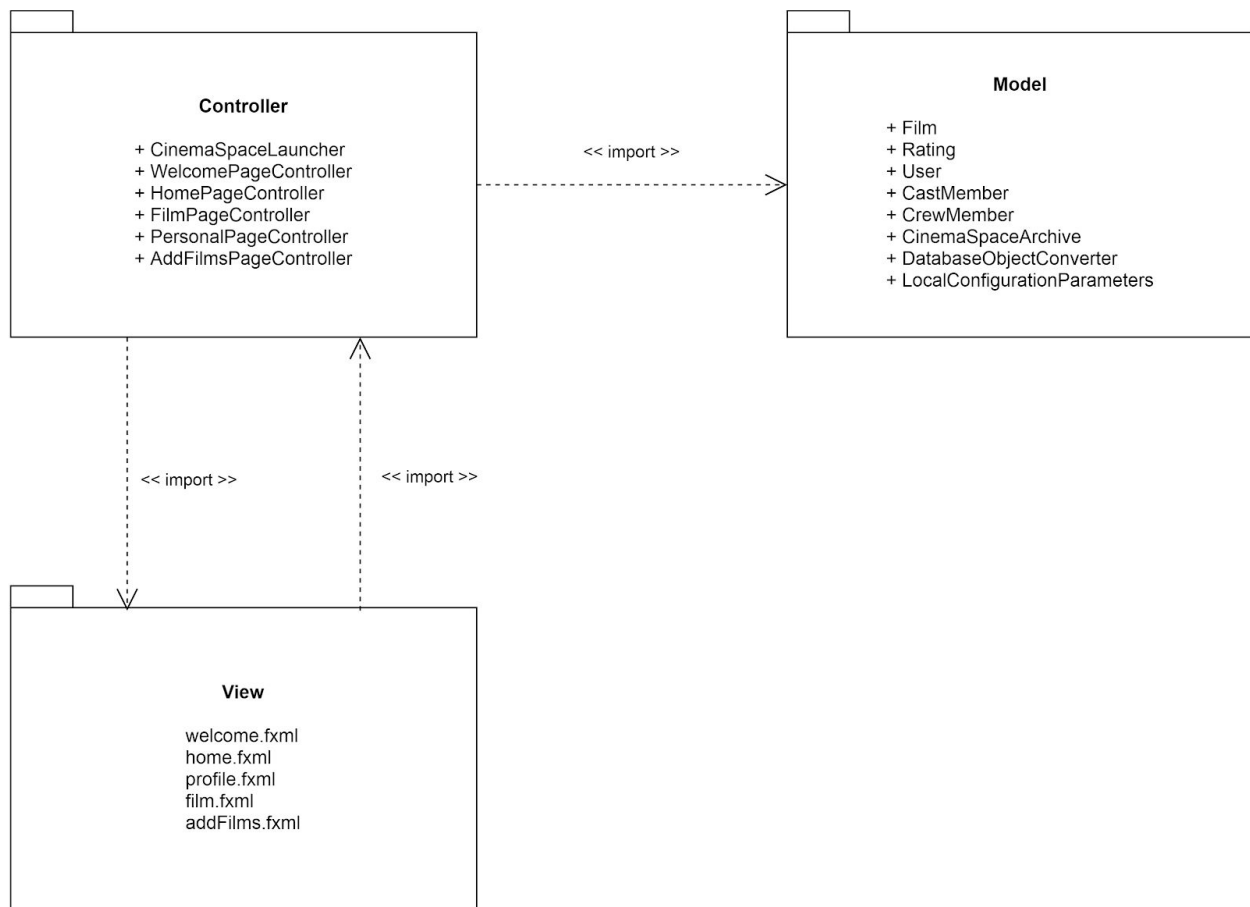
```
db.Rating.aggregate([
  { $match: {
    "_id.userId": ObjectId("5de0cc65bab5a6cf3157658f"),
    "rating": { $gte: 3, $lte: 4 }
  } },
  { $lookup: { from: "Film", localField: "_id.filmId", foreignField: "_id", as: "filmDetails" } },
  { $unwind: "$filmDetails" },
  { $project: { _id: 0, "filmId": "$_id.filmId", "genres": "$filmDetails.genres" } },
  { $unwind: "$genres" },
  { $group: {
    _id: "$genres",
    count: { $sum: 1 }
  } },
  { $project: { _id: 0, genre: "$_id", count: 1 } }
])
```

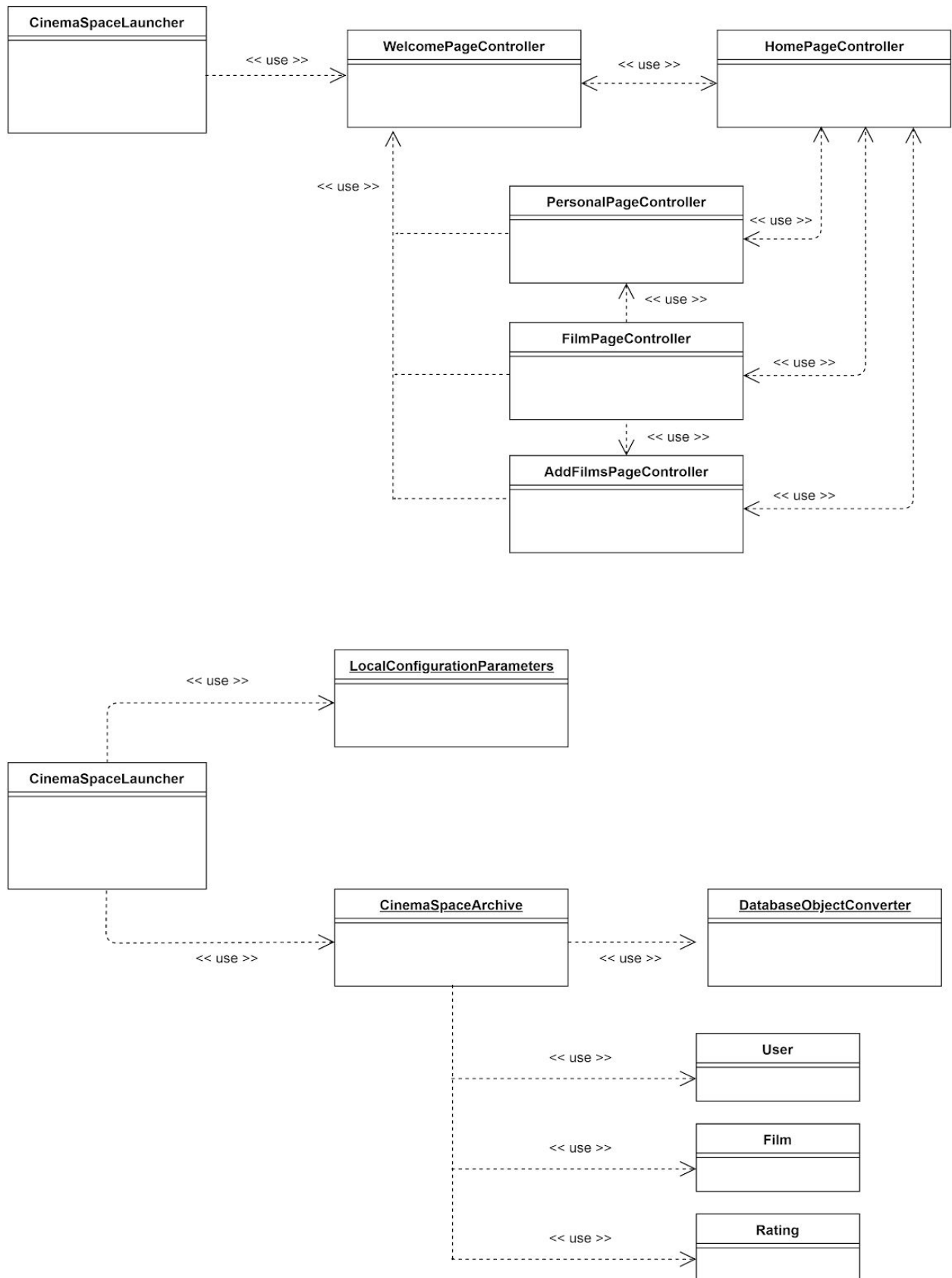
The response time of this aggregation can be improved defining a compound index on the `_id.userId` and `rating` fields, as follows:

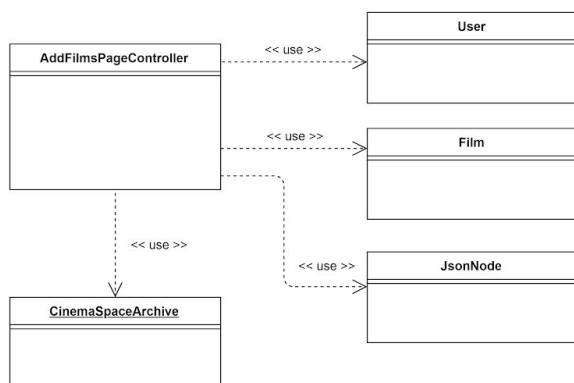
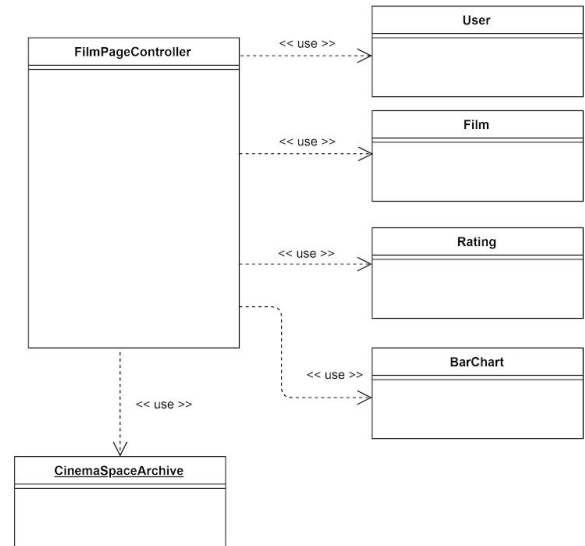
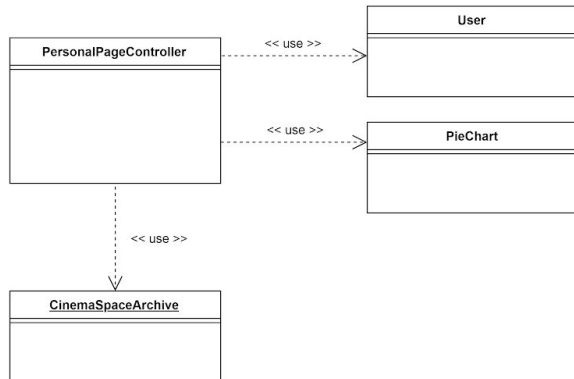
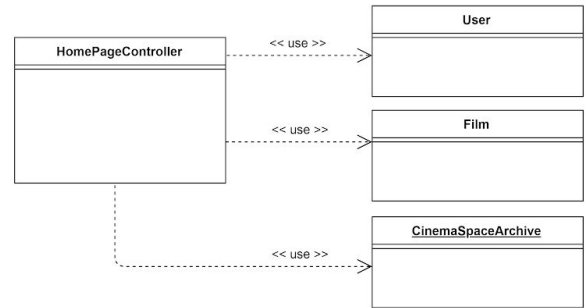
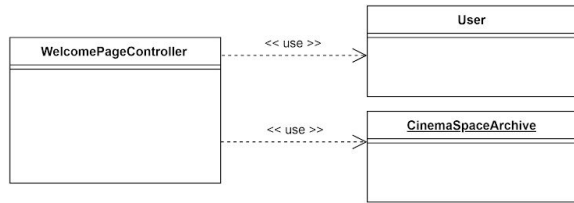
```
db.Rating.createIndex({ "_id.userId" : 1, rating: -1 }, { name: "userRating" })
```

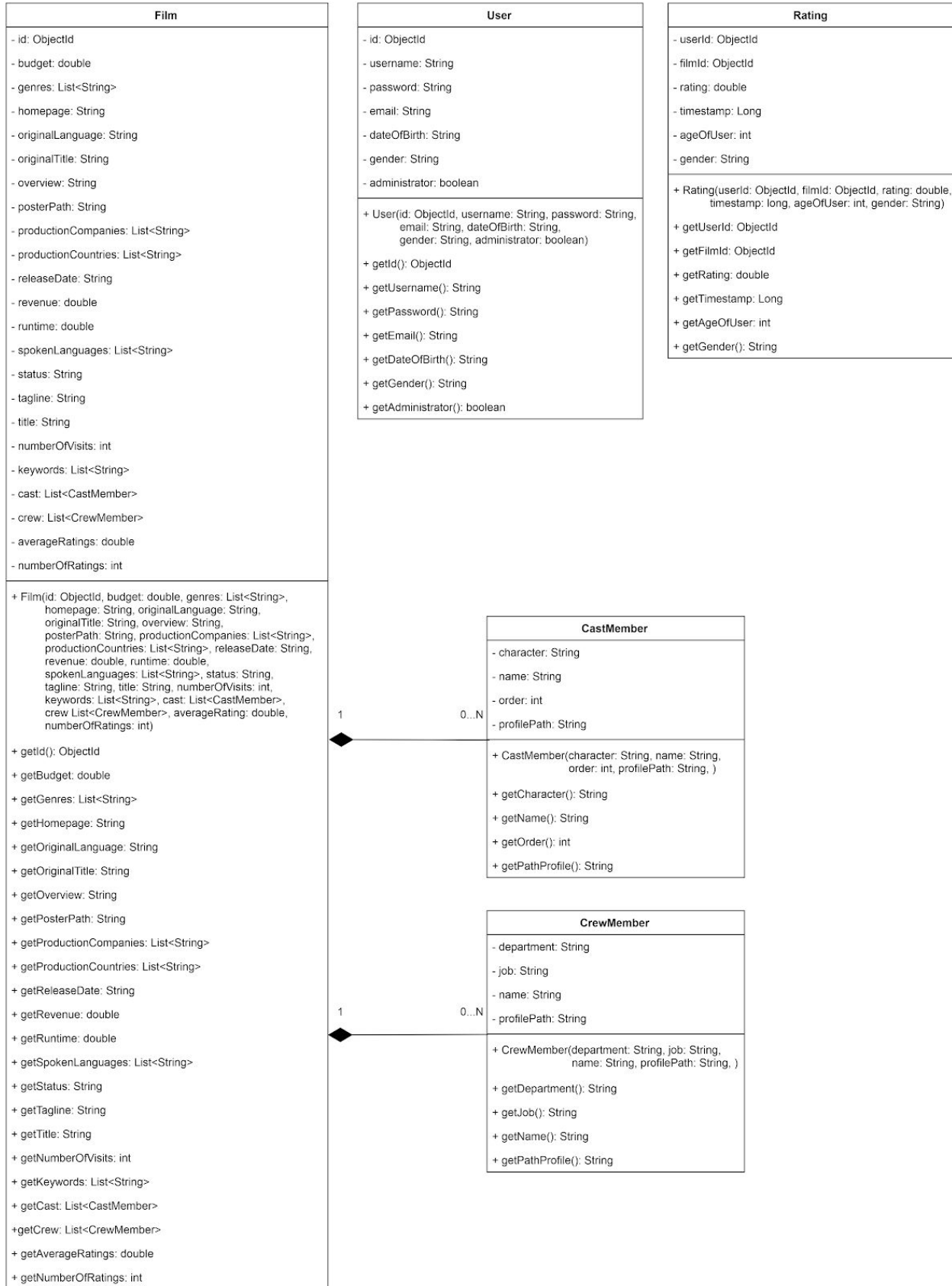
7 IMPLEMENTATION

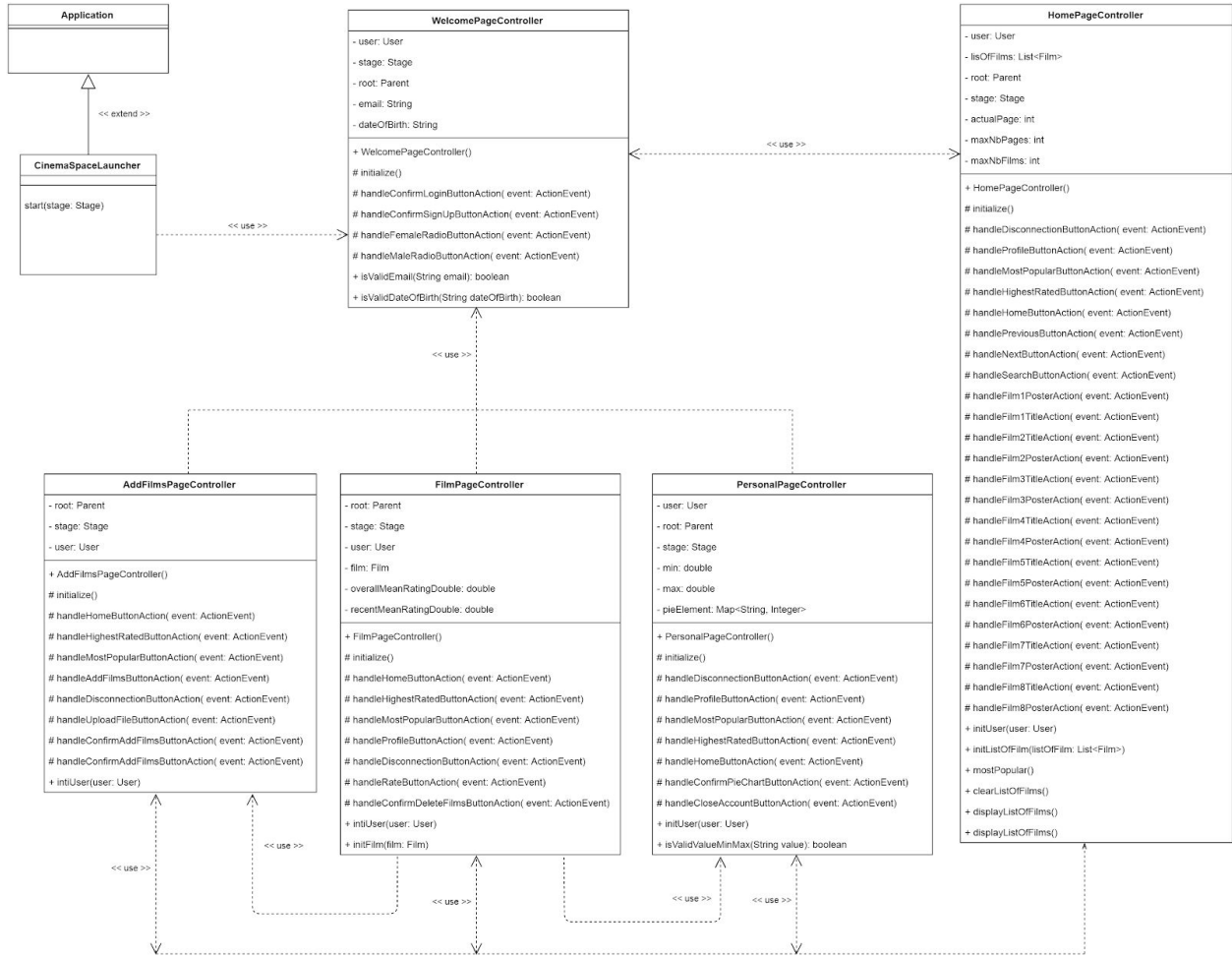
7.1 Class diagram

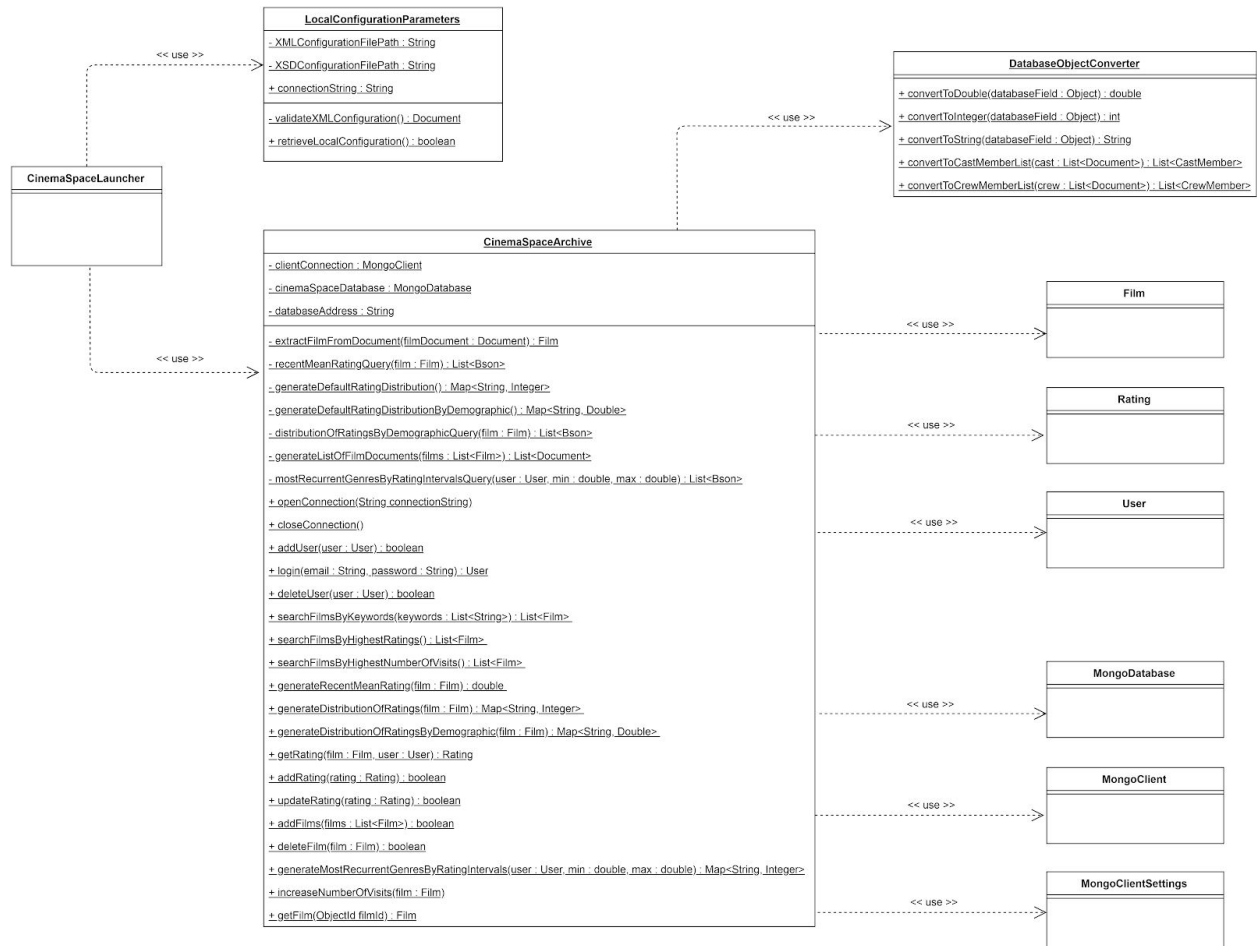












Class responsibilities

User: represents a user in the domain of CinemaSpace.

Film: represents a film in the domain of CinemaSpace.

Rating: represents a rating in the domain of CinemaSpace.

CinemaSpaceLauncher: manages the launch of the application, coordinating the retrieval of the configuration file, the connection to the database and the initialization of the GUI. After this phase, it leaves the control of the application flow to the WelcomePageController.

WelcomePageController: manages the welcome page, coordinating the signup and login services offered to the user. If a login is successfully requested, it leaves the control of the application flow to the HomePageController.

HomePageController: manages the home page and it is responsible for the visualization of the films searched by keywords, by highest ratings and by number of visits. It leaves the control of the application to the FilmPageController, PersonalPageController, AddFilmsPageController or WelcomePageController according to the user requests.

FilmPageController: manages the film page, showing the associated information and statistics, and gives the possibility to use the rating system of the community. It leaves the control of the application to the HomePageController, PersonalPageController, AddFilmsPageController or WelcomePageController according to the user requests.

PersonalPageController: manages the personal profile page of a user, showing the associated information/statistics and giving the possibility to close the account. It leaves the control of the application to the HomePageController or WelcomePageController according to the user requests.

AddFilmsPageController: manages the administrator page for adding films to the database from JSON files. It leaves the control of the application to the HomePageController or WelcomePageController according to the user requests.

CinemaSpaceArchive: manages the connection to the database, responding to the specific requests of the different page controllers. It returns opportune objects of type List<Film>, User, Rating and Map< > (the latter in the case of statistic distribution requests).

DatabaseObjectConverter: offers utility methods for the conversion of types. It is used by CinemaSpaceArchive to manage the retrieval of documents with non-fixed types from the database.

LocalConfigurationParameters: retrieves and validates the local configuration stored in an XML file, making available its parameters to all the classes.

Note: the connection to the database is managed by a pool, which is initialized at the start of the application and closed at the end: in particular, the latter is ensured by binding the opportune method to the *onCloseRequest* window event. Both these steps are executed in the CinemaSpaceLauncher class.

7.2 Replica set

As a solution for the fault tolerance, high availability and tolerance to network partitions requirements, the deployment of the same dataset with multiple copies on different database servers is used. MongoDB gives the opportunity to configure replica sets: for each set, the topology consists of a single primary server, which responds to read and write operations, and in a certain number of secondary servers, which ensure data replication.

In the chosen configuration:

- a replica set consisting of 3 servers is deployed. This means that a majority of 2 members is needed to elect a primary server (a primary can only stay primary if a majority can be reached);
- the secondary servers can respond to read operation;
- a read operation is completed when the data are received from one server;
- a write operation is completed when the data are written in one server.

Initialization of the mongod processes

Each node in the set must have a separate directory: in our case, the three directories *cinemaReplica1*, *cinemaReplica2* and *cinemaReplica3* are used, within the replica set domain *cinemaReplicaSet*. To initialize the set, the following commands must be executed in different shells:

```
mongod --replSet cinemaReplicaSet --dbpath cinemaReplica1 --port 27017 --oplogSize 200
mongod --replSet cinemaReplicaSet --dbpath cinemaReplica2 --port 27018 --oplogSize 200
mongod --replSet cinemaReplicaSet --dbpath cinemaReplica3 --port 27019 --oplogSize 200
```

where “--replSet” sets the replica set name, “--dbpath” sets the directory for the data files, “--port” sets the port where the service is going to be available and “--oplogSize” sets the size in MB to use for the oplog file.

Configuration of the set

To effectively start up the replica set, the creation of a configuration is needed: the later lists every member node and must be sent to one of the previously defined mongod processes, which will propagate it to the others. After connecting to a mongo shell, for instance using the command `mongo --port 27017`, the configuration is created as follows:

```
> rsconf = {
  _id: "cinemaReplicaSet",
  members: [
    { _id: 0, host: "localhost:27017"},
    { _id: 1, host: "localhost:27018"},
    { _id: 2, host: "localhost:27019"}
  ]
}
```

where “_id” is the name of the replica set and each member must be specified with a unique integer (“_id”) and a hostname. Then, the defined configuration can be applied:

```
> rs.initiate(rsconf)
```

The status of the replica set can be checked executing:

```
> rs.status()

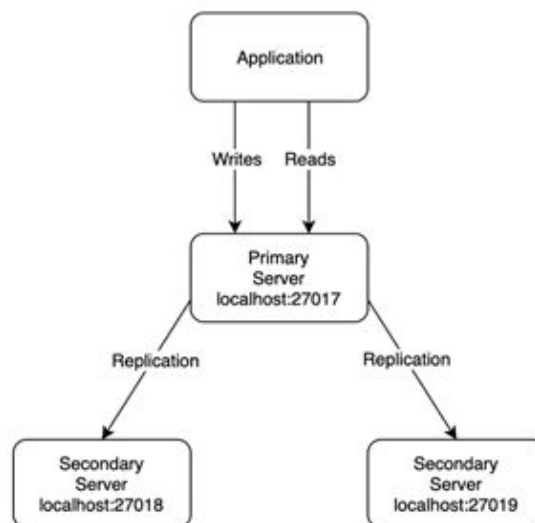
{
  "set" : "cinemaReplicaSet",
  "date" : ISODate("2019-12-10T09:04:16.002Z"),
  "myState" : 1,
  ...
  "members" : [
    {
      "_id" : 0,
      "name" : "localhost:27017",
      "health" : 1,
      "state" : 1,
      "stateStr" : "PRIMARY",
      ... },
    ...
  ]
}
```

```

{
  "_id": 1,
  "name": "localhost:27018",
  "health": 1,
  "state": 2,
  "stateStr": "SECONDARY",
  ...
},
{
  "_id": 2,
  "name": "localhost:27019",
  "health": 1,
  "state": 2,
  "stateStr": "SECONDARY",
  ...
}
],
...
}
}

```

In the “members” attribute, the servers participating in the replica are listed, where localhost:27017 was chosen as the primary. In the actual configuration, the situation is the following:



Import data into the replica set

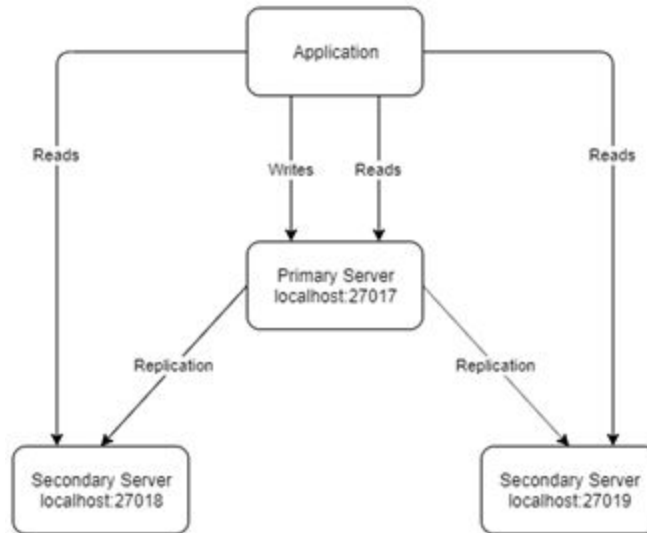
When all the servers are running, it is possible to import the database dump on the primary server, with an automatic replication process on the secondary servers.

Advanced configuration for read and write operations

By default, the primary server responds to read and write operations, while the secondary servers manage only the replication task. This behaviour cannot be modified within the replica set configuration, i.e. executing `rs.initiate()`, but can be manipulated selectively in order to serve client requests with specific needs. This means that the choice of how many servers should respond to a query must be specified in the application, exploiting the options offered by the MongoDB driver, with the only limitation of the impossibility to write on secondary servers. In the case of this application, the MongoDB Java driver offers three configuration objects: `ReadPreference`, `ReadConcern` and `WriteConcern`. They can be used as follows:

- `ReadPreference.primaryPreferred()` makes the client able to read from the primary if it is available; if it is unavailable, it reads from the secondary servers;
- `ReadConcern.LOCAL` makes the client able to get data with no guarantee that the data have been written to the majority of the replica set members. This increases the responsiveness of read operations;
- `WriteConcern(1)` makes the write operations acknowledged when they have been propagated to the primary in a replica set (not to the majority of the set).

```
1. String connectionString = "mongodb://" +
2. "localhost:27017,localhost:27018,localhost:27019/?replicaSet=cinemaReplicaSet";
3.
4. Builder connectionSettingsBuilder = MongoClientSettings.builder();
5. connectionSettingsBuilder.readPreference(ReadPreference.primaryPreferred());
6. connectionSettingsBuilder.readConcern(ReadConcern.LOCAL);
7. connectionSettingsBuilder.writeConcern(new WriteConcern(1));
8. connectionSettingsBuilder.applyConnectionString(new ConnectionString(connectionString));
9.
10. MongoClientSettings connectionOptions = connectionSettingsBuilder.build();
11. MongoClient clientConnection = MongoClient.create(connectionOptions);
```

Failure recovery test

- Enter to the mongo shell of the replica elected as the primary. In this example:

mongo --port 27017

> db.isMaster()

```

{
  "hosts" : [
    "localhost:27017",
    "localhost:27018",
    "localhost:27019"
  ],
  "setName" : "cinemaReplicaSet",
  "setVersion" : 1,
  "ismaster" : true,
  "secondary" : false,
  "primary" : "localhost:27017",
  "me" : "localhost:27017",
  "electionId" : ObjectId("7fffffff000000000000000005"),
  ...
}

```

- Proceed to shut down the server:

```
> db.adminCommand({"shutdown" : 1})
```

- Check one of the secondaries servers to see which one was elected as primary:

```
> secondaryConn = new Mongo("localhost:27019")
> secondaryDB = secondaryConn.getDB("CinemaSpace")
> secondaryDB.isMaster()
```

```
{
  "hosts" : [
    "localhost:27017",
    "localhost:27018",
    "localhost:27019"
  ],
  "setName" : "cinemaReplicaSet",
  "setVersion" : 1,
  "ismaster" : true,
  "secondary" : false,
  "primary" : "localhost:27019",
  "me" : "localhost:27019",
  "electionId" : ObjectId("7fffffff000000000000000006"),
  ...
}
}
```

"ismaster" is true inside the secondary server localhost:27019, which is the new primary. The replica set status is:

```
{
  "set" : "cinemaReplicaSet",
  "date" : ISODate("2019-12-12T18:53:06.236Z"),
  "myState" : 1,
  ...
  "members" : [
    {
      "_id" : 0,
      "name" : "localhost:27017",
      "ip" : "127.0.0.1",
      "health" : 0,
      "state" : 8,
```

```

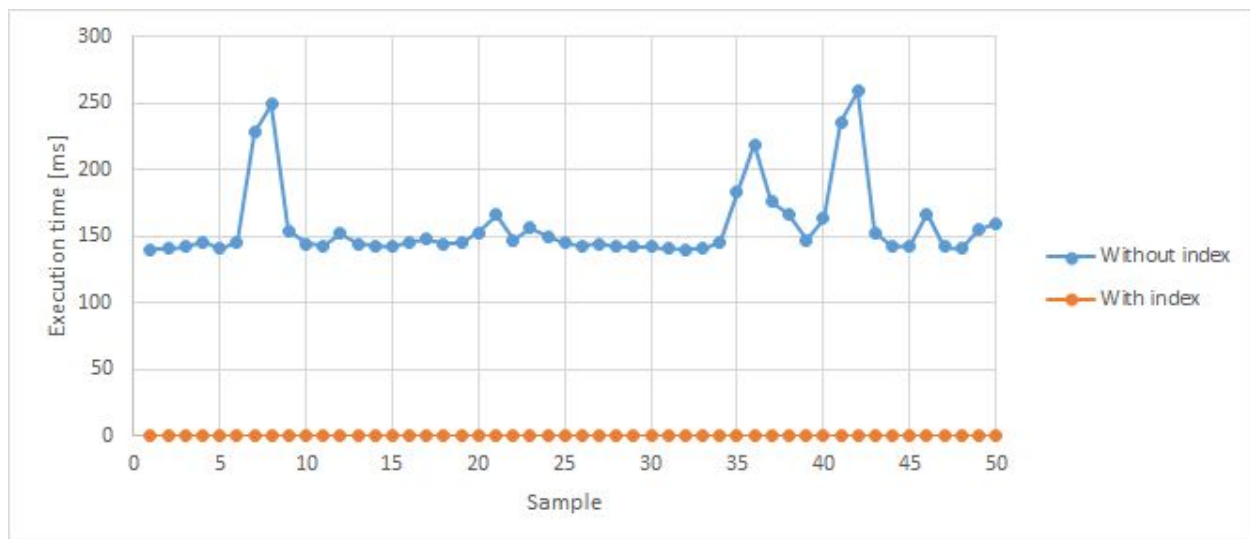
    "stateStr" : "(not reachable/healthy)",
    "uptime" : 0,
    ...
    "lastHeartbeatMessage" : "not running with --replSet",
    ...
  },
  {
    "_id" : 1,
    "name" : "localhost:27018",
    "ip" : "127.0.0.1",
    "health" : 1,
    "state" : 1,
    "stateStr" : "PRIMARY",
    "uptime" : 195,
    ...
  },
  {
    "_id" : 2,
    "name" : "localhost:27019",
    "ip" : "127.0.0.1",
    "health" : 1,
    "state" : 2,
    "stateStr" : "SECONDARY",
    "uptime" : 99,
    ...
  }
],
...
}

```

7.3 Performances of database indexes

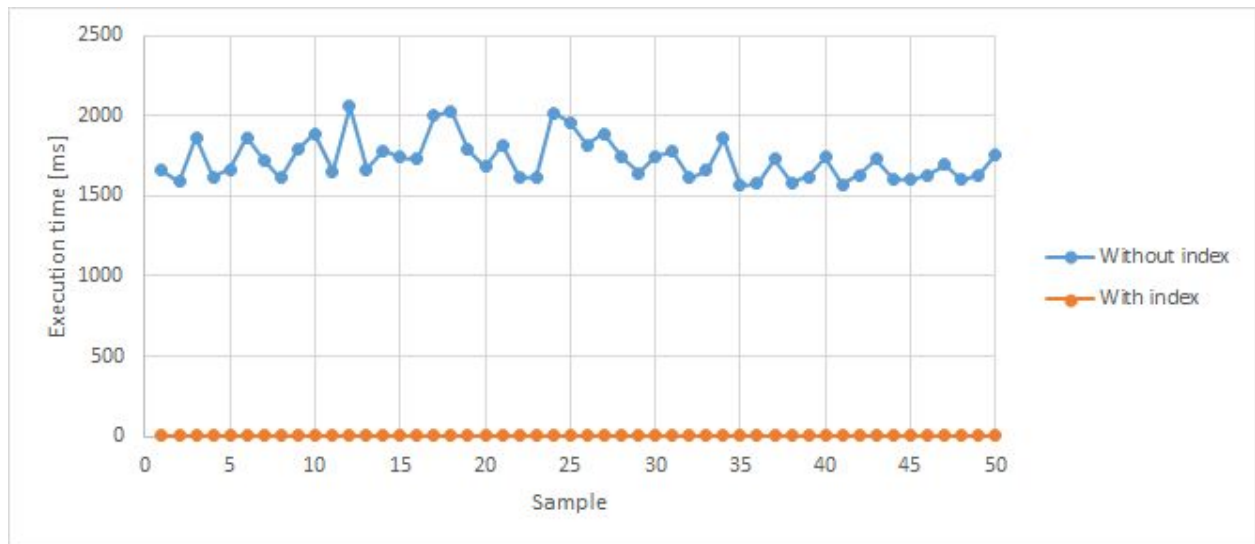
In the following, the execution times of the queries defined in the “Database structure” section and of insert/update/delete queries are presented, both in the case of a database without indexes and with indexes. The performance information were collected executing the queries directly in the mongo shell and exploiting the *explain("executionStats")* command; the only exception is represented by the delete() queries, where the execution times were obtained with a subtraction between two Javascript Date objects, since the previous command is not supported. To evaluate the in-memory performances, the measurements were collected after performing at least one time the related query.

Login



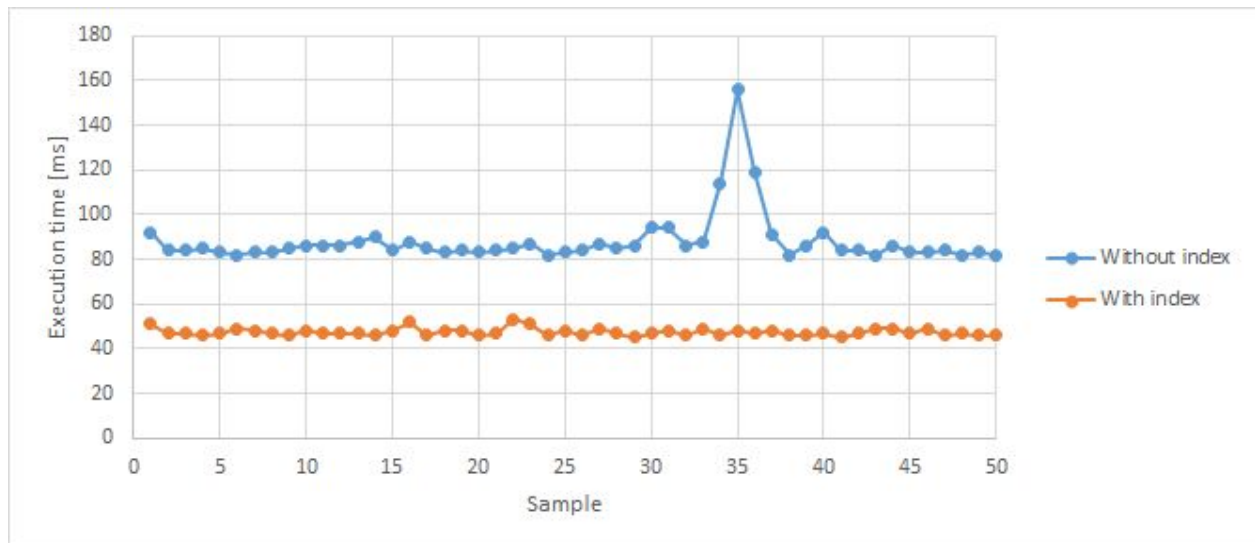
	Total keys examined	Total docs examined	Sample mean	95% CI
Without indexes	0	270897	157.78	[149.7237, 165.8363]
With indexes	1	1	0 [< 1 ms]	0 [<1 ms]

Search films by highest number of visits



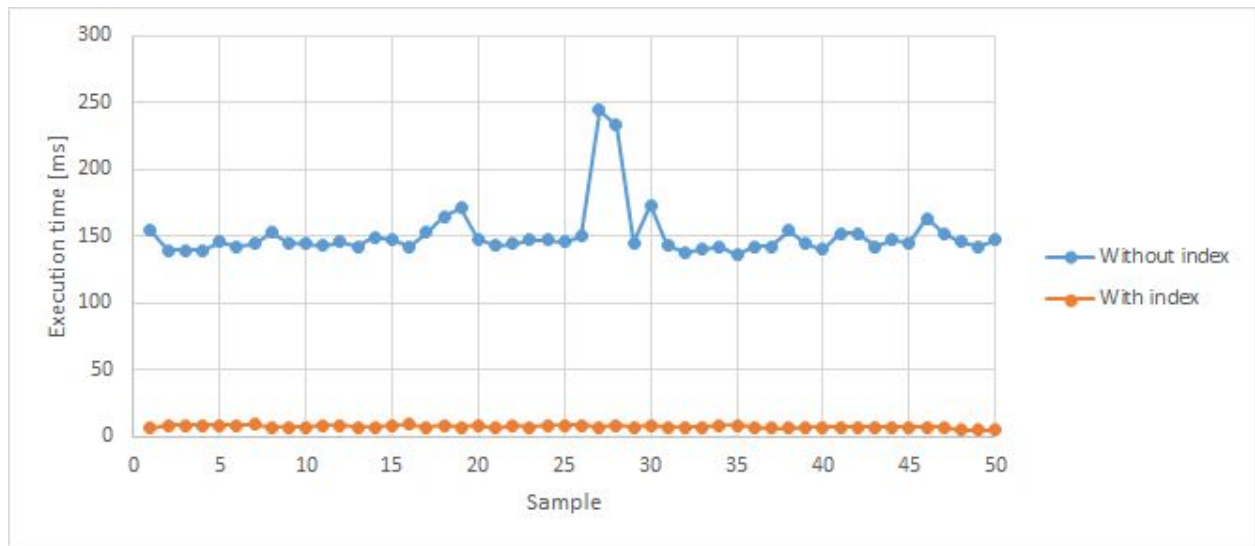
	Total keys examined	Total docs examined	Sample mean	95% CI
Without indexes	0	46629	1729	[1692.691, 1765.309]
With indexes	50	50	2	2

Search films by keywords



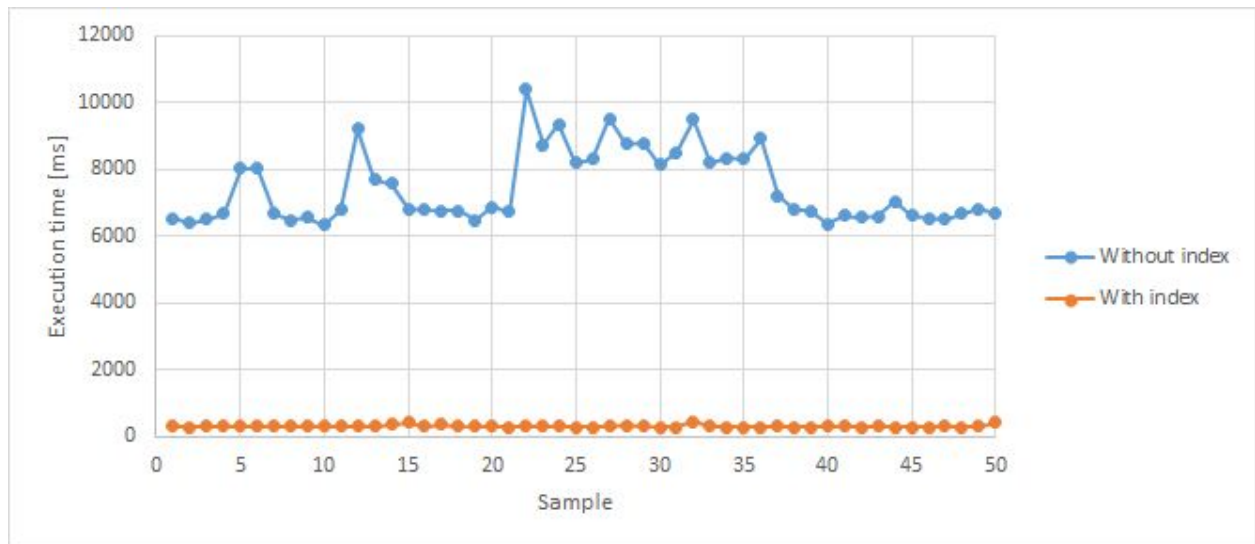
	Total keys examined	Total docs examined	Sample mean	95% CI
Without indexes	0	46629	88.04	[84.718, 91.362]
With indexes	12349	12349	47.44	[46.974, 47.905]

Search films by highest ratings



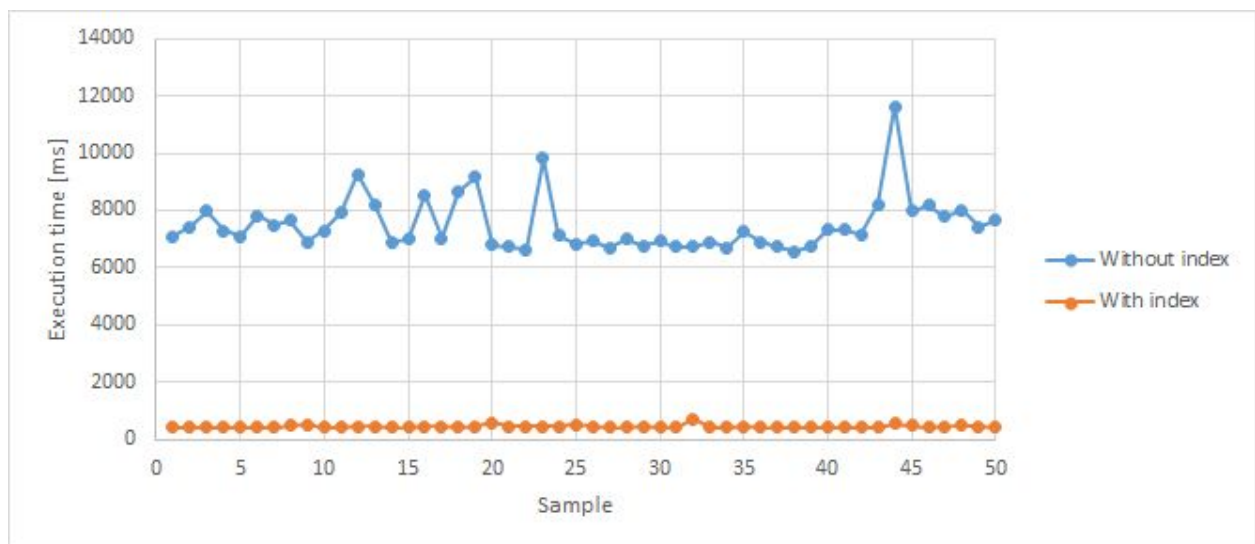
	Total keys examined	Total docs examined	Sample mean	95% CI
Without indexes	0	46629	150.64	[145.178, 156.102]
With indexes	140	140	7.24	[6.992, 7.487]

View distribution of ratings



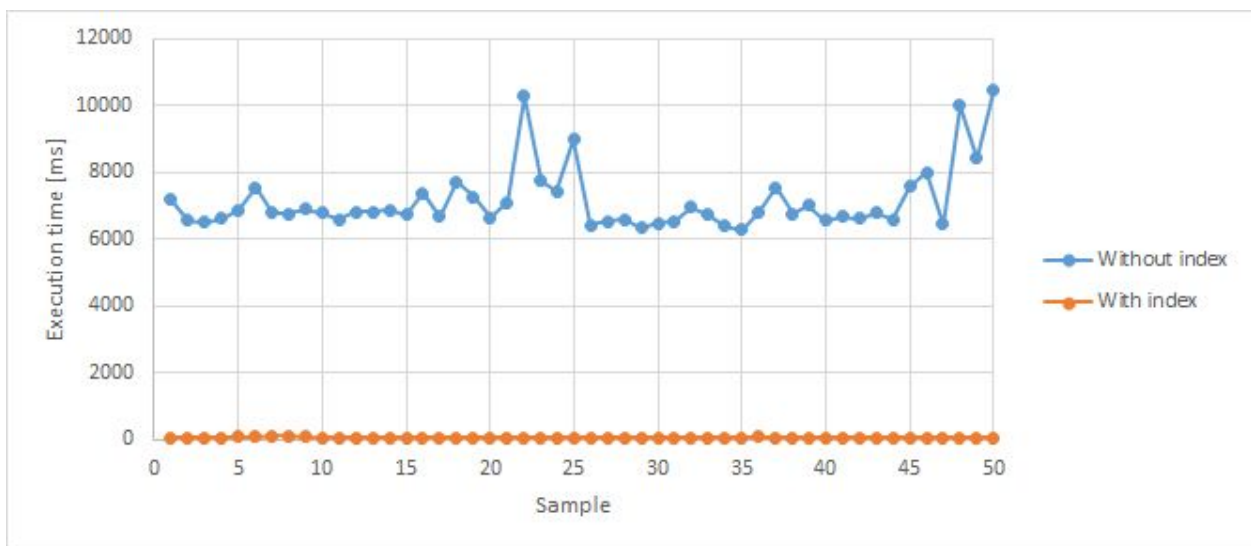
	Total keys examined	Total docs examined	Sample mean	95% CI
Without indexes	0	11436568	7442.96	[7145.145, 7740.775]
With indexes	91082	91082	305.92	[296.701, 315.139]

View distribution of ratings by demographic



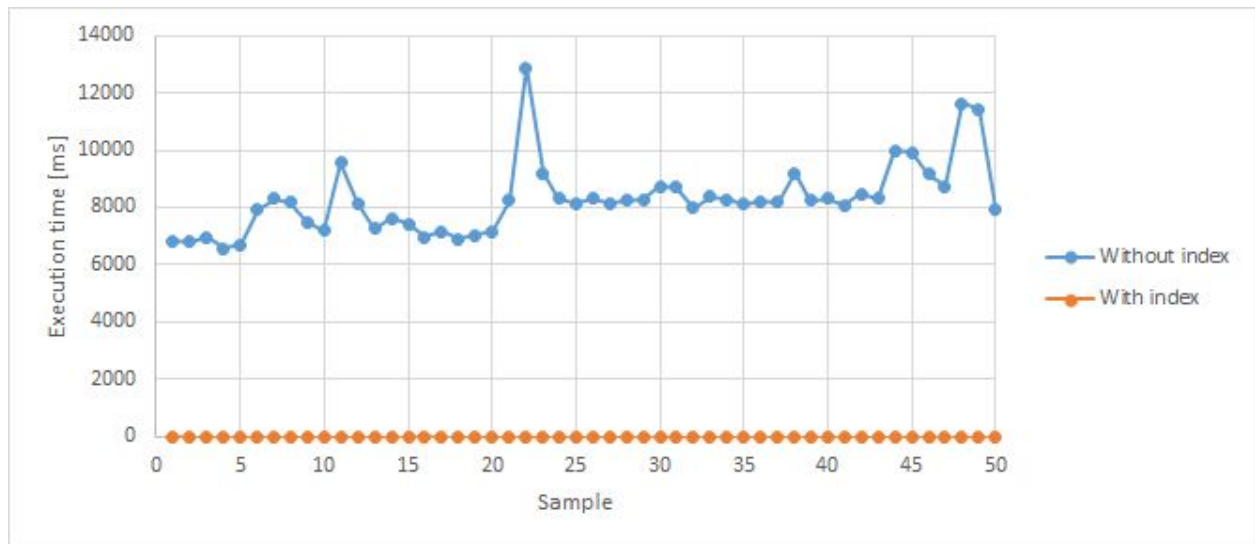
	Total keys examined	Total docs examined	Sample mean	95% CI
Without indexes	0	11436568	7498.3	[7234.312, 7762.288]
With indexes	91082	91082	450.08	[434.868, 465.292]

Calculate the recent mean rating



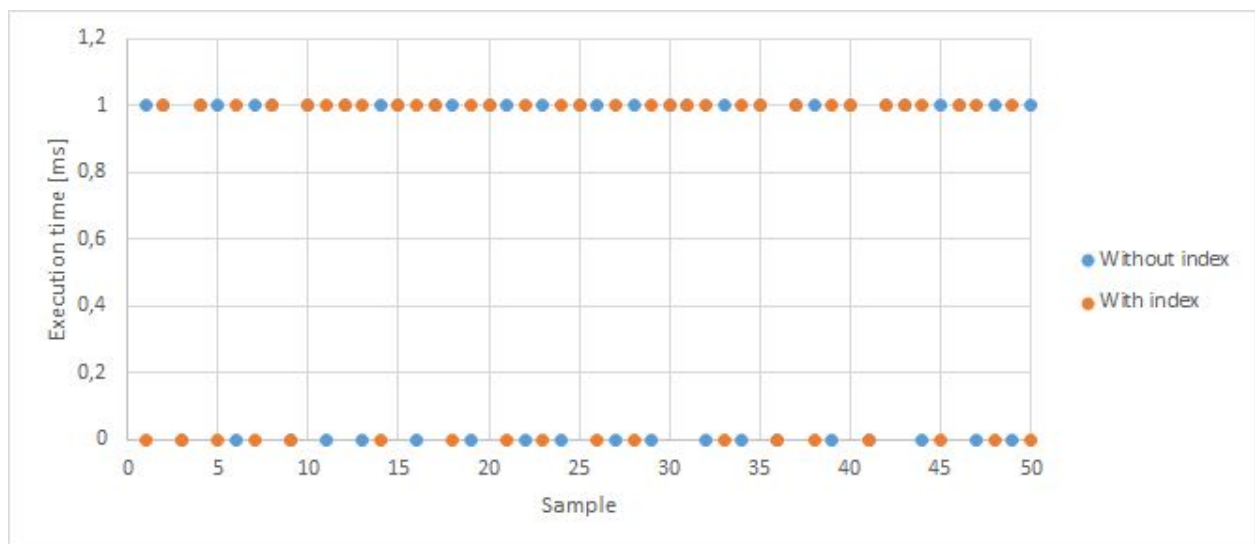
	Total keys examined	Total docs examined	Sample mean	95% CI
Without indexes	0	11436568	7140.74	[6874.866, 7406.614]
With indexes	15258	15258	64.62	[61.745, 67.495]

Calculate the most recurrent film genres in intervals of expressed ratings



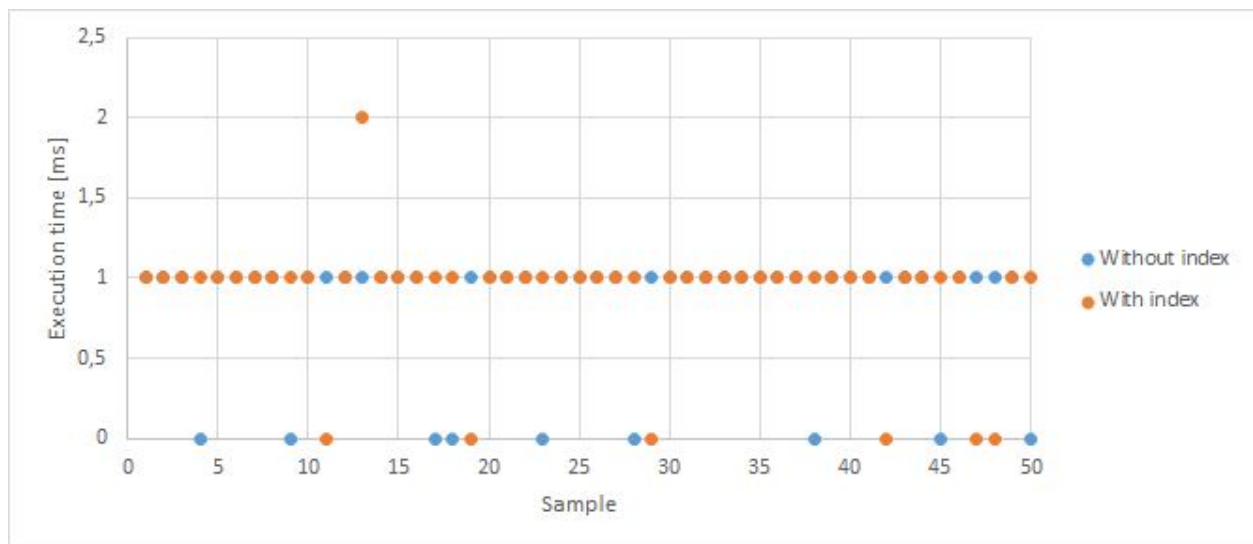
	Total keys examined	Total docs examined	Sample mean	95% CI
Without indexes	0	11436568	8276.92	[7930.609, 8623.231]
With indexes	13	13	2.04	[1.985, 2.095]

Insert a user



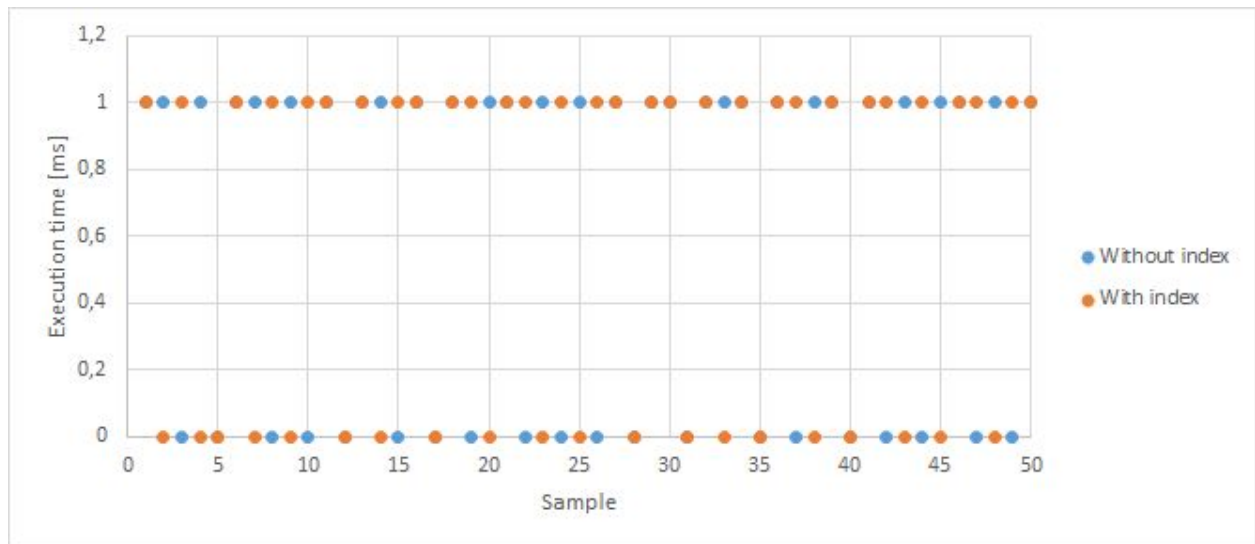
	Sample mean	95% CI
Without indexes	0.62	[0.484, 0.756]
With indexes	0.64	[0.506, 0.774]

Insert a film



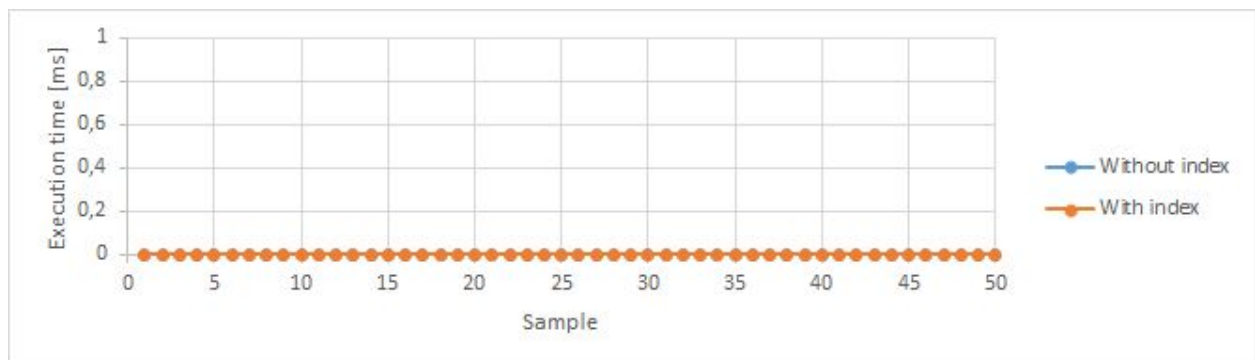
	Sample mean	95% CI
Without indexes	0.82	[0.712, 0.927]
With indexes	0.9	[0.799, 1.001]

Insert a rating



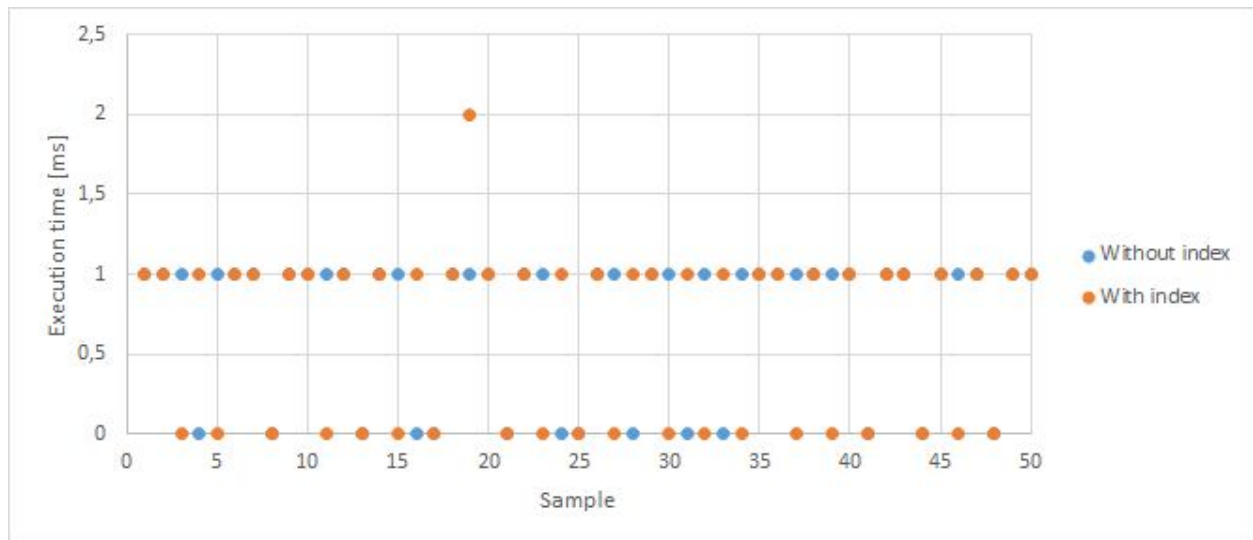
	Sample mean	95% CI
Without indexes	0.6	[0.463, 0.737]
With indexes	0.6	[0.463, 0.737]

Update the number of visits/average rating/rating value



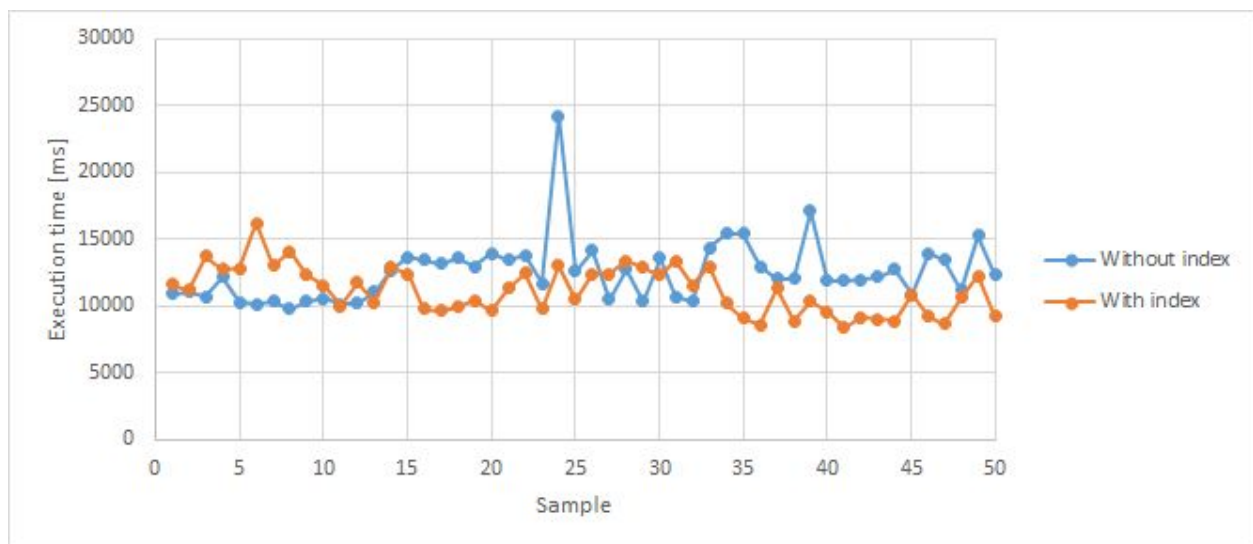
All the collected samples had a measured value equal to 0 (< 1 ms).

Delete a film



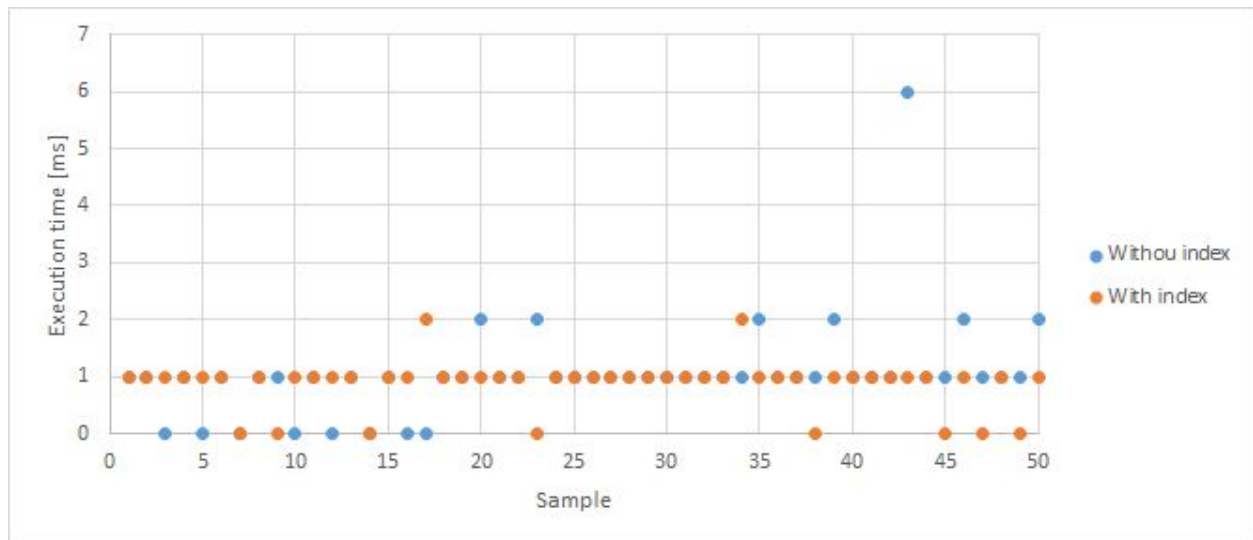
	Sample mean	95% CI
Without indexes	0.72	[0.594, 0.845]
With indexes	0.62	[0.473, 0.767]

Delete all the ratings of a film



	Sample mean	95% CI
Without indexes	12519.02	[11858.3, 13179.74]
With indexes	11156.84	[10669.45, 11644.23]

Delete a user



	Sample mean	95% CI
Without indexes	1.06	[0.813, 1.307]
With indexes	0.88	[0.759, 1.001]

The collected data show that the use of indexes gives a consistent boost in terms of response to read operations without noticeable losses of performance in write operations, hence it fits our requirements.

7.4 Test manual

To be able to connect the application with the database, an XML configuration file with the addresses of the replica set members must be specified, with the following format:



The configuration is validated at the start using a pre-defined XSD file.

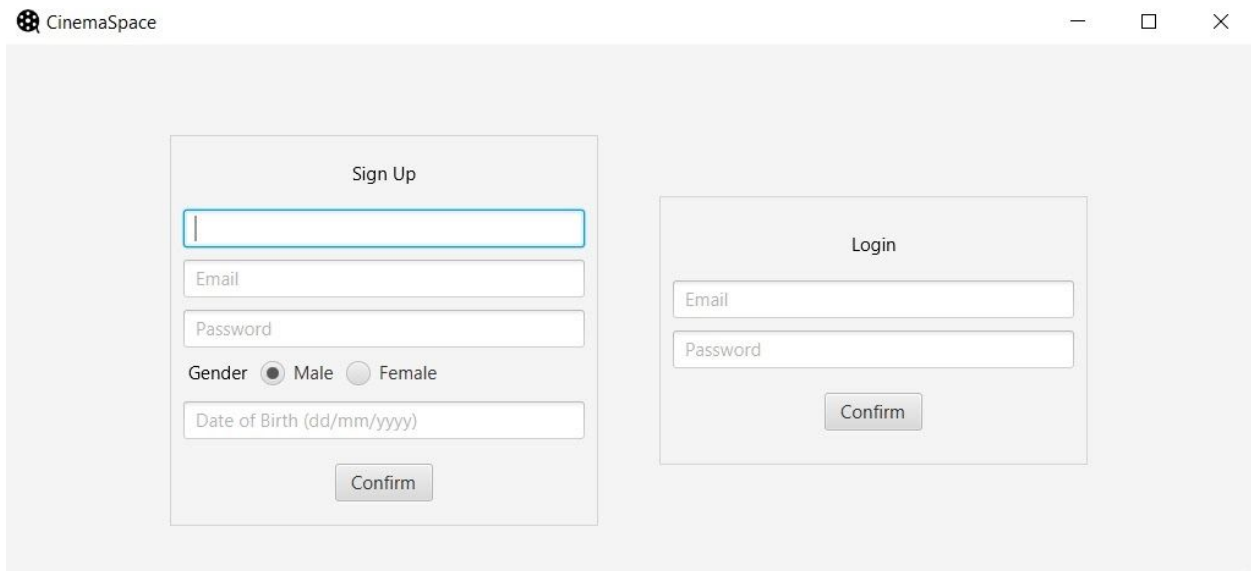
7.4.1 Client user

Sign Up

Preconditions: None

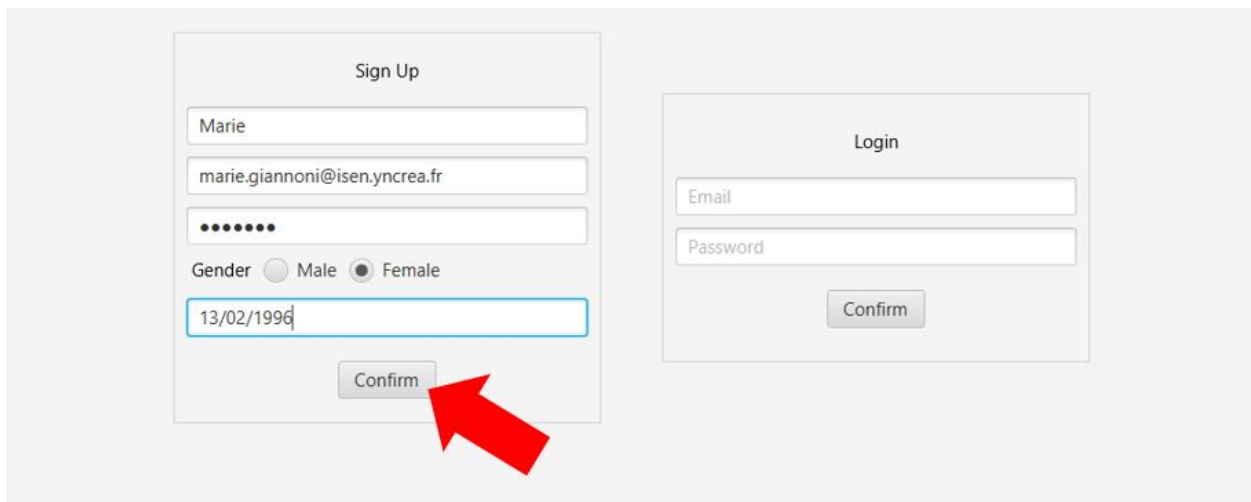
Steps:

1. The user opens the application.



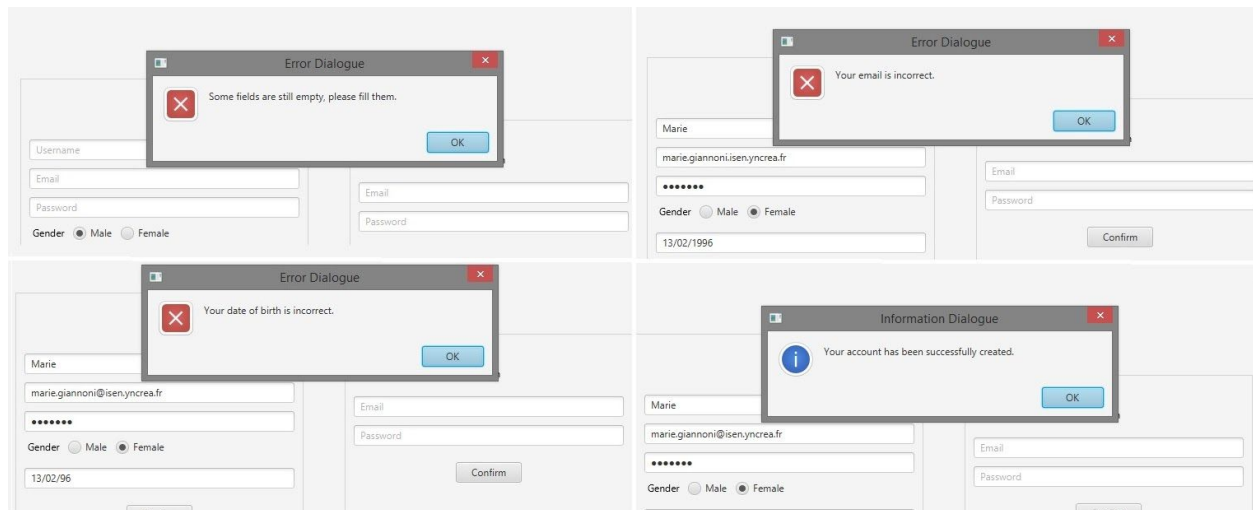
The screenshot shows a web application window titled "CinemaSpace". It contains two forms: "Sign Up" and "Login". The "Sign Up" form has fields for a first name (empty), email (empty), password (empty), gender (radio buttons for Male and Female, with Male selected), and date of birth (empty, with a placeholder "dd/mm/yyyy"). A "Confirm" button is at the bottom. The "Login" form has fields for email (empty) and password (empty), with a "Confirm" button below.

2. The user fills in all the fields in the Sign Up form and click on the “Confirm” button.

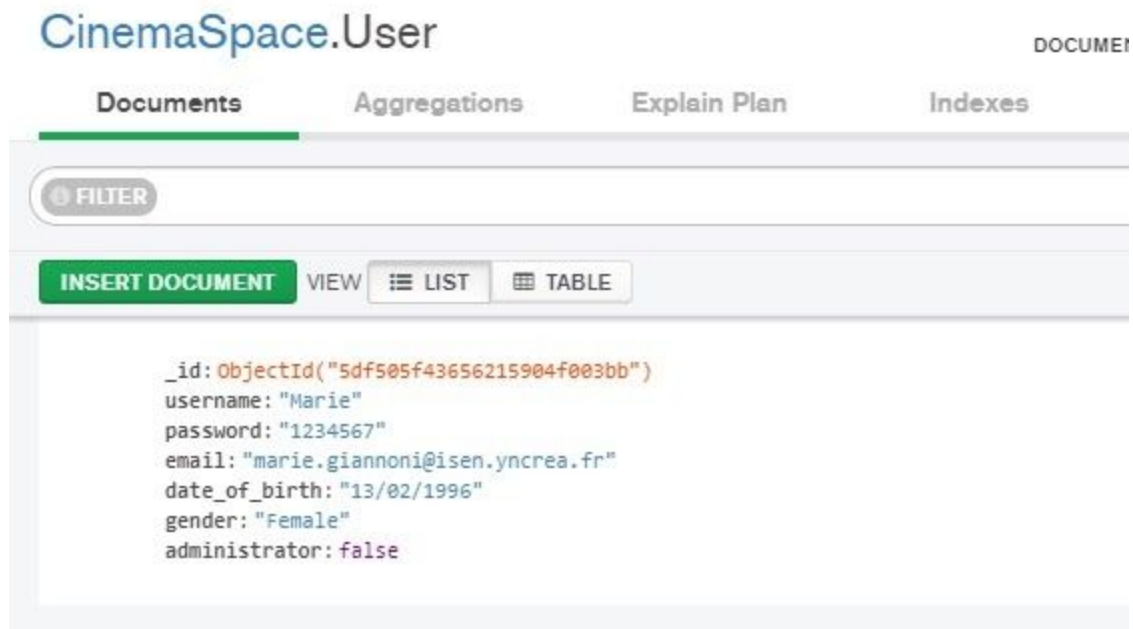


The screenshot shows the same "CinemaSpace" application window, but the "Sign Up" form is now filled out. The first name field contains "Marie", the email field contains "marie.giannoni@isen.yncrea.fr", the password field contains seven dots, the gender is now "Female" (radio button selected), and the date of birth field contains "13/02/1996". A red arrow points to the "Confirm" button at the bottom of the "Sign Up" form. The "Login" form remains empty.

The email must respect the pattern abc@efg.h, while the date of birth must be in the format dd/mm/yyyy. If these patterns are not respected, the application will show an error message.



Database status after the insertion

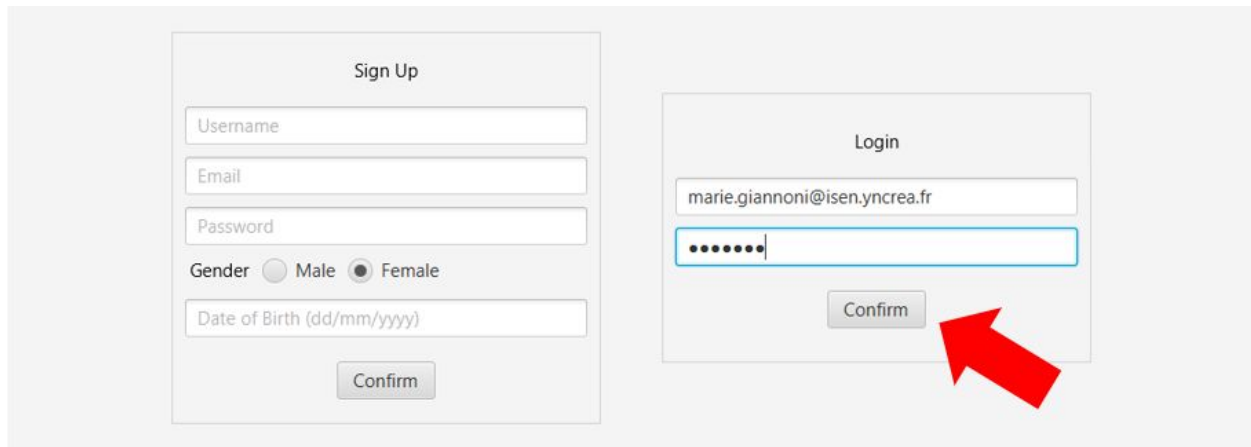


Log in

Preconditions: The user must have an account. The user must be logged in.

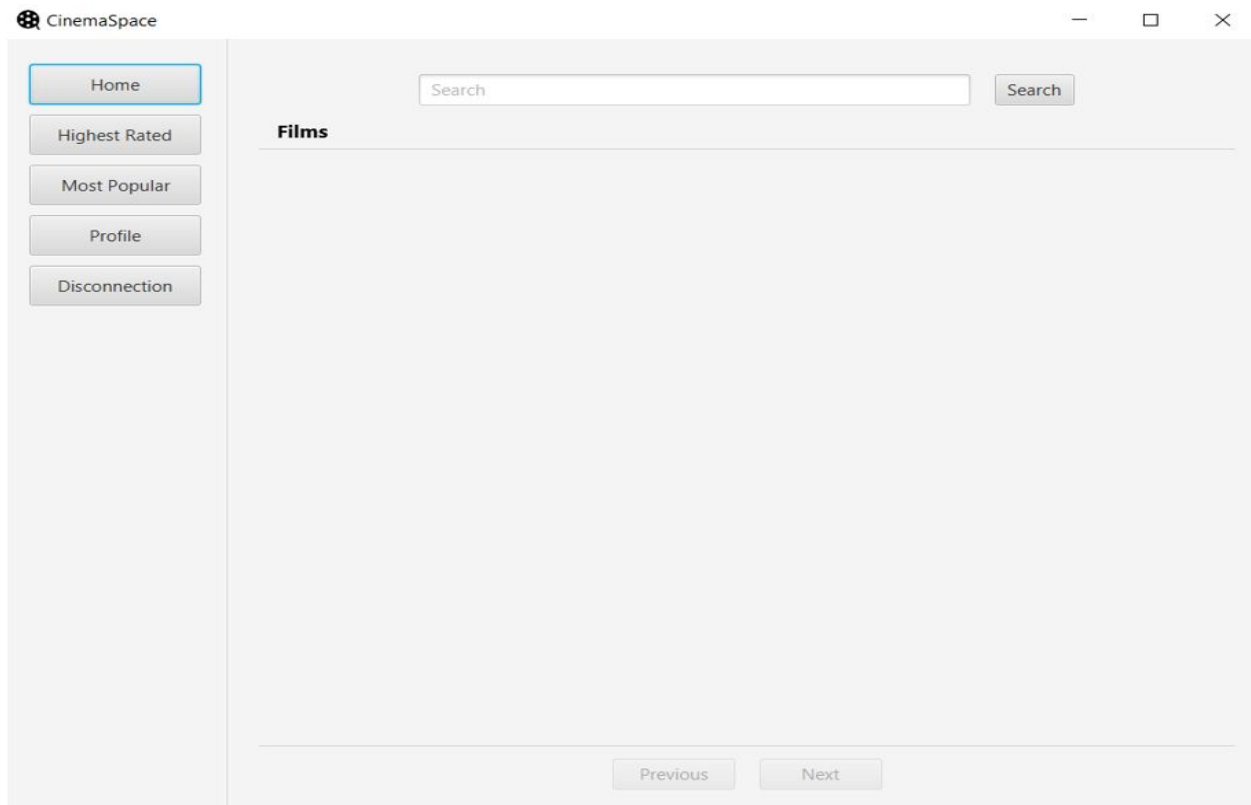
Steps:

1. The user opens the application.
2. The user enters email and password in the Login section and click on the “Confirm” button



The screenshot shows two side-by-side forms on a light gray background. The left form is titled 'Sign Up' and contains fields for 'Username', 'Email', 'Password', 'Gender' (with radio buttons for 'Male' and 'Female', where 'Female' is selected), and 'Date of Birth (dd/mm/yyyy)'. A 'Confirm' button is at the bottom. The right form is titled 'Login' and contains an email field with 'marie.giannoni@isen.yncrea.fr' and a password field with masked characters '.....'. A 'Confirm' button is below the password field. A large red arrow points to the 'Confirm' button in the Login form.

3. The “Home” page is displayed.

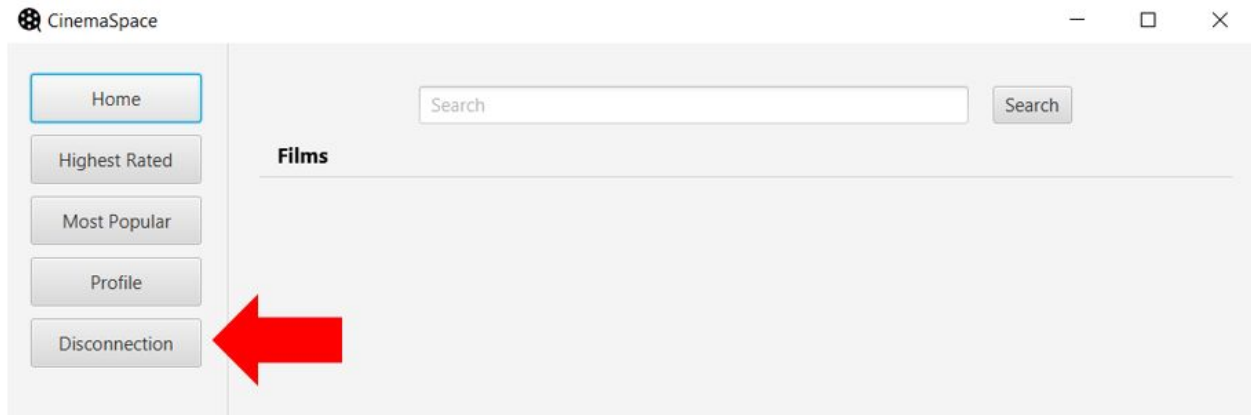


Log Out

Preconditions: The user must have an account. The user must be logged in.

Steps:

1. The user clicks on the “Disconnection” button



2. The navigation returns to the “Login” page.

Browse films by keywords

Preconditions: The user must have an account. The user must be logged in. The user must be either on the “Home” page, on the “Most Popular” page or on the “Highest Rated” page.

Steps:

1. The user writes one or many keywords in the search bar and clicks on the “Search” button.



2. A list of films corresponding to the search is displayed.

CinemaSpace

Home

Highest Rated

Most Popular

Profile

Disconnection

cinema

Search

Films matching with cinema

A movie poster for 'A PROPOSITO DE BUÑUEL' featuring a portrait of Luis Buñuel and a man in a suit.

Speaking of Bunuel

A movie poster for 'INGLOURIOUS BASTERDS' featuring Brad Pitt and other actors in a war setting.

IngLOURious Basterds

A movie poster for 'LIVING IN OBLIVION' featuring a woman's face and the title in large letters.

Living in Oblivion

A black and white movie poster for 'L'ARRIVÉE D'UN TRAIN À LA CIOTAT' showing a train arriving at a station.

The Arrival of a Train at La Ciotat

A movie poster for 'THE BLOB' with a pink background and a large, dark, blob-like hand.

The Blob

A movie poster for 'The Twilight Saga: New Moon' featuring the main cast in a dark, moody setting.

The Twilight Saga: New Moon

A movie poster for 'CAFÉ SOCIETY' featuring a stylized white silhouette of a woman's face on a black background.

Café Society

A movie poster for 'THE BATMAN SHOOTINGS' showing a man in a white shirt being held by another man.

The Batman Shootings

Previous

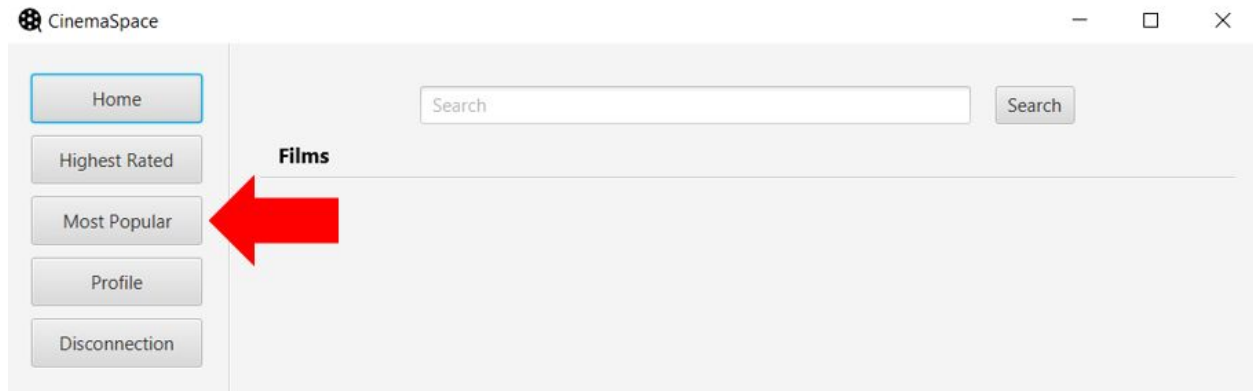
Next

Browse films by visits

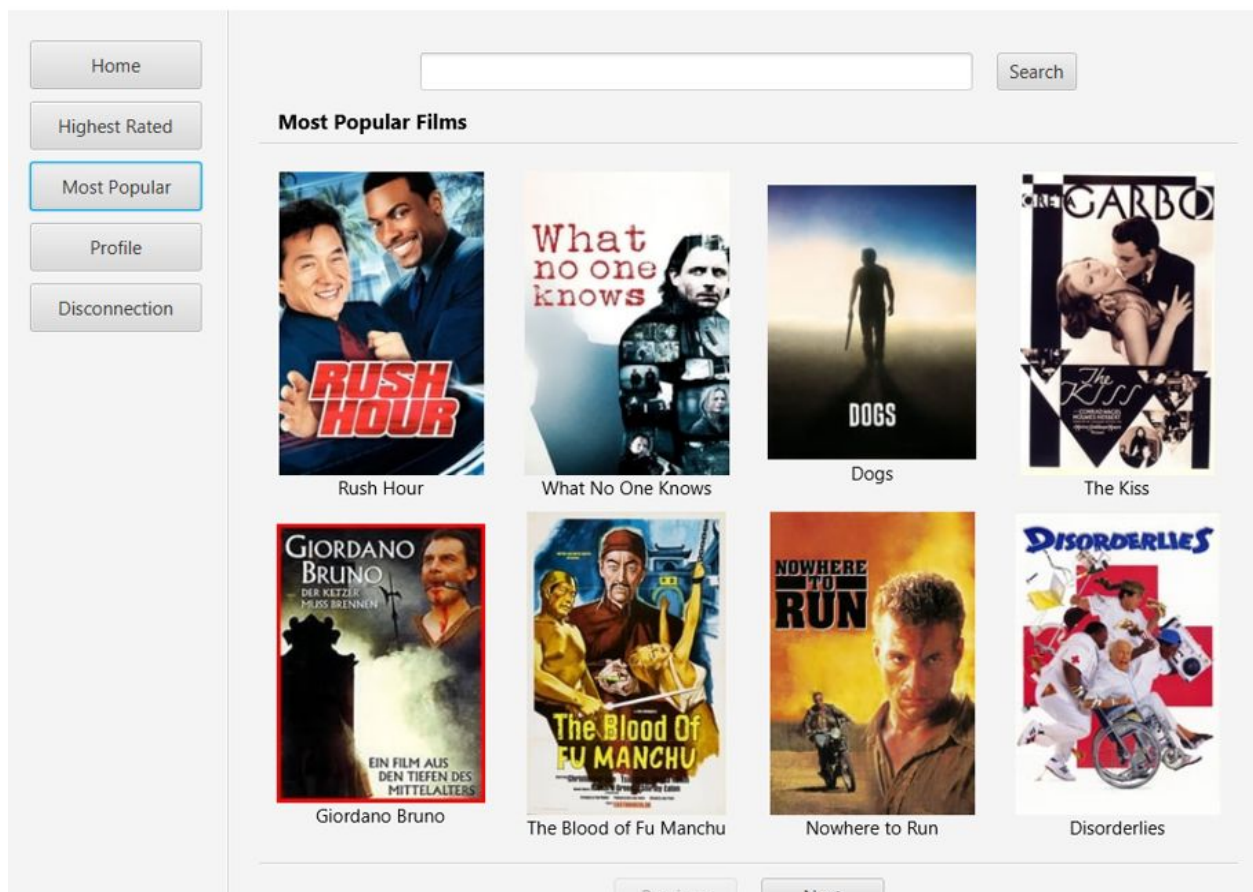
Preconditions: The user must have an account. The user must be logged in.

Steps:

1. The user clicks on the “Most Popular” button.



2. The films with the highest number of visits are displayed.

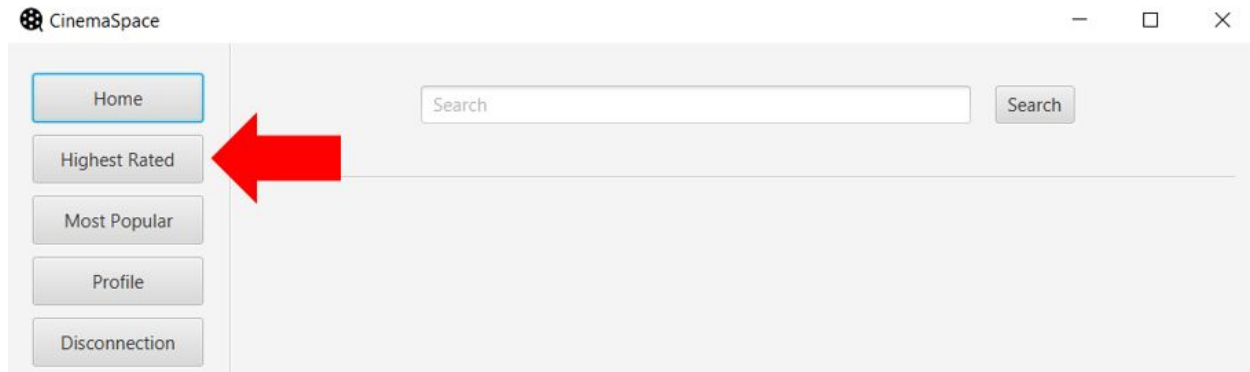


Browse films by ratings

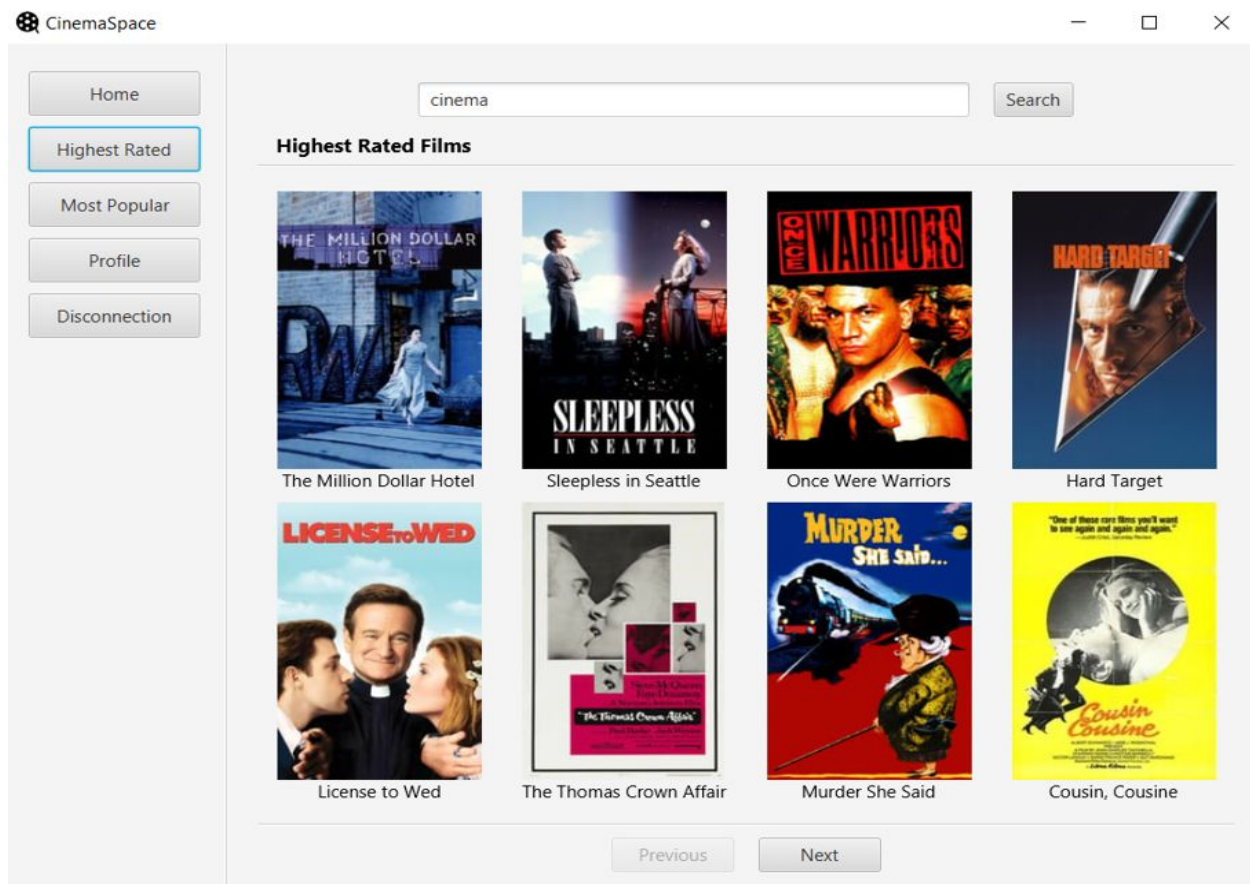
Preconditions: The user must have an account. The user must be logged in.

Steps:

1. The user clicks on the “Highest Rated” button.



2. The films with the highest rated are displayed.

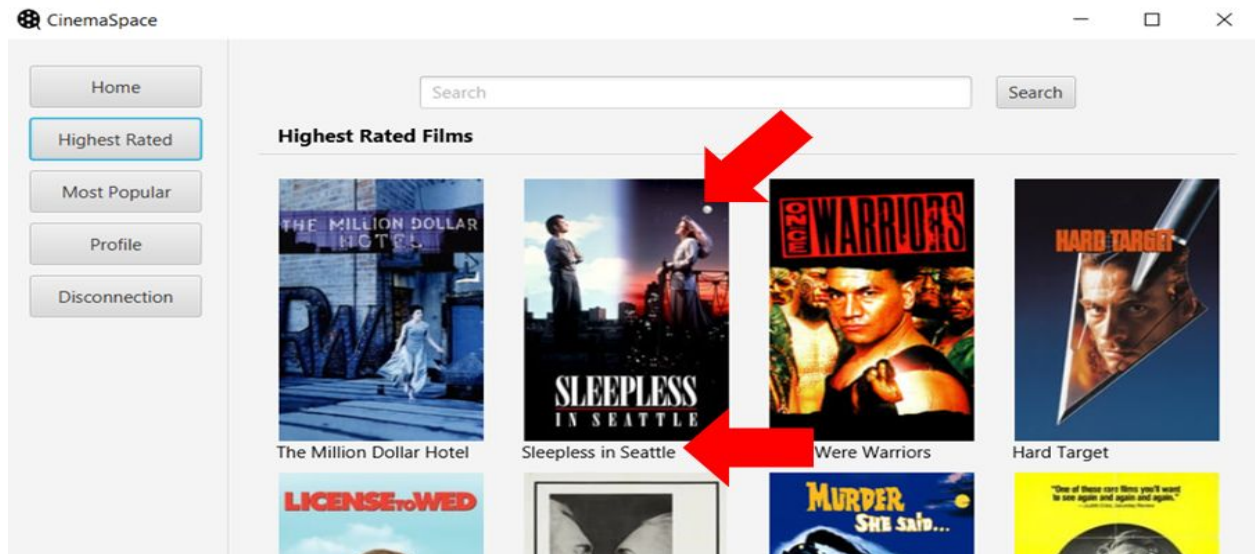


View a film

Preconditions: The user must have an account. The user must be logged in.

Steps:

1. The user clicks on the title or the poster of a film.



2. The film's information is displayed.

The screenshot displays the detailed information page for the movie 'Sleepless in Seattle' on the CinemaSpace website. The left sidebar remains the same. The main content area shows the movie's poster, title, and various details:

- Runtime:** 105.0
- Genres:** Comedy | Drama | Romance
- Release Date:** 1993-06-24
- Description:** A young boy who tries to set his dad up on a date after the death of his mother. He calls into a radio station to talk about his dad's loneliness which soon leads the dad into meeting a Journalist Annie who
- Director:** Nora Ephron
- Cast:** Tom Hanks | Meg Ryan | Bill Pullman | Ross Malinger | Rosie O'Donnell | Gaby Hoffmann ...
- Crew:** Nora Ephron | David S. Ward | Jeff Arch | Gary Foster | Marc Shaiman | Sven Nykvist ...
- Country:** United States of America
- Language:** English
- Budget:** 2.1E7
- Production Company:** TriStar Pictures

On the right side of the movie details, the overall mean rating is 4,34/5 and the recent mean rating (of the last two years) is 4,50/5.

Below the movie details is a 'Rating' section with a table showing ratings by category and a bar chart showing the distribution of ratings.

	All	< 18	18 - 45	45 +
All	4,34/5	4,35/5	4,34/5	4,34/5
Men	4,33/5	4,34/5	4,33/5	4,32/5
Women	4,35/5	4,35/5	4,35/5	4,35/5

The bar chart shows the number of ratings for each rating value from 0.0 to 5.0. The x-axis is labeled 'Ratings' and the y-axis shows the count from 0 to 30,000. The distribution is skewed towards higher ratings, with a peak at 5.0.

Rate a film

Preconditions: The user must have an account. The user must be logged in and on the film page.

Steps:

1. The user clicks on the combo box named “Rate”.

The screenshot shows the 'Rate' interface. At the top, there are fields for Country (United States of America), Language (English), Budget (2.1E7), and Production Company (TriStar Pictures). Below these is a 'Rating' section with a table and a bar chart. A red arrow points to the 'Rate' dropdown menu.

	All	< 18	18 - 45	45+
All	4,34/5	4,35/5	4,34/5	4,34/5
Men	4,33/5	4,34/5	4,33/5	4,32/5
Women	4,35/5	4,35/5	4,35/5	4,35/5

The bar chart shows the distribution of ratings from 0.0 to 5.0. The x-axis is labeled 'Ratings' and the y-axis shows the count of ratings, ranging from 0 to 30,000. The bars show a distribution peaking at 4.0 and 5.0.

2. The user selects a rating in the combo box.

The screenshot shows the 'Rate' dropdown menu with '4/5' selected. To the right, a confirmation dialog box is displayed with the message 'Your rating has been added.' and an 'OK' button. The background shows the film details for 'Sleeping in the Suburbs'.

Database status after the insertion

The screenshot shows the CinemaSpace.Rating database interface. The 'Documents' tab is selected, and the document structure is displayed. The document contains the following fields:

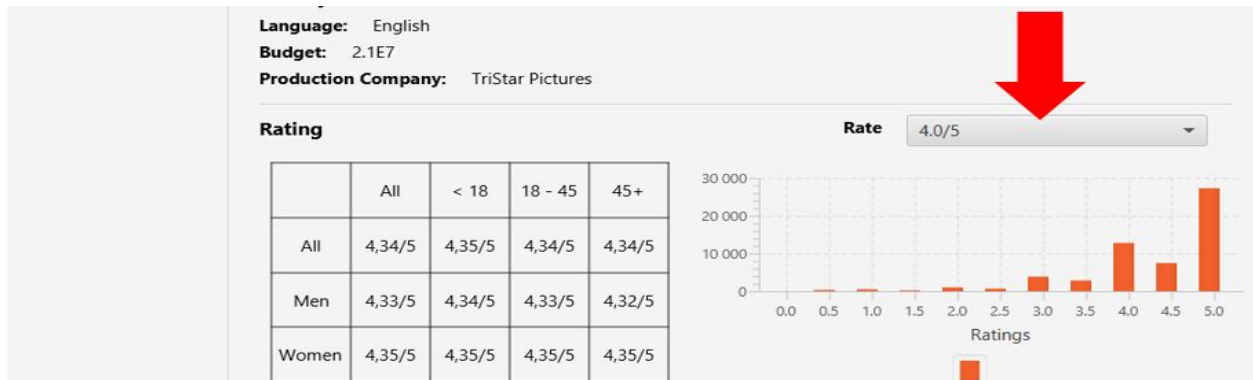
```
{
  "_id": Object,
  "userId": ObjectId("5df505f43656215904f003bb"),
  "filmId": ObjectId("5de0ca2109c85f7e66a01e81"),
  "rating": 4,
  "timestamp": 1576342576127,
  "age_of_user": 23,
  "gender": "Female"
}
```

Update a rating on a film

Preconditions: The user must have an account. The user must be logged in and on the page of a film she rated.

Steps:

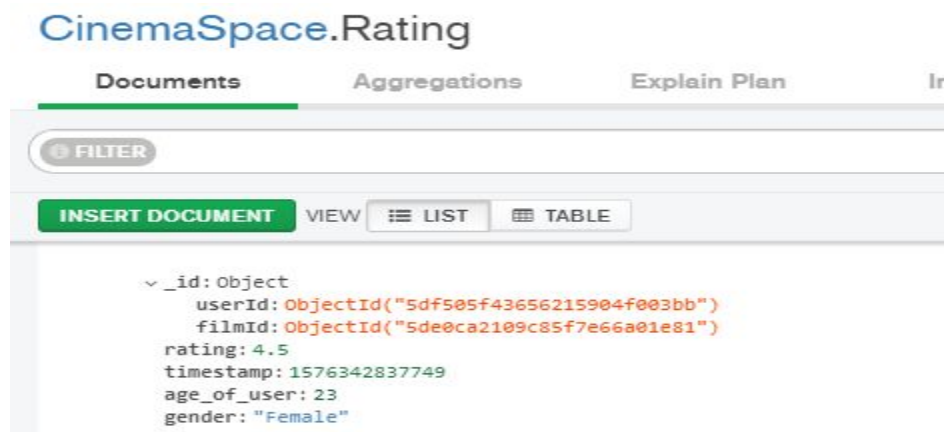
1. The user clicks on the combo box with the given rating.



2. The user selects a rating in the combo box items.



Database status after the insertion

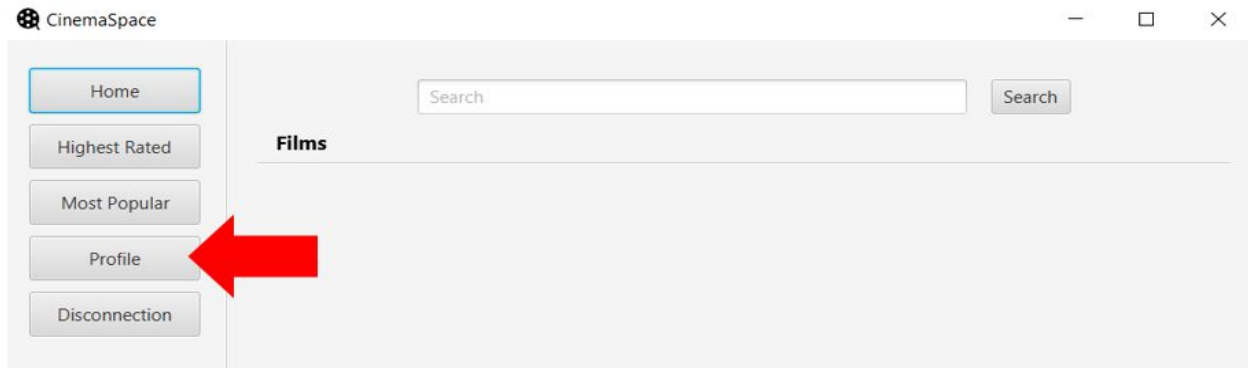


View personal profile

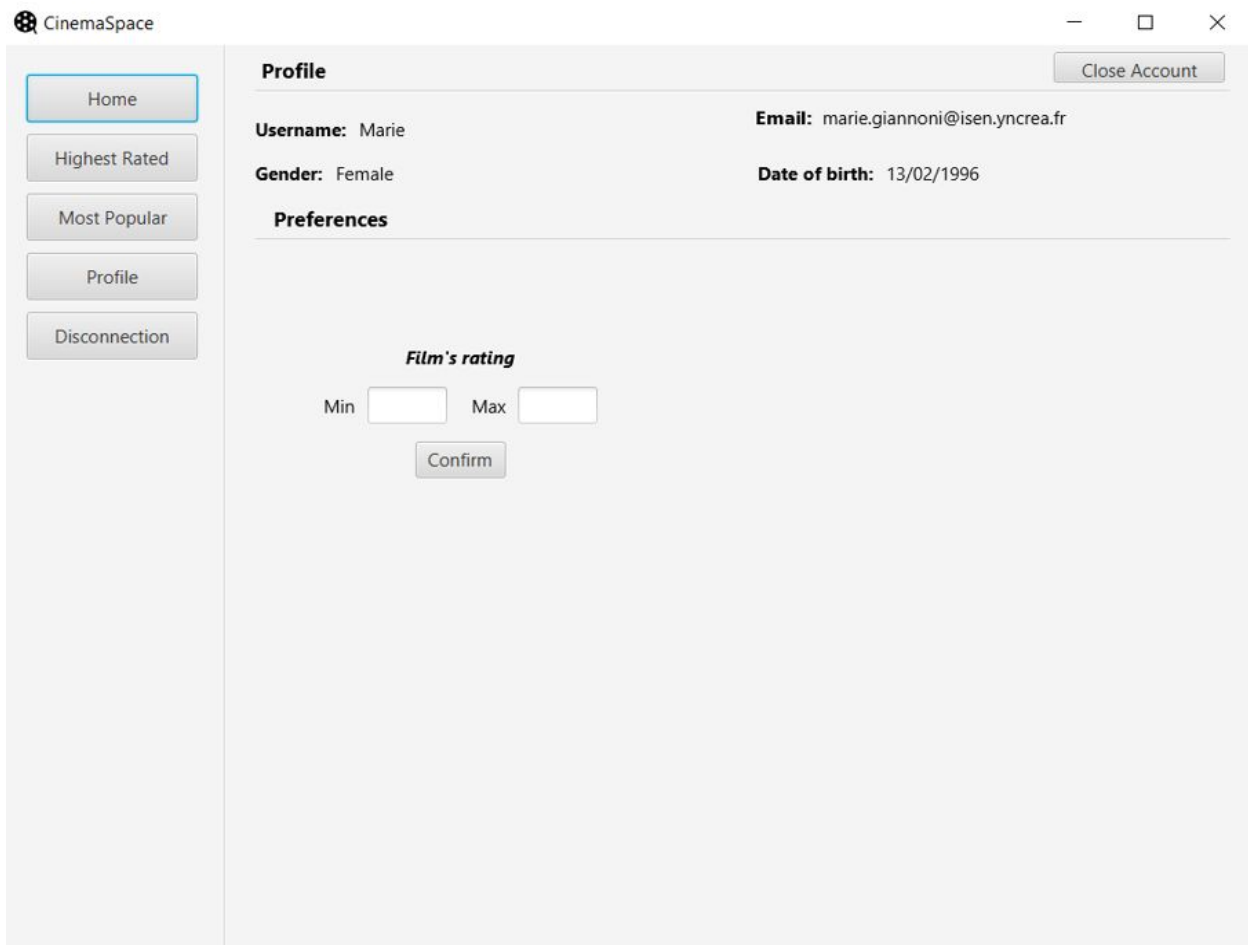
Preconditions: The user must have a profile. The user must be logged in.

Steps:

1. The user clicks on the “Profile” button.



2. The “Profile” page is displayed.



See the most recurrent film genres in intervals of expressed ratings

Preconditions: The user must have an account. The user must be logged in.

Steps:

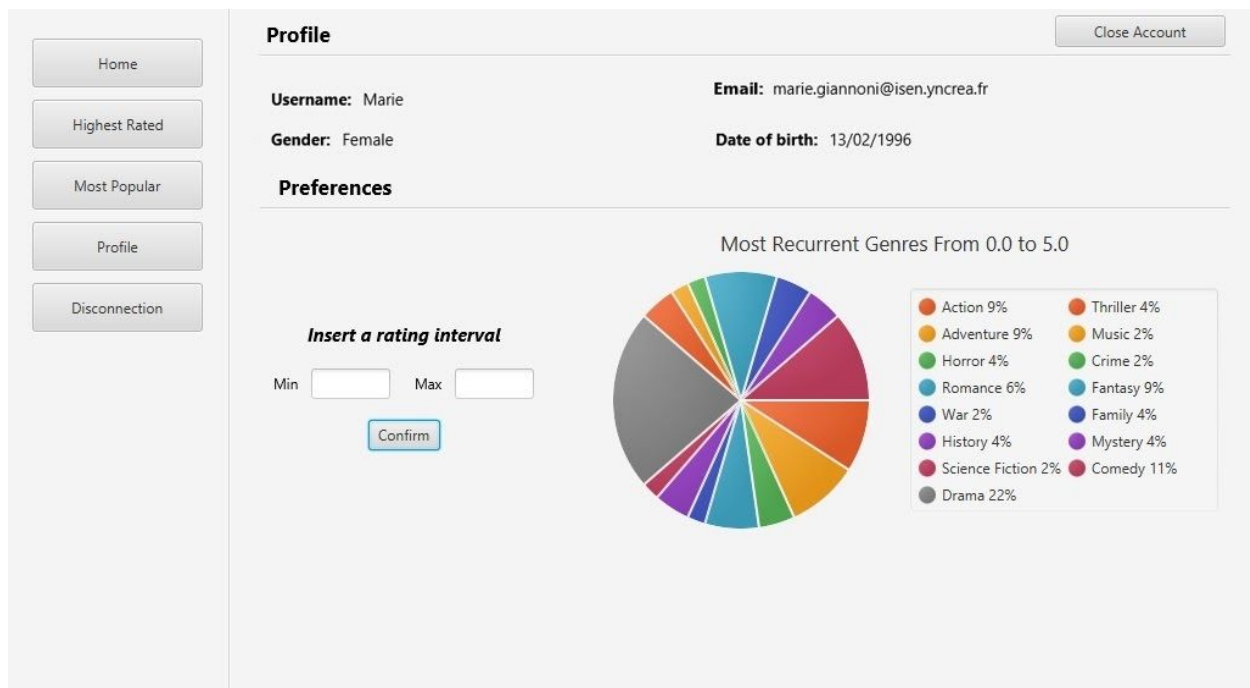
1. The user sets a minimum and maximum value in the text fields “Min” and “Max” and clicks on the “Confirm” button. The default values are min = 0 and max = 5.



Film's rating

Min Max

2. A pie chart with the genres proportion of the ratings is displayed.



Profile Close Account

Username: Marie **Email:** marie.giannoni@isen.yncrea.fr

Gender: Female **Date of birth:** 13/02/1996

Preferences

Most Recurrent Genres From 0.0 to 5.0

Insert a rating interval

Min Max

Genre Proportions:

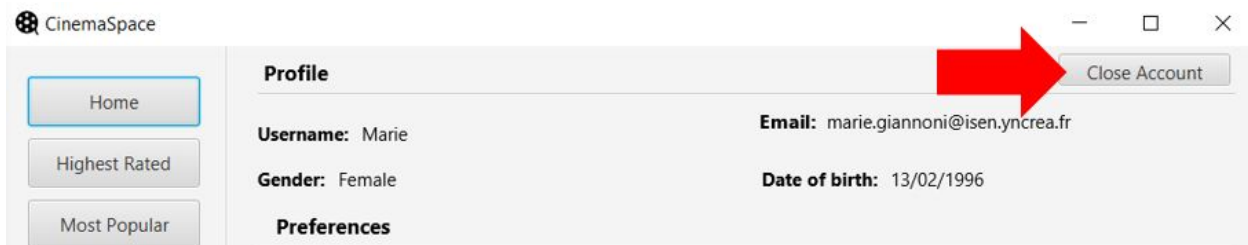
Action 9%	Thriller 4%
Adventure 9%	Music 2%
Horror 4%	Crime 2%
Romance 6%	Fantasy 9%
War 2%	Family 4%
History 4%	Mystery 4%
Science Fiction 2%	Comedy 11%
Drama 22%	

Close a user account

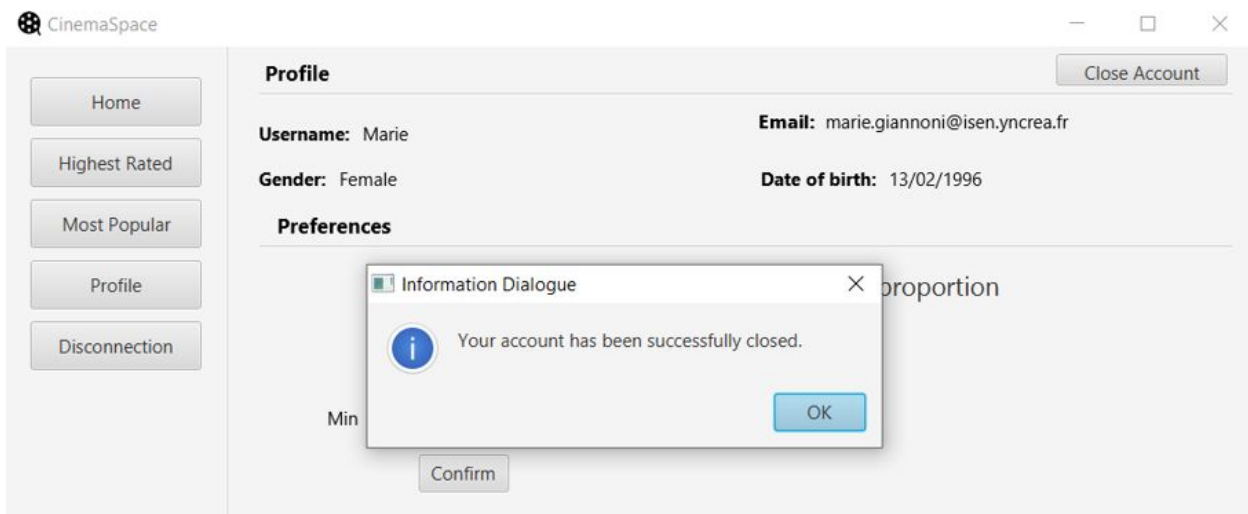
Preconditions: The user must have an account. The user must be logged in and on her profile page.

Steps:

1. The user clicks on the “Close Account” button.



2. An Information Dialogue informs that the account has been closed.



3. The navigation returns to the “Login” page.

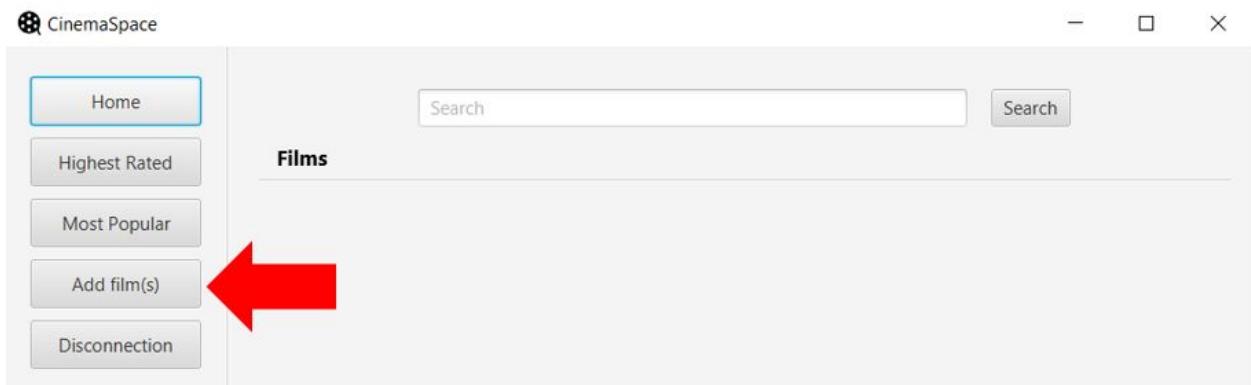
7.4.2 Administrator

Add a film or a collection of films

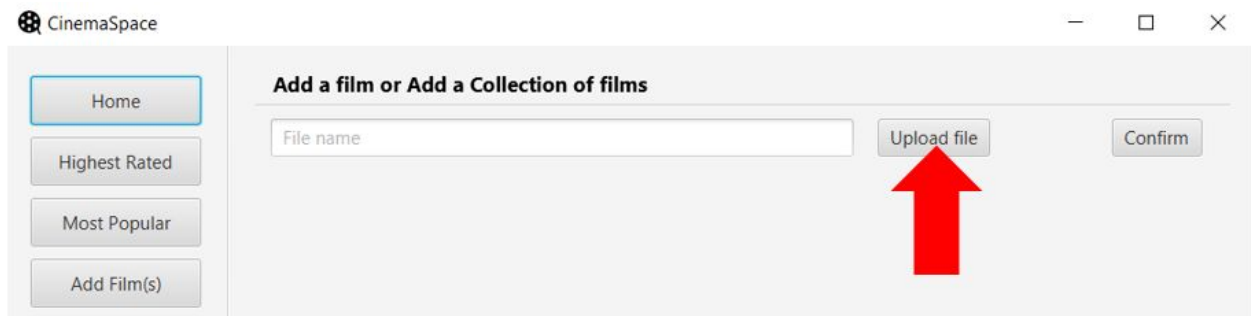
Preconditions: The user must have an account with the privilege of an administrator. The user must be logged in.

Steps:

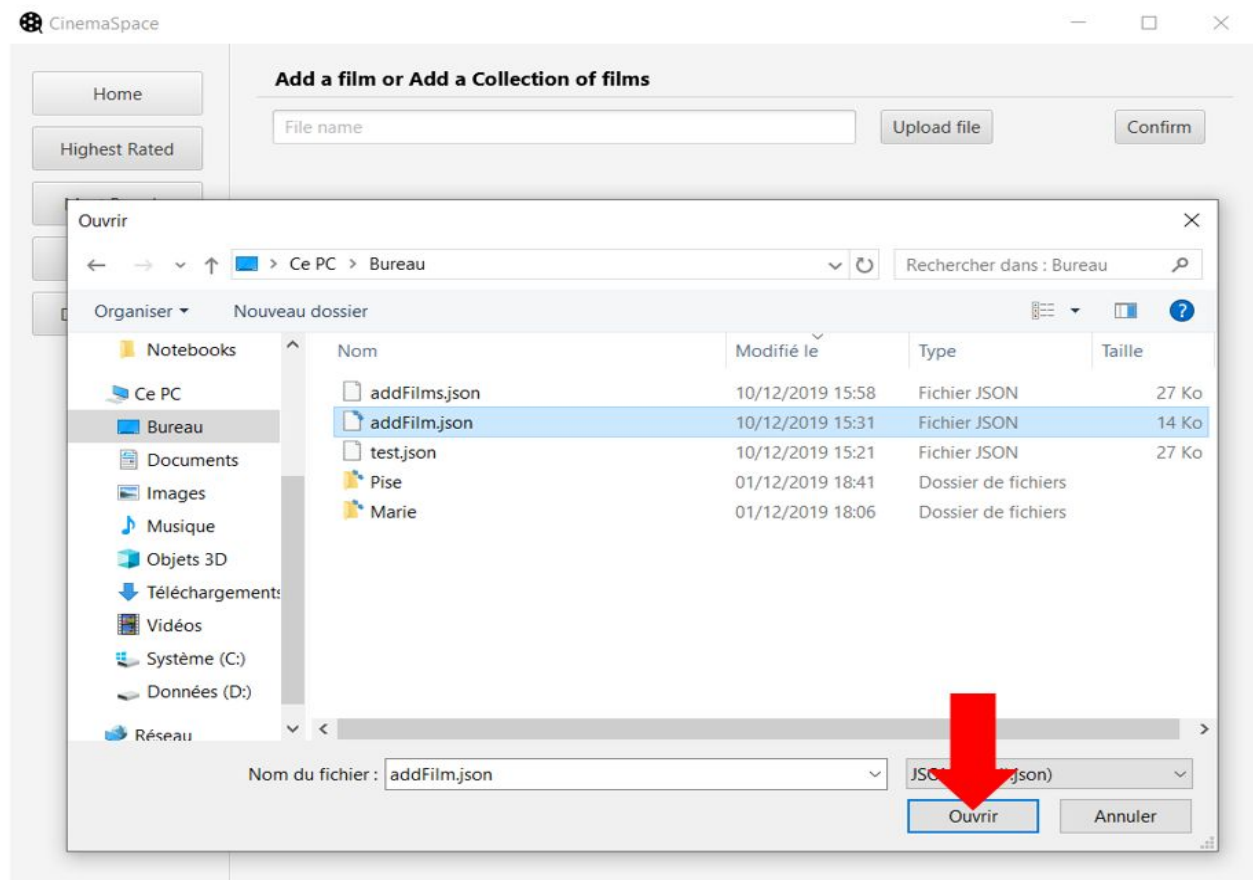
1. The user clicks on the “Add film(s)” button.



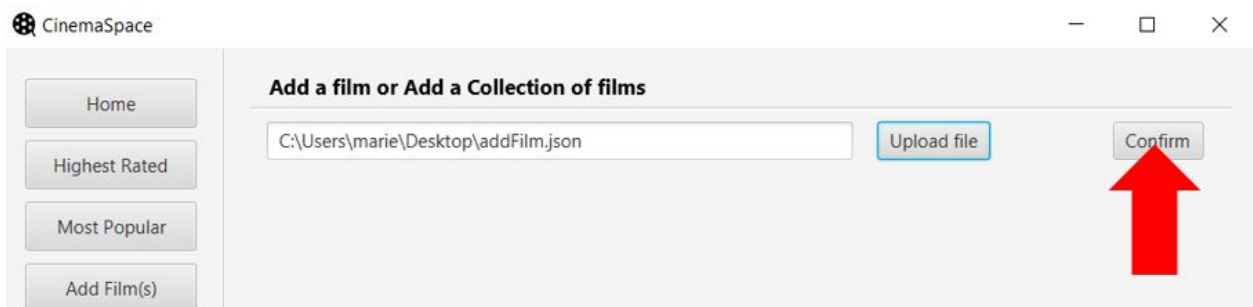
2. The user clicks on the “Upload file” button.



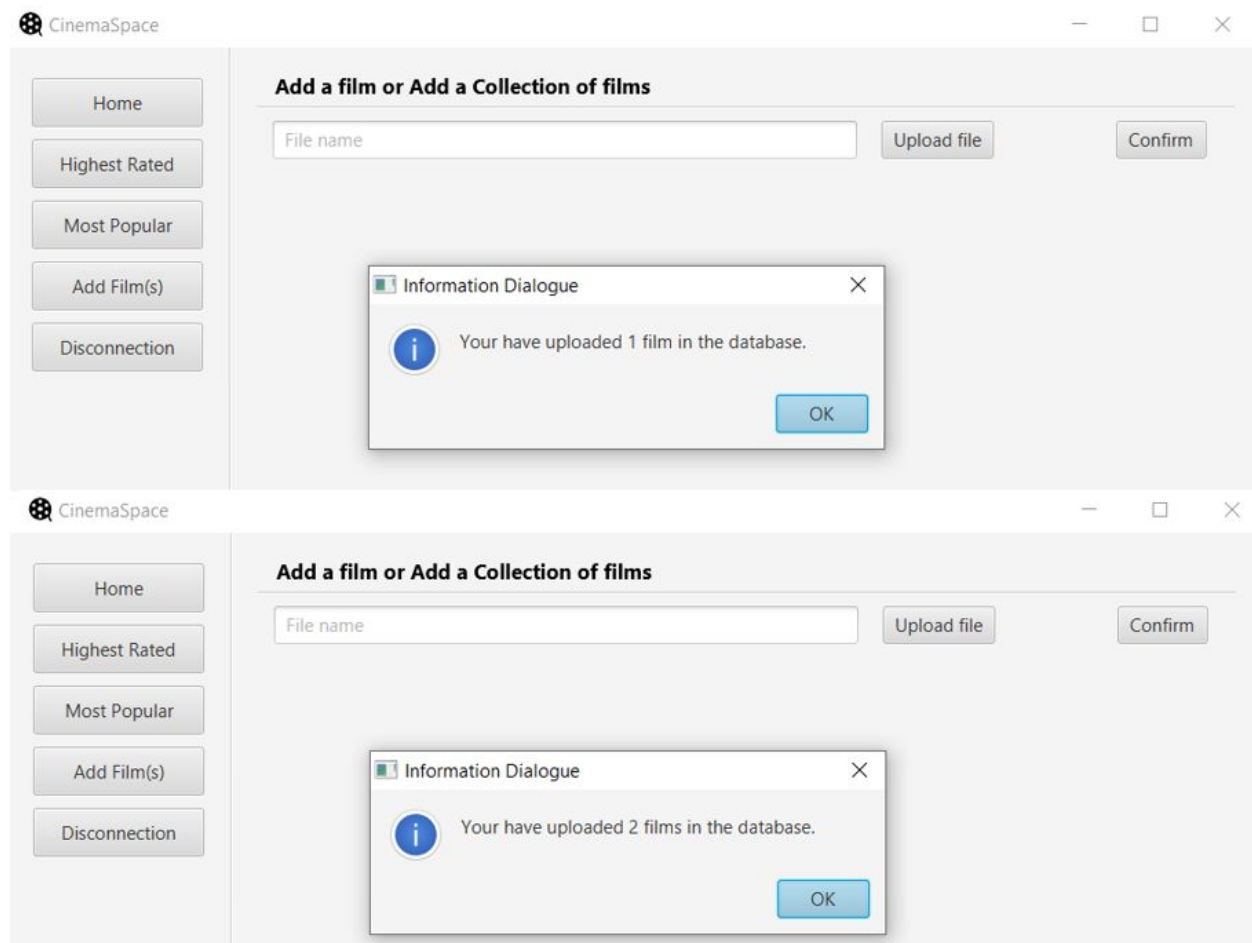
3. The user selects a JSON file with one or more films to add.



4. The user clicks on the "Confirm" button.



5. An Information Dialogue informs that the film or the collection of films were added.



Delete a film

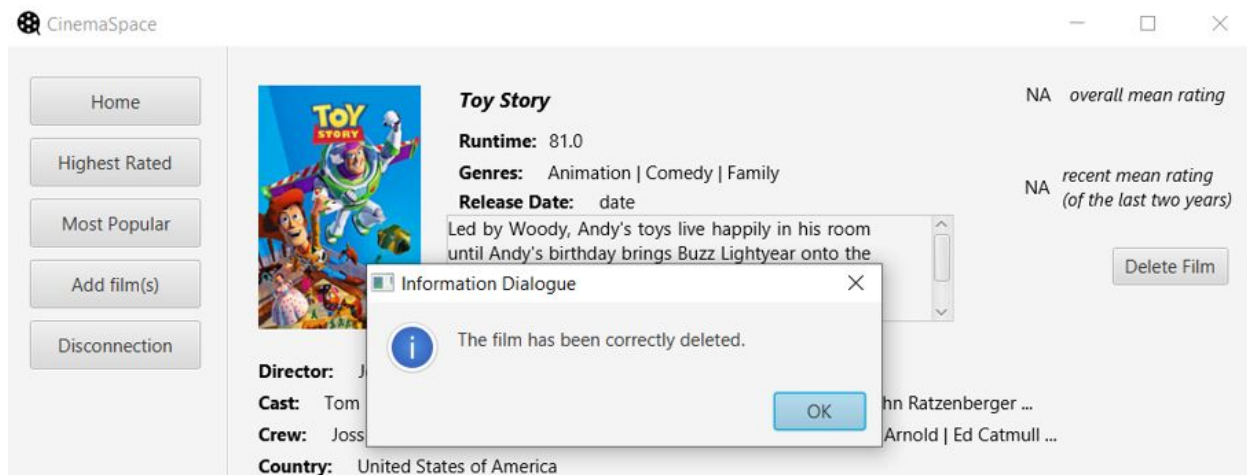
Preconditions: The user must have an account with the privilege of an administrator. The user must be logged in and on the film page that she wants to delete.

Steps:

1. Click on the “Delete a film” button



2. An Information Dialogue informs that the film has been deleted.



3. The “Home” page is displayed.

APPENDIX: DATASET CREDITS

The population of the database has been done starting from a pre-existing public dataset, called “*The Movies Dataset*”, available at www.kaggle.com/rounakbanik/the-movies-dataset.

The dataset has been manipulated using Python (Pandas and JSON libraries) in order to reflect the identified structure of the collections: the resulting JSON files and the MongoDB database dump are available at <https://mega.nz/#F!coRDGSTJ!cIL-u--JZJcsOJEN7JwS3w>.

The final set is characterized by:

- a total dump size of 1.52 GB;
- 46.6K films;
- 270.9K users;
- 11.4M ratings.