

Private Movie Collection

The screenshot shows a web application window titled "Private Movie Collection". The interface is divided into two main sections: "Categories" on the left and "Movies" on the right. The "Categories" section features a list of movie genres with their counts: "1, All categories", "4, Drama", "5, Thriller", "6, Horror", "14, Action", "18, Fantasy", "19, Science Fiction", "20, Comedy", and "24, Tragedy". Below this list are "Delete", "Add", and "Edit" buttons. The "Movies" section includes a search bar labeled "Enter movie title...", a "Minimum rating" dropdown, and "Set filter" and "Clear filter" buttons. A large "Play" button is also present. Below the search bar is a "Sort..." dropdown. The main area of the "Movies" section is a large empty box. At the bottom of the "Movies" section are "Delete", "Add", "Edit", and "Add Rating" buttons.

Private Movie Collection

Jonathan Hänsche Norre, Jonas Okholm Spatzek

Supervisors: Peter Gaarsmand Stegger Nielsen, Jeppe Moritz Led

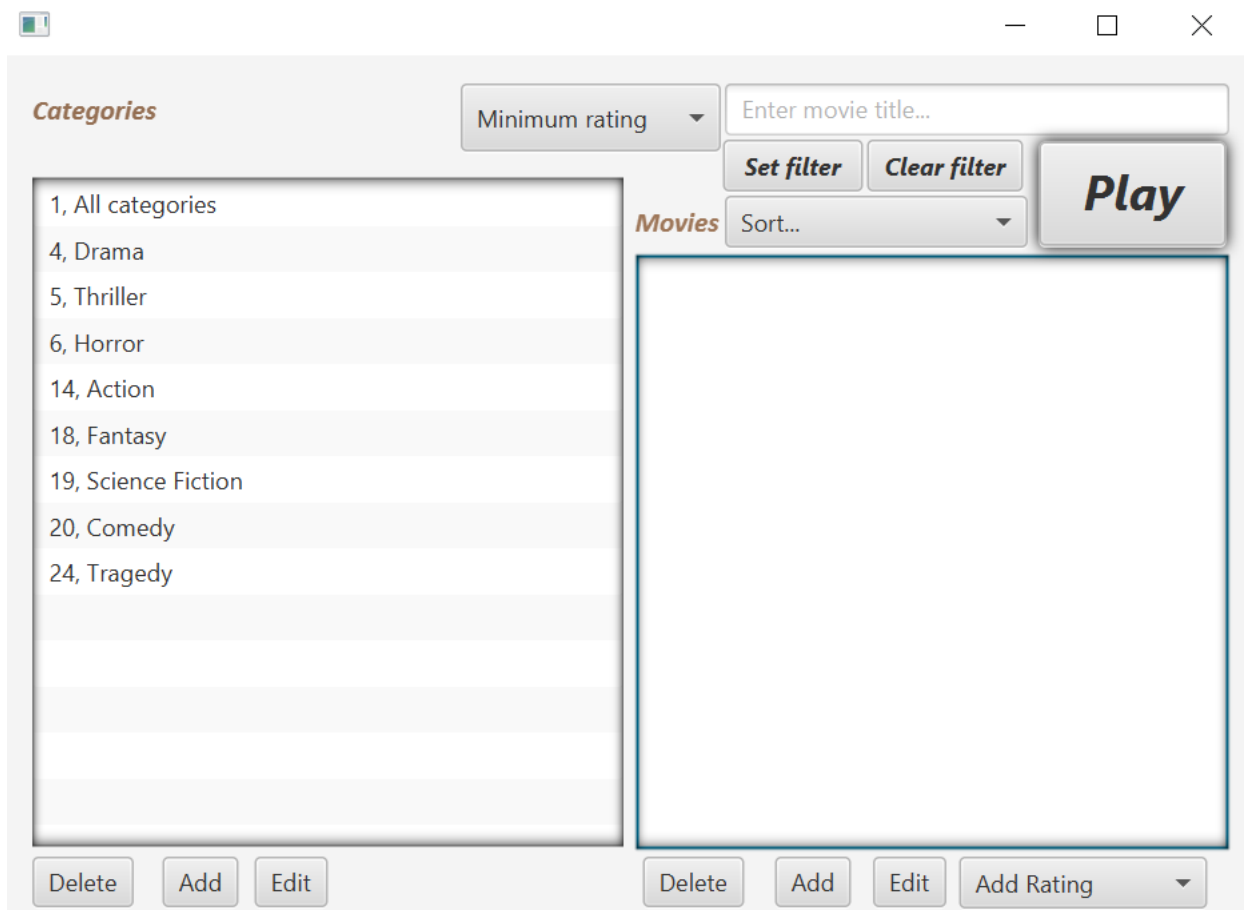
17. januar 2020

Business Academy Southwest, Esbjerg

A short introduction to the application

The application is intended to serve as an easy way to organize and display the user's collection of movie clips. The application does not play movie clips itself.

Below is (again) an overview of the application's main window:



When opening the program, the user will first be met with a question about whether to delete poorly rated, seldomly seen movies. The user can press "OK" to do so or cancel – in any case the user proceeds to the main window.

Private Movie Collection**Github:** <https://github.com/Spatzek/Movie>**Hand-in:** Jonathan Hänsche Norre, Jonas Okholm Spatzek

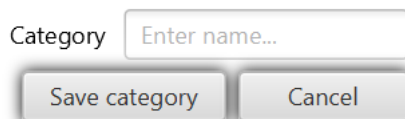
On the left side of the main window is a list of different categories as defined by the user. When clicking upon one or more categories (using CTRL-click), a list of movies belonging to one of the categories is displayed to the right side.

At the top of the list of categories is a unique category called “all categories”. When clicking this category, all movies in the collection are shown regardless of which other categories are selected.

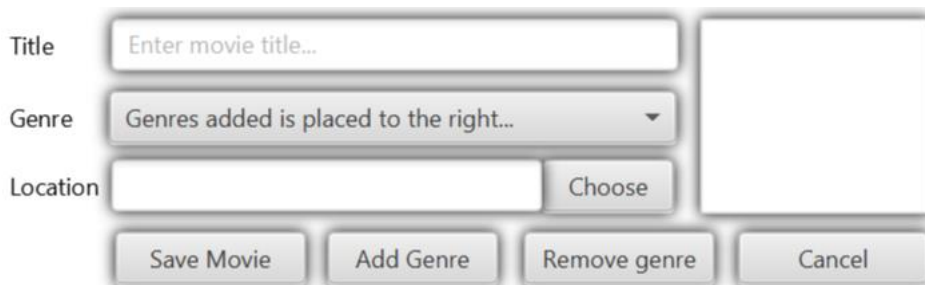
A category or movie can be added, edited or deleted by clicking one of the corresponding buttons below their respective lists. To edit or delete a category or movie, a single one of these must be selected first. The user will be warned if attempting to edit or delete without such a selection. Adding or editing will open a new window.

When adding or editing a movie, the user can choose which categories the movie should be considered to belong to. The user must also choose a unique name and a file location for the movie, with the file being required to be in a .mp4 or .mpeg4 format. Clicking “Save” will prompt the user to confirm the addition/change.

Below is shown the window when creating or editing categories:



When creating or editing movies:



Rating a movie must be done in the main window by clicking the movie and using the drop-down “Add rating” menu. Ratings are from 1 to 10.

Github: <https://github.com/Spatzek/Movie>

Hand-in: Jonathan Hänsche Norre, Jonas Okholm Spatzek

Other than by clicking through the categories, the list of displayed movies can be further customized via filtering and sorting options at the top. To filter the list of movies, the user has to click the “Set filter” button upon which the list will be filtered according to the entered search term and minimum rating, if any is set. The button’s text will then change to “New filter” with a red color, and this button has to be pressed again to refilter according to other criteria.

Furthermore, the movies can be sorted by title or rating via a drop-down menu. Filtering and sorting can be used in any combination with each other. To reset the chosen filter and sort method, the user can press the “Clear filter” button.

Finally, the user can click a movie and press the “Play” button to use the operating system’s default media player application to play the movie.

An overview of requirements and implementations

At the onset of the project, a list of expected features of the application was made available to us. What follows is this list along with comments on whether or not the finished application meets the expectations.

1) *“Add/remove categories.”*

DONE. Categories can be added, deleted and edited. At the top of the category list is the “all categories” category, which can however not be deleted or edited to the unique way it functions.

2) *“Add/remove movies. “*

DONE. Movies can be added, deleted and edited.

3) *“Set multiple categories per movie. “*

DONE.

4) *“Filter specific movie titles or part of title, one or multiple genres and/or specific minimum imdb rating.”*

DONE. However, the IMDB rating was deemed unnecessary to use.

5) *“Click a movie open the system’s media player and play the movie.”*

DONE.

- 6) *"Add/change a personal rating for each movie"*

DONE.

- 7) *"Sort movies by score, title or category, while filters are active."*

DONE. However, we didn't see the need to sort by category considering the ease with which categories are already displayed and chosen, and the problem with regards to which of several categories for a movie should take precedence. This omission is on purpose and in agreement with supervisor Jeppe Moritz Led.

Beyond this, there were three special requirements:

- *"Only files ending with .mp4 or mpeg.4 can be added."*

DONE.

- *"When starting, the application should warn users to remember to delete movies that have a personal rating under 6 and has not been opened from the application for more than 2 years."*

DONE. Also, two movies have been manually edited in the database to have old dates so as to facilitate a demonstration of this feature.

- *"The program should to the best of ability only register one movie once. No two movie should be the same title."*

DONE. This is accomplished in the most simple way possible though, by comparing new movie titles to already used ones. So it is possible to create a new movie object from the same file but with a different name.

Overall, we are fairly satisfied with the finished application, especially considering the size of our team. If we should single out an area in which the application could be improved if given more time, perhaps it would be further implementation of proper exception handling. To somewhat mitigate for this, we have at least taken measures to ensure that the user will be blocked from entering invalid input. With more time, we might also have done some refactoring and reviewed each method more closely to ascertain whether all or parts of it was placed correctly in accordance with 3-layer architecture.

Implementation procedure

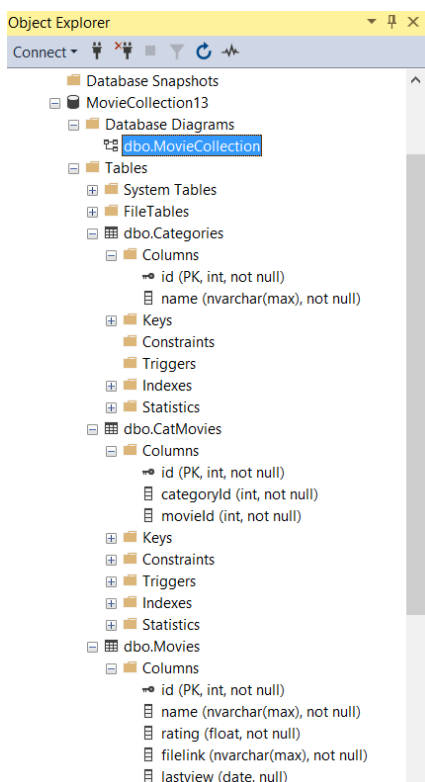
We began our work by setting up the database as shown in the case description, and proceeded to implement the methods needed to access and manipulate the database in the DAL layer – basic C.R.U.D. Much of this could be directly copied and modified from our earlier projects, and we deemed this satisfactory given both the similarity of some of the required features of the projects, and the fact that we had the smallest team. Nothing was copied without first understanding how it worked.

We then decided upon the look of the interface before going further, as we needed to agree which containers, controls and such to use, and this would affect how our code should be written. We then implemented required features piece by piece, carefully testing each feature before moving on to the next.

The order in which features were implemented was roughly this:

- View of all categories
- View of all movies
- Ability to delete, add and edit categories
- Ability to delete, add and edit movies
- Expanding from above, limiting the list of movies to those belonging to the currently selected category
- Ability to rate movies
- Ability to play movies
- Ability to sort movies by title or rating, and keeping the sorting in place when changing categories
- Ability to filter movies by title and/or minimum rating, concurrently with sorting, and keeping this in place when changing categories
- Ability to select multiple categories to retrieve a list of movies belonging to any of the selected categories

Data storage.



We decided to use the school's data storage to store our database, to edit to the database we use the **Microsoft SQL Server Management Studio 18 program**. This Program is only used for the administrative tasks to create the lists that we need for the **MovieCollector** has some data it can pull down for the client.

Via the Database our **MovieCollector** will be able to edit, add, delete and store the path of the movies folders on each computers movie folder.

Although the possibility to edit our data manually on the database is a possibility as an admin, on the client side of the **MovieCollector**, its very limited what changes can be made, things such as making genres add/edit movies and store their locations is pretty much the only changes that can be made client side.

```
classDiagram
    class Movie {
        id: int
        name: String
        runtime: double
        rating: double
        releaseDate: Date
        categories: List<Category>
        +Movie(String name, double rating, String runtime, Date lastWeek)
        +getId(): int
        +getName(): String
        +setName(String name): void
        +getRuntime(): double
        +setRuntime(double rating): void
        +getReleaseDate(): Date
        +setReleaseDate(Date lastWeek): void
        +getCategories(): List<Category>
        +setCategories(List<Category> categories): void
        +removeCategory(Category category): void
        +addRating(): String
    }

    class Category {
        id: int
        name: String
        +Category(String name)
        +getId(): int
        +setName(String name): void
        +addRating(): String
    }

    class MovieCollector {
        +MovieCollector()
        +startStage(stage): void
    }

    class CategoryManager {
        +CategoryManager()
        +readAllCategories()
        +readCategories(List<Category>)
        +readCategories(Category category)
        +readCategories(Movie movie)
        +readCategories(Category category, List<Movie>)
        +readCategories(Movie movie, List<Category>)
        +readCategories(Movie movie, List<Category>, String searchTerm)
        +readCategories(Category category, List<Movie>, String searchTerm)
        +readCategories(Category category, List<Movie>, String searchTerm, List<Movie>)
        +readCategories(Category category, List<Movie>, String searchTerm, List<Movie>, List<Category>)
        +readCategories(Category category, List<Movie>, String searchTerm, List<Movie>, List<Category>, List<Category>)
    }

    class MovieManager {
        +MovieManager()
        +readAllMovies()
        +readMovies(List<Movie>)
        +readMovies(Movie movie)
        +readMovies(Category category, List<Movie>)
        +readMovies(Movie movie, List<Category>)
        +readMovies(Movie movie, List<Category>, String searchTerm)
        +readMovies(Movie movie, List<Category>, String searchTerm, List<Movie>)
        +readMovies(Movie movie, List<Category>, String searchTerm, List<Movie>, List<Category>)
    }

    class MovieCollectorModel {
        +MovieCollectorModel()
        +startStage(stage): void
    }

    class CategoryManagerModel {
        +CategoryManagerModel()
        +readAllCategories()
        +readCategories(List<Category>)
        +readCategories(Category category)
        +readCategories(Movie movie)
        +readCategories(Category category, List<Movie>)
        +readCategories(Movie movie, List<Category>)
        +readCategories(Movie movie, List<Category>, String searchTerm)
        +readCategories(Category category, List<Movie>, String searchTerm)
        +readCategories(Category category, List<Movie>, String searchTerm, List<Movie>)
        +readCategories(Category category, List<Movie>, String searchTerm, List<Movie>, List<Category>)
    }

    class MovieManagerModel {
        +MovieManagerModel()
        +readAllMovies()
        +readMovies(List<Movie>)
        +readMovies(Movie movie)
        +readMovies(Category category, List<Movie>)
        +readMovies(Movie movie, List<Category>)
        +readMovies(Movie movie, List<Category>, String searchTerm)
        +readMovies(Movie movie, List<Category>, String searchTerm, List<Movie>)
        +readMovies(Movie movie, List<Category>, String searchTerm, List<Movie>, List<Category>)
    }

    class CategoryDBDAO {
        +CategoryDBDAO()
        +readAllCategories()
        +readCategories(List<Category>)
        +readCategories(Category category)
        +readCategories(Movie movie)
        +readCategories(Category category, List<Movie>)
        +readCategories(Movie movie, List<Category>)
        +readCategories(Movie movie, List<Category>, String searchTerm)
        +readCategories(Category category, List<Movie>, String searchTerm)
        +readCategories(Category category, List<Movie>, String searchTerm, List<Movie>)
        +readCategories(Category category, List<Movie>, String searchTerm, List<Movie>, List<Category>)
    }

    class MovieDBDAO {
        +MovieDBDAO()
        +readAllMovies()
        +readMovies(List<Movie>)
        +readMovies(Movie movie)
        +readMovies(Category category, List<Movie>)
        +readMovies(Movie movie, List<Category>)
        +readMovies(Movie movie, List<Category>, String searchTerm)
        +readMovies(Movie movie, List<Category>, String searchTerm, List<Movie>)
        +readMovies(Movie movie, List<Category>, String searchTerm, List<Movie>, List<Category>)
    }

    class DBSettings {
        +DBSettings()
        +readAllCategories()
        +readCategories(List<Category>)
        +readCategories(Category category)
        +readCategories(Movie movie)
        +readCategories(Category category, List<Movie>)
        +readCategories(Movie movie, List<Category>)
        +readCategories(Movie movie, List<Category>, String searchTerm)
        +readCategories(Category category, List<Movie>, String searchTerm)
        +readCategories(Category category, List<Movie>, String searchTerm, List<Movie>)
        +readCategories(Category category, List<Movie>, String searchTerm, List<Movie>, List<Category>)
    }

    class DBConnection {
        +DBConnection()
        +readAllCategories()
        +readCategories(List<Category>)
        +readCategories(Category category)
        +readCategories(Movie movie)
        +readCategories(Category category, List<Movie>)
        +readCategories(Movie movie, List<Category>)
        +readCategories(Movie movie, List<Category>, String searchTerm)
        +readCategories(Category category, List<Movie>, String searchTerm)
        +readCategories(Category category, List<Movie>, String searchTerm, List<Movie>)
        +readCategories(Category category, List<Movie>, String searchTerm, List<Movie>, List<Category>)
    }

    Movie "0" -- "0" Category
    MovieCollector "0" -- "0" Category
    CategoryManager "0" -- "0" Category
    MovieManager "0" -- "0" Category
    MovieCollectorModel "0" -- "0" Category
    CategoryManagerModel "0" -- "0" Category
    MovieManagerModel "0" -- "0" Category
    CategoryDBDAO "0" -- "0" Category
    MovieDBDAO "0" -- "0" Category
    DBSettings "0" -- "0" Category
    DBConnection "0" -- "0" Category
```