Chapter 1

Feasibility Study

The very first step was to perform is to look up for what type of data we need to create the application and how to handle it by performing a feasibility study.

1 Dataset analysis and creation

Initially we searched online for available dataset, we ended up with five different files coming from Kaggle and imdb with a combined storage of 4.18 GB:

- title_principal.tsv
- name_basic.tsv
- title_basic.tsv
- title_crew.tsv
- rotten_movies.csv
- rotten_reviews.csv

The first three were found on the imdb site containing general data about the title of the entries, type (not all were movies), cast, crew and other useful information. The last two came from Kaggle and they contain data scraped from the rotten tomatoes site regarding the movies rating and their reviews. All of these dataset were organized in a relational manner.

Initial steps and creation of the movie collection After a general look it was clear that we needed to process the data to obtain a less bloated dataset by deciding what to keep in base of our needs and specification; we decided to use python as programming language to trim the original file, this was also achieved through the use of Google Colab.

We started by taking all the tsv file and transforming them into a single file, we performed a join operation on all of them based on their id and then discarded the useless information. For the code see (Appendix A .1). After that we performed the same operation on the csv file coming from Rotten Tomatoes (Appendix A .2). We then proceed to join the two files based on the title of the movies (Appendix A .3).

Unfortunately we noticed that after the join there were more rows for the same movie, in fact, due to the relational nature of the first dataset, we had a different entry for personnel on each row, so we rollback to the start, but this time we collapsed all the information in a single row (Appendix A .4).

Due to the variability of the data in the string fields, like the content of the reviews, we decided to perform an escape on various elements to avoid crashes and failure during the import into the DBSM (Appendix B .1). We also had to do a similar process of normalization for the date fields (Appendix B .2)

Finally we achieved our goal of having a json file for the movie collection, the file occupied a space of 270MB.

Creation of the user collection The following step was to generate a collection for the user, this was achieved by starting from the movie one and for each different review author we generated a document later to be placed in a single json file of which the total storage size was 86,6 MB, this step was performed directly on the mongo shell (Appendix B .3).

2 Analysis result

With the dataset for the document database ready we finally had a better understanding on how to shape the models and the relative functionality in the application code. We will discuss later of the design of the collections and the methods used to interact with them. The same goes for the graph database of whom the structure was heavily influenced by the one in the MongoDB