Cinema Rating

Web Application using NoSQL MongoDB

Gustavo Fleury Soares

Induraj P. ramamurthy

Subject: NoSQL

Professor:

PhD. Rachid Chelouah

Cergy / FR

March, 2020

Summary

1 Introduction 1
2 Technical Solution 1
2.1 MongoDB 2
2.2 Django 2
2.3 Chart.js 3
2.4 Bootstrap 3
2.5 Docker 4
3 NoSQL Modeling 4
3.1 Classical SQL Model 4
3.2 NoSQL Model 5
4 Results and Screenshots 6
5 Conclusion 8
6 Bibliographic References 9
7 Appendix – Tutorial 10
7.1 Dockers 10
7.2 Django 12
7.3 MongoDB 14

1 Introduction

This project consists in developing a web application for display/update/collect user's ratings and comments for films, using a NoSQL database for store the data. The requirements proposed for the project are:

- Web environment using MVC model;
- Interface for list the films;
- Interface to add/update/delete the film information (CRUD) with interaction with the NoSQL database.

2 Technical Solution

For develop the project we decided to use MongoDB for a NoSQL database. The web framework for web development we select Django with Bootstrap and Chart.JS, like shown in the following figure. To facilitate the replication and use of the system in Cloud environment, we decided to use Docker Containers for run the database and application server.

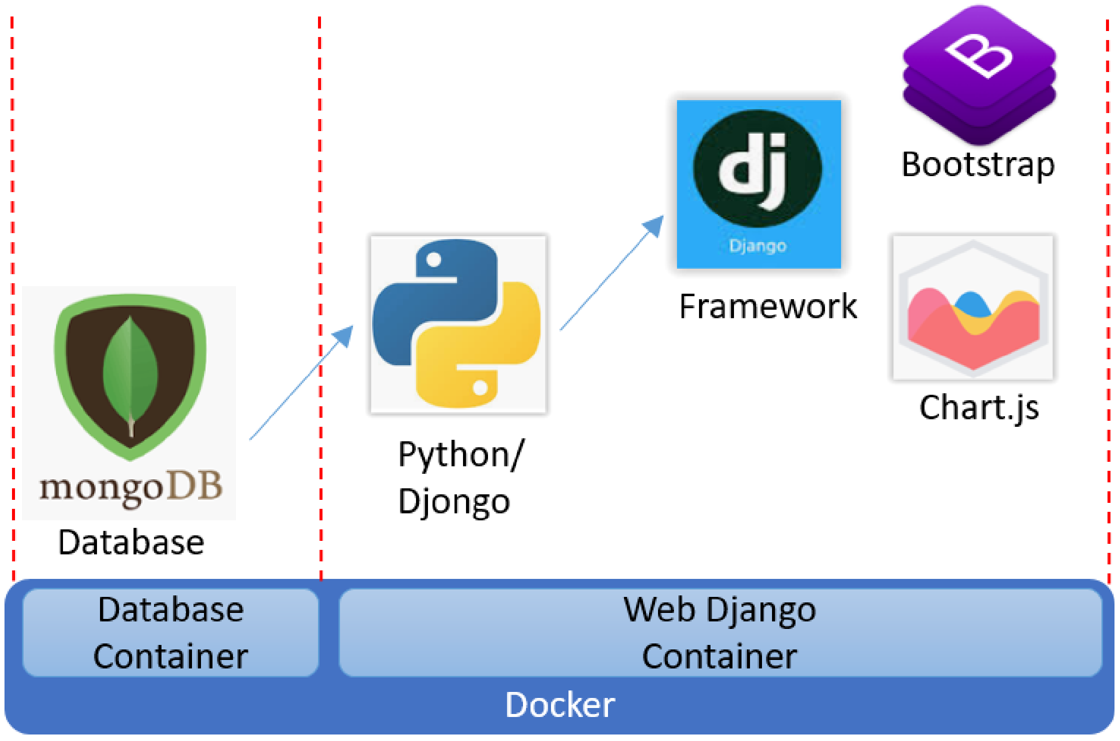


Figure 1. Technical Solution.

2.1 MongoDB

MongoDB is a document-oriented NoSQL database that uses JSON-like documents with schema. The advantage of use NoSQL database is not necessary define the schema, like in normal relational database. And, another advantage, is the possibility to improve the performance of queries, because is not necessary use joins [1].

2.2 Django

Django is a web framework, which permit use Model-View-Template (MVT) architectural pattern. We decided to select Django for development de system because it is the most common Python web framework, allowing reusability of components, rapid development and some user control interface [2].

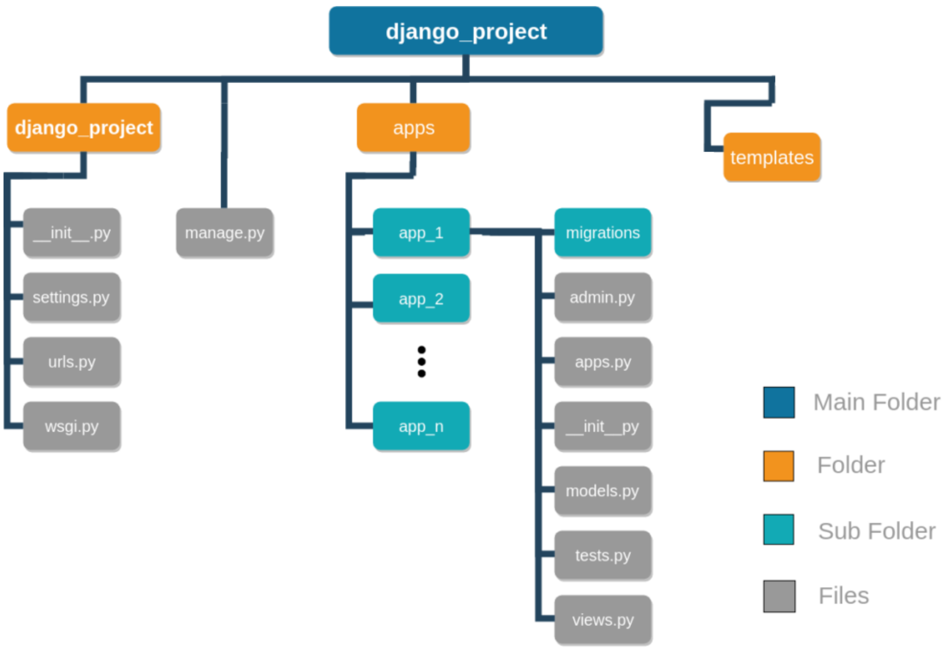


Figure 2. Django Directory and Files Structure. (**Adapted from**: [3])

The last figure exemplifies the Django directory structure. It is divide in a project, that could have several applications. Which application has models and views. The following graph explain the flux of data in which application.

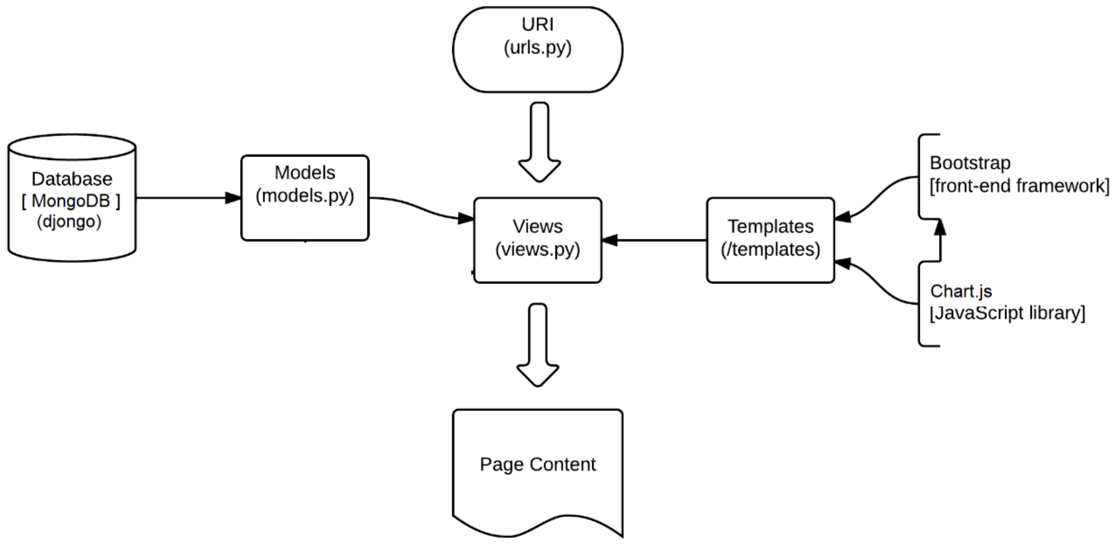


Figure 3. Django Application flux. (**Adapted from**: [4])

2.3 Chart.js

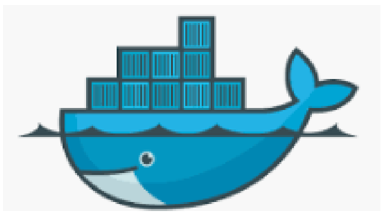
Chart.js

Chart.js is a JavaScript library that allows you to draw different types of charts by using the HTML5 canvas element [5].

2.4 Bootstrap

Bootstrap is an open source toolkit for developing with HTML, CSS, and Java Script. The main objective of Boostrap framework is build responsive and mobile-first websites. There are several templates and different objects to use in web interface [6].

2.5 Docker



Docker permit create isolated containers with light weighted and easy to reproduce. For example, it allows create simple database servers, similar a virtual machine, that could communicate with each other over simulated network [7].

This solution allows fast deploy of the application in cloud environment.

3 NoSQL Modeling

In this section we will show how one classical SQL Model looks like in a NoSQL implementation.

3.1 Classical SQL Model

The following data model shows the normalized structure necessary for data in the project. Notice that is necessary JOINs to link the information of the tables.

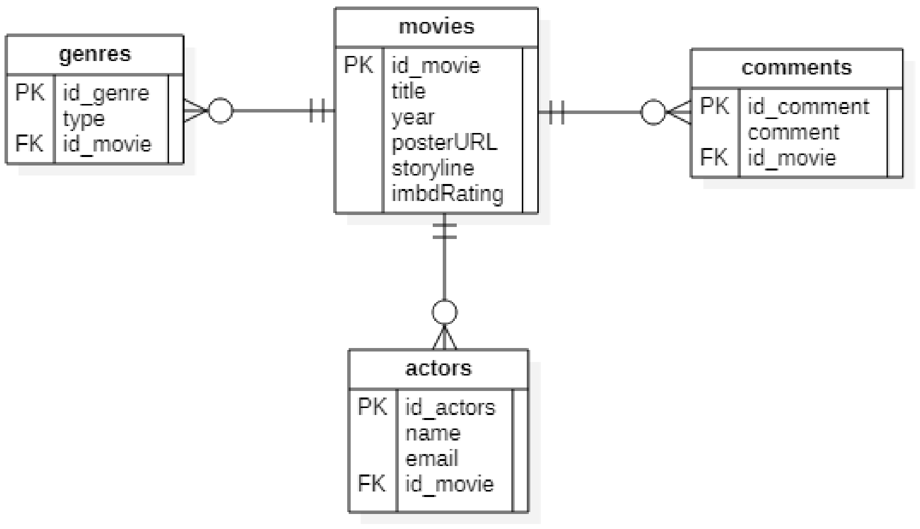


Figure 4. Relational Data Model for the Cinema Rating Project.

3.2 NoSQL Model

The following JSON exemplifies how the NoSQL Model are structured. We can observe that is much more simple than the relational model (figure 4), and it is possible to add new variables without necessity to change the schema of database. Other big advantage is not necessary use JOINs, because all the information is in one collection.

```
{
    "id" : 1,
    "title" : "Anand",
    "year" : "1971",
    "genres" : [
        { "genreType" : "Drama" },
        { "genreType" : "Fiction" }
    ],
    "ratings" : [
        { "ratingValue" : 5 },
        { "ratingValue" : 4 },
        { "ratingValue" : 5 }
    ],
    "posterURL" : "250_AL_.jpg",
    "storyline" : "Georgekutty (Mohanlal) is a cable TV …",
    "imdbRating" : 8.9,
    "actors" : [
        { "name" : "Gustav", "email" : "it@it.eu" },
        { "name" : "Induraj", "email" : "it2@it.eu" }
    ],
    "comments" : [
        { "commentT" : "This is the first comment." },
        { "commentT" : "This is the second comment." },
    ]
}
```

4 Results and Screenshots

This section exemplifies the main pages of the project:

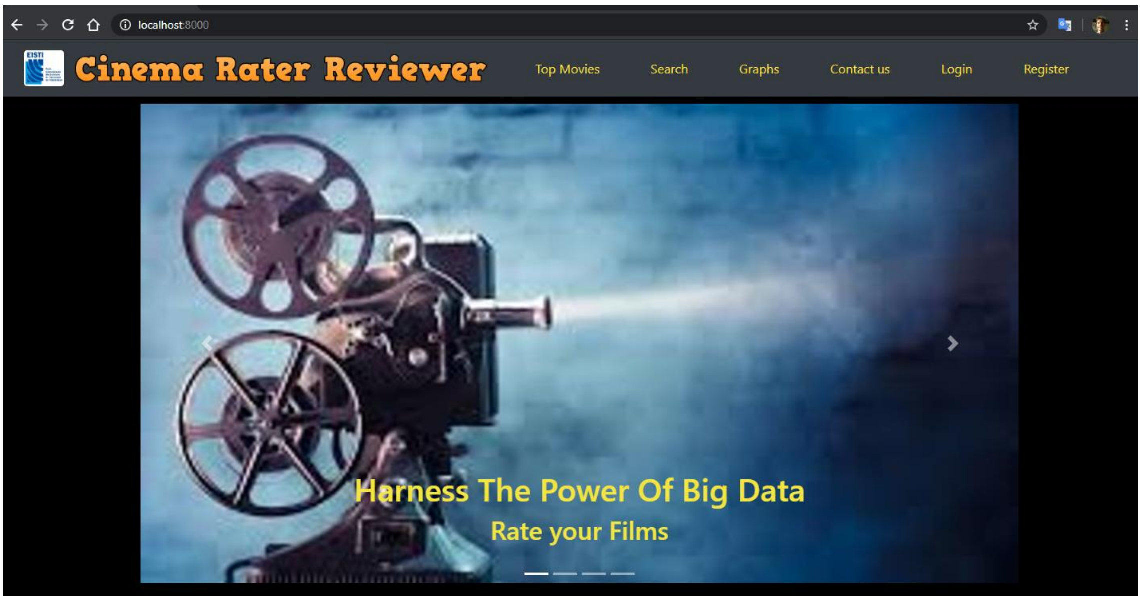
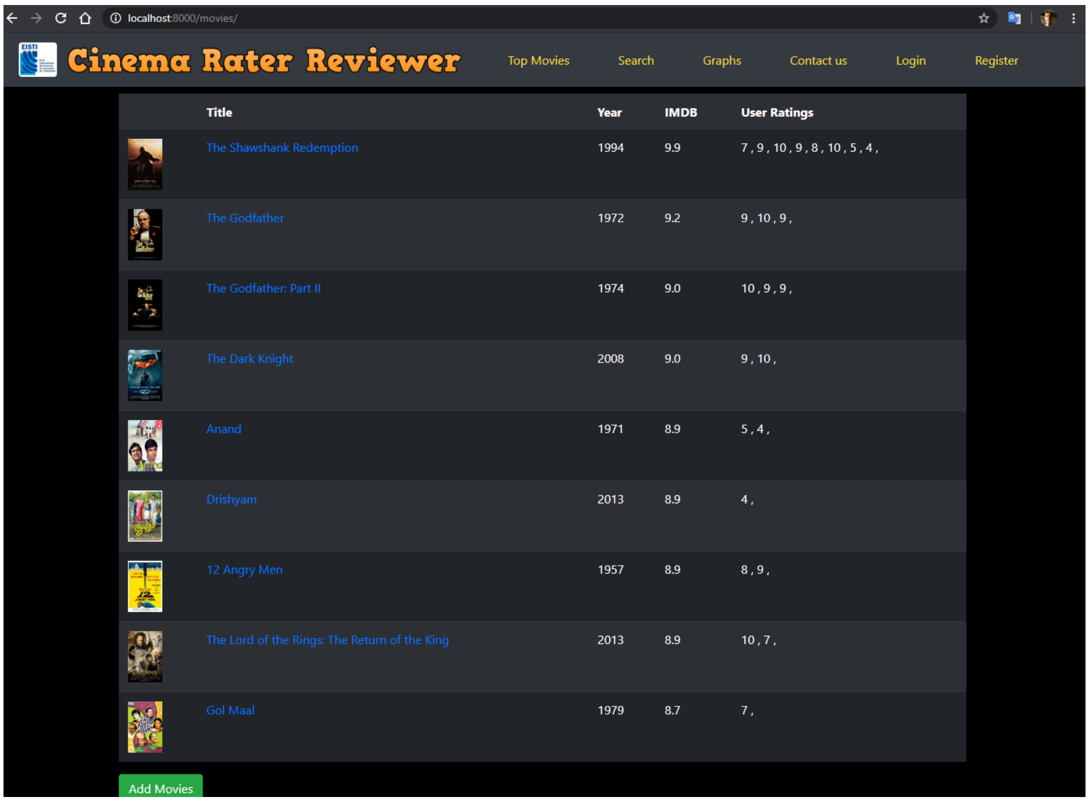
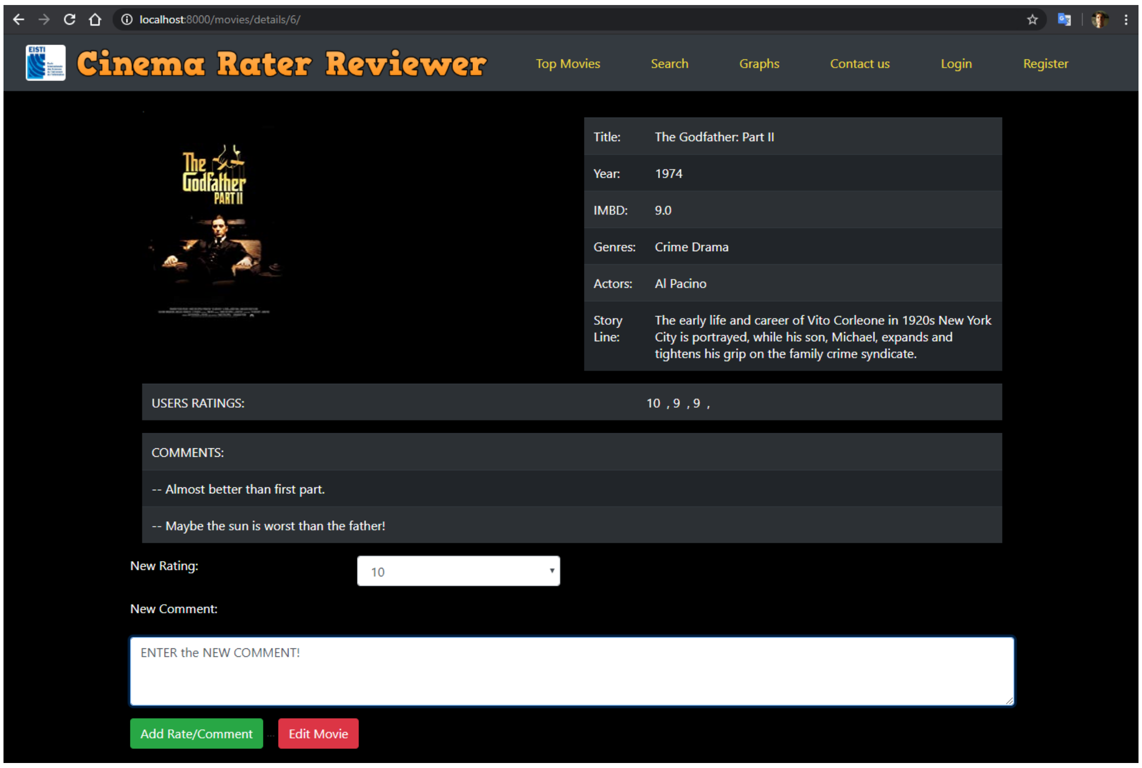


Figure 5. Relational Main Page.



	Title	Year	IMDB	User Ratings
	The Shawshank Redemption	1994	9.9	7 , 9 , 10 , 9 , 8 , 10 , 5 , 4 ,
	The Godfather	1972	9.2	9 , 10 , 9 ,
	The Godfather: Part II	1974	9.0	10 , 9 , 9 ,
	The Dark Knight	2008	9.0	9 , 10 ,
	Anand	1971	8.9	5 , 4 ,
	Drishyam	2013	8.9	4 ,
	12 Angry Men	1957	8.9	8 , 9 ,
	The Lord of the Rings: The Return of the King	2013	8.9	10 , 7 ,
	Gol Maal	1979	8.7	7 ,

Figure 6. Top Movies.



Title:	The Godfather: Part II
Year:	1974
IMBD:	9.0
Genres:	Crime Drama
Actors:	Al Pacino
Story Line:	The early life and career of Vito Corleone in 1920s New York City is portrayed, while his son, Michael, expands and tightens his grip on the family crime syndicate.

USERS RATINGS: 10 , 9 , 9 ,

COMMENTS:

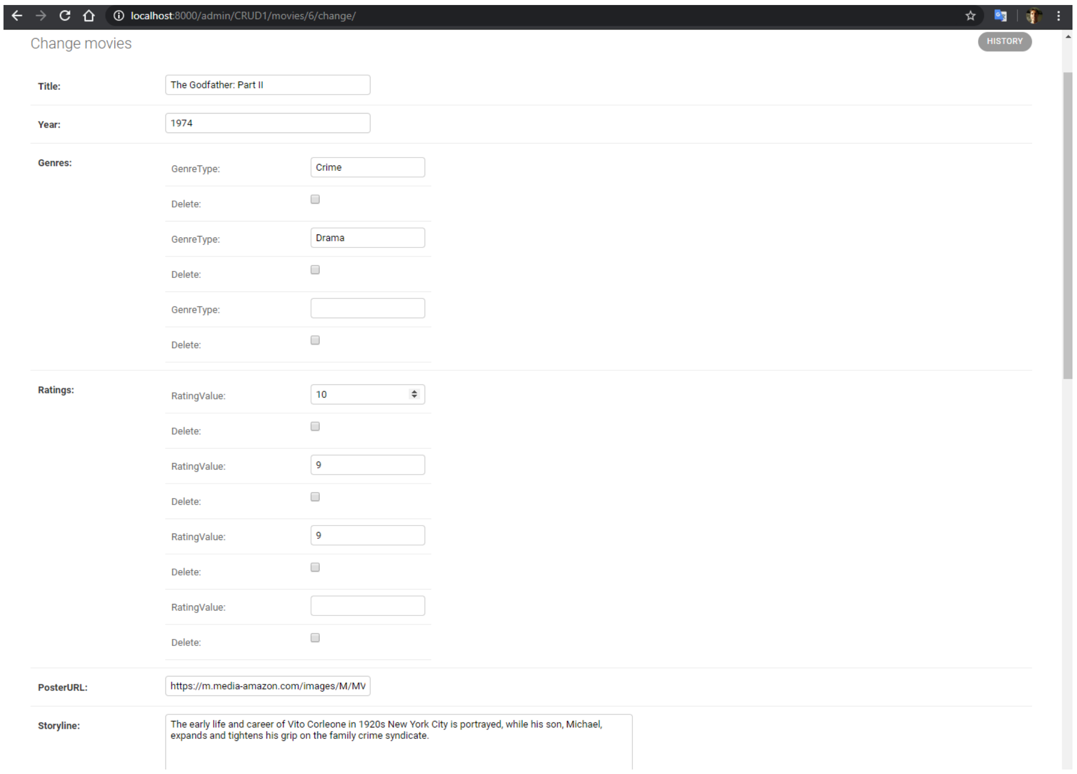
-- Almost better than first part.

-- Maybe the sun is worst than the father!

New Rating: 10

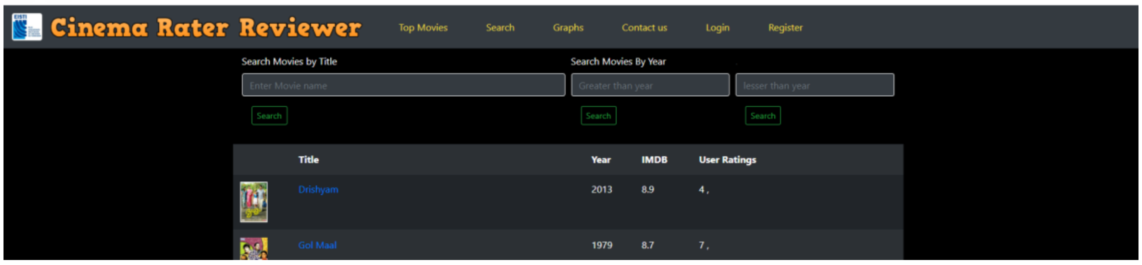
New Comment: ENTER the NEW COMMENT!

Figure 7. Movie Detail and Add new Ratings/Comments.



Title:	The Godfather: Part II
Year:	1974
Genres:	GenreType: Crime; Delete: ☐; GenreType: Drama; Delete: ☐; GenreType: ; Delete: ☐
Ratings:	RatingValue: 10; Delete: ☐; RatingValue: 9; Delete: ☐; RatingValue: 9; Delete: ☐; RatingValue: ; Delete: ☐
PosterURL:	https://m.media-amazon.com/images/M/MV
Storyline:	The early life and career of Vito Corleone in 1920s New York City is portrayed, while his son, Michael, expands and tightens his grip on the family crime syndicate.

Figure 8. Add/Edit/Delete Movie.



	Title	Year	IMDB	User Ratings
	Drishyam	2013	8.9	4 ,
	Gol Maal	1979	8.7	7 ,

Figure 9. Dynamic Filtering Query

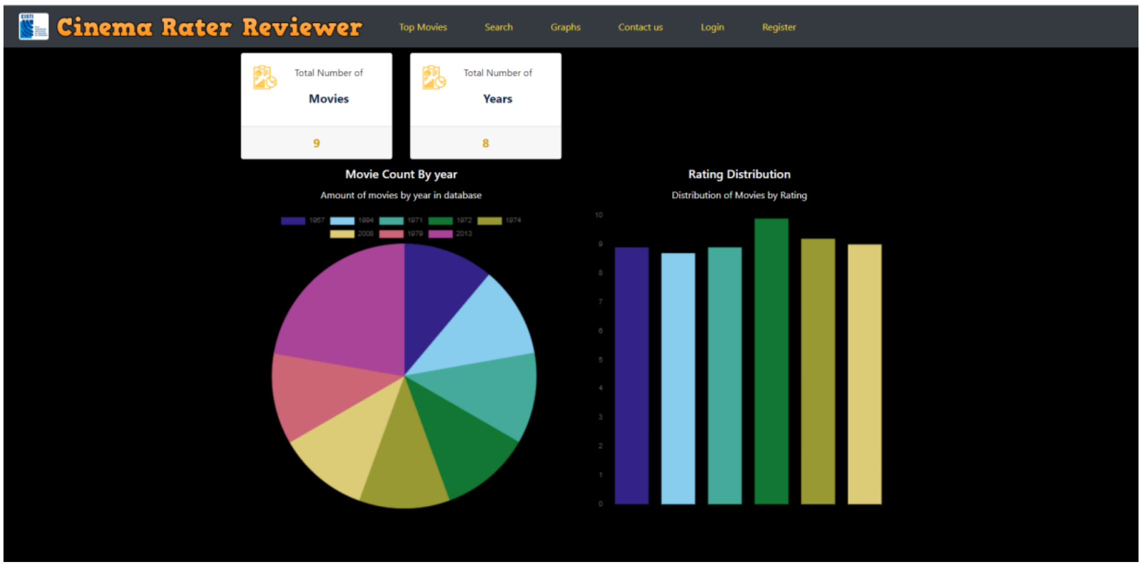


Figure 10. Movie Graphs.

5 Conclusion

The project allowed to work with NoSQL database (MongoDB) in a simulated enterprise environment, using containers in Docker and Django Web interface. Using the Docker containers, it is possible deploy the system in Cloud services like Amazon AWS, Google Cloud Platform or Microsoft Azure. Using the Django framework, we can fast develop a web system with users control and possibility of add values in fields.

6 Bibliographic References

[1] MongoDB, "MongoDB Documentation," [Online]. Available: https://docs.mongodb.com/. [Accessed March 2020].

[2] Django, "Django Documentation.," [Online]. Available: https://docs.djangoproject.com/en/3.0/. [Accessed March 2020].

[3] H. Sayyed, "Django Project Structure Best Practice 2019," [Online]. Available: https://studygyaan.com/django/best-practice-to-structure-django-project-directories-and-files. [Accessed March 2020].

[4] Blinkerz, "Bulletin Architeture," 2012. [Online]. Available: https://github.com/caylan/Bulletin/wiki/Architecture. [Accessed 2020].

[5] Chart.js, "Chart.js Documentation.," [Online]. Available: https://www.chartjs.org/docs/latest/. [Accessed March. 2020].

[6] Bootstrap, "Bootstrap Documentation.," [Online]. Available: https://getbootstrap.com/docs/4.1/getting-started/introduction/. [Accessed March 2020].

[7] Docker, "Docker Documentation.," [Online]. Available: https://docs.docker.com/. [Accessed March 2020].

[8] MIT, Heuristic Design and Optimization.

[9] C. Knauer, L. Schlipf, J. Schimidt and T. Hans, Largest inscribed rectangles in convex polygons, University Bayreuth, 2011.

7 Appendix – Tutorial

All the files listed in this tutorial are available in the Zip file of the project.

7.1 Dockers

We configured 3 containers:

- MongoDB server – access the mongo database over network on por 27107;
- Mongo Express – simple webserver for verify the status of MongoDB;
- Django Web Service – server with Django application and files from the Cinema Rating Review web site.

We used Docker-Compose to setup these containers. The docker-compose.yml:

```
version: '3.1'
services:
  mongo:
    image: mongo
    restart: always
    volumes:
      - .:/data/db
    ports:
        - 27017:27017
  mongo-express:
    image: mongo-express
    restart: always
    ports:
      - 8081:8081
  web:
    build: .
    restart: always
    command: python manage.py runserver 0.0.0.0:8000
    ports:
      - 8000:8000
    links:
        - mongo
```

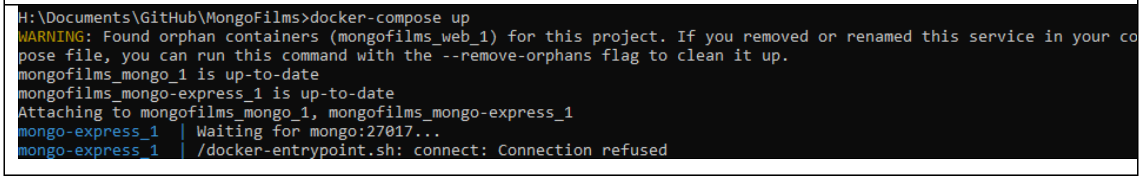
Coping the code directory and installing the python packages requirements in the file "Dockerfile" :

```
FROM python:3
ENV PYTHONUNBUFFERED 1
RUN mkdir /code
```

```
WORKDIR /code
COPY requirements.txt /code/
RUN pip install -r requirements.txt
COPY . /code/
```

For run the containers:

```
docker-compose up
```



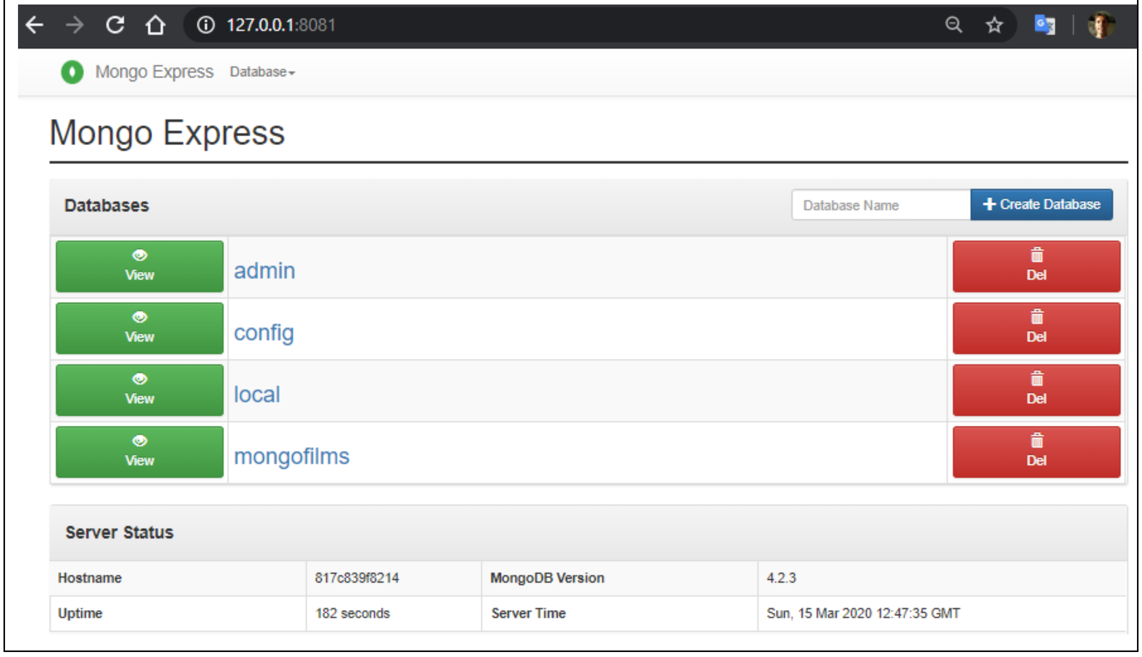
```
H:\Documents\GitHub\MongoFilms>docker-compose up
WARNING: Found orphan containers (mongofilms_web_1) for this project. If you removed or renamed this service in your co
pose file, you can run this command with the --remove-orphans flag to clean it up.
mongofilms_mongo_1 is up-to-date
mongofilms_mongo-express_1 is up-to-date
Attaching to mongofilms_mongo_1, mongofilms_mongo-express_1
mongo-express_1  | Waiting for mongo:27017...
mongo-express_1  | /docker-entrypoint.sh: connect: Connection refused
```

Observation: Sometimes the Django/Djongo does not update the value of HOST and always try to open the MongoDB server in the local host. To solve this problem is possible create a TCP IP PORT FORWARD between the Django and MongoDB servers. For this install SSH in Django Server and SSHD in MongoDB server. After setup the SSH server on MongoDB container (hostname "mongo"), start the port forward with:

```
ssh -L 127.0.0.1:27107:mongo:27107 mongo
```

Verify the services from the Docker Containers:

Mongo – Verify with Mongo Express:



Mongo Express

Databases

View	admin	Del
View	config	Del
View	local	Del
View	mongofilms	Del

Server Status

Hostname	817c839f8214	MongoDB Version	4.2.3
Uptime	182 seconds	Server Time	Sun, 15 Mar 2020 12:47:35 GMT

7.2 Django

Using Docker, the installation of Django and its dependences was done using the "Dockerfile" and "requirements.txt".

To create a project:

```
django-admin startproject cinema_review_rating .
```

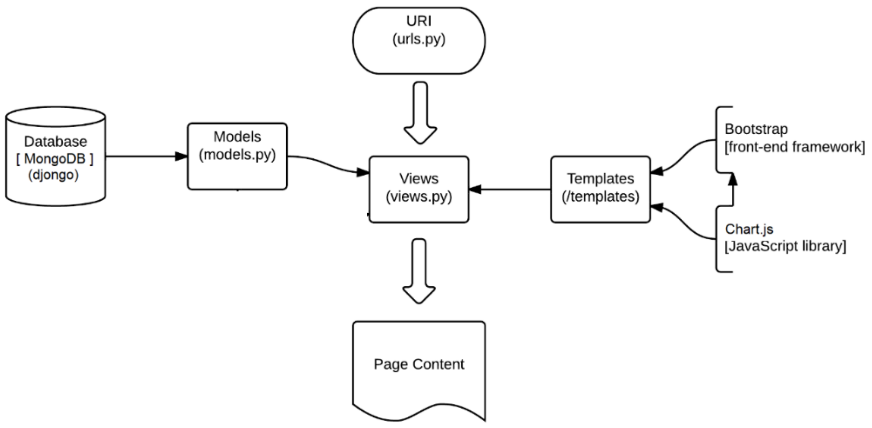
To create an application:

```
python manage.py startapp CRUD
```

Configure to access the Mongo Database in settings.py:

```
DATABASES = {
       'default': {
           'ENGINE': 'djongo',
           'NAME': 'cenimaRating',
           'PORT': '27017',
           'HOST': 'localhost',
       }
   }
```

The startAPP command will create a directory with the files: models.py, views.py and urls.py. This are the way the files interact within.



The models.py define the structure of database, that could be change in the future. Using MongoDB is possible to create embedded documents. The following lines exemplifies (full file is available in Zip file or Git link).

```
class Movies(models.Model):
    title = models.CharField(max_length=200)
    year = models.CharField(max_length=4)

    genres = models.ArrayField(
        model_container=Genre,
        model_form_class=GenreForm
    )
…
```

After create the models.py, is necessary migrate to sync the database.

```
python manage.py makemigrations CRUD

python manage.py migrate
```

The views.py references the models to get the data from database, treat, combine with the template (movies_details.html) and render the web page. The function above exemplify:

```
def movie_details(request, idMovie):
    movieTitle = Movies.objects.get(id=idMovie)
    moviess = Movies.objects.filter(title__icontains = movieTitle)
    return render(request, 'movies_details.html', {'moviess': moviess})
```

In template file, there are commands to iterate over the data passed (moviess) by the view and create the html file, like an example:

```
{% extends "base.html" %}
{% block content %}
<table class="table table-striped table-dark">
    {% for movie in moviess %}
    <tr>
      <td>
        <img src="{{  movie.posterURL }}" />
      </td>
      <td> {{ movie.title }}  </a></td>
    </tr>
    {% empty %}
    <tr>
        <td>No films in database yet.</td>
        <td></td>
    </tr>
    {% endfor %}
</table>
```

Run the Django application:

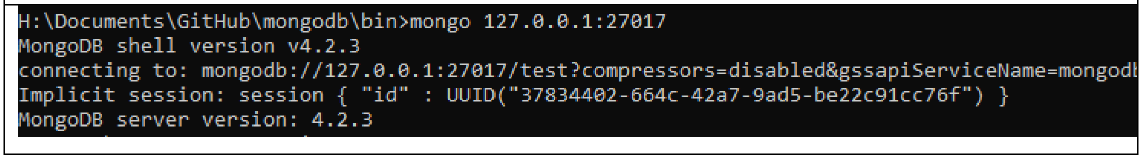
```
python manage.py runserver 0.0.0.0:8000
```

7.3 MongoDB

The installation was done in Dockers, and is not necessary another configuration. The server could be accessed by the hostname 'mongo' or by localhost from host machine. The follow commands are using the mongo client from Windows host.

Connect:

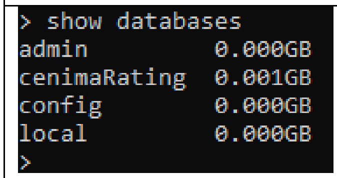
```
mongo 127.0.0.1:27017
```



```
H:\Documents\GitHub\mongodb\bin>mongo 127.0.0.1:27017
MongoDB shell version v4.2.3
connecting to: mongodb://127.0.0.1:27017/test?compressors=disabled&gssapiServiceName=mongod
Implicit session: session { "id" : UUID("37834402-664c-42a7-9ad5-be22c91cc76f") }
MongoDB server version: 4.2.3
```

Verify the databases created:

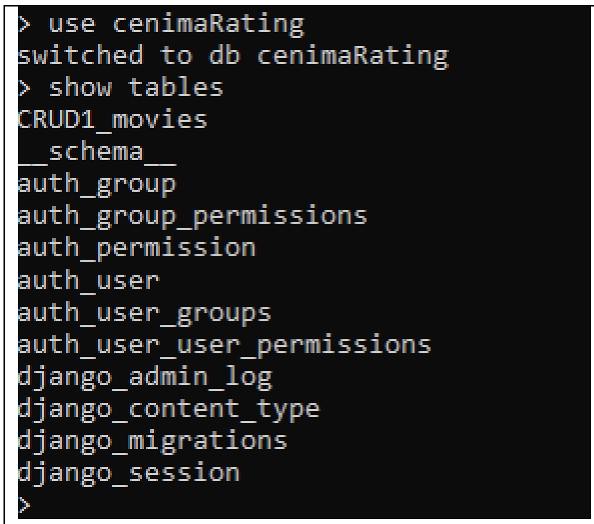
```
Show databases
```



```
> show databases
admin          0.000GB
cenimaRating   0.001GB
config         0.000GB
local          0.000GB
>
```

Verify the collections in database of applications CinemaRating:

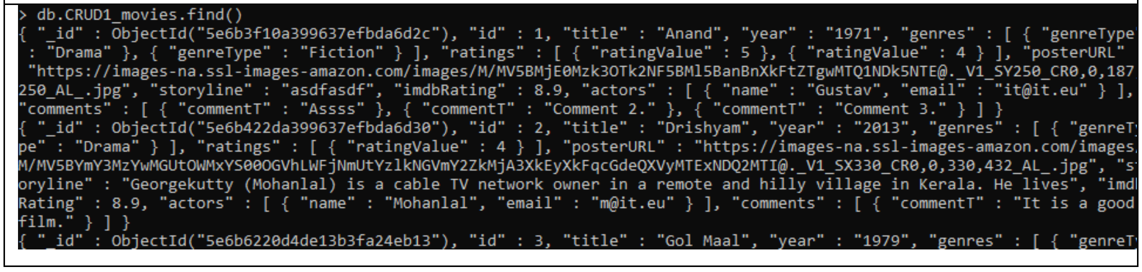
```
use cinemarating

show tables
```



```
> use cenimaRating
switched to db cenimaRating
> show tables
CRUD1_movies
__schema__
auth_group
auth_group_permissions
auth_permission
auth_user
auth_user_groups
auth_user_user_permissions
django_admin_log
django_content_type
django_migrations
django_session
>
```

Show documents in collection:

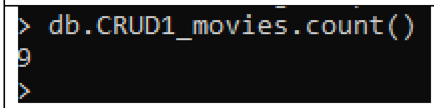
```
db.CRUD1_movies.find()
```



```
> db.CRUD1_movies.find()
{ "_id" : ObjectId("5e6b3f10a399637efbda6d2c"), "id" : 1, "title" : "Anand", "year" : "1971", "genres" : [ { "genreType
 : "Drama" }, { "genreType" : "Fiction" } ], "ratings" : [ { "ratingValue" : 5 }, { "ratingValue" : 4 } ], "posterURL"
 "https://images-na.ssl-images-amazon.com/images/M/MV5BMjE0Mzk3OTk2NF5BMl5BanBnXkFtZTgwMTQ1NDk5NTE@._V1_SY250_CR0,0,187
250_AL_.jpg", "storyline" : "asdfasdf", "imdbRating" : 8.9, "actors" : [ { "name" : "Gustav", "email" : "it@it.eu" } ],
"comments" : [ { "commentT" : "Assss" }, { "commentT" : "Comment 2." }, { "commentT" : "Comment 3." } ] }
{ "_id" : ObjectId("5e6b422da399637efbda6d30"), "id" : 2, "title" : "Drishyam", "year" : "2013", "genres" : [ { "genreTy
pe" : "Drama" } ], "ratings" : [ { "ratingValue" : 4 } ], "posterURL" : "https://images-na.ssl-images-amazon.com/images
M/MV5BYmY3MzYwMGUtOWMxYS00OGVhLWFjNmUtYzlkNGVmY2ZkMjA3XkEyXkFqcGdeQXVyMTExNDQ2MTI@._V1_SX330_CR0,0,330,432_AL_.jpg", "st
oryline" : "Georgekutty (Mohanlal) is a cable TV network owner in a remote and hilly village in Kerala. He lives", "imdb
Rating" : 8.9, "actors" : [ { "name" : "Mohanlal", "email" : "m@it.eu" } ], "comments" : [ { "commentT" : "It is a good
film." } ] }
{ "_id" : ObjectId("5e6b6220d4de13b3fa24eb13"), "id" : 3, "title" : "Gol Maal", "year" : "1979", "genres" : [ { "genreTy
```

Count the documents:

```
db.CRUD1_movies.count()
```



```
> db.CRUD1_movies.count()
9
>
```

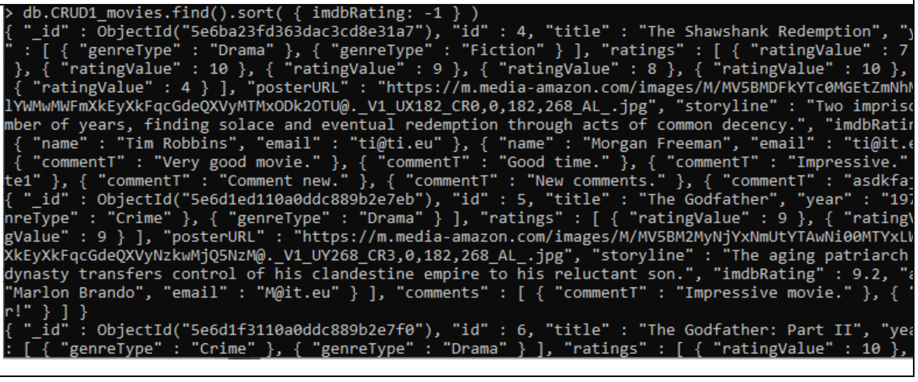
Export collection:

```
mongoexport /db cenimaRating /c CRUD1_movies /o movies.json
```

Import collection:

```
mongoimport /db cenimaRating /c CRUD1_movies /file:movies.json
```

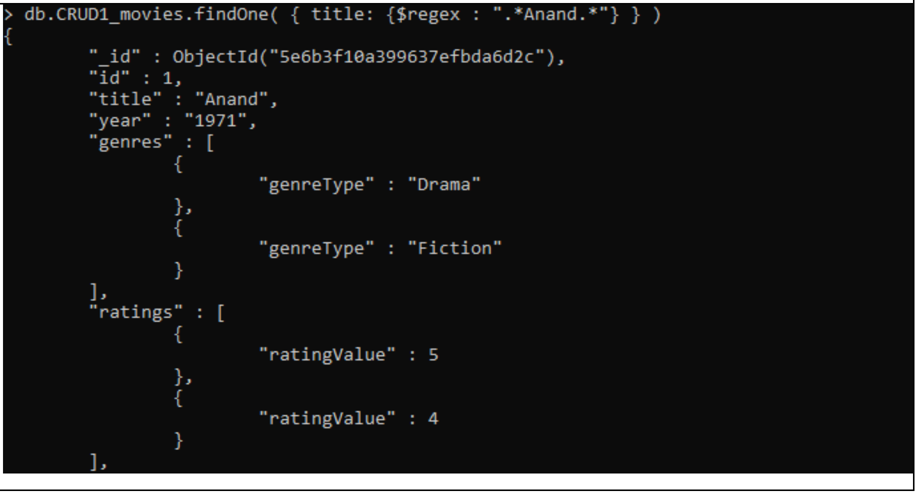
All the operations of query, filter, order are did using Python/Django/Djongo commands. The next commands show the Django/Djongo instruction (views.py) and respective MongoDB command and Result.

Description:	`Get list of movies ordered descending by imdbRating.`
Django/Djongo:	`moviess = Movies.objects.order_by('-imdbRating').all()`
Mongo:	db.CRUD1_movies.find().sort({ imdbRating: -1 })
Result:	

```
> db.CRUD1_movies.find().sort( { imdbRating: -1 } )
{ "_id" : ObjectId("5e6ba23fd363dac3cd8e31a7"), "id" : 4, "title" : "The Shawshank Redemption", "y
" : [ { "genreType" : "Drama" }, { "genreType" : "Fiction" } ], "ratings" : [ { "ratingValue" : 7
 }, { "ratingValue" : 10 }, { "ratingValue" : 9 }, { "ratingValue" : 8 }, { "ratingValue" : 10 },
 { "ratingValue" : 4 } ], "posterURL" : "https://m.media-amazon.com/images/M/MV5BMDFkYTc0MGEtZmNhM
lYWMwMWFmXkEyXkFqcGdeQXVyMTMxODk2OTU@._V1_UX182_CR0,0,182,268_AL_.jpg", "storyline" : "Two impriso
mber of years, finding solace and eventual redemption through acts of common decency.", "imdbRatir
 { "name" : "Tim Robbins", "email" : "ti@ti.eu" }, { "name" : "Morgan Freeman", "email" : "ti@it.e
 { "commentT" : "Very good movie." }, { "commentT" : "Good time." }, { "commentT" : "Impressive."
te1" }, { "commentT" : "Comment new." }, { "commentT" : "New comments." }, { "commentT" : "asdkfa
{ "_id" : ObjectId("5e6d1ed110a0ddc889b2e7eb"), "id" : 5, "title" : "The Godfather", "year" : "197
nreType" : "Crime" }, { "genreType" : "Drama" } ], "ratings" : [ { "ratingValue" : 9 }, { "rating\
gValue" : 9 } ], "posterURL" : "https://m.media-amazon.com/images/M/MV5BM2MyNjYxNmUtYTAwNi00MTYxLW
XkEyXkFqcGdeQXVyNzkwMjQ5NzM@._V1_UY268_CR3,0,182,268_AL_.jpg", "storyline" : "The aging patriarch
dynasty transfers control of his clandestine empire to his reluctant son.", "imdbRating" : 9.2, "a
"Marlon Brando", "email" : "M@it.eu" } ], "comments" : [ { "commentT" : "Impressive movie." }, { "
r!" } ] }
{ "_id" : ObjectId("5e6d1f3110a0ddc889b2e7f0"), "id" : 6, "title" : "The Godfather: Part II", "yea
: [ { "genreType" : "Crime" }, { "genreType" : "Drama" } ], "ratings" : [ { "ratingValue" : 10 },
```

Description:	`Search movie by name.`

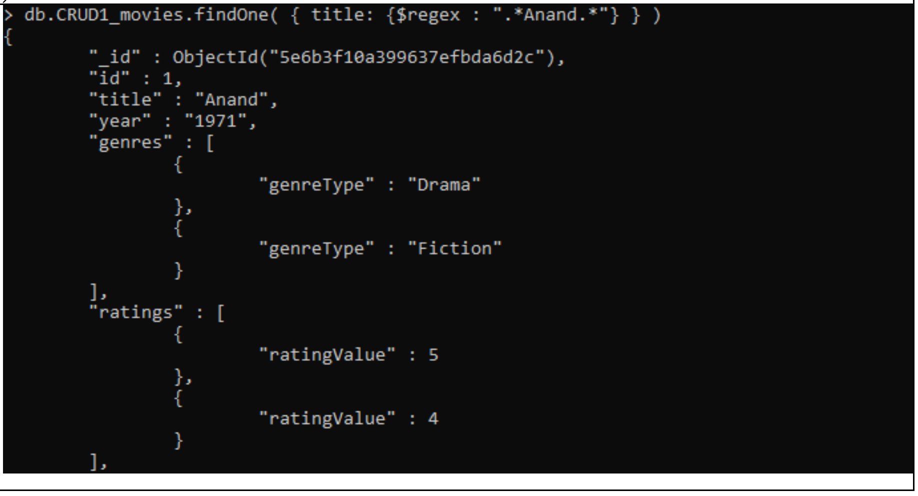
Django/Djongo:	`mTitle = 'Anand' //exemple`
	`moviess = Movies.objects.filter(title__icontains=mTitle)`

Mongo:	db.CRUD1_movies.findOne({ title: {$regex : ".*Anand.*"} })
Result:	

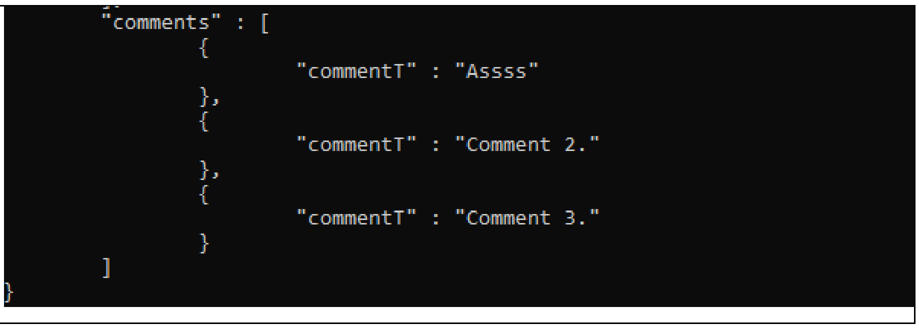
```
> db.CRUD1_movies.findOne( { title: {$regex : ".*Anand.*"} } )
{
        "_id" : ObjectId("5e6b3f10a399637efbda6d2c"),
        "id" : 1,
        "title" : "Anand",
        "year" : "1971",
        "genres" : [
                {
                        "genreType" : "Drama"
                },
                {
                        "genreType" : "Fiction"
                }
        ],
        "ratings" : [
                {
                        "ratingValue" : 5
                },
                {
                        "ratingValue" : 4
                }
        ],
```

Description:	`Save new Comments/Ratings.`

Django/Django:	`def addComment(request,idMovie):`
	`    if request.method == 'POST':`
	`        newC = Comment()`
	`        newC.commentT = request.POST['comment']`
	`        newR = Rating()`
	`        newR.ratingValue = request.POST['newRating']`
	
	`        movie = Movies.objects.get(pk=idMovie)`
	`        movie.comments.append(newC)`
	`        movie.ratings.append(newR)`
	`        movie.save()`
	
	`    movieTitle = Movies.objects.get(id=idMovie)`
	`    …`

Mongo:	db.CRUD1_movies. update({ id:1 }, { $addToSet: { ratings : { ratingValue: 5 } } }) db.CRUD1_movies. update({ id:1 }, { $addToSet: { comments: { commentT: "Comment 3." } } })
Result:	

```
> db.CRUD1_movies.findOne( { title: {$regex : ".*Anand.*"} } )
{
        "_id" : ObjectId("5e6b3f10a399637efbda6d2c"),
        "id" : 1,
        "title" : "Anand",
        "year" : "1971",
        "genres" : [
                {
                        "genreType" : "Drama"
                },
                {
                        "genreType" : "Fiction"
                }
        ],
        "ratings" : [
                {
                        "ratingValue" : 5
                },
                {
                        "ratingValue" : 4
                }
        ],
```

```
        "comments" : [
                {
                        "commentT" : "Assss"
                },
                {
                        "commentT" : "Comment 2."
                },
                {
                        "commentT" : "Comment 3."
                }
        ]
}
```