Student: Tomoiaga Anamaria

**Group:30235**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 3

3. System Architectural Design 3

4. UML Sequence Diagrams 3

5. Class Design 3

6. Data Model 3

7. System Testing 3

8. Bibliography 3

1. Requirements Analysis

# Assignment Specification

Proiectul reprezinta o aplicatie client-server care se ocupa cu administrarea vizualizarii show-rilor online cum ar fi filme, evenimente sportive. Aplicatia are trei tipuri de useri : basic user, premium user si administrator.

Basic user:

* Cauta show, selecta show, viziualiza detalii despre show
* Vizualiza istoricul show-rilor pe care le-a vazut
* Adauga raiting la un show
* Adauga comentariu la show

Premium user:

* Toate operatiile userului basic
* Recomanda un show unui prieten
* Arata interes pentru un show pe care vrea sa-l vada

Administrator:

* CRUD pe show
* CRUD pe user

# Functional Requirements

Ratingul trebuie sa fie de tip float.

Id-ul trebuie sa fie un numar intreg mai mare decat 0.

Tipul show-lui trebuie sa fie “movie”,”theatre” sau “sport event”.

# Non-functional Requirements

Securitate: sistemul este securizat prin cererea unui username si a unei parole pentru a putea utiliza aplicatia.

Performanta: sistemul trebuie sa raspunda repede la cererile userului.

2. Use-Case Model

Use case: <adaugare show>

Level: <user goal level>

Primary actor: <administrator>

Main success scenario: <Introducem toate informatiile despre show, care vor fi salvate in baza de date>

Extensions: <Daca nu introducem toate informatiile despre show vom primi un mesaj de eroare>

3. System Architectural Design

**3.1 Architectural Pattern Description**

Sistemul utilizeaza arhitectura pe layere. Fiecare layer are un scop anume si apeleaza functii din layerele de dupa el.

Layerele folosite sunt:

Controller: contin logica aplicatiiei si transmite datele utilizatorului.

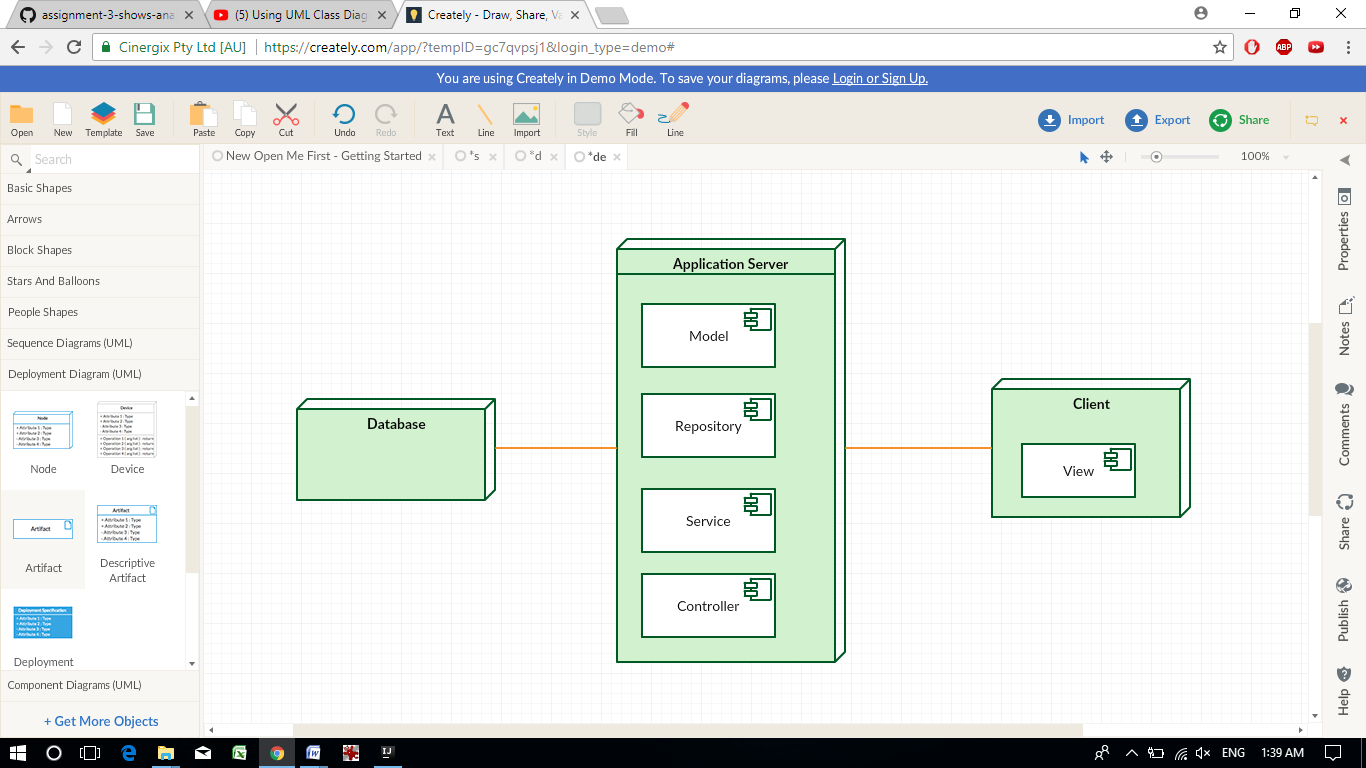
Service: colecatrea datelor de la controller si validarea intrarilor.

Repository: interactiunea cu modelele si efectuarea operatilor pe baza de date.

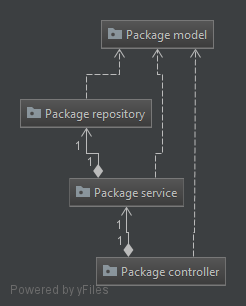
Model: responsabil pentru mentinerea datelor.

**3.2 Diagrams**

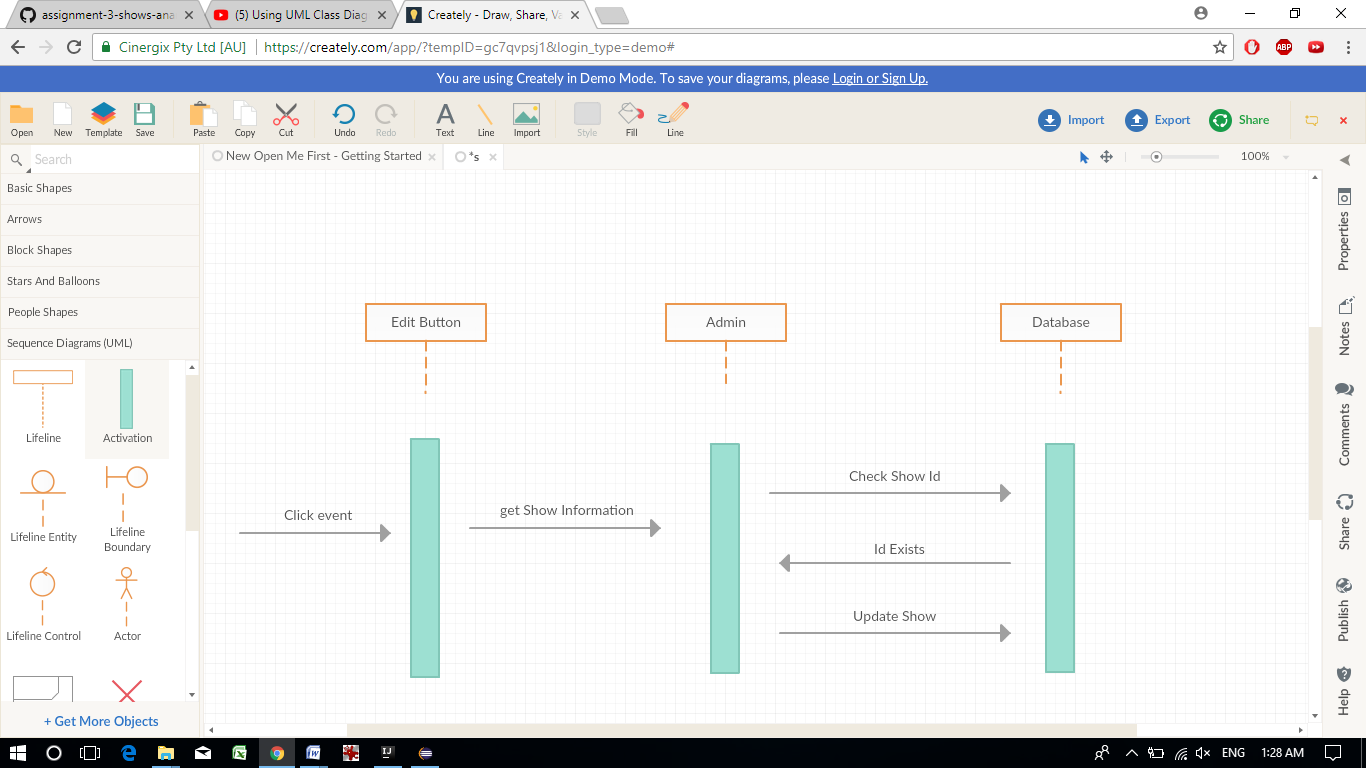
Deployment diagram



Package diagram

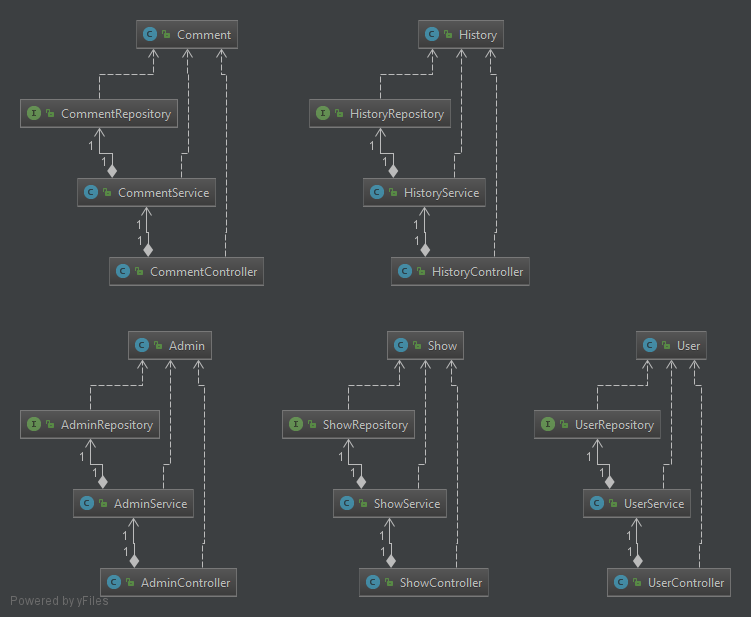


4. UML Sequence Diagrams

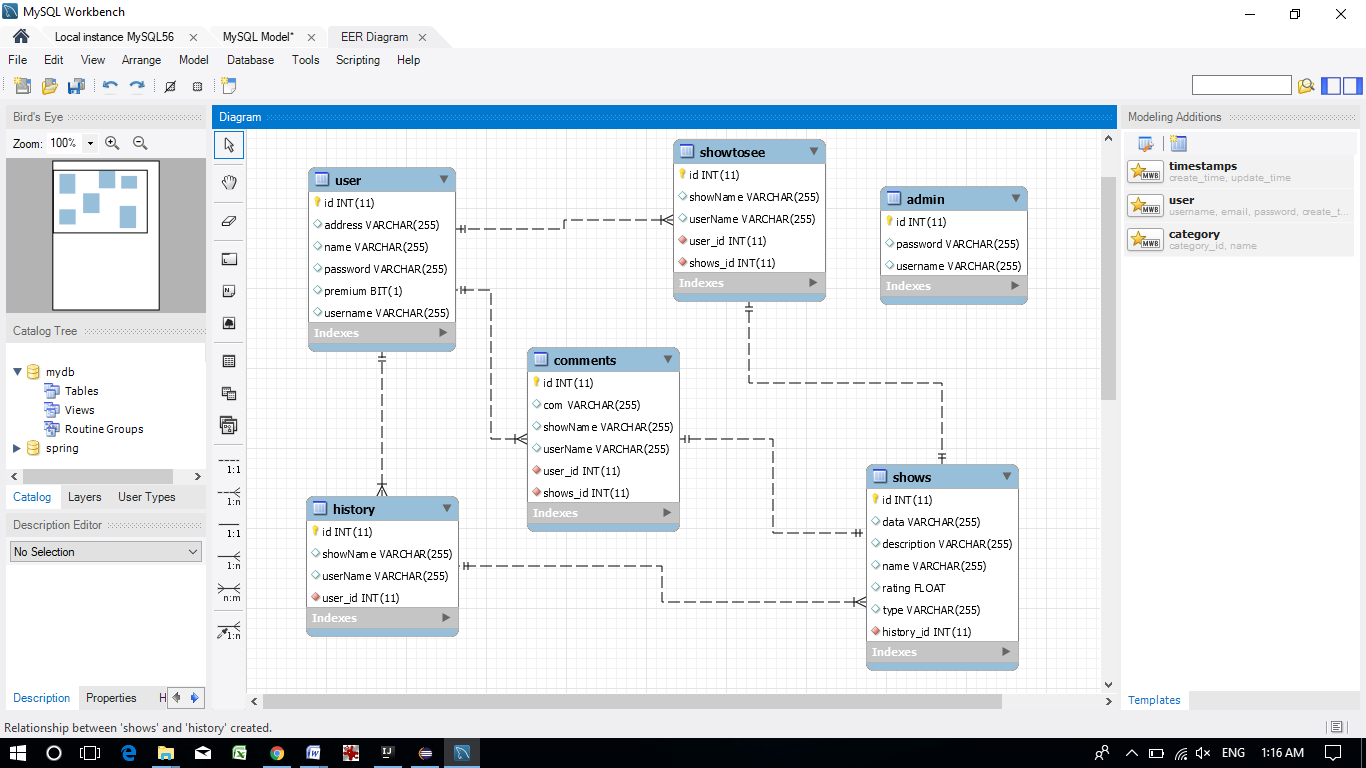


5. Class Design

**UML Class Diagram**

**

6. Data Model

**

7. System Testing

*[Present the used testing strategies (unit testing, integration testing, validation testing) and testing methods (data-flow, partitioning, boundary analysis, etc.).]*

8. Bibliography