National Theater of Cluj Web Application

Student:Chelemen Stefana-Bianca

**Group:30231**

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

The application is a Web application for the National Theater of Cluj-Napoca. The application has two types of users a cashier and an administrator, which must provide a username and a password to use the application. The administrator has control over the cashier, shows and over tickets for a show and a cashier has control over the tickets of a show.

# Functional Requirements

The functional requirements the application has are:

Login and Logout for two types of user.

CRUD on users and Shows

Get info about shows, tickets.

Sell ticket to a show.

Export details about tickets in a CSV or XML format.

# Non-functional Requirements

The non-functional requirements of the application are :

Availability

The application is available 99% of the time.

Performance

The application’s response time shouldn’t be more that 200 ms.

Security

The application should check the top ten OWASP requirements.

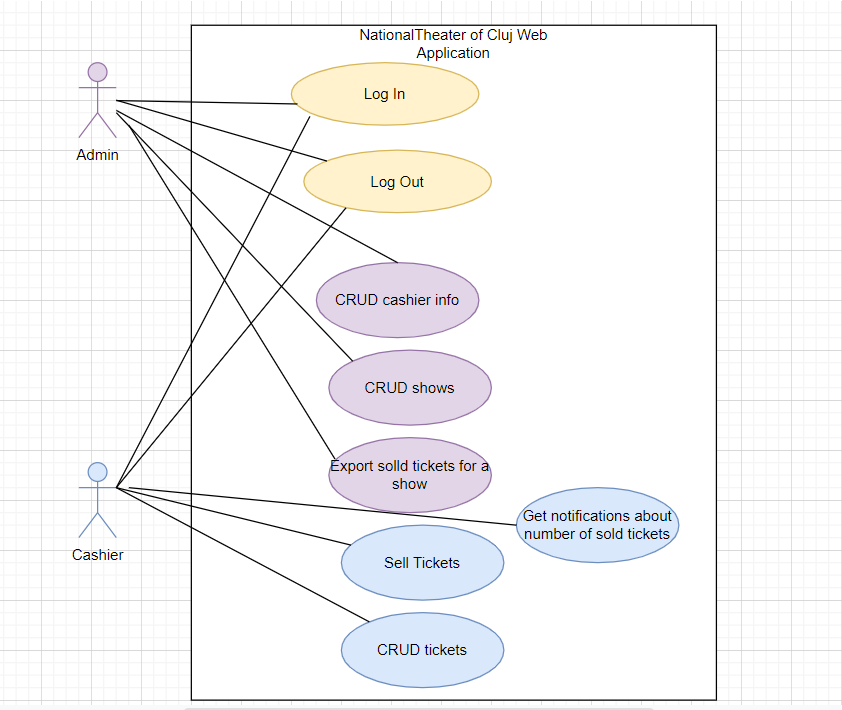
Testability

Number of defects of the application should be less than 5.

Design Constraints

For designing this application we consider the following constarints: C# and .NET framework, and the 3 Tier Architecture Model.

2. Use-Case Model

**

Use-Case description format:

Use case:CRUD cashier info

Level: user-goal level

Primary actor: Admin

Main success scenario: create, read, update or delete a cashier profile

Use case:CRUD shows

Level: user-goal level

Primary actor: Admin

Main success scenario: create, read, update or delete a show

Use case:Export sold tickets for a show

Level: user-goal level

Primary actor: Admin

Main success scenario: create a csv/xml files with data about sold tickets

**Extensions: display error message “Couldn’t createfile”**

Use case:CRUD tickets

Level: user-goal level

Primary actor: Cashier

Main success scenario: create, read, update or delete tickets

Use case:Sell tickets

Level: user-goal level

Primary actor: Cashier

Main success scenario: sell tickets for a show

**Extensions: display error message “There are no more tickets available”**

Use case:Log In

Level: application level

Primary actor: Admin,Cashier

Main success scenario: log in the application

**Extensions: display error message “ wrong username/password”**

Use case:Log Out

Level: application level

Primary actor: Admin,Cashier

Main success scenario: log out of the application

**Extensions: display error message “ couldn’ t log out”**

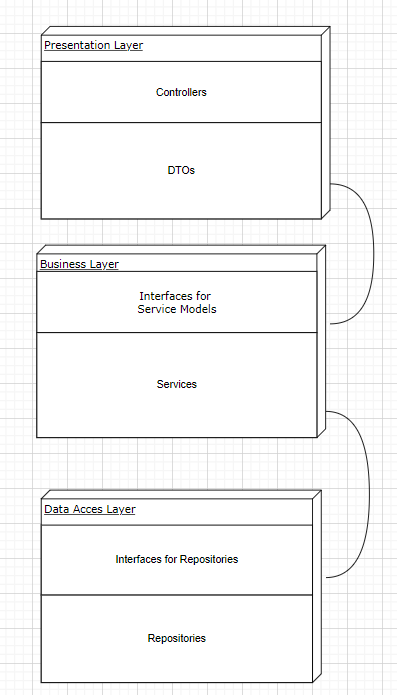
3. System Architectural Design

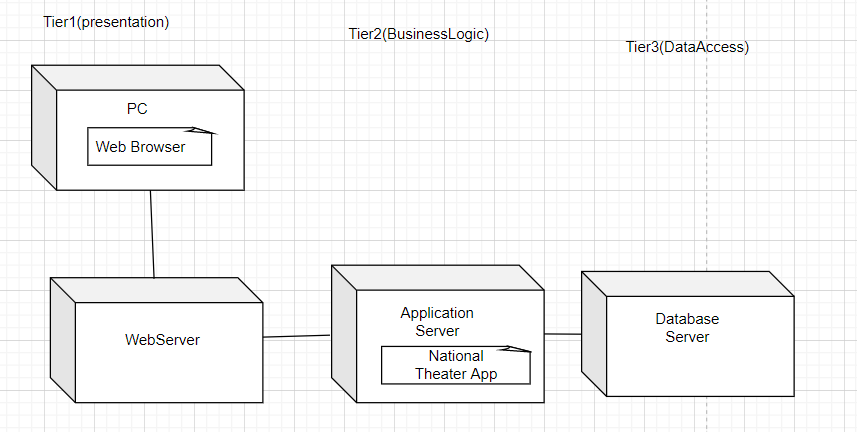
**3.1 Architectural Pattern Description**

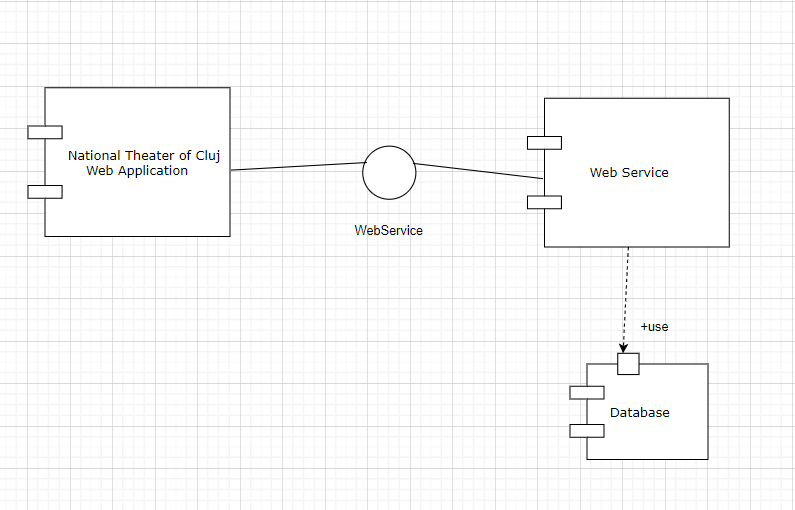
*[Describe briefly the used architectural patterns.]*

**3.2 Diagrams**

The architecture model used for designing this application was three-tier architecture. Three tier architecture is a well-established software application architecture that organizes applications into three logical and physical computing tiers: the presentation tier, or user interface; the application tier, where data is processed; and the data tier, where the data associated with the application is stored and managed.

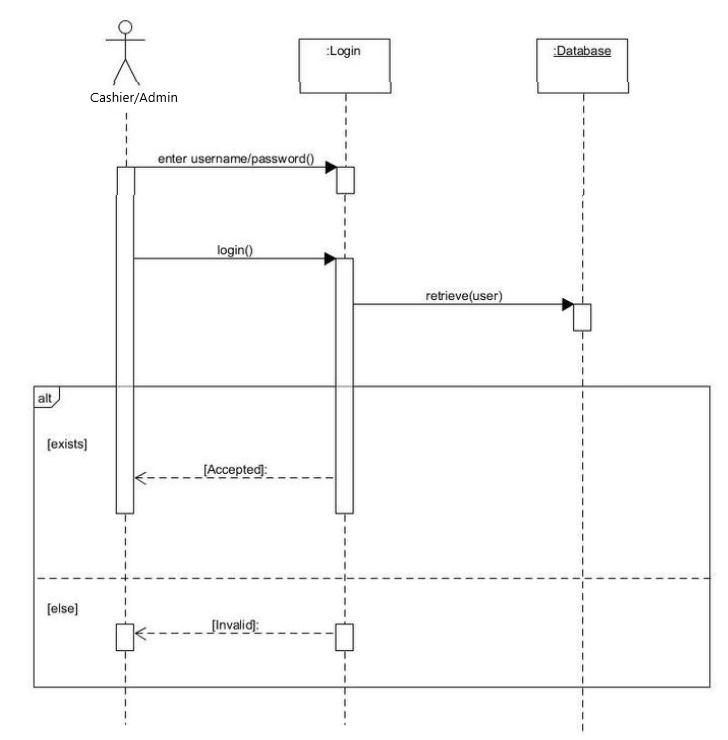
**

**

**

4. UML Sequence Diagrams

Sequence diagram for the login process:

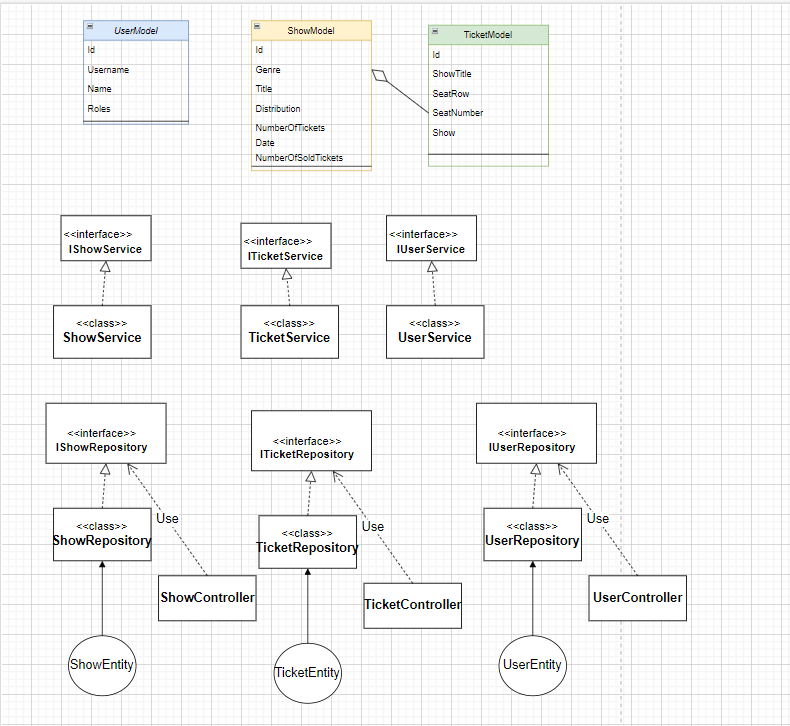


5. Class Design

**5.1 Design Patterns Description**

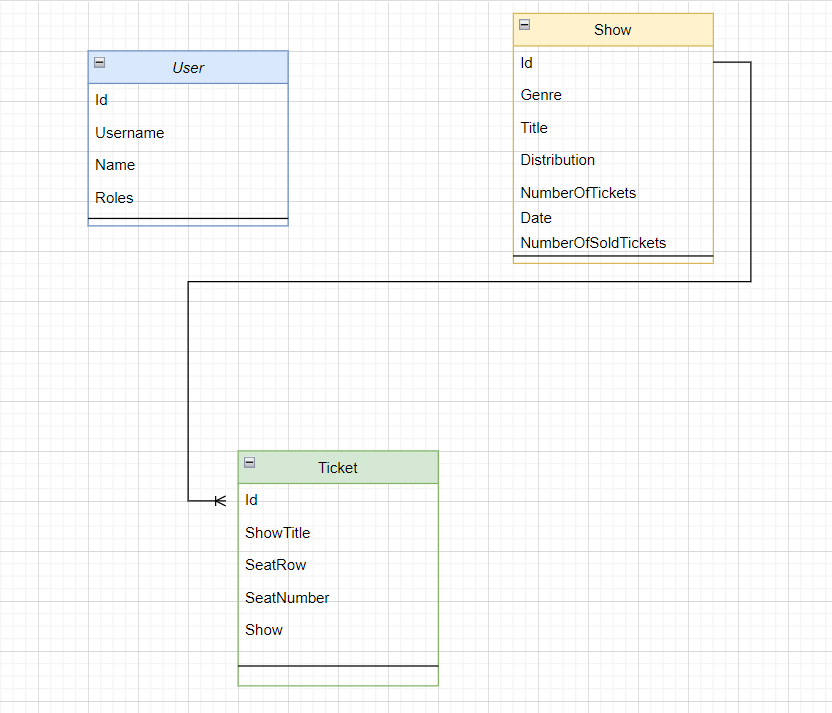
The design pattern that I used was Factory Method. Factory Method is a creational design pattern that provides an interface for creating objects in a superclass, but allows subclasses to alter the type of objects that will be created. I have used this pattern for export all the tickets to a show in CSV format or XML. I will have a interface IExport and then two classes that will implement this interface and implement the method in 2 different ways for xml and csv.

**5.2 UML Class Diagram**



6. Data Model

The application has the User Model for admin and Cashier, the Show model and the Ticket Model, and between Show and Ticket we have a one to many relationship

**

7. System Testing

8. Bibliography