Theater Management Application

Analysis and Design Document

Student: Marius Pop

**Group: 30431**

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

Use JAVA/C# API to design and implement an application for the National Theater of Cluj. The application should have two types of users (a cashier user represented and an administrator) which must provide a username and a password to use the application.

# Functional Requirements

Use JAVA/C# API to design and implement an application for the National Theater of Cluj. The application should have two types of users (a cashier user represented and an administrator) which must provide a username and a password to use the application.

The administrator user can perform the following operations:

CRUD on cashiers’ information.

CRUD on the list of shows that are performed at the theater. Keep track of the Genre (Opera, Ballet), Title, Distribution list (a long string is enough), Date of the show and the Number of tickets per show.

From time to time he can export all the tickets that were sold for a certain show (either in a csv or xml file).

The cashier can perform the following operations:

Sell tickets to a show. A ticket should hold information about the seat row and seat number.

The system should notify the cashier that the number of tickets per show was not exceeded.

A cashier can see all the tickets that were sold for a show, cancel a reservation or edit the seat.

# Non-functional Requirements

The data will be stored in a database.

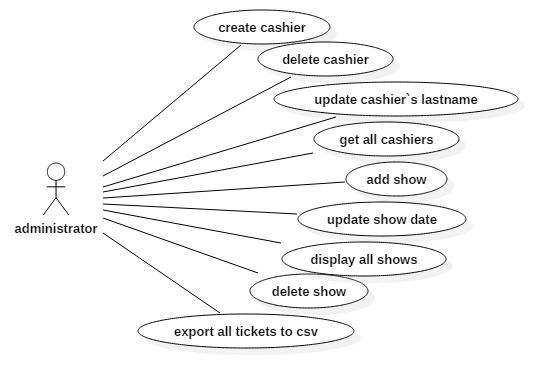
Use the Layers architectural pattern to organize your application.

Passwords are encrypted when stored to the database with a one-way encryption algorithm.

Provide unit tests for the number of tickets for show exceeded validation and the encryption algorithm.

Use **factory method (not factory)** for export to csv/xml.

2. Use-Case Model



*Use case:*

*🡪create a new cashier in the database*

*🡪delete cashier from the database*

*🡪update the last name of one of the existing cashiers*

*🡪display all cashiers that are currently in the database*

*🡪add a new show in the database*

*🡪update one of the already existing show`s date*

*🡪display all available shows*

*🡪delete one of the show that currently exists in the database*

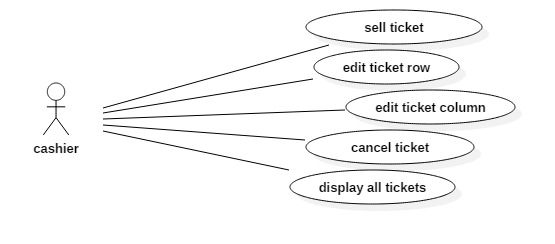
*🡪export all the tickets sold for shows in a .csv file*

*Level: user-goal level*

*Primary actor: administrator*

*Main success scenario: successfully executes all the functions that were mentioned above.*

*Extensions: failing while executing one or more of the functions mentioned above*



*Use case:*

*🡪sell a ticket for one of the shows available in the database*

*🡪edit the ticket row for an already bought ticket*

*🡪 edit the ticket column for an already bought ticket*

*🡪cancel one of the tickets from the database*

*🡪display all currently active tickets*

*Level: user-goal level*

*Primary actor: cashier*

*Main success scenario: successfully executes all the functions that were mentioned above.*

***Extensions: failing while executing one or more of the functions mentioned above***

3. System Architectural Design

**3.1 Architectural Pattern Description**

*The architectural design pattern used was Layers. There are present 3 layers :*

*🡪Presentation layer where are stored the classes belonging to the Graphical User Interface*

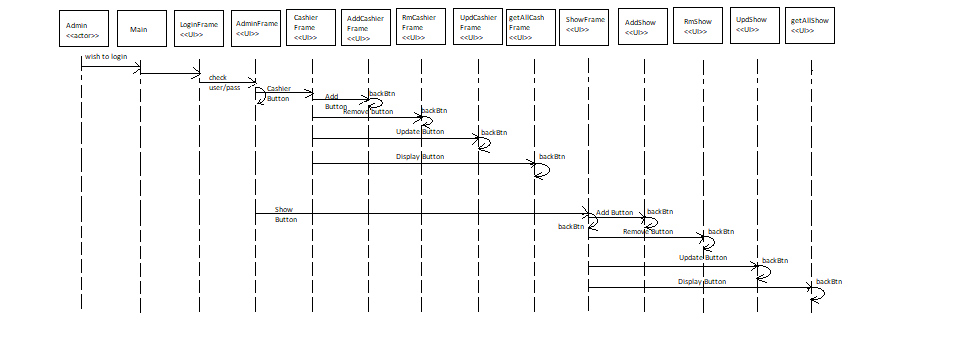
*🡪Business layer were the data related to the services can be found.*

*🡪Data Access layer were database relationships methods are present.*

**3.2 Diagrams**



4. UML Sequence Diagrams

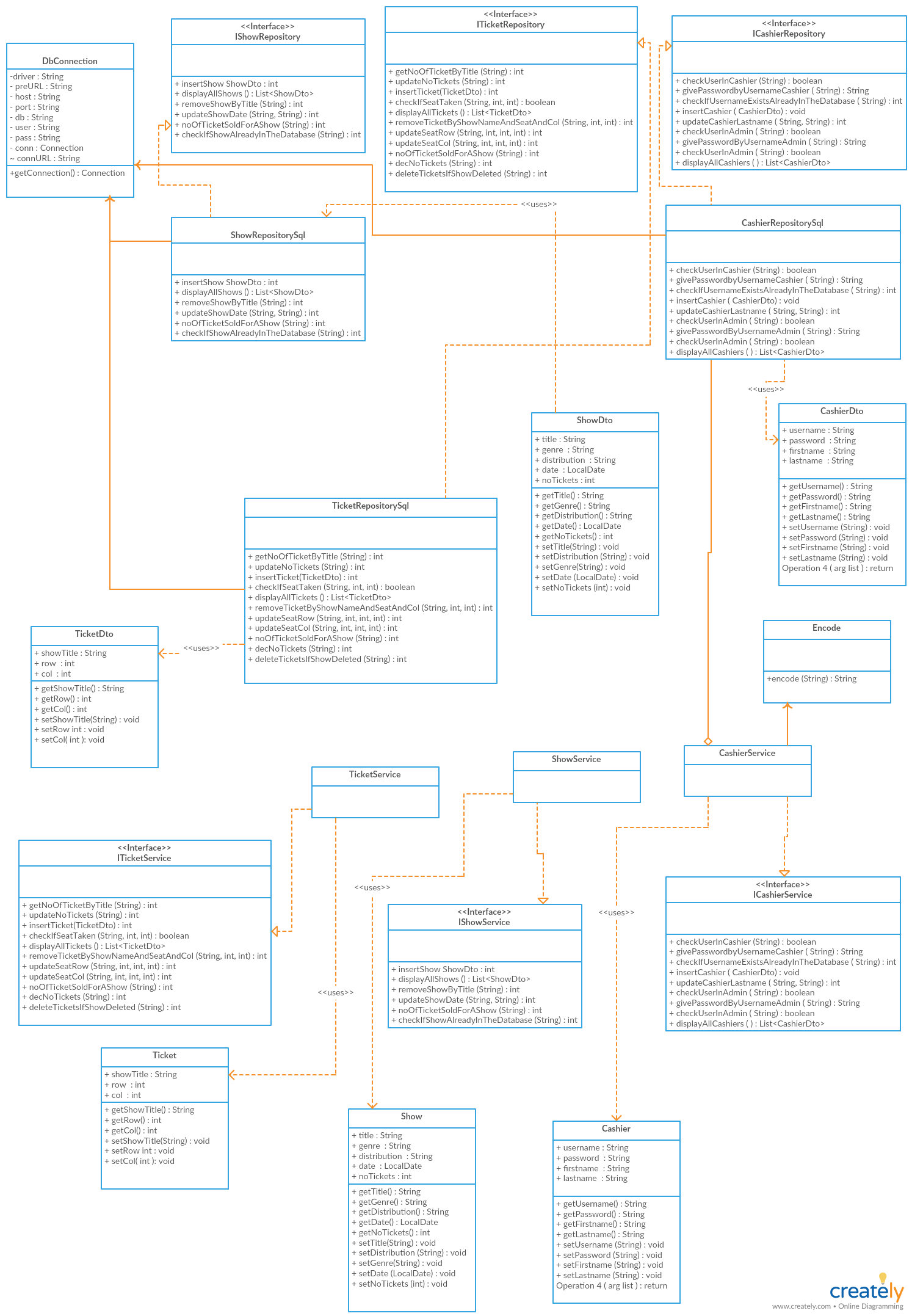


5. Class Design

**5.1 Design Patterns Description**

*[Describe briefly the used design patterns.]*

**5.2 UML Class Diagram**



6. Data Model

*In order to achieve the goal of this project, 2 models have been used :*

*🡪Data Transfer model including CashierDTO. ShowDTO, TicketDTO.*

*🡪Business model including Cashier, Show, Ticket.*

7. System Testing

*I used unit testing for the encryption method in order to check if after applying encode method for a given password, the result will match the corresponding password from the database. In case not, the test will fail and the red bar will be displayed in the class.*

8. Bibliography

<https://www.codeproject.com/Articles/654670/Layered-Application-Design-Pattern>

<https://www.safaribooksonline.com/library/view/software-architecture-patterns/9781491971437/ch01.html>

<https://en.wikipedia.org/wiki/Service_layer_pattern>