National Theater of Cluj-Napoca Web Application

Student: Izabella Bartalus

**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 user represented and an administrator) which provide a username and a password to use the application. The administrator has control over the cashier’s information, about the shows and over tickets for a show and a cashier has control over the tickets of a show.

# Functional Requirements

The functional requirements provided by the application are:

* Login and logout for the users;
* CRUD on cashier’s information;
* CRUD on shows;
* CRUD on tickets;
* Export the sold tickets for a show into a .csv file.

# 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 constraints: C# and .NET framework, and the Three Tier Architecture Model.

2. Use-Case Model

Use case: Log In

Level: application level

Primary actor: Admin, Cashier

Main success scenario: log in the application

Extensions: Display error message “Invalid 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 “Could not log out!”

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 files with data about sold tickets

Extensions: Display error message “Could not create file!”

Use case: CRUD tickets

Level: user-goal level

Primary actor: Cashier

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

Diagram

Description automatically generated

3. System Architectural Design

**3.1 Architectural Pattern Description**

Layered Architecture Pattern:

* Components within the layered architecture pattern are organized into horizontal layers, each layer performing a specific role within the application (e.g. presentation logic or business logic).

Model-View-Controller: also known as MVC pattern, divides an interactive application in to 3 parts as:

* model  - contains the core functionality and data;
* view - displays the information to the user (more than one view may be defined);
* controller - handles the input from the user.

We will use a relational database to store out data, and this is how we will be manipulating our data

**3.2 Diagrams**

Diagram

Description automatically generated

4. UML Sequence Diagrams

Sequence diagram for the add show process:

Diagram

Description automatically generated

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 have an interface IWriter and two classes WriteCSV and WriteXML that implement this interface and implement the method in 2 different ways for csv and xml.

**5.2 UML Class Diagram**

Diagram

Description automatically generated

6. Data Model

The Data Model used in the application are: UserModel, ShowModel and TicketModel.

Chart

Description automatically generated

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