Application - National Theatre of Cluj-Napoca

Analysis and Design Document

Student: Costea Ana-Maria

**Group: 30233**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <dd/mmm/yy> | <x.x> | <details> | <name> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 4

2. Architectural Design 4

2.1 Conceptual Architecture 4

2.2 Package Design 4

2.3 Component and Deployment Diagrams 4

III. Elaboration – Iteration 1.2 4

1. Design Model 4

1.1 Dynamic Behavior 4

1.2 Class Design 4

2. Data Model 4

3. Unit Testing 4

IV. Elaboration – Iteration 2 4

1. Architectural Design Refinement 4

2. Design Model Refinement 4

V. Construction and Transition 5

1. System Testing 5

2. Future improvements 5

VI. Bibliography 5

# Project Specification

Application for National Theatre of Cluj-Napoca

The clients will access the application only if they have an user account, they will see the weekly program, as well as the monthly one, they will get informations about the favorite theater play. They will be able to set a reminder for a theater play or buy a ticket online, using the credit card. Also, they will be able to choose if they will print the ticket or not. Other features will be added ulterior.

There will be 2 types of users : regular user (client) and admin.

Regular user will have access to the operations :

* Login or create new account
* Access the weekly or the monthly program of the theatre
* Buy a ticket for a theater play or set a reminder for it

Admin user will have access to the operations :

* Add, update or delete a theater play
* Generate raport of all users who bought a ticket

# Elaboration – Iteration 1.1

# Domain Model

In software engineering, a domain model is a conceptual model of the domain which incorporates both behavior and data. A domain model generally uses the vocabulary of the domain, but allows a representation of the model to be communicated to non-technical users. A domain model is generally implemented as an object model within a layer that uses a lower-level layer for persistence and “publishes” an API to a higher-level layer in order to gain access to the data and behavior of the model. In UML, a class diagram is used to represent the domain model.

For my application, the domain model is represented by the environment created by users and admins, by theater plays and information about them. A class diagram will be added later, after the implementation.

# Architectural Design

## Conceptual Architecture

The ideal architectural pattern for my application will be **layered pattern**. This will be used in order to structure the programs which can be composed in groups of subtasks, each one of these being at a particular level of abstraction. Each layer provides services to the next bigger layer. Most common layers which are used :

* Presentation Layer (UI layer)
* Application Layer (Service layer)
* Business Logic Layer (Domain layer)
* Data Access Layer (Persistence layer)

## Package Design



## Component and Deployment Diagrams



# Elaboration – Iteration 1.2

# Design Model

## Dynamic Behavior

Diagrams for the operation of buying a ticket at a theatre play.

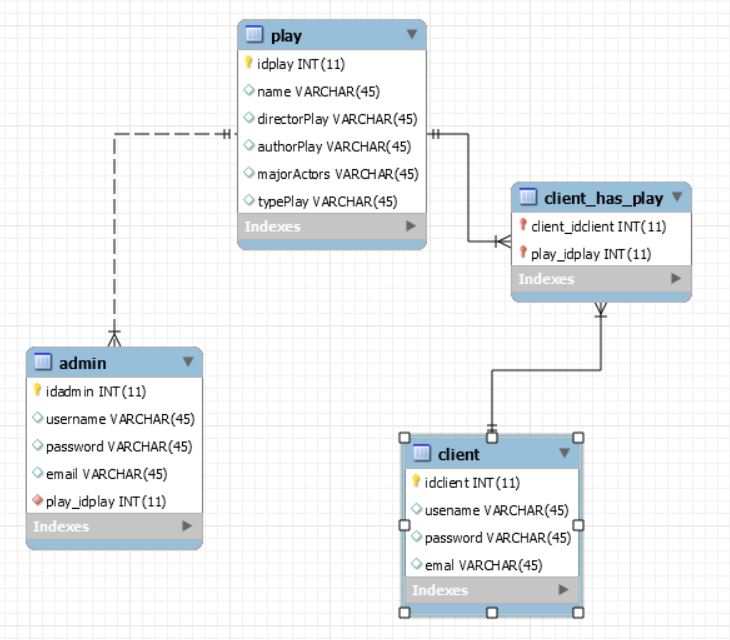




## Class Design

An UML class diagram will be added after the implementation of my application.

# Data Model



# Unit Testing

For testing, I will use Junit in order to take different test cases, such as : create a new client account, buy tickets at different plays, create/read/update/delete theatre plays as admin.

# Elaboration – Iteration 2

# Architectural Design Refinement

In my project, I will use **MVC** pattern (Model-View-Controller), for connecting different parts of the application in model, view, controller. The model represents Java objects, view represents the way data can be viewed, and the controller controls the data flow and modifies the view when the parts from the model change.

*[Refine the architectural design: conceptual architecture, package design (consider package design principles), component and deployment diagrams. Motivate the changes that have been made.]*

# Design Model Refinement

## *[Refine the UML class diagram by applying class design principles and GRASP; motivate your choices. Deliver the updated class diagrams.]*

# Construction and Transition

# System Testing

*[Describe how you applied integration testing and present the associated test case scenarios.]*

# Future improvements

As future improvements, I would design a web version of this application, using HTML and JavaScript for view. Also, I would add some new operations for the users, in order to improve the quality of the system.

# Bibliography

# [1] <https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013>

[2] <https://www.journaldev.com/1827/java-design-patterns-example-tutorial>