NBA Game Ticket App

Analysis and Design Document

Student: Zbucea Razvan

**Group:30431**

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

# This project aims to create an accessible NBA ticket game app, which can be used to buy tickets to basketball games. The main objectives are, for an user, to be able to: log into an account, select dates and games which the user would like to attend, make reservations to those games or cancel those reservations.

# Elaboration – Iteration 1.1

# Domain Model

# The domain model will consist of 4 main classes: User(there will be 1 type of user), Game, Payment, Seats.



# Architectural Design

## Conceptual Architecture

The **architecture** that I will be using for this web application is the **Layered Architecture**, consisting of the following layers:

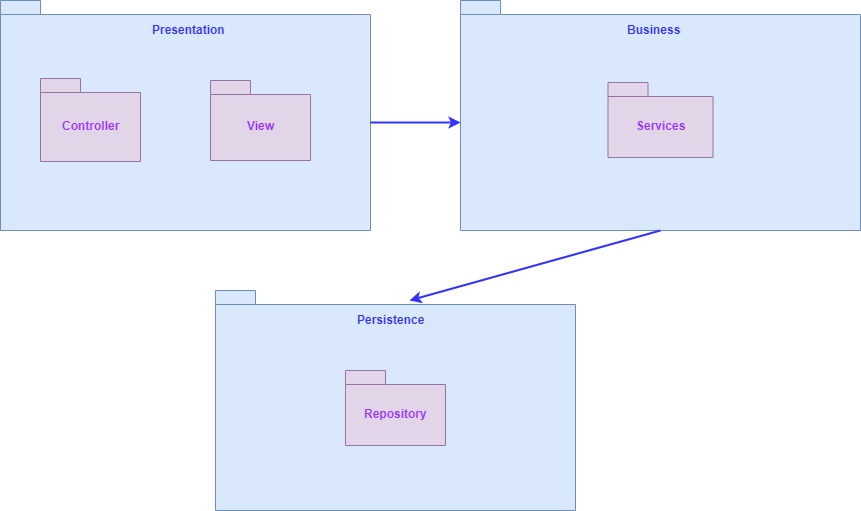
* **Presentation layer** – contains classes responsible for the User Interface
* **Business Layer** – contains rules that determine how data can be created, stored and changed
* **Persistence Layer –** deals with persisting(storing and retrieving) data from a database
* **Database Layer –** provides access to data stored in database

I have chosen this architecture because the web application that I am going to develop can be distributed into layers as described above. In this way, the dependency between classes is at minimum level, which means changes are easily done in a class without needing too much extra work in other classes.

The **design pattern** that I am going to use is the MVC design pattern: Model, View and Controller. The main advantages for using this pattern are:

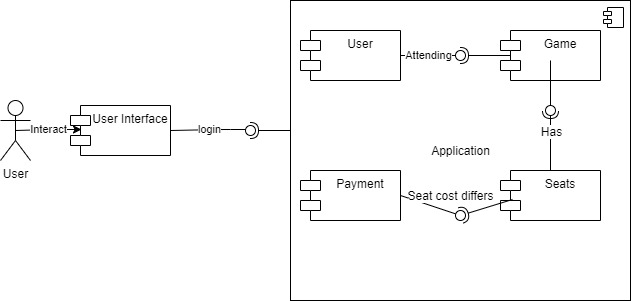
* Rapid application development
* Modification does not affect the entire model
* Ability to provide multiple levels

## Package Design

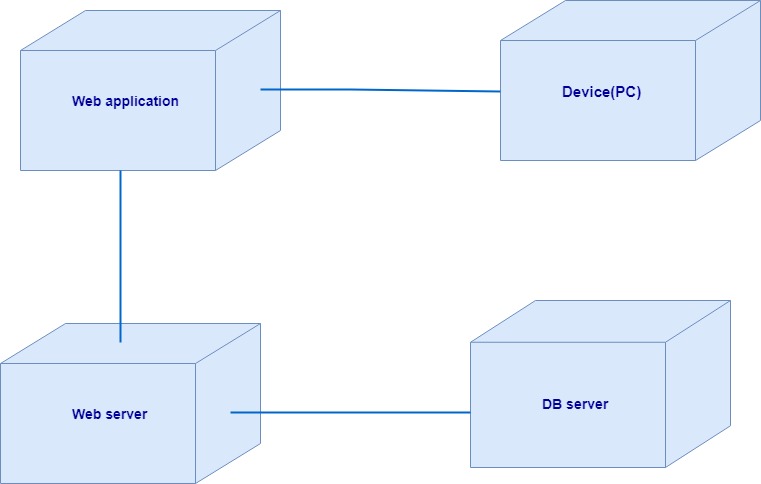
**

## Component and Deployment Diagrams

# Component diagram:



Deployment diagram:



# Elaboration – Iteration 1.2

# Design Model

## Dynamic Behavior

*[Create the interaction diagrams (1 sequence, 1 communication diagrams) for 2 relevant scenarios]*

## Class Design

*[Create the UML class diagram; apply GoF patterns and motivate your choice]*

# Data Model

*[Create the data model for the system.]*

# Unit Testing

*[Present the used testing methods and the associated test case scenarios.]*

# Elaboration – Iteration 2

# Architectural Design Refinement

*[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

# Future improvements

A livestream functionality could be added to the project. However, it would be a huge leap and it would probably change the scope of the project. A more realistic improvement would be by adding statistics and informations about certain teams or its players.

# Bibliography