Volleyball Manager

Analysis and Design Document

Student: Coca Sergiu

**Group: 30233**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 06/04/2017 | 1.0 | <details> | Coca Sergiu |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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

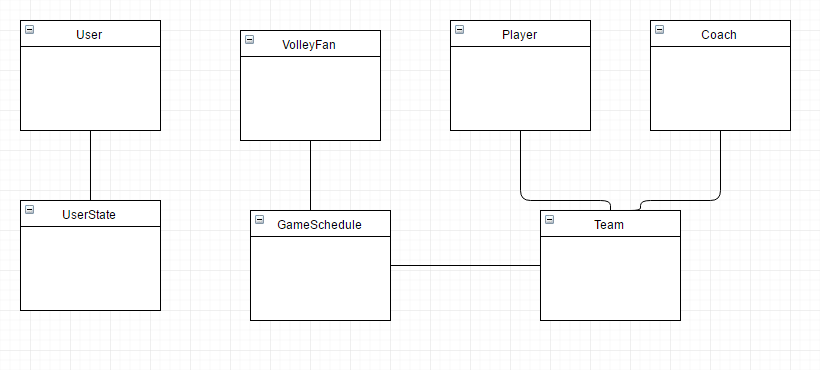
# Project Specification

This application it’s a perfect one for volleyball fans but also it’s useful for coaches. This application provide informations about a volleyball player like height, wheight, play team, etc. On the other hand if a coach want to transfer a player he can search him and if it’s available he can make an offer. When a player change his status to avaible, interested coaches are notified.

# Elaboration – Iteration 1.1

# Domain Model

The domain model should contain the following classes:



# Architectural Design

## Conceptual Architecture

Three-tier applications have been common since the growth of the database. A three-tier system satisfies the need for implementation isolation. Most frequently, this is desirable in any system where the storage/database layer of an application may need to be changed. However, this technological isolation is not restricted to just databases. It can, and should, be used whenever it is valuable to share code without requiring the application developer, or more importantly, the application maintainer, to have a detailed understanding of the implementation details of the lowest layer.

The application will be structured using the Layers architectural pattern and will also include MVC pattern

The Layers pattern will structure the application in four major modules:

* Business Layer – it will include all the functionality the application needs
* Presentation Layer – this layer will include everything that can be seen in a browser by the user
* DAO Layer – this layer will be responsible with the database access and CRUD operations requested by the business layer

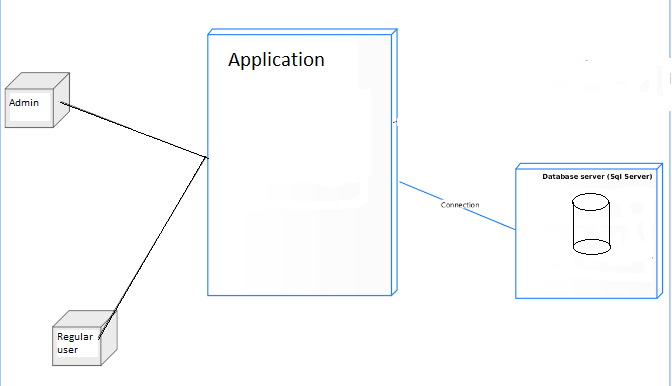
The MVC pattern will have a similar structure. The View and Controller components will be similar to the Presentation and Controller layers described above.

* **Model** - Model represents an object wich carrying data. It can also have logic to update controller if its data changes.
* **View** - View represents the visualization of the data that model contains.
* **Controller** - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.

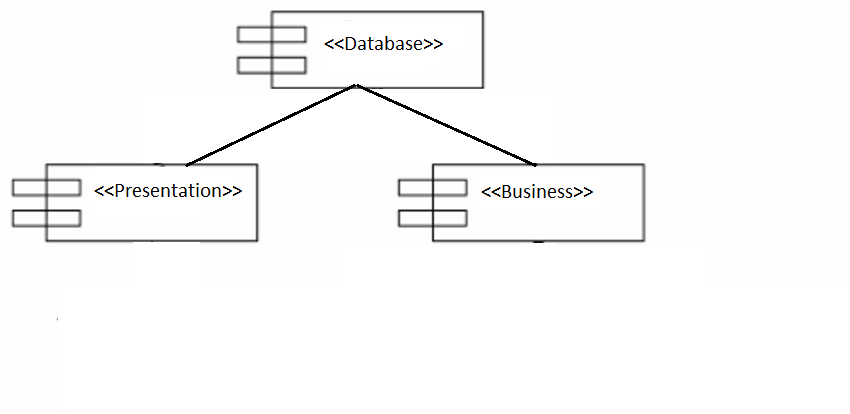
## Package Design

## Component and Deployment Diagrams

**Deployment**



**Component 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

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

# Future improvements

*[Present future improvements for the system]*

# Bibliography