Football Manager

Analysis and Design Document

Student: Milas Bogdan-Adrian

**Group:30233**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <06/04/2017> | <1.0> |  | Milas Bogdan-Adrian |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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 5

2.3 Component and Deployment Diagrams 5

III. Elaboration – Iteration 1.2 7

1. Design Model 7

1.1 Dynamic Behavior 7

1.2 Class Design 9

2. Data Model 9

IV. Construction and Transition 10

1. System Testing 10

2. Future improvements 10

V. Bibliography 10

# Project Specification

Scopul acestui “Football Manager” este de a produce soft care manageriaza si gestioneaza meciurile unui turneu de fotbal care contine mai multe echipe.Mai concret ,modul in care echipele se vor duela intre ele , locul de desfasurare al meciurilor. dar si altele. Datele despre turneu si echipele participante vor fi stocate intr-o baza de date. Detaliile cu privire la modul în care Football Managerul își îndeplinește actiunile sunt detaliate în cazurile de utilizare și specificatiile suplimentare.

Aplicatia va trebui sa aiba 2 tipuri de utilizatori (un administrator si un utilizator obisnuit) care vor trebui sa se autentifice printr-un nume si printr-o parola.Utilizatorul normal se va putea loga , va putea cumpara bilete , chema alti prieteni sau comenta meciurilela care participa.Administratotul , in schimb , se va putea si el loga ,va putea adauga un turneu ,va putea tipari bilete si ocazional poate active anumite oferte pentru primii client.

# Elaboration – Iteration 1.1

# Domain Model

Modelul nostru va fi structurat(aproximativ) cum arata urmatoarea diagrama de clase:



# Architectural Design

## Conceptual Architecture

Aplicatia noastra va folosi MVC

MVC pattern:

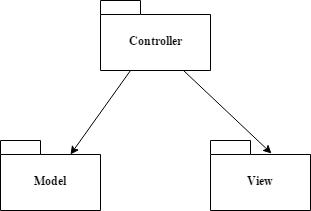
-Model va fi folosit pentru structura aplicatiei.De exemplu : felul in care se reprezinta clasele (cu metode , constructori) sau un table din baza de date.

-View si Controller vor fi utilizate pentru tot ce vede cel ce foloseste aplicatia.

Observer pattern:

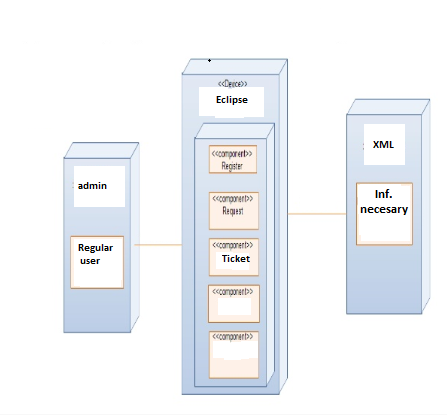
-pentru notificarea clientului atunci cand un bile teste disponibil

## Package Design

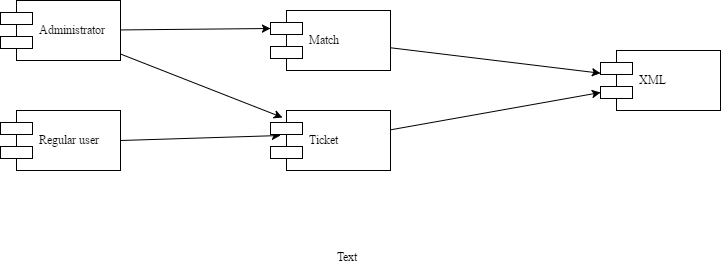


## Component and Deployment Diagrams

Deployment Diagram



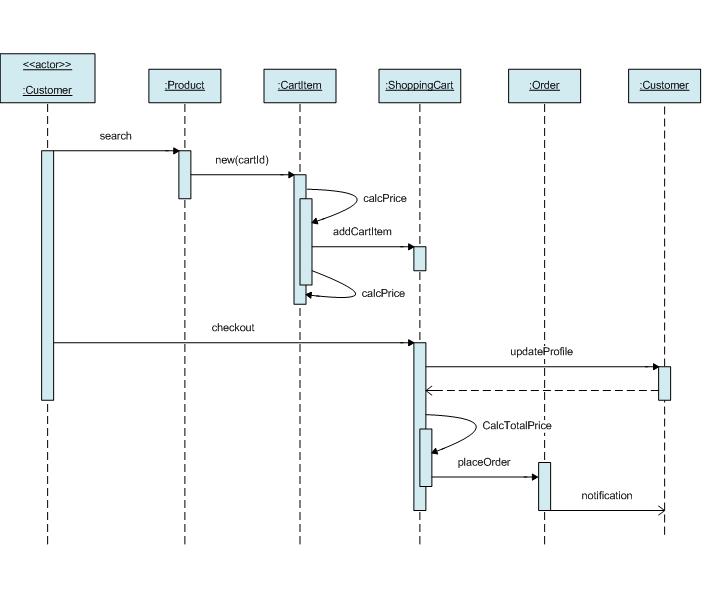
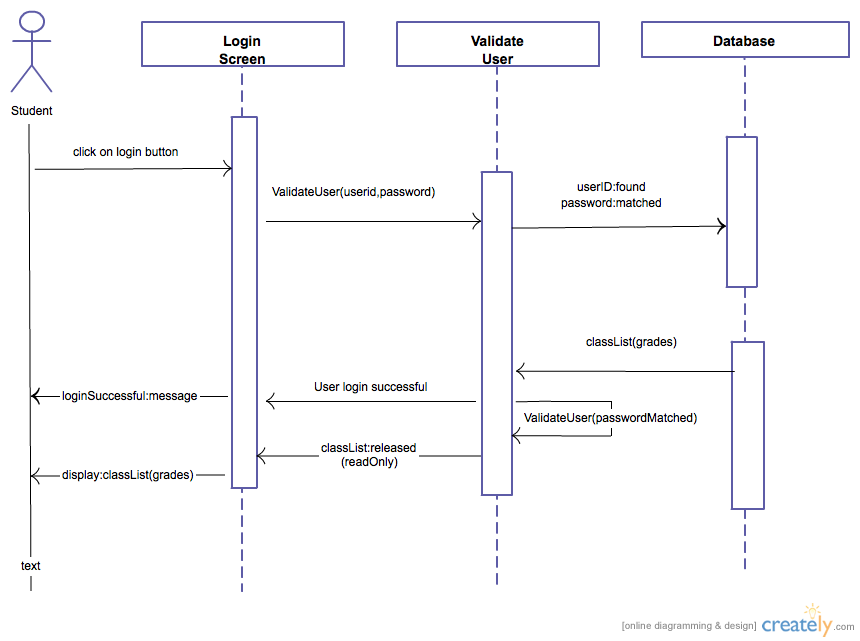
Component Diagram



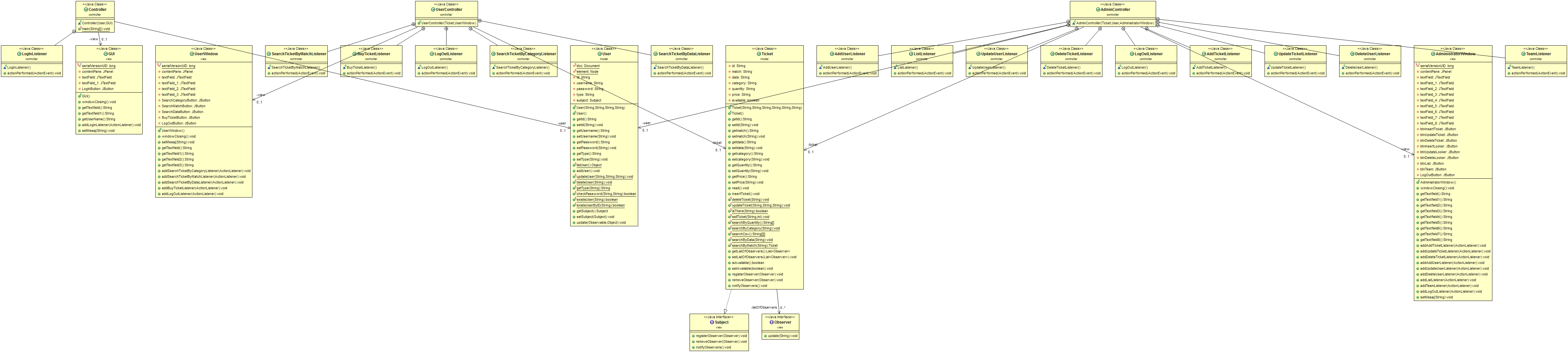
# Elaboration – Iteration 1.2

# Design Model

## Dynamic Behavior



## Class Design



# Data Model

Datele sunt pastrate in 3 fisiere xml :ticket.xml , user.xml si selling.xml. Fisierul ticket.xml contine pentru fiecare bilet un id , meciul care se joaca ,data de desfasurare , categoria(vip,peluza sau tribuna) ,cantitatea si pretul .Fisierul user.xml contine pentru fiecare utilizator un id ,username-ul ,parola si tipul(administrator/regular user).Ultimul fisier , selling.xml contine informatii despre biletele care trebuie cumparate.

# Construction and Transition

# System Testing

Pentru o buna functionalitate a aplicatii am implementat cateva teste care verifica ca datele de intrare sa fie valide si astfel sa nu putem introduce date gresite . Pentru inceput am creat teste la momentul logarii ,astfel ca doar introducand date care exista in fisierul user.xml se va putea face autentificarea.

# Future improvements

Acestui proiect ii pot fi aduse alte imbunatatiri.In primul rand am putea trece de la aplicatie desktop la aplicatie web.

Am putea implementa mult ai bine modul de vanzare si emitere al biletelor. S-ar putea folosi o baza de date de unde sa se preia informatiile necesare dar si altele.

# Bibliography

* <https://stackoverflow.com/>
* <https://docs.oracle.com/javase/tutorial/>
* <http://docs.oracle.com/javase/tutorial/uiswing/>
* <http://docs.oracle.com/javase/tutorial/jdbc/basics/index.html>
* <https://www.tutorialspoint.com/>