|  |  |  |
| --- | --- | --- |
| sigla MI cu text | **Universitatea TRANSILVANIA din Braşov**  **Facultatea de Matematicǎ şi Informatică**  **Programul de studiu Informatică aplicată** | sigla unitbv transparenta |

Lucrare de licenţă

Autor: **Trifanov Paula**

Coordonator ştiinţific: **Prof. univ. dr. Bocu Dorin**

Braşov

Iulie,2016

Aplicaţie suport pentru constituirea de echipe echilibrate şi customizabile

**FIŞA LUCRĂRII DE ABSOLVIRE/ LUCRĂRII DE LICENŢĂ/ PROIECTULUI DE DIPLOMĂ/ DISERTAŢIE**

|  |  |
| --- | --- |
| Universitatea *Transilvania* din Braşov | Lucrare de absolvire |
| Facultatea Matematică şi Informatică Brasov |
| Departamentul | Viza facultăţii |
| Programul de studii:Informatică aplicată | Anul universitar 2015-2016 |
| Candidat Trifanov Paula | Promoţia 2016 |
| Cadrul didactic îndrumător Prof. univ. dr. Bocu Dorin |  |

|  |
| --- |
| **LUCRARE DE ABSOLVIRE/ LUCRARE DE LICENŢĂ/ PROIECT DE DIPLOMĂ/**  **DISERTAŢIE** |
| Titlul lucrării:Aplicaţie suport pentru constituirea de echipe echilibrate şi customizabile |
| Problemele principale tratate:  Organizarea persoanelor din cadrul unui grup  Planificarea activităţilor în cadrul unui grup  Consemnarea schimbărilor apărute în cadrul organizatiei  Consultarea istoricului evenimentelor  Consemnarea evoluţiei membrilor grupului |
| Locul şi durata practicii:  Siemens, 20 iunie 2015-iulie 2016 |
| Bibliografie:   * <http://en.wikipedia.org/wiki/JavaServer_Faces> * <http://www.roseindia.net/jsf/> |
| Aspecte particulare:  (desene, aplicaţii practice, metode specifice etc.) |

|  |
| --- |
| Primit tema la data de: |

|  |
| --- |
| Data predării lucrării: |

|  |  |  |
| --- | --- | --- |
| Director departament, Cadru didactic îndrumător,  *Conf.univ.dr. MARIUS PĂUN, (*nume,prenume,semnătură)  *semnătura* | | |
| Candidat,  *(nume,prenume, semnătura* | | |
| **LUCRARE DE ABSOLVIRE/ LUCRARE DE LICENŢĂ/ PROIECT DE DIPLOMĂ/**  **DISERTAŢIE – VIZE** | | | |
| Data  vizei | | Capitole/ problemele analizate | Semnătura cadrului didactic îndrumător |
|  | |  |  |
|  | |  |  |
|  | |  |  |
|  | |  |  |
|  | |  |  |

|  |  |  |
| --- | --- | --- |
| **APRECIEREA ŞI AVIZUL CADRULUI DIDACTIC ÎNDRUMĂTOR** | | |
|  | | |
| Data: | ADMIS pentru susţinere/ RESPINS | CADRU DIDACTIC ÎNDRUMĂTOR  (nume,prenume,semnătură) |

|  |  |  |
| --- | --- | --- |
| **AVIZUL DIRECTORULUI DE DEPARTAMENT** | | |
| Data: | ADMIS pentru susţinere/ RESPINS | Director departament  Conf.univ. dr. Marius Păun,  semnătură |

|  |  |
| --- | --- |
| **SUSŢINEREA LUCRĂRII DE ABSOLVIRE/ LUCRĂRII DE LICENŢĂ/**  **PROIECTULUI DE DIPLOMĂ/ DISERTAŢIEI** | |
| Sesiunea | |
| Rezultatul  susţinerii | PROMOVAT cu media: |
| RESPINS **cu** refacerea lucrării |
| RESPINS **fără** refacerea lucrării |
| PREŞEDINTE COMISIE  (nume, prenume, semnătura) | |

Cuprins

[1 Introducere 1](#_Toc454825215)

[2 Descriere problemei 2](#_Toc454825216)

[3 Tehnologii folosite 3](#_Toc454825217)

[3.1 Java 3](#_Toc454825218)

[3.2 Maven 5](#_Toc454825219)

[3.2.1 Descriere 5](#_Toc454825220)

[3.2.2 POM 5](#_Toc454825221)

[3.3 Java Server Faces 6](#_Toc454825222)

[3.3.1 Prezentare generală 6](#_Toc454825223)

[3.3.2 Caracteristici 7](#_Toc454825224)

[3.3.3 Arhitectura JSF 8](#_Toc454825225)

[3.3.4 Utilitatea framework-ului 10](#_Toc454825226)

[3.3.5 Utilizarea în cadrul unei aplicaţii 11](#_Toc454825227)

[3.4 Primefaces 12](#_Toc454825228)

[3.4.1 Introducere 12](#_Toc454825229)

[3.4.2 Setup 13](#_Toc454825230)

[3.4.3 Dependinţe 13](#_Toc454825231)

[3.4.4 Componente Primefaces 14](#_Toc454825232)

[3.5 Bootstrap 17](#_Toc454825233)

[3.5.1 Prezentare generală 17](#_Toc454825234)

[3.5.2 Structură şi funcţionalitate 17](#_Toc454825235)

[3.5.3 Descărcare şi instalare 18](#_Toc454825236)

[3.6 PostgreSQL 18](#_Toc454825237)

[3.6.1 Caracteristici 18](#_Toc454825238)

[3.6.2 pgAdmin 19](#_Toc454825239)

[3.7 JPA(Java Persistence API) 20](#_Toc454825240)

[3.7.1 Componente JPA 20](#_Toc454825241)

[4 Descrierea soluţiei 26](#_Toc454825242)

[4.1 Stabilirea obiectivelor 26](#_Toc454825243)

[4.2 Cerinţele utilizatorilor fată de aplicaţie 26](#_Toc454825244)

[4.3 Diagrama claselor 28](#_Toc454825245)

[4.4 Consideraţii relativ la implementare 37](#_Toc454825246)

[4.4.1 Suprascrierea stilului CSS implicit din Bootstap 37](#_Toc454825247)

[4.4.2 Persistenţa datelor 39](#_Toc454825248)

[4.4.3 Algoritmul de generare de echipe ‘echilibrate’ 42](#_Toc454825249)

[4.4.4 Actualizarea modificărilor 43](#_Toc454825250)

[4.4.5 Actualizarea rating-urilor 44](#_Toc454825251)

[4.5 Ghid de utilizare 45](#_Toc454825252)

[4.6 Index de figuri 61](#_Toc454825253)

[5 Concluzii 64](#_Toc454825254)

[6 Bibliografie 65](#_Toc454825255)

# Introducere

Problema organizării unui grup de persone, indiferent de domeniul de activitate sau de scopul propus, poate fi destul de complexă. Poate fi desemnată o persoană căreia să îi revină această sarcină, însă îndeplinirea obiectivului poate fi destul de complicată, având în vedere faptul că persoana responsabilă ar trebui să aibă acces la o multitudine de date despre fiecare persoană agregată în grup.

Pentru a se putea organiza într-un mod eficient, un grup are nevoie de informații ce privesc membrii grupului în mod general, dar și informații particulare pentru fiecare membru în mod individual. De asemenea, membrii grupului ar trebui să își poată exprima opinia în legătură cu modul în care se face organizarea, acest lucru presupunând găsirea unei modalități eficiente de comunicare între leader-ul grupului şi restul membrilor.

O primă problemă care apare o reprezintă împărţirea grupului în mai multe subgrupuri, pentru a desfăsura diferite activităţi. Trebuie stabilite nişte reguli care să stea la baza formării subgrupurilor, reguli acceptate şi validate de către toţi membrii grupului. De asemenea, evidenţa acţiunilor grupului este greu de păstrat şi întretinut, daca nu există o modalitate flexibila şi dinamică, precum un tool sau un soft corespunzător.

Proiectul se dorește a fi unul care eficientizează modul în care se face organizarea în cadrul unui grup de persone, utilizatorii săi având posibilitatea de a cconsulta evoluţia grupului.

Lucrarea cuprinde 5 capitole, fiecare având un rol semnificativ în întelegerea logicii și a modurilor de folosire a tehnologiilor pentru realizarea proiectului.

În cel de-al doilea capitol este realizată o descriere a problemei. Acesta va cuprinde o prezentare a principalelor probleme întâmpinate în organizarea unor activități cu caracter sportiv la nivel de grup structurat pe diferite criterii, dar şi o descriere a potențialilor utilizatori.

Al treilea capitol este destinat prezentării tehnologiilor folosite pentru realizarea aplicaţiei, urmând ca în următorul capitol să se facă o prezentare a modului în care au fost folosite tehnologiile pentru realizarea produsului final, precum și o prezentare a principalelor moduri în care un utilizator poate interacționa cu aplicația.

# Descriere problemei

Problema concretă o reprezintă realizarea unor activități cu caracter sportiv la nivel de grup structurat pe diferite criterii. Se dorește realizarea a doua echipe care să fie “echilibrate”, echilibrul fiind obținut pe baza mai multor criterii. De asemenea, se dorește stabilirea programului pentru activitățile în desfășurare sau activitățile următoare.

Una dintre principalele dificultăți întâlnite în cadrul organizării unui grup de persoane o reprezintă preluarea şi folosirea datelor fiecărui individ din grup. Folosirea datelor personale ale fiecărui individ ar trebui făcută cu informarea şi confirmarea acestuia, numai de către persoane autorizate.

O altă problemă o reprezintă delegarea responsabilităților în cadrul grupului. Pentru acceptarea unor noi membrii în cadrul grupului, aceștia trebuie sa îndeplinească anumite condiții, impuse de leader-ul grupului şi confirmate de ceilalți membrii.În cazul unei aplicații web, acest lucru presupune existența mai multor nivele de acces: utilizator de tip administrator sau utilizator simplu. Un utilizator simplu nu are posibilitatea de a modifica structura curentă a grupului, nici de a introduce noi membrii în cadrul acestuia, sau a stabili un eveniment următor. Acest lucru trebuie să fie făcut de către un membru care deține aceste drepturi.

De asemenea, se dorește păstrarea unei evidențe a acțiunilor desfășurate în cadrul grupului, precum și existența unui feedback pentru membrii grupului care au participat la un eveniment. După fiecare acțiune, se dorește consemnarea detaliilor acesteia, pentru a se ține cont ulterior de consemnările făcute pentru o mai bună organizare a urmatoarelor evenimente.

Un alt punct de interes îl reprezintă consultarea schimbărilor apărute de-a lungul timpului, pentru deciderea următoarelor acțiuni. Scopul înregistrării detaliilor îl reprezintă formarea unei ierarhii în cadrul indivizilor. Cu cât un membru participă la mai multe acțiuni și primește feedback pozitiv, cu atât poziția sa în cadrul ierarhiei crește.

Potențialii utilizatori ar putea fi membrii unui grup, în cazul acesta al unei companii, interesați de activități sportive, precum meciurile de fotbal. Aceștia vor forma un grup ce va fi organizat în două subgrupuri, echipe, ce vor participa la activitățile sportive planificate. Grupul se dorește a fi unul dinamic, putând fi modificată structura acestuia în orice moment, prin introducerea de noi membrii. Subgrupurile sunt de asemenea dinamice, configurația acestora fiind influențată de mai mulți factori, precum poziția fiecărui individ în cadrul ierarhiei.

# Tehnologii folosite

## Java

Java este un limbaj de programare dezvoltat de Sun Microsystems, inițiat de James Gosling și lansat în 1995. Acesta rulează pe mai multe platforme, precum Windows, Mac OS și diferite versiuni de Linux.

Cea mai recentă versiune de Java Standard Edition este Java SE 8. Odată cu avansarea rapidă a crescut și popularitatea limbajului, fiind realizate mai mult configurații, potrivite pentru diferite platforme: ex:J2EE pentru aplicațiile Enterprise sau J2ME pentru aplicațiile de mobil.

Limbajul împrumută o mare parte din sintaxă de la C și C++, dar are un model al obiectelor mai simplu și prezintă mai puține facilități de nivel jos. Un program Java compilat, corect scris, poate fi rulat fără modificări pe orice platformă care e instalată o mașină virtuală Java (Java Virtual Machine, prescurtat JVM). Acest nivel de portabilitate (inexistent pentru limbaje mai vechi cum ar fi C) este posibil deoarece sursele Java sunt compilate într-un format standard numit cod de octeți (byte-code) care este intermediar între codul mașină (dependent de tipul calculatorului) și codul sursă.

Mașina virtuală Java este mediul în care se execută programele Java. În prezent, există mai mulți furnizori de JVM, printre care Oracle, IBM, Bea, FSF. În 2006, Sun a anunţat că face disponibilă varianta sa de JVM ca open-source.

Există 4 platforme Java furnizate de Oracle:

* Java Card - pentru smartcard-uri (carduri cu cip)
* Java Platform, Micro Edition (Java ME) — pentru hardware cu resurse limitate, gen PDA sau telefoane mobile
* Java Platform, Standard Edition (Java SE) — pentru sisteme gen workstation
* Java Platform, Enterprise Edition (Java EE) — pentru sisteme de calcul mari, eventual distribuite

Deoarece folosește obiecte, Java este un limbaj de programare obiect orientat. Programarea Orientată pe Obiecte este o paradigmă de programare care utilizează **obiecte** și **interacțiuni** între acestea pentru a modela arhitectura unui program. Obiectele din POO modelează obiecte din lumea reală. **Obiectele** din POO sunt **instanțe** ale unui tip de date numit **clasă**.

În lumea reală, există niște tipare, care grupează niște **atribute** ale obiectelor cu **acțiunile** lor, pentru a forma un tot ce definește obiectul respectiv. Pe acest concept, numit **încapsulare**, se sprijină programarea orientată obiect.

Se consideră drept principale caracteristici ale obiectelor încapsularea, moștenirea și polimorfismul.

Încapsularea este proprietatea obiectelor de a-și ascunde o parte din date si metode. Din exteriorul obiectului sunt accesibile numai datele si metodele publice. Putem deci să ne imagiăm obiectul ca fiind format din două straturi.

Obiectul se comporă ca şi când ar avea două "învelişuri": unul "transparent", care permite accesul la datele si metodele publice ale obiectului, şi un al doilea înveliş "opac", care cuprinde datele si metodele invizibile (inaccesibile) din exterior. Starea obiectului depinde atât de datele publice, cât şi de cele încapsulate. Metodele publice ale obiectului au acces la datele şi metodele încapsulate (ascunse) ale acestuia. În consecinţă, starea obiectului poate fi modificată atât prin modificarea directă, din exterior, a valorilor variabilelor publice, fie prin utilizarea unor metode publice care modifică valorile variabilelor încapsulate. În mod similar, valorile variabilelor încapsulate pot fi obţinute numai utilizând metode publice ale obiectului respectiv.

Încapsularea obiectelor prezintă avantaje importante în programare, deoarece măreşte siguranţa şi fiabilitatea programelor, prin eliminarea posibilităţii modificării accidentale a valorilor acestora ca urmare a accesului neautorizat din exterior. Din această cauză, programatorii evită în general să prevadă într-un obiect date publice, preferând ca accesul la date să se facă numai prin metode.

Partea vizibilă (publică) a obiectului constituie interfaţa acestuia cu "lumea exterioară". Este posibil ca două obiecte diferite să aibă interfeţe identice, adică să prezinte în exterior aceleaşi date şi metode. Datorită faptului că partea încapsulată diferă, astfel de obiecte pot avea comportament diferit. Moştenirea este proprietatea unei clase de a conţine toate atributele (variabilele) şi metodele superclasei sale. În consecinţă, trecerea de la clasă la subclasă se face prin adăugarea de atribute şi/sau de metode.

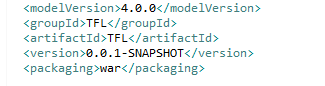
În general, în programarea orientată pe obiecte, moştentirea poate fi simplă sau multiplă. În cazul moştenirii simple fiecare clasă are cel mult o superclasă. În limbajul Java se admite numai moştenirea simplă. Lipsa moştenirii multiple este compensată în limbajul Java prin introducerea conceptului de interfaţă.

## Maven

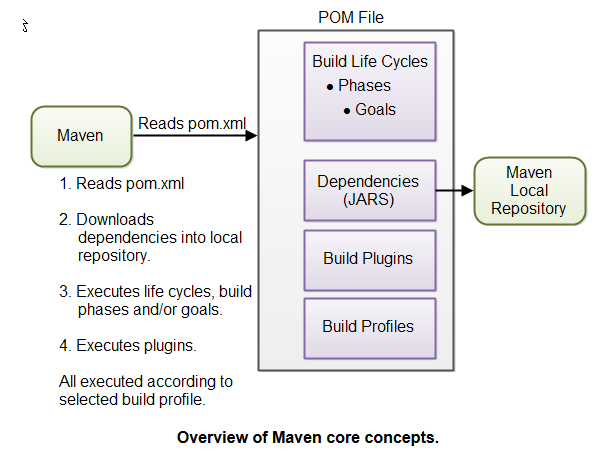
### Descriere

Maven este un sistem de build și management al proiectelor, scris în Java. Face parte din proiectele găzduite de Apache Software Foundation. Funcționalitățile sale principale sunt descrierea procesului de build a software-ului și descrierea dependențelor acestuia. Un fișier XML descrie proiectul care urmează să fie build-uit, dependențele acestuia sau ale modulelor și componente de care depinde, ordinea în care se execută build-ul, directoarele și plug-in-urile necesare. Maven descarcă dinamic bibliotecile Java si plug-in-uri necesare, din unul sau mai multe repository-uri.

### POM

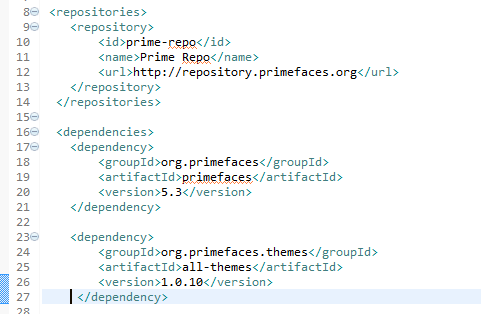
POM reprezintă abrevierea de la Project Object Model. Este un fisier xml şi reprezintă componenta principală în lucrul cu Maven. Pentru fiecare proiect Maven, există un singur fişier pom.xml. Înainte de a crea un fisier POM, trebuie să alegem un nume pentru proiect şi un groupId.

Figură 1. Crerea unui proiect Maven

Fişierul pom.xml conţine referinţe către toate resursele proiectului: Jar-URI externe,dependinţe,directoarele în care se găseşte codul de testare sau codul sursă. Acesta ar trebui să fie salvat în folderul principal al proiectului.

Figură 2. Conceptele Maven

În diagrama din Figura2 este evidențiat modul în care Maven utilizează fişierul pom.xml, precum şi principalele componente ale fişierului.

De exemplu, pentru a folosi framework-ul Primefaces, vom aduce dependinţele sale cu ajutorul Maven-ului. În fişierul pom.xml din cadrul proiectului, adaugăm dependințele, precum în Figura 3.

Figură 3. Dependinţe funcţionale incluse cu Maven

La acţiunea de build, se vor aduce dependinţele proiectului. De asemenea, se poate face build în diferite moduri: de exemplu pe un computer local, pentru dezvoltare şi testare sau un build pentru implementare într-un sistem de producţie. Pentru acest lucru se folosesc diferite profiluri de build, specificând la execuţia Maven-ului profilul folosit.

## Java Server Faces

### Prezentare generală

Java Server Faces este un framework folosit pentru dezvoltarea de aplicaţii web. A fost creat de Java Community Process (JPC) format din experţi în dezvoltarea de aplicaţii web din diferite grupuri, ca: Jakarta Struts, Oracle, Sun, IBM, ATGetc.

JSF face parte din şabloanele Web bazate pe MVC (Model-View-Controller) design pattern. Aplicaţiile create folosind framework-ul JSF sunt uşor de dezvoltat şi de întreţinut comparativ cu alte aplicaţii create folosind JSP şi Servlet.

Ce oferă JSF:

* permite crearea UI folosind componente standard, reutilizabile
* permite accesarea şi manipularea componentelor UI folosind taguri JSP
* salvează starea componentelor UI când clientul face o solicitare pentru o nouă pagină şi o restaurează când solicitarea este returnată (asigură persistenţa stării la nivel Web)
* oferă un model de interactiune bazat pe evenimente
* furnizează mecanisme pentru dezvoltarea de componente proprii
* separă prezentarea componentelor de funcţionalitate, astfel încât acestea să poataă fi utilizate în pagini HTML, WML, etc
* simplifică modul de creare a IDE-urilor pentru dezvoltarea de aplicaţii Web

Componente JSF:

* Un set de API pentru reprezentarea componentelor interfeţei utilizatorului (UI) şi administrarearea stării lor, tratarea evenimentelor şi validarea intrărilor, convetirea valorilor, definirea navigării în pagini, şi suport pentru internationalizare şi accesibilitate
* Librarii de taguri JSP folosite la crearea componentelor UI

### Caracteristici

JSF este un framework orientat pe partea de server, nu pe partea de client. Acest lucru înseamnă că în arhitectura JSF, majoritatea evenimentelor legate de UI management sunt tratate pe partea de server. Un exemplu de framework UI orientat pe partea de client este Swing.

O caracteristica importantă a framework-ului JSF este faptul că se separă tipurile de activităţi efectuate la crearea unei aplicaţii web:

* Design- editarea paginilor JSP, HTML
* Dezvoltare- implementarea logicii aplicaţiei
* CreareacomponentelorUI- componente ce vor fi utilizate de designeri pentru realizarea UI
* Arhitectura- asigurarea fluxului aplicaţiei, a scalabilităţii, configurare, etc.

Caracteristici generale pe care functionează framework-ul:

* Definirea regulilor de navigare între pagini – JSF are propriile componente care facilitează navigarea printe pagini şi oferă posibilitatea controlului navigării.
* Validator – este componenta care se ocupă cu validarea datelor de intrare
* Convertors - JSF converteşte valori de tip date şi numere în valori ce pot fi afişate cu uşurinţă (strings)
* Messages – folosite pentru afişarea de mesaje de success/failure utilizatorului
* Beans – se ocupă de logica aplicaţiei sunt obiecte de pe partea de server asociate unor UI components folosite în pagini web
* Event handling – JSF simplifică tratarea evenimentelor
* Internaţionalizare
* Custom GUI controls – JSF oferă un set de API şi taguri associate pentru a crea forme HTML cu interfeţe complexe

### Arhitectura JSF

JSF este un User Interface framework orientat pe partea de server. În cazul În care un client face o cerere, cererea ajunge prin intermediul reţelei la server, unde framework-ul JSF construieste o reprezentare UI şi o afişează clientului în mark-up language-ul corespunzator clientului. Userul interacţionează cu pagina web şi submite o cerere de procesare serverului. Framework-ul JSF interpretează parametrii cererii, îi decodează şi îi converteşte la evenimente apoi le propagă către logica de tratare a evenimentelor.

JSF este unul din framework-urile care se bazează pe structura Model-View-Controller(MVC). Ca orice alt framework bazat pe MVC, arhitectura JSF are propriul Front controller, numit FacesServlet. Rolul acestui controller este de gatekeeper.

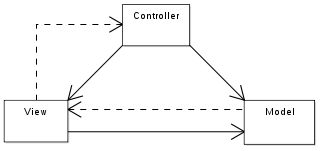
MVC este un model arhitectural folosit în ingineria software. El separă interfața aplicațiilor web în trei părți: model, view și controller, rezultând astfel o aplicație unde este mai ușor de modificat aspectul vizual sau nivelele inferioare ale regulilor de business fără a afecta alte nivele.Datele introduse de user, modelarea lumii externe și feedback-ul visual către ușer sunt tratate separat de fiecare componența a modelului. Controlerul interpretează intrările de la mouse și tastatură și mapeaza aceste acțiuni la comenzi care sunt trimise către model sau/și view pentru a efectua schimbările corespunzătoare. Modelul prelucrează unul sau mai multe elemente, răspunde la interogări asupra stării lui și răspunde la instrucțiuni de schimbare a stării. View-ul controlează o zonă din suprafața de afișat și este responsabil cu prezentarea datelor către user printr-o combinație de grafică și text.

**Model-ul** este responsabil cu reprezentarea datelor şi a acţiunilor ce operează asupra datelor de la nivelul Web al aplicaţiei.Este folosit pentru a controla informaţiile şi a notifica observatorii când aceastea se modifică. Pe langă controlarea informaţiilor şi notificarea observatorilor când starea acestora se modifică, modelul mai este folosit ca o abstractizare a proceselor sau sistemelor din lumea reală. El surprinde nu numai starea unui proces sau a unui sistem, ci şi cum acesta funcţionează.

**View-ul** este reponsabil cu prezentarea rezultatelor către utilizator. View-ul are o relaţie de correspondenţă de 1 la 1 cu suprafaţa de afişare şi ştie cum să afişeze conţinutul către utilizator. În plus, când modelul se modifică, view-ul automat redesenează partea afectată pentru a reflecta modificările apărute. Componenta View este creată uzual folosind: pagini JSP, taguri proprii, taguri definite de utilizator, JSTL, şabloane Velocity, Transformari XSLT.

**Controler-ul** reprezintă componenta prin care userul interacţionează cu aplicaţia. Controller-ul primeşte informaţii de la user şi instruieşte model-ul şi view-ul să execute acţiuni bazate pe input-ul primit. În principiu, controller-ul este responsabil cu maparea acţiunilor efectuate de user la răspunsul care trebuie oferit de aplicaţie. De exemplu, dacă userul selectează un meniu item, controlerul trebuie să determine cum ar trebui aplicaţia să răspundă acestei acţiuni.

Figură 4. Arhitectura MVC (Model-View-Controller)

****

Modelul, view-ul şi controller-ul sunt strâns legate între ele şi într-un continuu contact. Diagrama de mai sus ilustrează modul de comunicare dintre cele trei componente. Modelul comuniă cu view-ul, fără să ştie informaţii despre acesta, transmitându-i notificări asupra schimbărilor întâmplate, iar view-ul comunică la rândul lui cu modelul, dar acesta ştie tipul modelului observat, ceea ce îi oferă posibilitatea să apeleze metodele modelului. View-ul comunică de aseamenea şi cu controlerul, dar acesta nu-îi poate apela decât metodele din clasa de bază. Controlerul comunică cu amândouă: model-ul şi view-ul şi ştie tipul ambelor componente, deoarece acesta trebuie să ştie să răspundă corect la orice input primit de la user.

Arhitectura componentelor UI cuprinde cinci tipuri de modele:

* Modelul claselor UI
* Modelul de prezentare
* Modelul de conversie
* Modelul de tratare a evenimentelor
* Modelul de validare

Modelul claselor UI :

* Descrie funcţionalitatea componentelor UI prin extinderea clasei UIComponent (ex. UICommand, UIForm, UIPanel, UIOutput, UIInput, UIMessage, UIColumn, etc.)
* Implementează interfeţe comportamentale precum: ActionSource, ValueHolder, StateHolder, NamingContainer
* Nu definesc reprezentarea grafică a componentelor

Modelul de prezentare:

* Defineşte reprezentarea grafică a componentelor UI prin folosirea de obiecte de tip Renderer – converteşte componente din şi în un limbaj markup specific
* Claselor de tip UIComponent le pot fi asociate mai multe tipuri de reprezentări

Modelul de conversie:

* Componentele UI pot avea asociate date, memorate la nivelul serverului într-un bean
* Datele componentelor sunt organizate astfel:

- model: informaţia propriu-zisă, reprezentată printr-un tip de date Java (int, long, String);

- prezentare: reprezentarea la nivel de client astfel încât să permită citirea/editarea informaţiei

* JSF va face automat conversia între model şi prezentare, şi invers.

Modelul Event-Listener

* Interacţiunea utilizatorului cu componente UI generează evenimente
* La apariţia unui eveniment, sunt notificate metodele claselor listener înregistrate pentru componenta sursa a evenimentului

Modelul pentru validare:

* Tipuri de validatori:

-standard: referiţi prin intermediul tagurilor de validare

-JSF, include în corpul unui tag unei componente UI;

-proprii: validarea din cadrul unei componente bean

### Utilitatea framework-ului

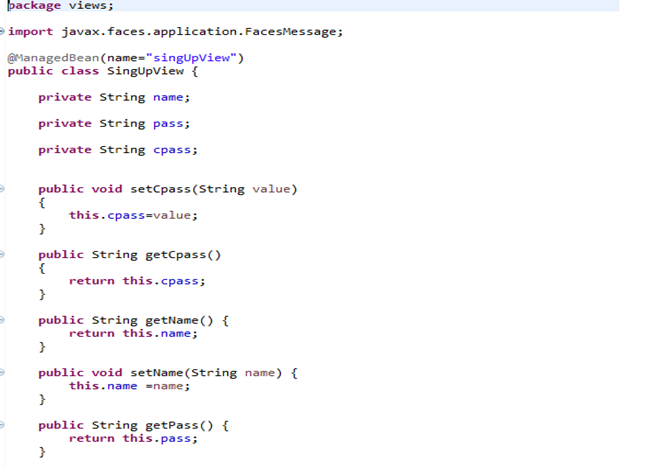
Când experţii JSF au început să lucreze la specificaţiile JSF, aceştia voiau să creeze un framework care să îndeplinească anumite condiţii. În primul rând să fie user-friendly, şi să ofere posibilitatea de a crea aplicații web prin drag and drop de componente UI. O altă condiție este aceea de a se face distincție clară între UI component model și afișarea componentelor folosind orice tip de client şi orice tip de protocol. În plus, JSF ar trebui să poată lucra cu JSP, dar și cu alte tehnologii.

### Utilizarea în cadrul unei aplicaţii

Pentru a folosi JSF într-o aplicaţie, în primul rând trebuie să îi aducem dependinţele. În cazul aplicaţiei curente, aducem dependinţele cu ajutorul Maven-ului.

Figură 5. Dependinţele JSF

După încărcarea dependinţelor se vor crea containerele (“Managed bean”) , clasele în care se va găsi informaţia cu care lucrează utilizatorii şi care poate fi accesată de către paginile JSF. Pentru a indica faptul că o clasă este un container se va folosi adnotarea ‘@ManagedBean’’.

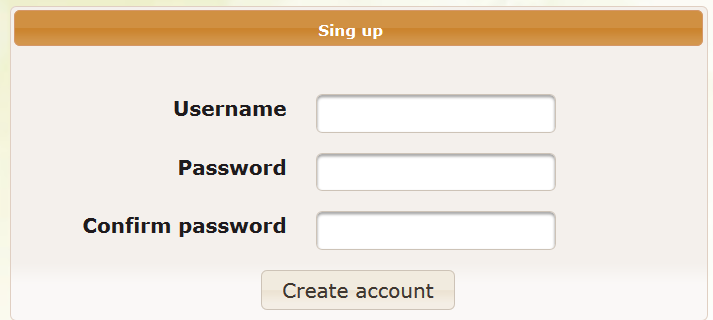


Figură 6. Crearea unui ManagedBean

În final, putem folosi JSF în paginile HTML, ca în exemplul din Figura7.

Figură 7. Exemplu de folosire a JSF în HTML

Pagina web se conectează la container prin Expression Language (EL), prin expresia “#{singUpView.name}” se obţine valoarea name din containerul ‘singUpView’. Dacă pentru clasă nu se specifică un nume alături de adnotarea @ManagedBean, containerul poate fi accesat prin numele clasei cu prima literă transformată din majusculă în literă mică.

În browser, se va obţine:

Figură 8. Exemplu formular JSF

## Primefaces

### Introducere

Primefaces este o bibliotecă ce furnizează componente UI care pot fi incluse în paginile JSF.Acesta a fost dezvoltat pentru Java Server Faces 2.0, și se bazează în mare măsură pe jQueryUI.

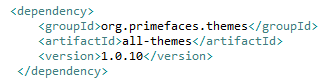
PrimeFaces adaugă suport suplimentar pentru tehnologii moderne, pe partea de sus a jQuery UI, adăugarea unor widget-uri suplimentare, permițând dezvoltatorilor să construiască interfeţe bogate, interactiv cu utilizatorul.

### Setup

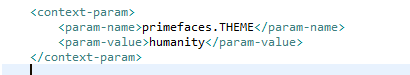
Primefaces are un singur jar numit ‘primefaces-{version}.jar’. Acesta poate fi downloadat de pe <http://www.primefaces.org/downloads> sau poate fi downloadat cu ajutorul Maven-ului, adăugând dependinţa sa în fişierul pom.xml.

### Dependinţe

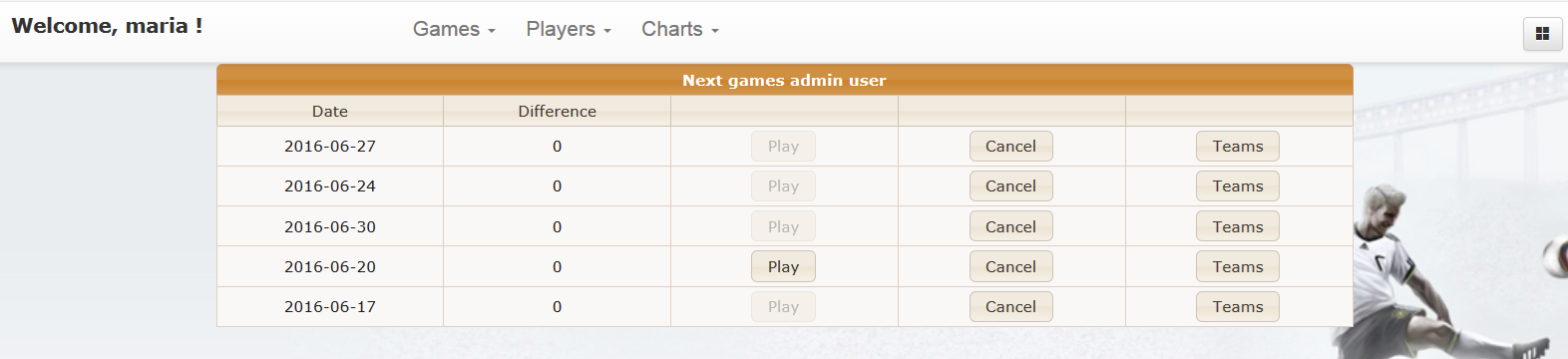
Pentru a putea fi folosit în dezvoltarea aplicaţiilor, framework-ul necesită Java 5+ şi JSF 2.x. Pentru mai multe funcţionalităţi, se pot include şi dependinţe opţionale. Nu avem nevoie de configuraţii suplimentare, însă putem face acest lucru în funcţie de cerinţe.

Primefaces cuprinde mai mult de 30 de teme predefinite. Pentru a putea folosi o temă pentru paginile web, putem adăuga dependinţa în fişierul pom.xml.

Figură 9. Adăugarea dependinţei pentru temele Primefaces

Pentru a seta o temă pentru paginile web, o adăugăm în fişierul web.xml.

Tema setată pentru pagina web va arăta precum în Figura10.:



Figură 10. Tema 'Humanity' Primefaces

### Componente Primefaces

Componentele Primefaces pot fi folosite alături de mai multe atribute. Câteva dintre atributele ce pot fi folosite sunt:

* Id( string), reprezentând un identificator unic pentru componentă
* Rendered(Boolean), în cazul în care este false componenta nu este vizibiă în pagină
* Binding(Object) reprezintă componenta din container asociată butonului
* ActionListener(MethodObj) reprezintă acţiunea ce va fi executată atunci când se dă click pe buton
* Style(String) stilul ce poate fi declarat inline
* StyleClass(String) reprezintă clasa stilului pentru element
* Multiple(Boolean) pentru componente de selecţie în cazul în care se doreste selectarea multiplă

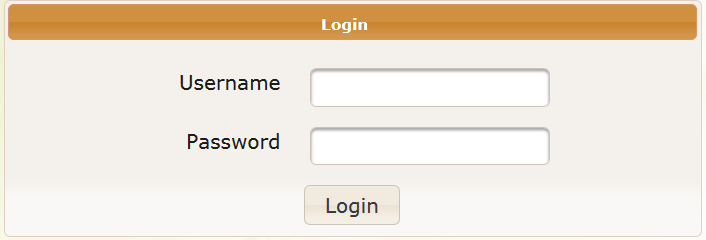
#### Panel

Pentru autentificare, se completează câmpurile din formularul ce va conţinte un panel, componenta Primefaces folosită pentru a grupa elementele, care are sintaxa:

<p:panel>

//child components here…

</p:panel>

Rezultatul obtinut va fi cel din Figura11.

Figură 11. Primefaces Panel

#### CommandButton

Dupa autentificare, din meniu, utilizatorul poate alege acţiunile pe care doreste să le execute, prin click pe un item al meniului, reprezentat de un commandButton.Tag-ul Primefaces este <p:commandButton></p:commandButton> şi poate avea diferite atribute şi stiluri css.

#### AutoComplete

Componenta folosită pentru a afişa o listă de sugestii în timpul scrierii unui input de către utilizator. Sintaxa este:

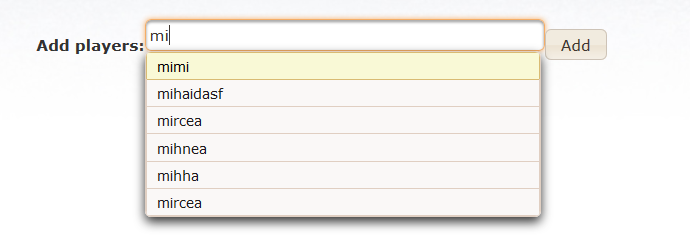
<p:autoComplete id="themes" multiple="true" value="#{autoCompleteView.selectedPlayers}" completeMethod="#{autoCompleteView.completeTheme}" var="theme" itemLabel="#{theme.username}" itemValue="#{theme}" converter="playerConverter" forceSelection="true">

<p:column style="width:80%">

<h:outputText value="#{theme.username}"/>

</p:column>

</p:autoComplete>

Rezultatul afişat pe interfaţa va fi:

Figură 12. Primefaces Autocomplete

#### Chart

Chart-ul este o componentă grafică care foloseşte librăria jqplot pentru afişarea datelor într-un format interactiv. Există mai multe tipuri de chart-uri: Pie Chart, Line Chart, Bar Chart, Area.

Codul html pentru un LineChart ar putea fi:

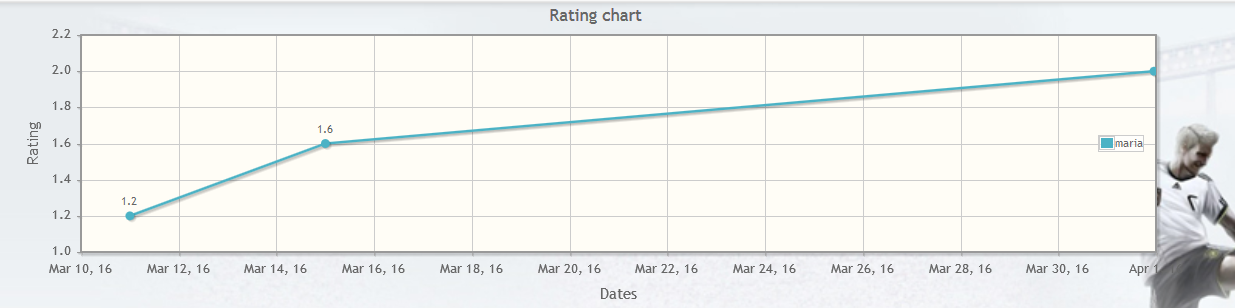
<div class=*"col-sm-12 col-md-12"*>

<p:chart type=*"line"* model=*"#{chartView.lineModel}"* style="height:*300px*;"/>

</div>

Unde ‘lineModel’ este creat în controller, clasa adnotată cu @ManagedBean, ChartView, şi care conţine atributul:

private LineChartModel lineModel;

Rezultatul obţinut este:

Figură 13. Primefaces Chart

#### FileUpload

FileUpload adaugă un plus de inovaţie şi flexibilitate faţa de componenta <input> de tip “file” <input type=”file” > din HTML5. Pentru partea de server pe care se face upload se poate folosi servlet 3.0 sau alt tip de server pe care se poate face upload. Acesta detectează automat cel mai potrivit tip de server pe care să se faca upload, dar se pot adăuga şi alte configurări:

<context-param>

<param-name>primefaces.UPLOADER</param-name>

<param-value>auto|native|commons</param-value>

</context-param>

#### PickList

PickList este folosită pentru a putea transfera date între două colecţii diferite.

Sintaxa componentei este:

<p:pickList id=*"pick"* value=*"#{teamsView.players}"* var=*"theme"* effect=*"bounce"* itemValue=*"#{theme}"* itemLabel=*"#{theme.username}"* showCheckbox=*"true"* responsive=*"true"*filterMatchMode=*"contains"* converter=*"playerConverter"*>

</p:picklist>

Figură 14. Primefaces PickList

## Bootstrap

### Prezentare generală

Bootstrap este un framework de front-end, folosit pentru dezvoltarea aplicaţiilor web şi a website-urilor. Conţine template-uri de HTML şi CSS pentru componente din interfaţa, precum input-uri, butoane, formulare, componente pentru navigarea în pagină, dar şi extensii de JavaScript. Spre deosebire de alte framework-uri, Bootstap este orientat în special către partea de interfaţa şi dezvoltarea front-end.

Numit iniţial **Twitter Blueprint,** a fost dezvoltat de către Mark Otto şi Jacob Thornton ca şi framework ce avea ca scop consecvenţa între diferite tool-uri interne,înainte de Bootstrap fiind folosite mai multe librării pentru a realiza partea de interfaţa. Folosirea unei multitudini de tipuri de librării avea ca rezultat aplicaţii greu de menţinut şi cu o flexibilitate redusa.

Bootstrap este compatibil cu cele mai noi versiuni de Google Chrome, Firefox, Opera, Internet Explorer şi Safari, deşi nu toate browsere-le sunt suportate de diferite platforme.

Începând cu versiunea 2.0, Bootstap are de asemenea suport pentru un design web responsive. Acest lucru înseamnă că paginile îşi ajustează conţinutul în funcţie de tipul de device de pe care sunt accesate(desktop, tablete, telefoane mobile).

Framework-ul este disponibil open-source pe GitHub, dezvoltatorii fiind încurajati să contribuie la dezvoltarea acestuia.

### Structură şi funcţionalitate

Bootstrap este împărtit pe module, şi constă în principal dintr-o serie de stylesheet-uri care implementează o multitudine de componente din setul de instrumente. Dezvoltatorii pot adapta Bootstrap-ul prin selectarea componentelor pe care doresc să le utilizeze. În plus faţă de elementele de bază de HTML, framework-ul aduce elemente speciale, precum butoane cu aspect şi funcţionalitate specifică(ex: butoane grupate, butoane cu opţiunea de drop-down), liste de navigare sau tab-uri orizontale sau verticale. Componentele sunt implementate ca şi clase CSS, care trebuie atribute elementelor din HTML.

Framework-ul mai conţine şi o serie de scripturi JavaScript,incluse ca şi plug-in-uri de JQuery. Acestea aduc în plus alte elmente de interacţiune cu utilizatorul, precum ferestre de dialog, tooltip-uri sau un mod interactiv de vizualizare a imaginilor, numit carusel.

### Descărcare şi instalare

Pentru a putea folosi Bootstap, îl putem descărca de pe repository-ul de GitHub, accesând link-ul: <http://getbootstrap.com/getting-started/> , sau îl putem include dependinţele in tag-ul <head> din pagina html. Dependinţele ce trebuie incluse sunt următoarele:

<!-- Latest compiled and minified CSS -->

<link rel="stylesheet"

href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css" integrity="sha384-1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjWPGmkzs7" crossorigin="anonymous">

<!-- Optional theme -->

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap-theme.min.css" integrity="sha384-fLW2N01lMqjakBkx3l/M9EahuwpSfeNvV63J5ezn3uZzapT0u7EYsXMjQV+0En5r" crossorigin="anonymous">

<!-- Latest compiled and minified JavaScript -->

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js" integrity="sha384-0mSbJDEHialfmuBBQP6A4Qrprq5OVfW37PRR3j5ELqxss1yVqOtnepnHVP9aJ7xS" crossorigin="anonymous"></script>

## PostgreSQL

PostgreSQL este un sistem de baze de date relaţionale. Este disponibil gratuit sub o licentă open-source de tip BSD. PostgreSQL nu este controlat de nici o companie, își bazează dezvoltarea pe o comunitate răspândită la nivel global, precum și câteva companii dezvoltatoare.

### Caracteristici

Sistemul are cateva limitări generale:

* Dimensiunea maximă a bazei de date: nelimitat
* Dimensiunea maximă a unei tabele: 32 TB
* Dimensiunea maximă a unei înregistrări: 1,6 TB
* Dimensiunea maximă a unui câmp: 1 GB
* Număr maxim de înregistri într-o tabelă: nelimitat
* Număr maxim de coloane într-o tabelă: 250 - 1600 în funcție de tipul coloanelor

PostgreSQL permite folosirea limbajelor procedurale pentru a executa blocuri de cod direct în serverul de baze de date. Se pot folosi pentru a crea funcții definite de utilizator (subrutine, trigerre, agregate și funcții fereastră) sau pentru a crea blocuri ad hoc "DO". Instalarea standard a PostgreSQL permite utilizarea urmatoarelor limbaje:

* PL/pgSQL un limbar asemănator cu PL/SQL existent in Oracle.
* PL/Tcl pune la dispoziție Tcl
* PL/Perl pune la dispoziție Perl
* PL/Python pune la dispoziție Python, versiunea 2 sau 3.

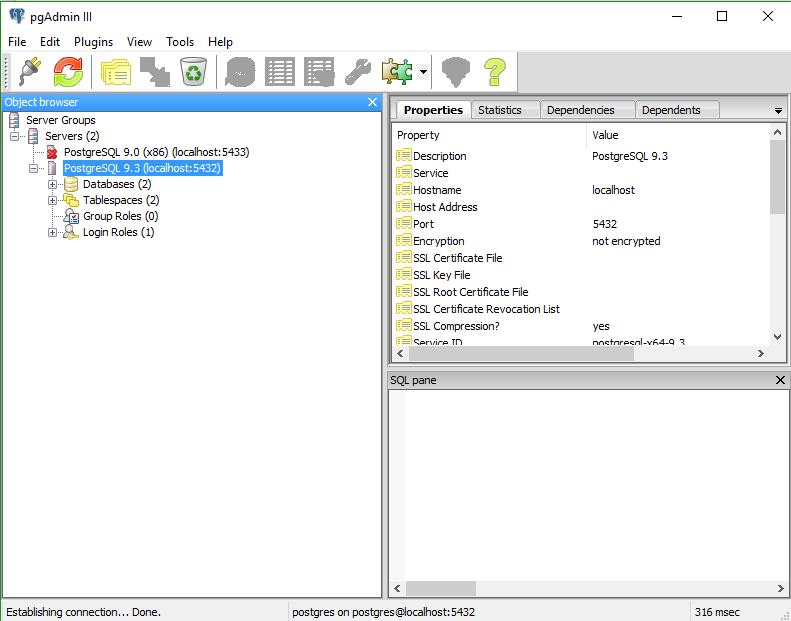
Alte limbaje disponibile în afara pachetului de bază includ: PL/Java, PL/php, PL/Ruby, etc.

La fel ca și alte baze de date, PostgreSQL permite utilizarea indecșilor pentru accelerarea interogarilor. Suportă mai multe tipuri de indecși: B-tree, Hash, GiST sau GIN. Daca tipul nu este specificat la crearea indexului, se utilizează B-tree.

Alte caracteristici suportate:

* Valorile indecșilor pot fi calculate printr-o expresie sau o funcție.
* Indecși parțiali permit să se indexează doar o parte dintr-o tabelă. Pot fi creați prin specificarea unei clauze WHERE la sfârșitul unei comenzi CREATE INDEX.
* Planificatorul este capabil să folosească mai mulți indecși pentru a executa interogări complexe.

### pgAdmin

PgAdmin este una dintre cele mai populare platforme open-source de administrare şi dezvoltare a bazelor de date PostgreSQL. PgAdmin pune la dispoziţie un design capabil să rezolve o multitudine de probleme ale utilizatorilor, de la query-uri simple, până la baze de date foarte complexe.

Figură 15. Interfaţa pgAdmin

Interfaţa grafică suportă toate caracteristicile principale ale limbajului şi face ca administrarea bazei de date să devină un proces simplu şi intuitiv. Conexiunea la baza de date se poate face cu ajutorul protocoalelor precum TCP/IP sau IPC( inter-process communication socket) şi poate fi criptată folosind SSL pentru asigurarea securitaţii datelor.

## JPA(Java Persistence API)

JPA este un framework lightweight ce foloseşte POJO (Plain Old Java Objects) pentru a persista obiecte Java ce reprezintă date relaţionale. JPA 1.0 a început ca parte a specificatiilor EJB 3.0 pentru a standardiza un model pentru ORM (object-relational mapping). JPA 2.0 (JSR-317) imbunatateste specificaţiile originale.

POJO face parte, de asemenea, din specificaţiile EJB 3.0. Orice obiect normal este un obiect POJO

Beneficiile utilizării JPA:

* Nu trebuie să cream obiecte complexe de acces la date (DAO)
* API-ul este folosit pentru a gestiona tranzacţii
* Codul de interacţiune cu baza de date este standard, indiferent de vendorul bazei de date relaţionale
* Putem evita SQL şi în schimb putem folosi un query language, orientat pe obiect
* Putem folosi JPA pentru persistenţa aplicaţiilor desktop

JDBC a fost primul mecanism pe care dezvoltatorii Java l-au folosit pentru a persista date. JPA este un framework de mapare ce păstrează abilitatea de a manipula baza de date direct.

Conceptele fundamentale ale JPA sunt:

* Entitate, este utilizată pentru a reprezenta un tabel relaţional într-un obiect Java
* Unitate de persistenţă, defineşte mulţimea tuturor claselor ce au legatură cu aplicaţia şi care sunt mapate unei singure baze de date
* Context persistent, este o mulţime de instanţă de entităţi în care este o unică instanţă a entităţii pentru orice identificator de entitate persistentă
* Entity manager, face munca de creare, citire şi scriere a entităţilor

### Componente JPA

Persistenţa este mecanismul utilizat de aplicaţii pentru a păstra datele, date ce altfel ar fi pierdute la închiderea aplicaţiei sau a calculatorului, într-un context persistent, precum o bază de date.

JPA stabileşte relaţionări statice ale modelului persistent, prin definirea componentelor entitate. API-ul defineşte clasa entitate ca un echivalent al unei tabele din baza de date, pe partea de business a aplicaţiei. O instanţă a entităţii este definită ca un obiect, echivalent al unei linii din tabela bazei de date.

JPA stabileşte relaţionarea dinamică a modelului persistent prin definirea unui obiect entity manager. Acesta este însărcinat cu sincronizarea datelor continute într-o instanţă entitate cu datele conţinute în linia echivalentă din baza de date. Spre exemplu, daca pe timpul execuţiei aplicaţiei un camp al instanţei entitate este modificat, entity managerul modifica linia echivalenta din baza de date.

JPA furnizeaza programatorilor Java facilitatea de mapare obiect/relaţie, pentru a gestiona modelul relaţional al bazelor de date implicate în aplicaţiile Java. Persisţenta în Java constă din:

* JPA
* Limbajul de interogare (JPQL)
* Java Persistence Criteria API
* Metadatele de mapări obiect/relaţie

Figură 16.Arhitectura JPA

In Figura16 avem:

* Entity, obiectul persistent ce reprezintaă o înregistrare din tabela bazei de date. Este un POJO cu anotaţii
* Interfaţa EntityManager, furnizează API-ul de interacţiune cu entitatea
* EntityManagerFactory este folosit pentru a crea o instanţă de EntityManager.

Un obiect Java poate cuprinde date parţiale dintr-o tabela sau date din mai multe tabele.Framework-urile ORM gestionează maparea între tabelele bazei de date relaţionale şi obiectele Java astfel încât programatorii să codeze cât mai puţin. EclipseLink şi Hibernate sunt exemple de soft-uri ORM.

#### Entităţi

Aşadar, o entitate reprezintă o structură de date prin intermediul căreia se asigură persitenţa. O entitate este asociată unui tabel al unei baze de date relaţionale, iar fiecare instanţă a entităţii corespunde unei linii din tabela. Câmpul unei clase entitate corespunde unei coloane din tabela bazei de date.

O entitate:

* Este un POJO creată utilizând cuvântul new
* Suportă moştenire şi polimorfism
* Este serializată şi poate fi utilizată ca un obiect detached

#### Relaţionarea entităţilor

Primul pas în modelarea asocierii la nivelul datelor este definirea relaţionării ca o mulţime de proprietăţi. Aceste proprietăţi pot fi apoi utilizate pentru a implementa relaţionarea obiectelor prin clasele entitate.

Următoarele patru proprietăţi sunt folosite pentru a descrie asocierea între obiecte:

* Cardinalitatea unei relaţii, care specifică numărul de relaţionări între două entităţi relaţionate, poate fi de următoarele tipuri: unu-unu, unu-mulţi, mulţi-unu, mulţi-mulţi.
* Direcţia unei relaţii, care determină navigabilitatea şi vizibilitatea, poate fi:

Bidirecţională, fiecare entitate poate vedea cealaltă entitate din relaţie. Putem astfel include cod în oricare dintre entităţi pentru a naviga către cealaltă entitate în vederea obţinerii de informaţii şi servicii de la această. În JPA natura bidirecţională a unei relaţii este specificată prin utilizarea unei anotaţii de cardinalitate în ambele clase entitate situate de cele doua părţi ale relaţiei.

Unidirecţională, doar o singură entitate dintre cele doua implicate în relaţie poate vedea cealaltă entitate. Natura unidirecţională a relaţiei este specificată prin folosirea uneia dintre anotaţiile de cardinalitate într-una dintre entităţi.

Proprietarul relaţiei, specifică partea de owning a relaţionării, care la rândul său conţine maparea fizică. Cealaltă parte a relaţiei se numeşte inverse side.Tipul de propagare a operaţiei (cascading type), se referă la propagarea efectului unei operatii entitatilor asociate. Se exprimă prin termenii: All, Persist, Merge, Remove, Refresh, None.

Într-o relaţie bidirectională fiecare entitate are un câmp sau o proprietate ce referă la cealaltă entitate din relaţie. Prin acest câmp codul entităţii poate accesa obiectul relaţionat.

Relaţiile bidirecţionale trebuie să urmărească regulile:

* inverse side trebuie să refere către owning side utilizând elementul mappedBy al uneia dintre anotaţiile: @OneToOne, @OneToMany sau @ManyToMany. Valoarea acestui element este câmpul sau proprietatea din entitatea ce joacă rol de owner al relaţiei
* partea “multi” a unei relaţii multi-unu, într-o relaţionare bidirectională, nu trebuie să definească elementul mappedBy. Această parte este considerată întotdeauna owner
* într-o relaţie unu-unu bidirecţioanaă owning side corespunde părţii ce conţine cheia străină corespunzătoare
* într-o relaţie multi-multi ambele părti pot fi owning side

Într-o relaţie unidirecţională doar o singură entitate are un câmp sau o proprietate ce referă la cealaltă entitate.

Direcţionalitatea este necesară pentru a putea naviga între relaţii.Entităţile ce utilizează relaţiile au deseori dependinţe relativ la existenta celeilalte entităţi din relaţie.

#### Gestiunea entităţilor

Entităţile sunt gestionate printr-un entity manager reprezentat de o instanţă a interfeţei javax.persistence.EntityManager. Fiecare entity manager este asociat unui context

persistent, adică instanţe ale unora dintre entităti ce există mapate dintr-o bază de date. Contextul persistent defineşte domeniul sub care instanţele entităţilor sunt create, salvate sau şterse. EntityManager defineşte metodele utilizate pentru a interacţiona cu contextul persistent.

În cazul unui entity manager gestionat de container, o instanţă a EntityManager este automat propagată de către container către toate componentele aplicaţiei ce utilizează instanţa în cadrul unei tranzacţii JTA.

Tranzacţiile JTA implică deseori apeluri peste mai multe componente. Pentru a executa o tranzacţie JTA aceste componente au nevoie de acces către un singur context persistent. Aceasta apare atunci când EntityManager este injectat în componentele aplicaţiei, prin anotaţia javax.persistence.PersistenceContext. Acest context persistent este propagat automat împreună cu tanzacţia curentă JTA, iar referinţele lui EntityManager sunt mapate aceluiaşi context persistent ca şi tranzacţia. Prin propagarea automată a contextului persistent componenetele aplicaţiei nu au nevoie să-şi trimită referinţele către instanţele EntityManager în cadrul unei tranzacţii. Containerul Java EE gestionează ciclul de viată entity managerilor gestionaţi de container.

Obţinerea unei instanţe se face prin construcţia sintactică:

@PersistenceContext

EntityManager em;

#### Folosirea JPA într-o aplicaţie Java SE

Paşii ce trebuie urmaţi pentru a include şi utiliza JPA într-o aplicaţie sunt următorii:

1. Creăm baza de date pe serverul de baze de date
2. Adăugăm bibliotecile EclipseLink şi conectorul la BD
3. Definim entităţile în aplicaţie
4. Creăm persistence.xml şi-l configurăm: definim o unitate de persistenţă şi un tip de tranzacţie, furnizăm nume claselor entitate, definim proprietăţile conexiunii JDBC
5. Creăm instanţe ale EntityManagerFactory şi EntityManager
6. Scriem cod pentru a efectua persistenţa entităţilor folosind instanţa entity manager

#### Tranzacţii

O tranzacţie este un mecanism ce manipulează un grup de operaţii ca şi cum ar fi o singură operaţie. Într-o tranzacţie toate operaţiile se executp sau niciuna. Operaţiile implicate într-o tranzacţie se pot baza pe mai multe baze de date.

O tranzacţie este formal definită ca o mulţime de proprietăţi incluse în acronimul ACID:

* Atomicitate: o tranzacţie este efectuată integral sau deloc. În caz de eşec operaţiile şi procedurile se consideră neefectuate şi toate datele revin la starea anterioară tranzacţiei
* Consistenţă: o tranzacţie transformă un sistem dintr-o stare consistentă într-o altă stare consistentă
* Izolare: fiecare tranzacţie se efectuează independent de alte tranzacţii ce se efectuează în acelaşi timp
* Durabilitate: tranzacţiile efectuate cu succes devin permanente chiar daca sistemul eşueaza

Operaţiile pe entitate sunt tranzacţionale, adică este necesar să fie parte a unei tranzacţii. Avem două modele tranzacţionale suportate de JPA:

* resource-local, reprezintă tranzacţiile native suportate de driver-ele JDBC în unitatea de persistenţa
* JTA, sunt parte a server-ului Java EE
* Interfaţa EntityTransaction suportă tranzacţii resource-local, şi este obţinută din EntityManager prin apelul metodei getTransaction().

Metode obişnuite ale unei tranzacţii includ:begin(), pornesc un nou context tranzacţional,

commit(), încheie contextul tranzacţional curent şi scrie orice modificări nescrise înca în baza de date,

rollback(), readuce tranzacţia la starea iniţială.

# Descrierea soluţiei

## Stabilirea obiectivelor

Aplicaţia are ca obiectiv principal formarea unor echipe de fotbal “echilibrate” şi stabilirea programului pentru meciurile de fotbal. Acest lucru va duce la organizarea mai eficientă a grupului,fiecare membru fiind la curent cu toate evenimentele următoare şi cu schimbările apărute, putând de asemenea interveni şi participa la organizarea meciurilor de fotbal.

Un alt obiectiv îl reprezintă comunicarea eficientă în cadrul unui grup de indivizi, grup structurat pe diferite criterii. Se doreşte ca apicaţia să faciliteze transmiterea corectă si rapidă de informaţii în cadrul grupului.

De asemenea , aplicaţia se doreste a fi un instrument pentru păstrarea istoricului informaţiilor, atunci când are loc o modificare a acestora. Utilizatorii vor putea consulta istoricul datelor, facând modificări ce vor fi persistate într-o bază de date.

Dinamicitatea grupului este un alt obiectiv important. Pentru a păstra această caracteristică, membrii grupului trebuie să aibă posibilitatea de a modifica structura grupului, prin introducerea de noi membrii, sau prin eliminarea celor care nu mai doresc să facă parte din grup.

## Cerinţele utilizatorilor fată de aplicaţie

În ceea ce priveşte utilizatorii aplicaţiei, aceştia utilizează aplicaţia pentru a se putea organiza într-un mod eficient, urmând totodată anumite reguli unanim acceptate.

În primul rând, se stabilesc nişte criterii conform cărora o echipă să se poată considera ‘echilibrată’ sau nu. Astfel, putem spune că două echipe sunt cu atât mai “echilibrate” cu cât diferenţa dintre suma rating-urilor jucătorilor din cele două echipe este mai mica. Se doreşte ca acest lucru să fie realizat atât în funcţie de rating-ul fiecărui jucător, cât şi în funcţie de preferinţele acestuia, în mod automat, fără a fi nevoie ca împărţirea pe echipe să se faca manual, până la obţinerea “echilibrului”.

Un altă cerinţă este reprezentată de păstrarea istoricului jocurilor, dar şi evidenţa evoluţiei jucătorilor, aceştia primind o evaluare după fiecare meci caştigat sau pierdut. După fiecare meci jucat, se doreşte păstrarea rezultatului. Rezultatul va fi un număr, ce va reprezenta diferenţa dintre numărul de goluri înscrise de fiecare echipă. În funcţie de rezultat, atât pentru jucătorii echipei câştigătore, cât şi pentru cei din echipa învinsă, ratingul se va modifica. Astfel, după un meci jucat, fiecare jucator va avea un nou rating, reprezentând evoluţia, sau involuţia acestuia.

De asemenea, se doreşte ca accesul la datele din cadrul aplicaţiei să se facă ţinând cont de drepturile de acces ale fiecărui utilizator. Astfel, la înscriere, un utilizator nu are dreptul de a stabili şi organiza echipe, el poate doar să vadă următoarele meciuri şi să se înscrie la cele la care doreşte să participe. Poate vizualiza istoricul meciurilor sale, precum şi graficul ce va reprezenta evoluţia/involuţia sa, realizat pe baza ratingului pe diferite perioade de timp. Un utilizator cu drepturi de administrator are acces la date despre toti jucătorii, poate elimina(logic) un jucător din baza de date, sau poate adăuga noi jucători. Tot el este cel care va stabili datele pentru meciurile următoare, va introduce scorul obţinut de echipe şi va putea să vizualizeze diagrame de rating pentru alţi utilizatori, înafară de el însuşi.

Pentru a se genera echipele în mod automat, trebuie ca la meci sa fie inscrisi minim 4 jucatori. Numărul maxim de jucători cu care se pot forma echipe este 16. În momentul în care s-a atins numărul de jucători necesari, un utilizator cu drepturi de administrator poate alege opţiunea de a genera echipele. În acest mod, se vor genera toate perechile de echipe posibile, care vor fi sortate crescător, dupa diferenţa dintre suma ratingurilor jucătorilor fiecărei echipe, astfel încât un utilizator de tip administrator poate alege oricare din perechile generate, chiar daca aceasta nu este cea mai “echilibrata”.

Înregistrarea unui nou jucător se poate face atat de către un utilizator al aplicaţiei care deţine drepturi pentru a face acest lucru, cât şi de user-ul care doreşte crearea unui cont.

Un alt scop al aplicaţiei este păstrarea istoricului datelor, pentru a putea fi consultate mai târizu, dar şi adăugarea de date noi atât pentru userii existenţi, cât şi pentru activităţile în desfăşurare. Datele pot fi vizualizate sub diferite forme: liste, diferite tipuri de grafice şi pot fi salvate în diferite formate.

Fiecare utilizator are un drept de acces, care îi permite să realizeze diferite interacţiuni cu ajutorul interfeţei web. Astfel, unui utilizator i se pot atribui drepturi de administrator de către alţi utilizatori de acest tip, pentru a putea realiza interacţiuni suplimentare precum ştergerea logică a unui utilizator din baza, anularea unui joc sau realizarea echipelor.

La înregistrare, viitorul utilizator trebuie să îşi aleagă un username şi o parolă, rating-ul sau fiind setat initial 5 în mod automat. Dacă înregistrarea s-a făcut cu succes, user-ul se va putea autentifica şi va avea acces la datele din aplicaţie.

Opţiunile pe care acesta le are sunt structurate sub forma unui meniu. Astfel, un user fără drepturi de admin poate să vadă meciurile următoare şi istoricul meciurilor, şi se poate înscrie pentru a participa la un meci următor sau poate anula participarea sa la un joc. Îşi poate consulta datele personale şi îşi poate actiualiza poza de profil, disponibilitatea, sau parola. Are de asemenea acces la datele publice ale celorlalţi utilizatori, precum numele, rating-ul curent al acestora şi poza de profil, însa doar pentru vizualizare. O alta opţiune pe care o poate alege este cea de a-şi vizualiza diagrama de rating.

Unui utilizator de tip administrator, i se pun la dispoziţie câteva opţiuni în plus, precum adăugarea unui nou jucător şi stabilirea datei/datelor pentru următoarele meciuri. De asemenea, acesta are acces la diagramele de rating pentru toţi utilizatorii, poate vizualiza şi poate elimina ( la nivel logic), din baza de date un jucător. În plus, are acess la opţiunea de generare a echipelor, validând formatul echipelor şi informând ceilalţi participanţi înscrişi. Tot el este cel care poate anula un meci următor şi poate introduce rezultatul unuia anterior. Are posibilitatea de a schimba configuraţia echipelor stabilite până în momentul în care se loghează şi de a le persista.

## Diagrama claselor

Codul sursă al server-ului este structurat în patru pachete:’dataAccessLayer’,’helpers’,’model’ şi ‘views’, fiecare cuprinzând clase specifice unei anumite functionalităţi.

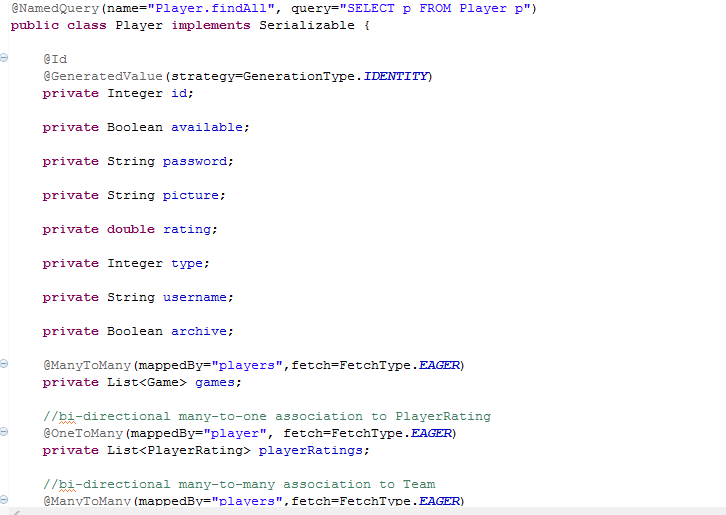
#### Pachetul ‘Model’

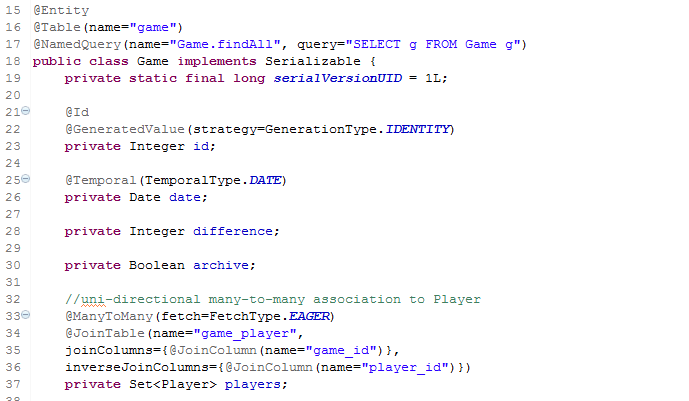
Pachetul ‘model’ cuprinde 4 clase, reprezentând principalele entităti cu care lucrează aplicaţia: Player, Game, PlayerRating şi Team.

Clasa ‘Player’ cuprinde atribute care reprezintă informaţiile pentru fiecare jucător, precum şi metode publice pentru interacţiunea cu entitatea.

Clasa Player cuprinde următoarele atribute:

* Int id, identificatorul unic al jucătorului
* String username, numele jucătorului
* String password, parola utilizatorului
* String picture, poza de profil în cazul în care utilizatorul a încarcat una, altfel se setează o poza de profil default
* Boolean available, atribut ce reţine disponibilitatea jucătorului; un jucator available se poate înscrie la meciuri şi poate fi inclus în formarea echipelor
* Double rating, conţine rating-ul curent al jucătorului
* List<Games> games, va conţine lista tuturor jocurilor unui user, care cuprinde atât jocurile următoare cât şi jocurile care s-au desfăşurat
* List<PlayerRating> playerRatings, păstrează istoricul ratingurilor pentru un utilizator

 Figură 17.Clasa Player

Clasa Game este prezentată în Figura18.

Figură 18. Clasa 'Game'

Atributele sale sunt:

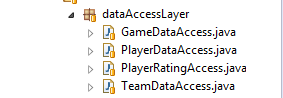
* Integer id, identificatorul unic pentru joc
* Date date, reţine data la care este programat jocul
* Integer difference, rezultatul jocului, reprezentând modulul dintre numărul de goluri înscrise de fiecare echipă
* Boolean archive, variabilă booleană ce va fi setată false în baza de date în cazul în care jocul este anulat
* Set<Player> players, o listă ce va conţine jucătorii înscrişi la joc
* Team firstTeam, prima echipă a meciului
* Team secondTeam, cea de-a doua echipă a meciului

Clasa ‘Team’ operează cu date referitoare la echipele fiecărui joc. Aceasta conţine informaţii atât despre jucători, cât şi despre modul în care sunt împărţiţi în cele două echipe.

Clasa ‘PlayerRating’ este folosită pentru a păstra istoricul rating-urilor pentru fiecare jucător. Fiecare entitate de tip ‘PlayerRating’ conţine următoarele informatii: id-ul jucătorului, rating-ul său şi data la care a fost setat acesta.

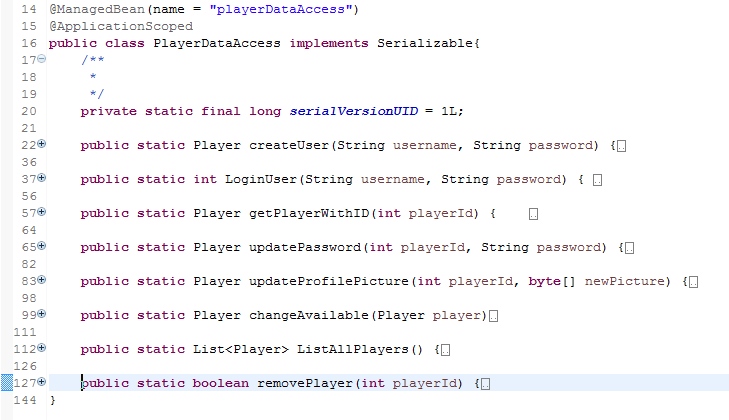
#### Pachetul ‘DataAccessLayer’

Pachetul ‘dataAccessLayer’ cuprinde clase care ajută la salvarea modificărilor şi persistarea entităţilor. Clasele conţin metode statice ce asigură nivelurile de acces la entităţi.



Figură 19. Pachetul 'DataAccessLayer'

Clasa ‘PlayerDataAccess’ cuprinde următoarele metode:

Figură 20. Clasa PlayerDataAccess

Astfel, cu ajutorul acestei clase, se pot înregistra noi utilizatori în baza de date, se pot face diverse modificări asupra utilizatorilor existenti sau se pot lista utilizatorii curenţi.

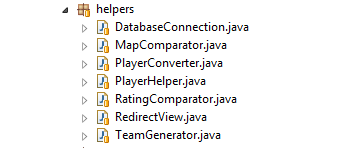
Clasa ‘GameDataAccess’ reprezintă un nivel de acces pentru clasa ‘Game’. Cu ajutorul acesteia se pot persista sau modifica entităţi de tip ‘Game’.

Funcţionalitatea clasei ‘PlayerRatingAccess’ este aceea de a asigura actualizarea şi salvarea ratingurilor pentru fiecare jucător participant la un meci de fotbal, dupa ce acesta a avut loc.

‘TeamDataAccess’ realizează salvarea configuraţiei echipelor de fotbal, iar în cazul unor modificări, noile date sunt actualizate tot cu ajutorul acestei clase.

#### Pachetul ‘Helpers’

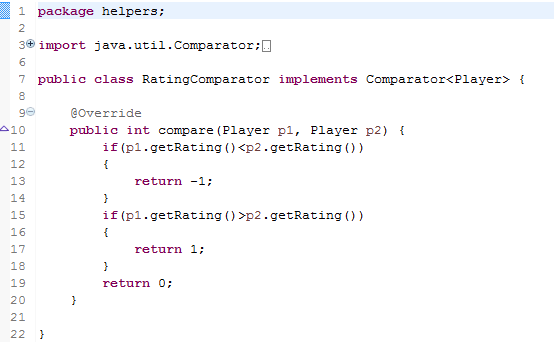
Pachetul ‘helpers’ cuprinde clase responsabile de realizarea legăturilor dintre clase sau generarea de informaţii ce vor fi utilizate ulterior pentru îndeplinirea altor funcţii. Clasele au funcţionalităţi bine delimitate, şi reprezintă obiecte importante pentru funcţionarea corectă a întregului program.



Figură 21. Pachetul ‘Helpers’

Clasa ‘DatabaseConnection’ este o clasă de tip Singleton, existând o singură instanţă a acesteia pe parcursul execuţiei programului. Deoarece crearea unei conexiuni la baza de date este o operaţie destul de costisitoare din punct de vedere al resurselor şi al performanţei, crearea conexiunii se va face o singură dată, urmând să fie folosită de clase unde este necesară această conexiune. Obţinearea unei conexiuni de către o clasă se face prin apelarea metodei GetConnection(), ce returnează un obiect de tip EntityManager, necesar pentru realizarea managementului entităţilor. În general, clasele care folosesc un obiect de acest tip sunt clasele din pachetul ‘dataAccessLayer’.

Clasele RatingComparator şi MapComparator implementează interfaţa IComparable şi suprascriu metoda Compare. Sunt folosite în realizarea sortărilor unui grup de jucători după ratingul acestora, resprectiv în compararea echipelor în scopul obţinerii celei mai ‘echilibrate’ echipe.



Figură 22. Clasa RatingComparator

PlayerConverter este clasa ce reprezintă un convertor custom pentru obiectul de tip Player. Pentru a crea o clasă de tip convertor al unui obiect, clasa trebuie să implementeze interfaţa Converter şi să suprascrie metodele:

@Override

public Object getAsObject(FacesContext context, UIComponent component,

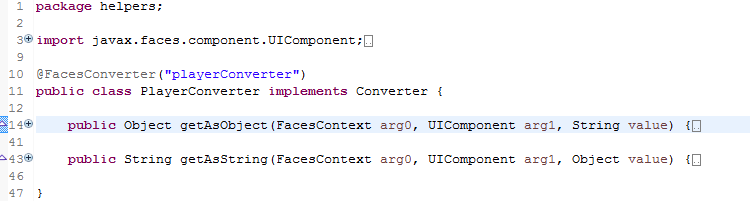
String value) { //...}

@Override

public String getAsString(FacesContext context, UIComponent component,

Object value) {//...}

Pentru a atribui convertorului un ID, putem folosi adnotarea „@FacesConvertor’. În cazul clasei PlayerConverter, am utilizat adnotarea pentru a-i atribui id-ul ‚playerConverter’.



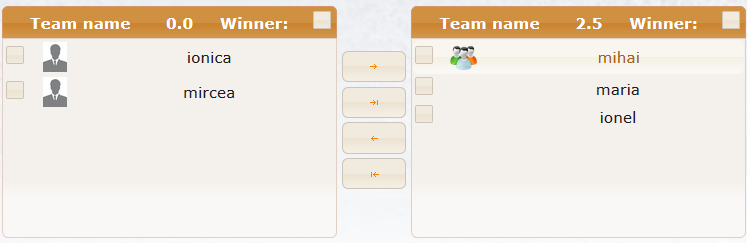
Figură 23. Clasa PlayerConverter

În cadrul aplicaţiei, este utilizată atunci când se doreşte mutarea unui jucaăor dintr-o echipă în alta, folosind interfaţa web. Echipele sunt afişate dinamic, sub formă a două tabele, utilizatorul având posibilitatea să facă schimbări de elemente din tabel, realizând astfel un schimb de jucători între echipe. Acest lucru este implementat cu ajutorul elementului PickList din framework-ul Primefaces, căruia i se plasează ca şi atribut convertorul custom ‘PlayerConverter’.



Figură 24. Sintaxa elementului Picklist cu atributul ‘converter’

În pagina web accesată de client, cele două echipe sunt reprezentate precum în Figura25.



Figură 25. Reprezentarea echipelor cu ajutorul elementului PickList

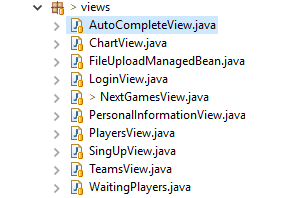
Clasa ‘RedirectView’ este folosită în cadrul claselor din pachetul Views, pentru redirectarea către diferite pagini, în funcţie de tipul de utilizator. Aceasta conţine două metode statice:

* void Redirect (Player player, String pageForNormalUser, String pageForAdmin), ce realizează redirectarea către pagini custom pentru fiecare tip de utilizator
* void Redirect (String page), redirecţionează toate categoriile de utilizatori către aceeaşi pagină web

Un exemplu de utilizare a clasei este atunci când se face autentificarea unui utilizator. După validarea datelor, utilizatorul este redirecţionat către o pagină, ce va cuprinde conţinut accesibil şi util pentru acesta.

#### Pachetul ‘Views’

În cadrul acestui pachet sunt cuprise clasele de tip ‘ManagedBean’, cele care sunt responsabile de logica aplicaţiei. Ele reprezintă practic obiecte pe partea de server, associate unor componente UI folosite în paginile web. Clasele regăsite în pachet sunt cele din Figura26.



Figură 26. Pachetul ‘views’

Fiecare clasă este asociată cu componente UI din paginile web, iar la interacţiunea utilizatorului cu aplicaţia, acestea sunt responsabile de preluarea şi prelucrearea, daca este cazul, a datelor cu care lucrează utilizatorul.

De exemplu, pentru a adăuga noi jucători în lista de jucători deja existenţi ai unui joc, se foloseşte elementul Primefaces AutoComplete, căruia i se asociază o listă de jucători din managed bean-ul ‘AutoCompleteView’. Asocierea se face prin setarea atributului value din autoComplete:

<p:autoComplete multiple="true" **value="#{autoCompleteView.selectedPlayers}"** converter="playerConverter" forceSelection="true">

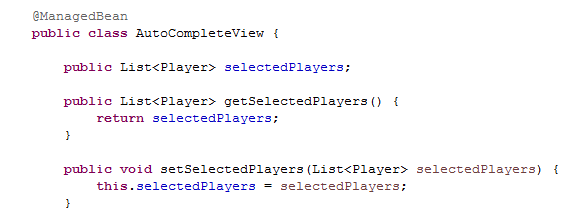
<p:column style="width:80%">

<h:outputText value="#{theme.username}"/>

</p:column>

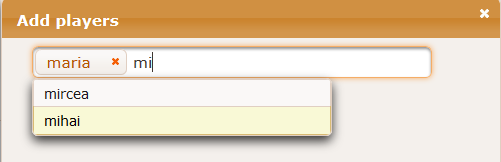
</p:autoComplete>

unde ‚selectedPlayers’ este o listă din clasa ’AutoCompleteView’.



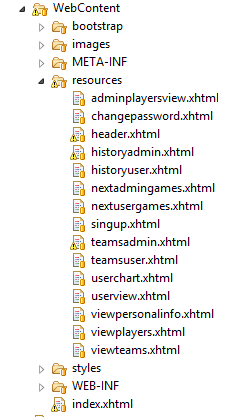
Figură 27. ManagedBean-ul AutoCompleteView

În cadrul paginii web, utilizatorul va interacţiona cu fereastra:



Figură 28. Fereastra pentru adăugarea unui nou jucător

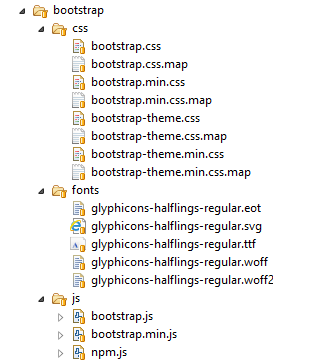
Partea de client este reprezentată de paginile web, interfaţa web cu care interacţionează clientul. Paginile web sunt cuprinse în folderul ‘WebContent->resources’. Conţinutul din ‘WebContent’ este ilustrat in Figura29.



Figură 29. Conţinutul folderului 'WebContent'

Paginile au extensia xhtml, fiecare având un scop bine stabilit în interacţiunea cu utilizatorul. Folderul mai conţine în plus alte foldere ce cuprind librării sau resurse necesare pentru buna funcţionare a aplicaţiei.

În folderul ‘bootstrap’ se află librăria Bootstrap, ce va fi inclusă ulterior în paginile web pentru a putea fi folosită. Librăria conţine jar-uri, pagini .css dar şi pagini .js.



Figură 30. Conţinutul folderului 'bootstrap'

Folderul ‘images’ conţine pozele de profil ale jucătorilor, încarcate pe server. De fiecare dată când se face update pozei de profil, aceasta este salvată în acest folder, existând o singură fotografie pentru fiecare user.

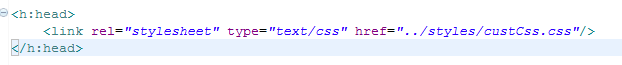
În folderul ‘styles’ sunt cuprinse câteva pagini .css, care sunt folosite pentru a suprascrie stilurile implicite fie din Bootstap sau cele din Primefaces.

## Consideraţii relativ la implementare

“Aplicaţia suport pentru constituirea de echipe echilibrate şi customizabile” este o aplicaţie web. Pe parte de server este folosit Tomcat, codul fiind scris în limbajul de programare Java, folosind mediul de dezvoltare Eclipse. Partea de Client, care reprezintă interacţiunea cu utilizatorul, este dezvoltată folosind limbajul HTML, împreună cu framework-urile Bootstrap şi Primefaces.

### Suprascrierea stilului CSS implicit din Bootstap

Pentru a se face suprascrierea unui stil CSS din Bootstrap sau pentru a adăuga noi elemente de stil se creează o nouă pagină cu extensia .css. Apoi se adaugă referinţa către pagina creată în elementul <head></head> al paginii:



Pentru a aplica paginii stilul custom dorit, este de ajuns să adăugăm descrierea acestuia în pagina .css creată anterior. De exemplu, dacă dorim să schimbăm stilul elementului div ce are clasa Bootstap ‘navbar navbar-default’ din secvenţa de cod următoare:

<div class=*"navbar navbar-default"* style="margin-bottom:*0px*">

<div class=*"container-fluid"*>

<div class=*"row"*></div>

</div>

</div>

adăugam în pagina .css descrierea stilului:

. navbar navbar-default

{

padding-top: 10px;

float: none;

text-align: center;

font-size: 18px;

}

Putem defini propriile clase de stil ce pot fi aplicate elementelor. În “Aplicaţia suport pentru constituirea de echipe echilibrate şi customizabile” am definit clasa ‘withBackground’, care setează un fundal unui element html daca este aplicată acestuia.

.withBackground

{

background: url('../images/d.jpg') no-repeat center center fixed;

-webkit-background-size: cover;

-moz-background-size: cover;

-o-background-size: cover;

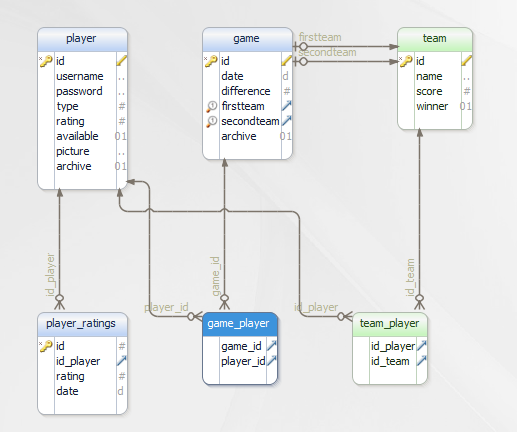
background-size: cover;

}

Pentru a adăuga o clasa de stil unui element, se adaugă numele clasei custom la numele clasei din Bootstap:

<div class="container-fluid withBackground" ></div>

### Persistenţa datelor

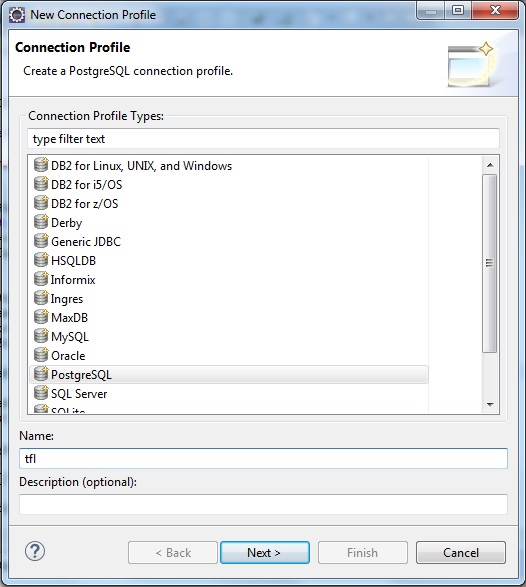
Datele fiecărui utilizator sunt salvate într-o bază de date, realizată cu ajutorul sistemului de baze de date relaţional PostgreSQL şi administrată cu ajutorul tool-ului pgAdmin. Aceasta este formată din 6 tabele, ce reprezintă entităţile principale cu care operează aplicaţia.

Figură 31. Baza de date a aplicatiei

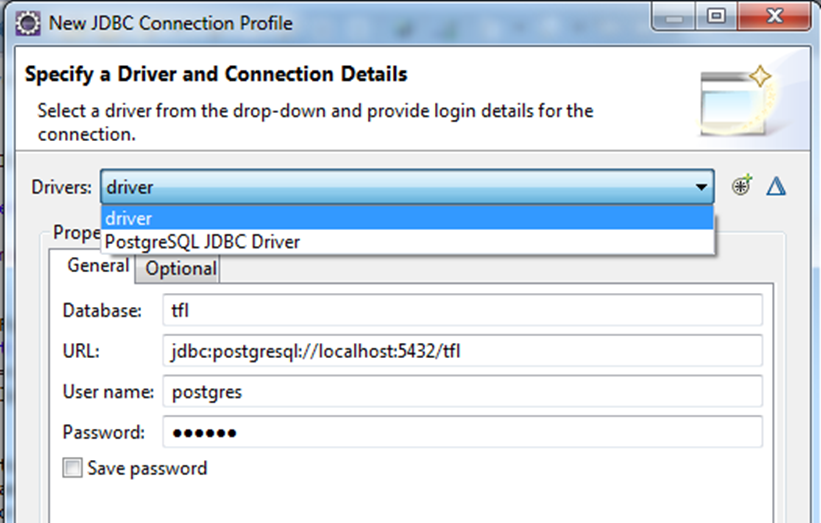
Tabelele sunt mapate cu ajutorul framework-ului JPA, devenind clase cu care se va realiza logica aplicaţiei. Clasele ce reprezintă entităţile din baza de date sunt integrate în pachetul ‘model’.

#### Maparea entităţilor cu ajutorul JPA

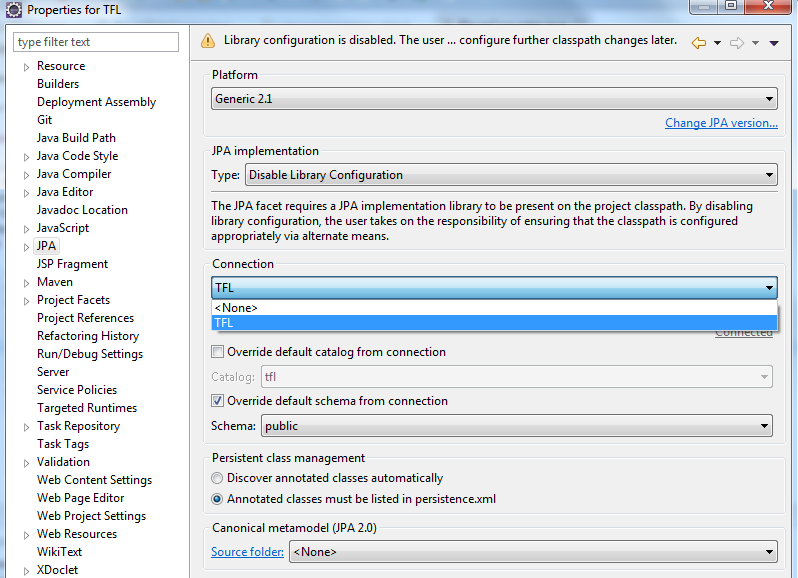
Pentru a se obţine clase pornind de la o bază de date existentă cu ajutorul JPA, în primul rând trebuie să creem o conexiune la baza de date. Folosind Eclipse, se urmează paşii:

1. Din tab-ul Eclipse ‘Data Source Explorer’ se alege opţiunea Database Connections-> New…
2. Se alege driver-ul PostgreSQL pentru a se realiza conexiunea

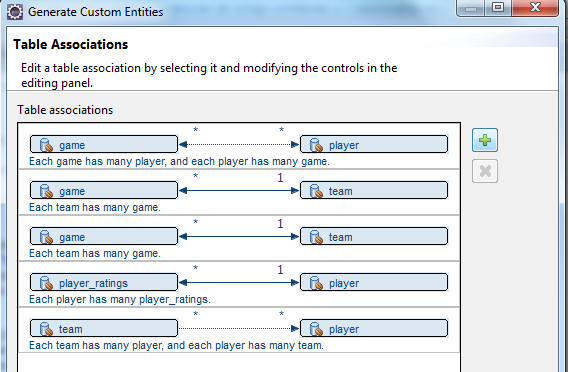
Figură 32. Crearea conexiunii la baza de date PostgreSQL



Figură 33. Alegerea driver-ului pentru conexiune

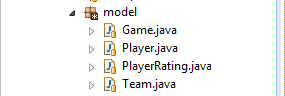
1. Se verifică existenţa JPA în cadrul proiectului. Acest lucru se poate face accesând: Click dreapta pe proiect->Proprietes->JPA

Figură 34.Verificarea existenţei JPA în cadrul proiectului

1. Se generează entităţile prin accesarea: Click dreapta pe proiect-> New-> JPA Entities from tables, apoi din fereastra aparută se aleg entităţile pentru care se doreşte generarea claselor.

Figură 35. Maparea entitatilor din baza de date

În această etapă se pot adăuga noi asocieri între tabele sau se pot customiza cele existente.

În final, pachetul ‘model’ este format din 4 clase: Game, Player, PlayerRating şi Team, rămânând două tabele de legătura pentru care nu a fost necesară crearea unei entităţi.

Figură 36.Clasele rezultate în urma mapării JPA a entităţilor

### Algoritmul de generare de echipe ‘echilibrate’

Algorimul de generare al echipelor este cuprins în clasa ‘TeamGenerator’. Acesta reprezintă o formă mai complexă a algorimului de generare a tuturor combinărilor din n elemente luate câte k. Formula generală a combinărilor este:



Figură 37. Formula generala combinari

Algoritmul implementat se foloseşte de această formulă, generând în primă instanţă toate combinările din n elemente luate câte n/2, pentru ca echipele să aibă şi acelaşi număr de jucători.

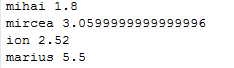
Principalele atribute ale clasei ‘TeamGenerator’ sunt:

* List<Player> list, reprezintă lista de jucători ce se doreşte a fi împărţită în două echipe
* Double totalValue, însumează rating-ul tuturor jucătorilor împărtit la 2 şi va fi necesar în algoritm pentru compararea echipelor
* TreeMap<Double,List<Player>> resultMap, va conţine echipele sortate în ordine crescătoare, în funcţie de gradul de ‚echilibru’ obţinut

O echipă se consideră cu atât mai echilibrată cu cât modulul diferenţei dintre totalValue şi suma rating-urilor jucătorilor din echipă este mai mic. Considerăm:

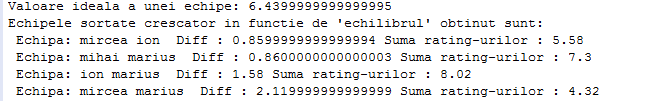
calculată pentru fiecare echipă. În TreeMap se va reţine fiecare grup obţinut împreună cu valoarea sa *diff.* Deoarece TreeMap primeşte pe constructor comparatorul de echipe ‘MapComparator’,în final obţinem toate grupurile posibile formate din lista iniţială de jucători, sortate crescător.

De exemplu, în Figura38, avem lista de jucători şi rating-ul fiecăruia, listă ce se doreşte a fi împărţită în două echipe.



Figură 38 Lista jucatori pentru împărţirea ăn echipe

După aplicarea algoritmului, se obţine rezultatul:



Figură 39. Rezultatul împărţirii in echipe e listei

Se observă că suma rating-urilor pentru grupul format din jucătorii ‘Mircea’ şi ‘Ion’, 5.58, este cea mai apropiată valoare de valoarea ideală a unei echipe în cazul acestui grup, avand diff=0.859. Evident, al doilea grup va fi format din jucătorii rămaşi, ‘Mihai’ şi ‘Marius’, cu diff=0.86.

### Actualizarea modificărilor

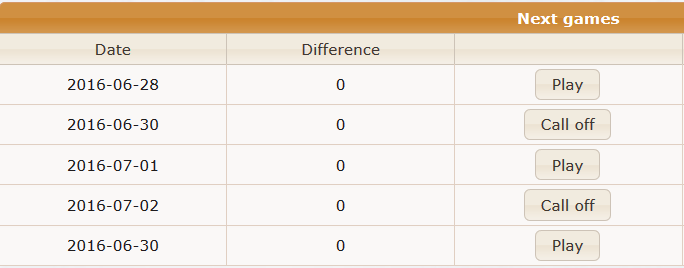
După fiecare meci jucat, apar modificări ce trebuie înregistrate, acest lucru necesitând şi o actualizare a datelor ce depind de fiecare schimbare apărută. Una dintre schimbările ce poate apărea este rearanjarea echipelor în cadrul terenului. Chiar daca echilibrul dintre echipe este influenţat într-un mod negativ, aplicaţia se doreşte a fi una flexibilă, ca urmare pot apărea schimbări de ultim moment, deşi echipele au fost salvate în baza de date.

În cazul acesta, un utilizator ce deţine drepturi poate accesa prin intermediul aplicaţiei datele despre meci, şi poate actualiza configuraţia celor două echipe, recalculându-se rating-ul total al echipei. De asemenea, utilizatorii au dreptul să îşi aleagă un nume personalizat pentru fiecare echipă, numele putând fi schimbat de mai multe ori în cadrul unui joc, însă numai de către utilizatori autorizaţi.

În plus, un user admin poate adăuga noi jucători, în cazul în care nu este depăşit numărul maxim de jucători înscrişi. În această situaţie, se păstrează echipele existente, juătorii fiind schimbaţi după aplicarea algoritmului de generare de echipe.

Orice utilizator are opţiunea de a se înscrie pentru a participa la un meci de fotbal, sau poate opta să nu mai participe, prin intermediul meniului pus la dispoziţie de pagina web. În cazul înscrierii unui jucător algoritmul urmează următorii paşi:

1. Se verifică dacă utilizatorul este disponibil pentru înscriere
2. Daca este disponibil, se verifică dacă numărul maxim de jucători ce se pot înscrie nu a fost depăşit
3. Dacă se mai pot înscrie jucători, user-ul se adaugă în lista de jucători ai jocului curent şi se reconfigurează echipele
4. În cazul în care s-a depăşit numărul maxim de locuri, utilizatorul este adăugat în lista de aşteptare a meciului, pentru cazurile în care alţi jucători renunţă.



Figură 40.Înscrierea unui jucător în cadrul unui meci

În situaţia în care un jucător nu mai poate participa, are disponibilă opţiunea ‘Call off’. El este eliminat din lista de jucători ai meciului, şi este adăugat primul jucator din lista de aşteptare, în cazul în care există. Apoi se face o reconfigurarea a echipelor, pentru a restabili echilibrul.

### Actualizarea rating-urilor

După fiecare meci jucat, utilizatorii au opţiunea de a introduce rezultatul, sub formă de număr, ce reprezintă numărul de goluri pentru fiecare echipă. În funcţie de diferenţa dintre numărul de goluri, rating-urile jucătorilor sunt actualizate, putându-se astfel urmări evoluţia sau involuţia utilizatorilor.

Pentru a se realiza actualizarea rating-urilor algoritmul urmează următorii paşi:

1. Se calculează modulul diferenţei dintre numărul de goluri dintre prima echipă şi a doua. Fie diff diferenţa dintre numarul de goluri dintre cele doua echipe
2. Se desemnează echipa caştigătoare (echipa cu cele mai multe goluri)
3. Pentru fiecare jucător se recalculează ratingul. Dacă acesta face parte din echipa caştigătoare, rating-ul său va creşte după formula:

Altfel, noul rating va fi:

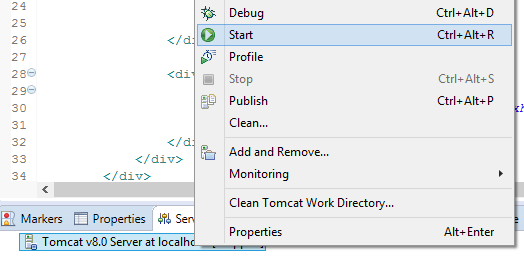
O problemă apare în cazul în care se doreşte actualizarea numărului de goluri ale unui meci, posibil înregistrat greşit anterior. Deoarece rating-ul curent al fiecărui utilizator a fost actualizat în urma consemnării eronate a rezultatului precedent, o nouă întregistrare a rezultatelor ar putea duce la un update necorespunzător al fiecărui jucător. În cazul acesta se vor şterge toate datele începând cu data meciului pentru care se face actualizarea şi se vor recalcula din nou pentru fiecare meci, păstrându-se astfel continuitatea şi acurateţea înregistrărilor.

De exemplu, dacă rating-ul unui jucător care se înscrie la un meci este 5.0, iar după terminarea jocului echipa sa a înscris 4 goluri, câştigând cu diferenţa de 3, noul rating va fi 5.01.

## Ghid de utilizare

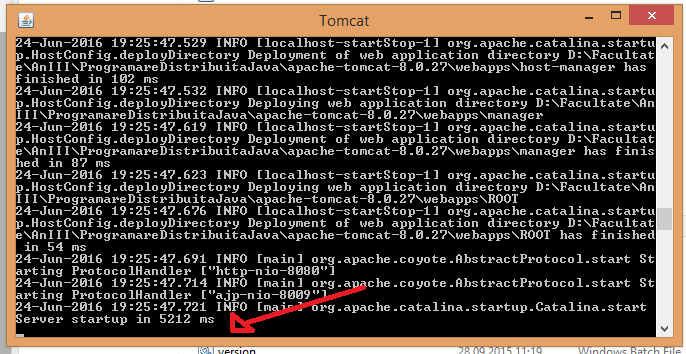
Pentru a putea utiliza aplicaţia, trebuie să pornim serverul, care va prelua datele de la client.

Serverul Tomcat se poate integra în mediul de dezvoltare Eclipse şi se poate porni cu ajutorul acestuia, prin Click dreapta pe server->Start.



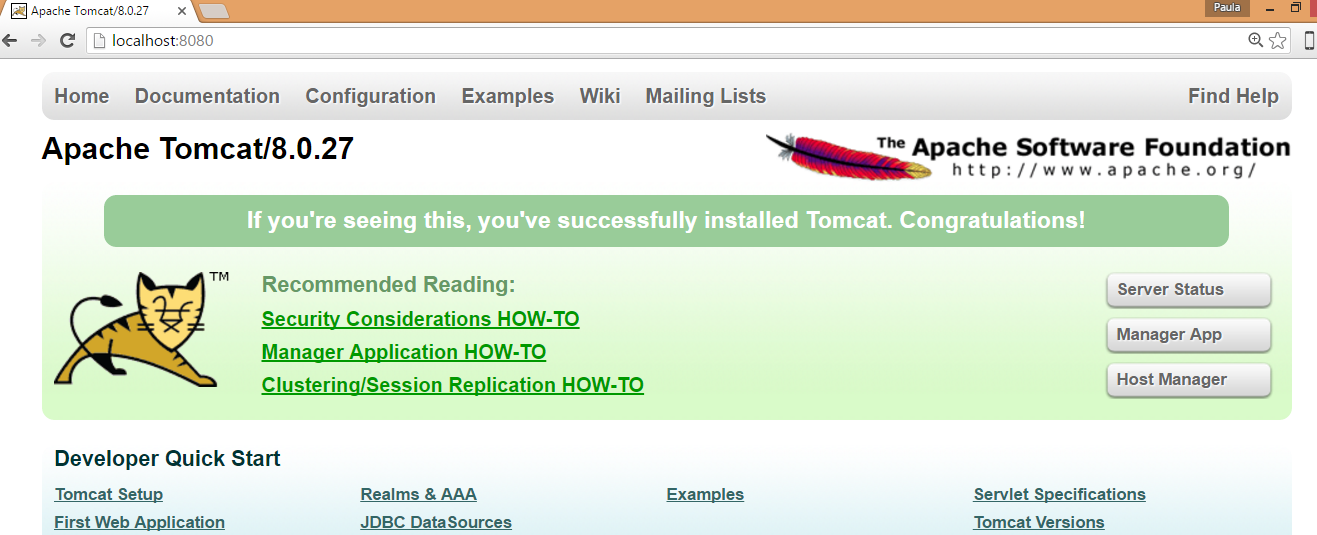
Figură 41. Pornirea serverului Tomcat din mediul de dezvoltare Eclipse

Dacă nu este integrat în mediul Eclipse, se poate porni prin fişierul startup.bat, accesat fie din linia de comandă, fie din folderul unde este localizat, prin dublu click.



Figură 42. Pornirea serverului Tomcat din linia de comandă

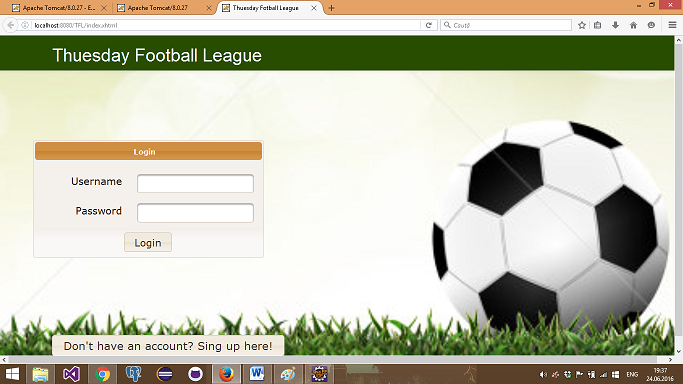
O altă modalitate de a ne asigura că serverul este pornit este aceea de a accesa într-un browser web pagina: localhost:8080, 8080 fiind portul pe care Tomcat rulează implicit.



Figură 43. Verificarea pornirii corecte a serverului Tomcat

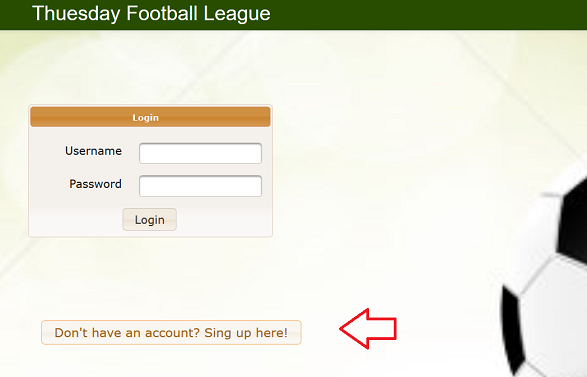
Accesul în aplicaţie se face tot prin intermediul browser-ului.





Figură 44. Accesul la aplicatie prin intermediul unui browser web

Următorul pas este completarea datelor necesare pentru autentificare: numele de utilizator şi parola, în cazul în care există un cont creat, altfel se poate crea un cont nou, prin accesarea butonului din partea de jos a paginii.



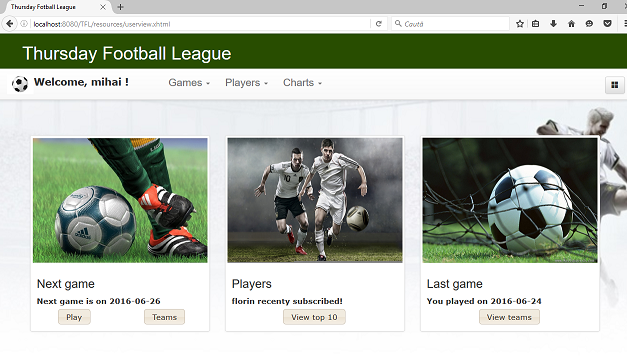
Figură 45. Înregistrarea unui nou jucător

După completarea datelor, se verifică daca acestea sunt corecte, în caz contrar se afişează un mesaj de atenţionare:



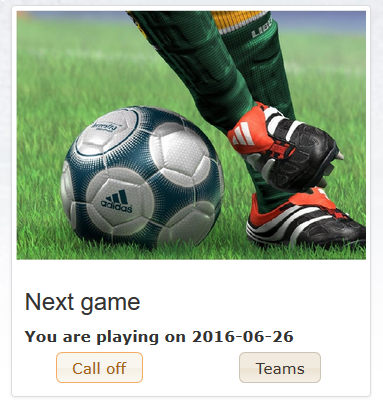
Figură 46. Exemplu autentificare cu date greşite

În cazul în care datele sunt corecte, utilizatorul este redirectat către o altă pagină, unde va putea interacţiona cu aplicaţia prin intermediul unui meniu.



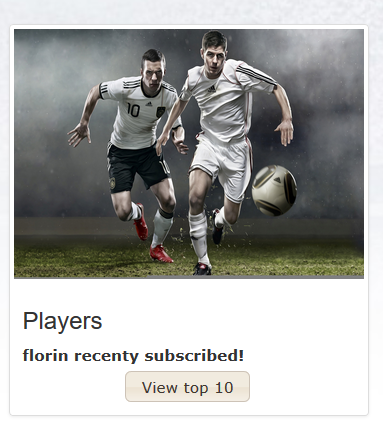
Figură 47. Pagina către care este redirectat un utilizator după autentificare

În pagina principală acesta are informaţii despre următorul meci, la care se poate înscrie sau poate vedea echipele formate până în momentul curent. În cazul în care acesta este înscris la un meci următor, sunt afişată informaţii despre acel meci, iar user-ul are opţiunea de anulare a meciului.



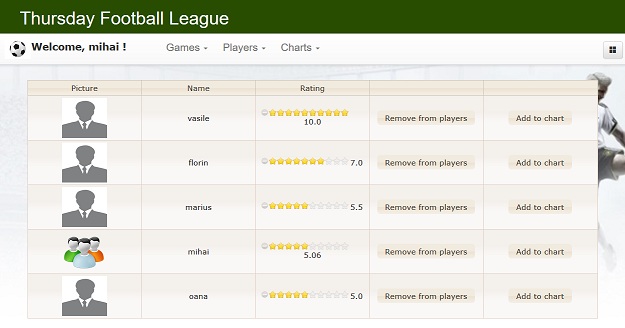
Figură 48. Prezentarea informaţiilor despre meciul următor

Pagina principală conţine de asemenea o secţiune în care se găsesc informaţii despre jucători. Este afişat ultimul user înregistrat în ultima săpămână, dacă există. Se poate accesa pagina ce va conţine top 10 jucători, ordonaţi după rating.



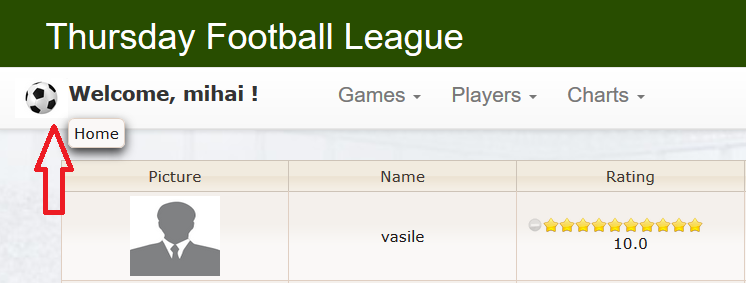
Figură 49. Prezentarea informaţiilor despre ultimii jucători înscrişi

La ckick pe butonul ‘View top 10’, utilizatorul este redirectat către vizualizarea top-ului.



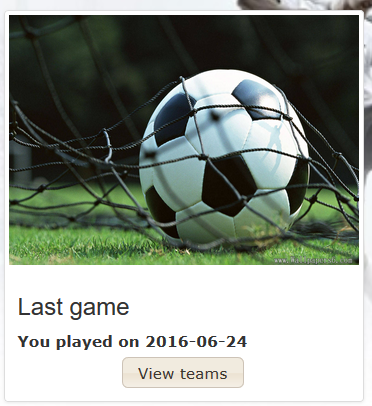
Figură 50. Top 10 jucători

Revenirea la pagina de start se poate face prin click pe imaginea din stânga, care este prevăzută cu un tool-tip util pentru navigarea în pagină.



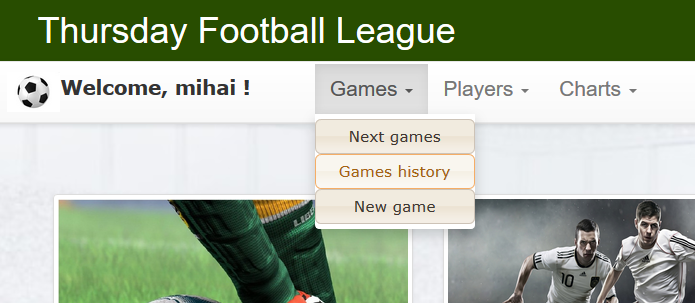
Figură 51. Revenirea la pagina Home a aplicaţiei

Ultima secţiune prezentă în pagina principală este cea dedicată meciurilor ce au avut loc. Aici avem informaţii despre utlimul meci la care a participat user-ul logat. La click pe butonul ‘View teams’ utilizatorul este redirectat către pagina de vizualizare a echipelor, unde poate introduce şi rezultatul meciului.



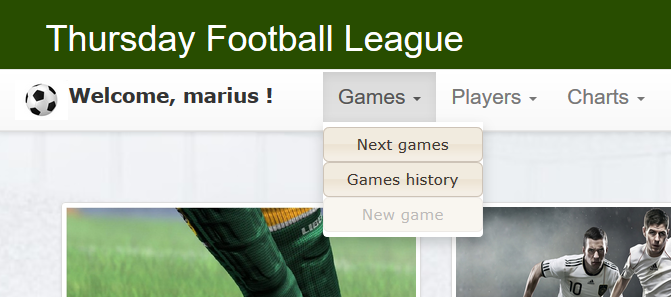
Figură 52. Prezentarea informaţiilor despre meciul anterior

Utilizatorii pot naviga în cadrul aplicaţiei prin intermediul meniului din partea de sus a paginii, care cuprinde secţiunile: Games, Players, Charts. Fiecare secţiune are mai multe subsecţiuni, unele dintre acestea nefiind active în cazul în care utilizatorul nu are drepturi de administrator. Meniul pentru un utilizator de tip administrator, are toate butoanele active.



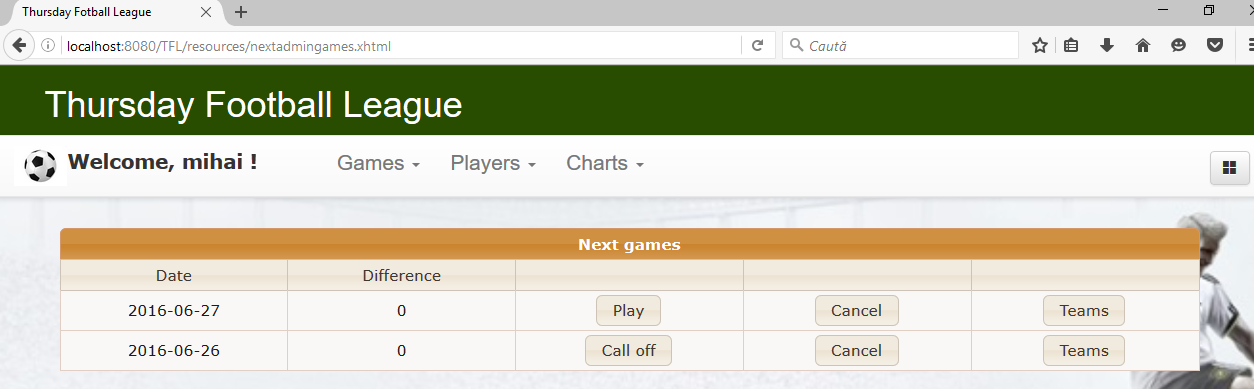
Figură 53. Prezentarea meniului pentru un utilizator de tip admin

Un utilizator fără drepturi de administrator nu are activă opţiunea pentru adăugarea unuj joc nou.



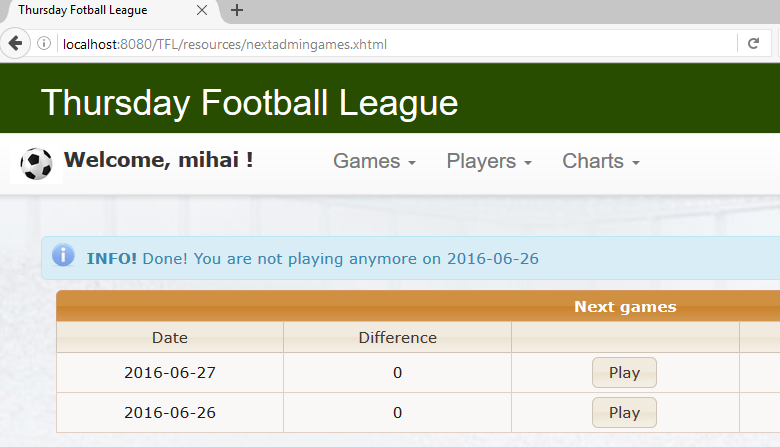
Figură 54. Prezentarea meniului pentru un utilizator fără drepturi de admin

Jucătorul are opţiunile de vizualizare a jocurilor următoare, vizualizare a istoricului jocurilor, sau stabilirea datei pentru un meci următor. La click pe butonul Next games, se va afişa o listă cu următoarele jocuri. Pentru fiecare joc, există 3 opţiuni în cazul unui user admin: se poate înscrie sau poate anula înscrierea făcută, poate anula jocul, sau poate consulta echipele.



Figură 55. Opţiunilepentru meciurile anterioare

După fiecare acţiune efectuată, utilizatorul este informat în legătura cu reuşita acesteia. De exemplu, dacă utilizatorul alege să nu mai joace în jocul din data de 26-06-2016, va primi un mesaj de confirmare şi va fi de asemenea actualizată pagina web.



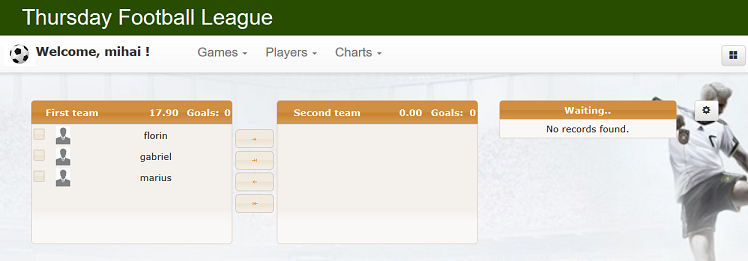
Figură 56. Informarea utilizatorului cu privire la acţiunea solicitată

Dacă se alege anularea unui joc, utilizatorului i se cere mai întâi o confirmare, pentru a avea siguranţa că butonul nu a fost apăsat din greseală. Aşadar, meciul de fotbal este anulat numai în cazul în care este confirmat acest lucru, prin apăsarea butonului Yes. În caz contrar, nu are loc nicio acţiune.



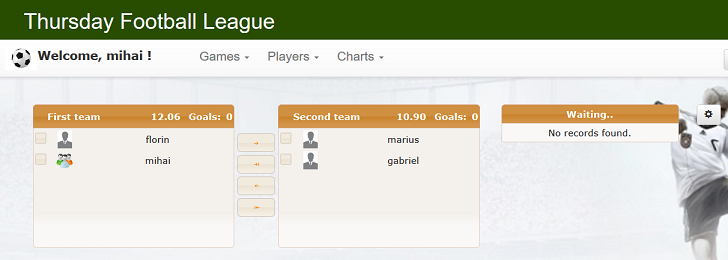
Figură 57. Confirmarea anulării unui joc

La apăsarea butonului ‘Teams’, utilizatorul poate vedea echipele formate până în momentul actual, echipe care pot fi modificate însă până la data jocului. Împărţirea jucătorilor pe echipe se face doar în momentul în care sunt minim 4 utilizatori inscrişi la joc. De exemplu, la meciul din data de 26.06.2016 din Figura58, sunt inscrişi 3 jucatori.



Figură 58. Vizualizarea meciului înainte de a se atinge numărul minim de jucători

Se poate observa faptul că echipele nu sunt formate încă. Dacă utilizatorul logat în acest moment se înscrie la acest joc, se face împărţirea jucătorilor în două echipe echilibrate.



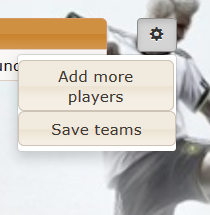
Figură 59. Echipele meciului după adăugarea unui nou jucător

Deoarece utilizatorul logat în acest moment are drepturi de administrator, el poate schimba configuraţia curentă a echipelor, scorul fiecărei echipe actualizând-se, şi poate salva noua configuraţie. Scorul primei echipe este 12.06, iar a celei de-a doua 10.90. După interschimbarea jucătorilor Mihai şi Marius, noul scor pentru fiecare echipă este:12.50, respectiv 10.46.



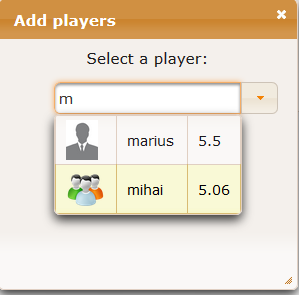
Figură 60. Interschimbarea de jucători în cadrul echipelor

Salvarea noii configuraţii se face prin click pe butonul din dreapta tabelelor cu echipe, apoi ‘Save teams’.



Figură 61. Salvarea configuraţiei curente

Se pot adăuga şi noi jucători echipelor curente, daca nu s-a depăşit numărul maxim de jucători permişi, prin click pe butonul ‘Add more players’. Va apărea o fereastră, de unde se poate alege jucătorul ce se doreşte a fi adăugat, prin selectarea acestuia din lista de jucători aparută.



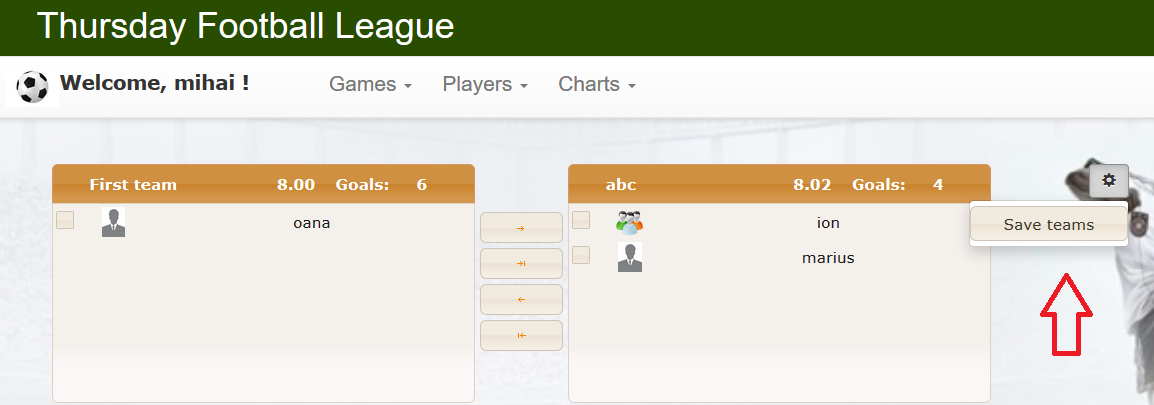
Figură 62. Adăugarea de noi jucători in cadrul echipelor

Prin selectarea opţiunii ‘Games history’ din meniu, se va afişa un istoric al jocurilor, utilizatorii având opţiunea de a vedea echipele şi de a introduce rezultatul jocului, prin click pe butonul ‘Add result’.



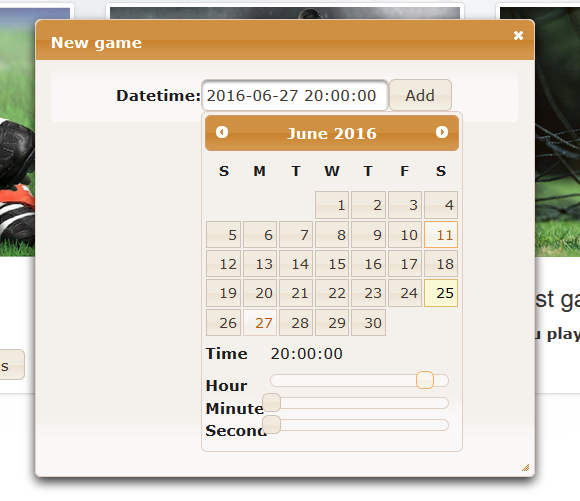
Figură 63. Opţiunile pentru un meci anterior

Se poate introduce numărul de goluri pentru fiecare echipă, se pot schimba numele echipelor, sau jucătorii între echipe, totate aceste schimbări persistându-se în baza de date, rating-ul jucătorilor fiind de asemenea recalculat în funcţie de diferenţa de goluri. În exemplul din Figura64, am schimbat numele celei de-a doua echipe şi am introdus numărul de goluri marcate de fiecare dintre echipe.



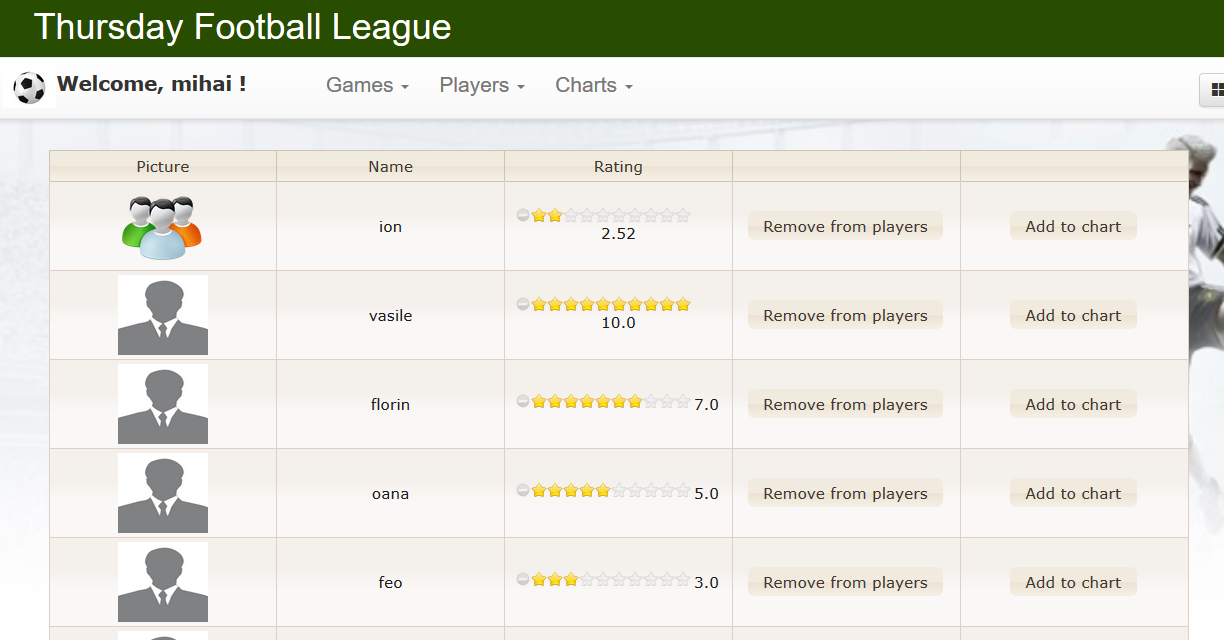
Figură 64. Exemplu introducere rezultat meci

Dacă se doreşte stabilirea datei pentru următorul meci de fotbal, din meniu se alege opţiunea ‘New game’, iar în fereastra apăruta se alege data şi ora jocului. În Figura65, am stabilit un joc pentru data de 27.06.2016 la ora 20:00.



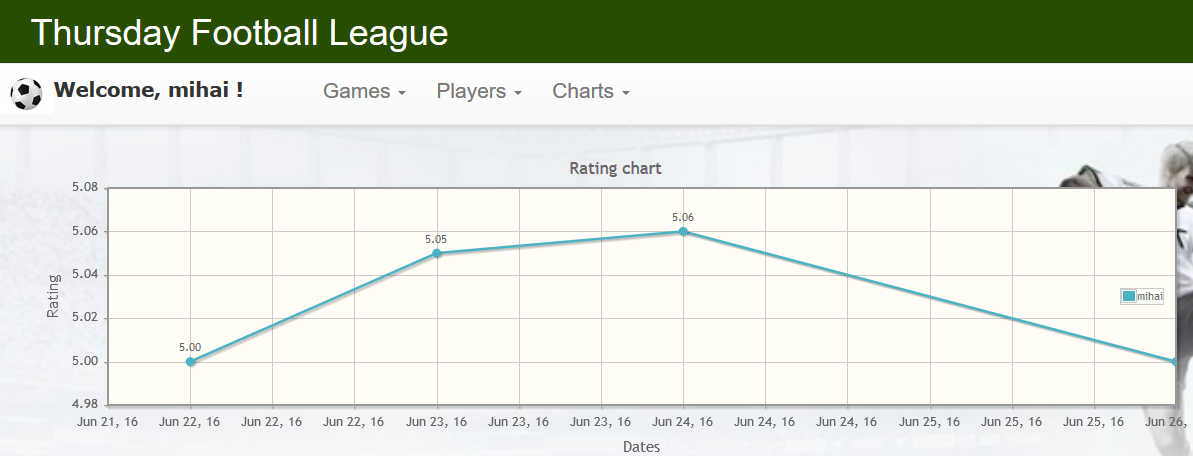
Figură 65. Stabilirea următorului meci de fotbal

Secţiunea destinată jucătorilor cuprinde două opţiuni: vizualizarea tuturor jucătorilor şi adăugarea unui nou jucător, disponibilă numai pentru administratori. Pentru a vedea jucătorii, se face click pe ‘View players’, utilizatorul fiind redirectat către pagina ce va conţine informaţii despre jucători.



Figură 66. Vizualizarea tuturor jucătorilor

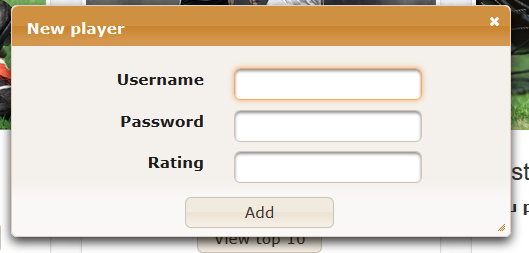
Pentru fiecare jucator, un user de tip admin poate efectua următoarele acţiuni, poate sterge (logic) din baza de date, jucătorul, sau îl poate adăuga la diagrama jucătorilor, pentru a vedea graficul evoluţiei acestuia. De exemplu, prin adăugarea jucătorului Mihai la diagrama, apoi accesarea diagramei din Charts->View chart, obţinem următorul grafic:



Figură 67. Exemplu grafic jucător

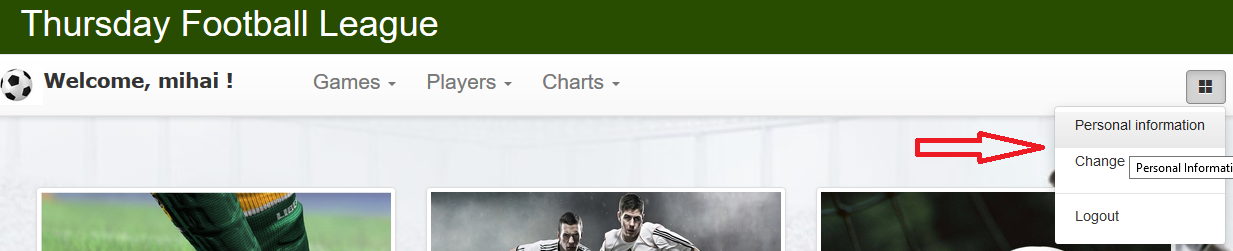
În diagramă se poate observa o evoluţie a rating-ului jucătorului Mihai, de la data de 22 iunie pană pe 24 iunie, apoi o scădere în data de 26 iunie.

Dacă se doreşte adăugarea unui nou jucător, după click pe tabul ‘Players’ din meniu şi alegerea opţiunii ‘Add new player’, se vor completa câmpurile: nume, parola şi rating. Câmpul rating nu este obligatoriu, în cazul în care este lăsat liber, utilizatorul înregistrat va avea rating-ul 5, rating-ul maxim fiind în jurul valorii 10.



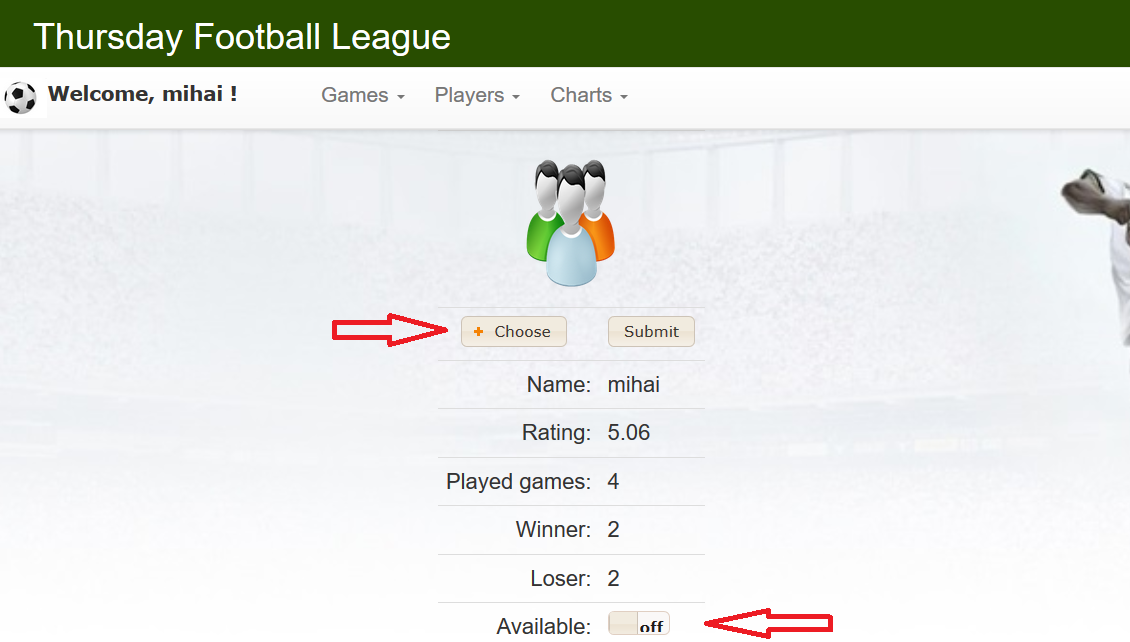
Figură 68. Exemplu adăugare jucător nou

Utilizatorii mai au opţiunile de a-şi schimba parola şi a-şi vizualiza şi modifica datele persoanle. Pentru a putea vizualiza datele sale personale, user-ul trebuie să aleagă din meniu, optiunea ‘Personal information’, precum in Figura69.



Figură 69. Accesarea informaţiilor personale

Informaţiile cuprinse în pagina încărcată sunt: numele jucătorului logat, rating-ul, numărul de jocuri jucate, numărul de jocuri câştigate, numărul de jocuri pierdute, precum şi disponibilitatea acestuia. De asemenea, tot din aceasta pagină, jucatorii işi pot schimba fotografia de profil, prin click pe butonul ‘Choose’. Userii pot schimba şi disponibilitatea acestora. Prin setarea opţiunii available ‘false’, un jucător nu se mai poate înscrie la meciuri, şi nu poate fi inclus în echipe de alţi utilizatori.



Figură 70. Schimbarea pozei de profil si a disponibiităţii

Dacă se doreşte schimbarea parolei curente, din meniu se alege ‘Change password’ şi se completează câmpurile din formular, precum în Figura71.



Figură 71. Schimbare parolă

În final, pentru ieşirea din aplicaţie se face click pe ‘Logout’, sesiunea curentă se închide şi se face redirectarea către pagina de start a aplicaţiei.

22

## Index de figuri

[Figură 1. Crerea unui proiect Maven 12](file:///C:\Users\luchi\Desktop\paula\Aplicaţie-suport-pentru-constituirea-de-echipe-echilibrate-şi-customizabile-2.docx#_Toc454793611)

[Figură 2. Conceptele Maven 12](#_Toc454793612)

[Figură 3. Dependinţe funcţionale incluse cu Maven 13](#_Toc454793613)

[Figură 4. Arhitectura MVC (Model-View-Controller) 16](file:///C:\Users\luchi\Desktop\paula\Aplicaţie-suport-pentru-constituirea-de-echipe-echilibrate-şi-customizabile-2.docx#_Toc454793614)

[Figură 5. Dependinţele JSF 18](file:///C:\Users\luchi\Desktop\paula\Aplicaţie-suport-pentru-constituirea-de-echipe-echilibrate-şi-customizabile-2.docx#_Toc454793615)

[Figură 6. Crearea unui ManagedBean 18](#_Toc454793616)

[Figură 7. Exemplu de folosire a JSF în HTML 19](file:///C:\Users\luchi\Desktop\paula\Aplicaţie-suport-pentru-constituirea-de-echipe-echilibrate-şi-customizabile-2.docx#_Toc454793617)

[Figură 8. Exemplu formular JSF 19](file:///C:\Users\luchi\Desktop\paula\Aplicaţie-suport-pentru-constituirea-de-echipe-echilibrate-şi-customizabile-2.docx#_Toc454793618)

[Figură 9. Adăugarea dependinţei pentru temele Primefaces 20](file:///C:\Users\luchi\Desktop\paula\Aplicaţie-suport-pentru-constituirea-de-echipe-echilibrate-şi-customizabile-2.docx#_Toc454793619)

[Figură 10. Tema 'Humanity' Primefaces 20](file:///C:\Users\luchi\Desktop\paula\Aplicaţie-suport-pentru-constituirea-de-echipe-echilibrate-şi-customizabile-2.docx#_Toc454793620)

[Figură 11. Primefaces Panel 21](file:///C:\Users\luchi\Desktop\paula\Aplicaţie-suport-pentru-constituirea-de-echipe-echilibrate-şi-customizabile-2.docx#_Toc454793621)

[Figură 12. Primefaces Autocomplete 22](file:///C:\Users\luchi\Desktop\paula\Aplicaţie-suport-pentru-constituirea-de-echipe-echilibrate-şi-customizabile-2.docx#_Toc454793622)

[Figură 13. Primefaces Chart 23](file:///C:\Users\luchi\Desktop\paula\Aplicaţie-suport-pentru-constituirea-de-echipe-echilibrate-şi-customizabile-2.docx#_Toc454793623)

[Figură 14. Primefaces PickList 23](file:///C:\Users\luchi\Desktop\paula\Aplicaţie-suport-pentru-constituirea-de-echipe-echilibrate-şi-customizabile-2.docx#_Toc454793624)

[Figură 15. Interfaţa pgAdmin 26](file:///C:\Users\luchi\Desktop\paula\Aplicaţie-suport-pentru-constituirea-de-echipe-echilibrate-şi-customizabile-2.docx#_Toc454793625)

[Figură 16.Arhitectura JPA 28](file:///C:\Users\luchi\Desktop\paula\Aplicaţie-suport-pentru-constituirea-de-echipe-echilibrate-şi-customizabile-2.docx#_Toc454793626)

[Figură 17.Clasa Player 36](#_Toc454793627)

[Figură 18. Clasa 'Game' 36](file:///C:\Users\luchi\Desktop\paula\Aplicaţie-suport-pentru-constituirea-de-echipe-echilibrate-şi-customizabile-2.docx#_Toc454793628)

[Figură 19. Pachetul 'DataAccessLayer' 37](#_Toc454793629)

[Figură 20. Clasa PlayerDataAccess 38](#_Toc454793630)

[Figură 21. Pachetul ‘Helpers’ 39](#_Toc454793631)

[Figură 22. Clasa RatingComparator 39](#_Toc454793632)

[Figură 23. Clasa PlayerConverter 40](#_Toc454793633)

[Figură 24. Sintaxa elementului Picklist cu atributul ‘converter’ 40](#_Toc454793634)

[Figură 25. Reprezentarea echipelor cu ajutorul elementului PickList 41](#_Toc454793635)

[Figură 26. Pachetul ‘views’ 42](#_Toc454793636)

[Figură 27. ManagedBean-ul AutoCompleteView 42](#_Toc454793637)

[Figură 28. Fereastra pentru adăugarea unui nou jucător 43](#_Toc454793638)

[Figură 29. Conţinutul folderului 'WebContent' 43](#_Toc454793639)

[Figură 30. Conţinutul folderului 'bootstrap' 44](#_Toc454793640)

[Figură 31. Baza de date a aplicatiei 46](file:///C:\Users\luchi\Desktop\paula\Aplicaţie-suport-pentru-constituirea-de-echipe-echilibrate-şi-customizabile-2.docx#_Toc454793641)

[Figură 32. Crearea conexiunii la baza de date PostgreSQL 47](file:///C:\Users\luchi\Desktop\paula\Aplicaţie-suport-pentru-constituirea-de-echipe-echilibrate-şi-customizabile-2.docx#_Toc454793642)

[Figură 33. Alegerea driver-ului pentru conexiune 47](file:///C:\Users\luchi\Desktop\paula\Aplicaţie-suport-pentru-constituirea-de-echipe-echilibrate-şi-customizabile-2.docx#_Toc454793643)

[Figură 34.Verificarea existenţei JPA în cadrul proiectului 48](file:///C:\Users\luchi\Desktop\paula\Aplicaţie-suport-pentru-constituirea-de-echipe-echilibrate-şi-customizabile-2.docx#_Toc454793644)

[Figură 35. Maparea entitatilor din baza de date 48](file:///C:\Users\luchi\Desktop\paula\Aplicaţie-suport-pentru-constituirea-de-echipe-echilibrate-şi-customizabile-2.docx#_Toc454793645)

[Figură 36.Clasele rezultate în urma mapării JPA a entităţilor 49](file:///C:\Users\luchi\Desktop\paula\Aplicaţie-suport-pentru-constituirea-de-echipe-echilibrate-şi-customizabile-2.docx#_Toc454793646)

[Figură 37. Formula generala combinari 49](#_Toc454793647)

[Figură 38 Lista jucatori pentru împărţirea ăn echipe 50](#_Toc454793648)

[Figură 39. Rezultatul împărţirii in echipe e listei 50](#_Toc454793649)

[Figură 40.Înscrierea unui jucător în cadrul unui meci 51](#_Toc454793650)

[Figură 41. Pornirea serverului Tomcat din mediul de dezvoltare Eclipse 52](#_Toc454793651)

[Figură 42. Pornirea serverului Tomcat din linia de comandă 53](#_Toc454793652)

[Figură 43. Verificarea pornirii corecte a serverului Tomcat 53](#_Toc454793653)

[Figură 44. Accesul la aplicatie prin intermediul unui browser web 54](#_Toc454793654)

[Figură 45. Înregistrarea unui nou jucător 54](#_Toc454793655)

[Figură 46. Exemplu autentificare cu date greşite 55](#_Toc454793656)

[Figură 47. Pagina către care este redirectat un utilizator după autentificare 55](#_Toc454793657)

[Figură 48. Prezentarea informaţiilor despre meciul următor 56](#_Toc454793658)

[Figură 49. Prezentarea informaţiilor despre ultimii jucători înscrişi 56](#_Toc454793659)

[Figură 50. Top 10 jucători 57](#_Toc454793660)

[Figură 51. Revenirea la pagina Home a aplicaţiei 57](#_Toc454793661)

[Figură 52. Prezentarea informaţiilor despre meciul anterior 58](#_Toc454793662)

[Figură 53. Prezentarea meniului pentru un utilizator de tip admin 58](#_Toc454793663)

[Figură 54. Prezentarea meniului pentru un utilizator fără drepturi de admin 59](#_Toc454793664)

[Figură 55. Opţiunilepentru meciurile anterioare 59](#_Toc454793665)

[Figură 56. Informarea utilizatorului cu privire la acţiunea solicitată 60](#_Toc454793666)

[Figură 57. Confirmarea anulării unui joc 60](#_Toc454793667)

[Figură 58. Vizualizarea meciului înainte de a se atinge numărul minim de jucători 61](#_Toc454793668)

[Figură 59. Echipele meciului după adăugarea unui nou jucător 61](#_Toc454793669)

[Figură 60. Interschimbarea de jucători în cadrul echipelor 62](#_Toc454793670)

[Figură 61. Salvarea configuraţiei curente 62](#_Toc454793671)

[Figură 62. Adăugarea de noi jucători in cadrul echipelor 63](#_Toc454793672)

[Figură 63. Opţiunile pentru un meci anterior 63](#_Toc454793673)

[Figură 64. Exemplu introducere rezultat meci 64](#_Toc454793674)

[Figură 65. Stabilirea următorului meci de fotbal 64](#_Toc454793675)

[Figură 66. Vizualizarea tuturor jucătorilor 65](#_Toc454793676)

[Figură 67. Exemplu grafic jucător 65](#_Toc454793677)

[Figură 68. Exemplu adăugare jucător nou 66](#_Toc454793678)

[Figură 69. Accesarea informaţiilor personale 66](#_Toc454793679)

[Figură 70. Schimbarea pozei de profil si a disponibiităţii 67](#_Toc454793680)

[Figură 71. Schimbare parolă 67](#_Toc454793681)

# Concluzii

În concluzie, Aplicaţia suport pentru constituirea de echipe echilibrate şi customizabile, are utilitate în lumea reală, putând fi folosită de grupul format din angajaţii unei companii, care doresc să se organizeze pentru a desfăsura activităti sportive, precum meciurile de fotbal. Reprezintă o modalitate interactivă si utila de organizare.

Ca şi perspective de viitor, se pot introduce mai multe criterii pe baza cărora să se facă împărţirea jucătorilor în cele două echipe. Un criteriu de care se mai poate ţine cont ar putea fi preferinţele jucătorilor pentru alţi jucători. Astfel, la formarea echipelor se va ţine cont de jucătorii pe care ar dori fiecare utilizator să îi aibă în echipă.

Pe o durată mai mare de timp, aplicaţia ar putea fi utilă chiar pentru organizare unor campionate de fotbal, participantii la campionate putând fi grupuri diferite de persoane.

# Bibliografie

* <http://en.wikipedia.org/wiki/JavaServer_Faces>
* <http://www.roseindia.net/jsf/>
* <http://profs.info.uaic.ro/~acf/tap/slides/jsf_slide.pdf>
* [http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller](http://en.wikipedia.org/wiki/Model–view–controller)
* [http://web.princeton.edu/sites/isapps/jasig/2004summerWestminster/Presentations/java%20ser ver%20faces.pdf](http://web.princeton.edu/sites/isapps/jasig/2004summerWestminster/Presentations/java%20ser%20ver%20faces.pdf)
* <http://www.primefaces.org/docs/guide/primefaces_user_guide_5_3.pdf>
* <https://en.wikipedia.org/wiki/Bootstrap_%28front-end_framework%29>
* <https://en.wikipedia.org/wiki/Java_Persistence_API>
* <http://www.tutorialspoint.com/maven/>
* <https://en.wikipedia.org/wiki/PostgreSQL>