Information System Project

Football Archive System

Task 2

Università di Pisa

Nicola Mota, Giacomo Pellicci, Bruk Tekalgne

April, 2020

1 Introduction 3

2 Design 3

2.1 Functional Requirements 3

2.2 Non Functional Requirements 5

2.3 User Use Cases 6

2.4 Administrator Use Cases 7

2.5 Analysis Classes 8

2.6 Software Architecture 8

3 Web Scraping 9

3.1 Obtaining data to populate the database 9

3.2 Scrapers 10

3.2.1 Java scraper for calcio.com 10

3.2.2 Python scraper for sofascore.com 16

4 Data Model 20

4.1 League collection 20

4.2 Player statistics collection 23

4.3 Team statistics collection 23

4.4 Account collection 24

5 Implementation 25

5.1 MongoDB indexes 25

5.1.1 Index 1 25

5.1.2 Index 2 25

5.1.3 Index 3 26

5.2 Replica setup 27

5.3 Analytics and Statistics 28

5.3.1 Top 20 scorers 28

5.3.2 Top 20 vote players 29

5.3.3 Player statistics (goals and assists) 30

5.3.4 Team statistics 31

5.4 Main software modules 33

5.5 Create 34

5.6 Read 34

5.7 Update 35

5.8 Delete 35

6 User manual 36

7 Admin manual 43

1 Introduction

A television company wants to add a new service to allow subscribers to enjoy the content of football matches related to the four major European championships: the Italian Serie A, the English Premier League, the Spanish La Liga and the German Bundesliga. In particular, the company requires the development of a service to store information relating to the last four seasons. For each season of a league the service must make available the results of the matches of each league day, the participating teams and the final ranking of the teams. For each team should be available the general information and the roster of players with their relative information (number, role, nationality and date of birth). For each match should be available the information on goals (author and assist-man), line-ups, cards and substitutions, the main statistics of the match and other secondary information such as the date and time of the match, the stadium, the number of spectators and the referee. Moreover, to provide support to people who play the fantasy football italian game, for each Serie A match should be available the votes of the players

For each season of a certain league the service offers some global statistics on teams and players.
In particular, for teams the service will provide rankings on ball possession, shots, shots on target and corners.
For players the service must provide rankings for the best 20 players for a number of goals, assists, yellow or red cards. For the Italian league the service must provide also rankings for the best 20 players and for the 20 worst players for average vote.

In addition, the system should allow users to compare two players of the same league season

Reference sites: *calcio.com, sofascore.com, fantacalcio.it*

2 Design

2.1 Functional Requirements

- The system shall allow users to sign-up providing username, password and their favourite league
- The system shall allow users to log-in providing username and password
- The system shall allow users to browse the Serie A, Premier League, La Liga and Bundesliga leagues
- The system shall allow users to browse the past four league seasons
- The system shall allow users to browse the league rounds played
- The system shall allow users to browse the league matches played
- A match shall include the date, home team, away team and the final result
- The system shall allow users to browse the match details
    - The match details shall include the teams lineups, goals, cards, substitutions,match statistics and player votes only for the Italian league and other secondary information such as the stadium, the number of spectators and the referee
- The system shall allow users to browse the team ranking
- The system shall allow users to browse the league teams
- The system shall allow users to browse the team infos

- The system shall allow users to browse the team rosters
- The system shall allow users to browse team statistics like:
    - Average ball possession
    - Average shots
    - Average shots on target
    - Average shots out
    - Average blocked shots
    - Average corners
    - Average offsides
    - Average fouls
    - Average yellow cards
    - Average red cards
    - Average shot from penalty area
    - Average shots from outside penalty area
    - Average goalkeeper saves
    - Average passages
    - Average passage precision
    - Average tackles
    - Average air tackles won

- The system shall allow users to browse
    - The ranking of the 20 players with the highest average rating only for the Italian league
    - The ranking of the 20 players with the lowest average rating only for the Italian league
    - The ranking of the 20 players with the most goals
    - The ranking of the 20 players with the most assists
    - The ranking of the 20 players with the most yellow cards
    - The ranking of the 20 players with the most red cards

- The system shall allow users to compute player statistics based on:
    - matches played
    - starting games
    - minutes played
    - goalt
    - assist
    - yellow cards
    - red cards
    - assist
    - highest vote
    - lowest vote
    - average vote

- The system shall allow users to browse team ranking based on
    - Average ball possession
    - Average shots
    - Average shots on target

    - Average corners

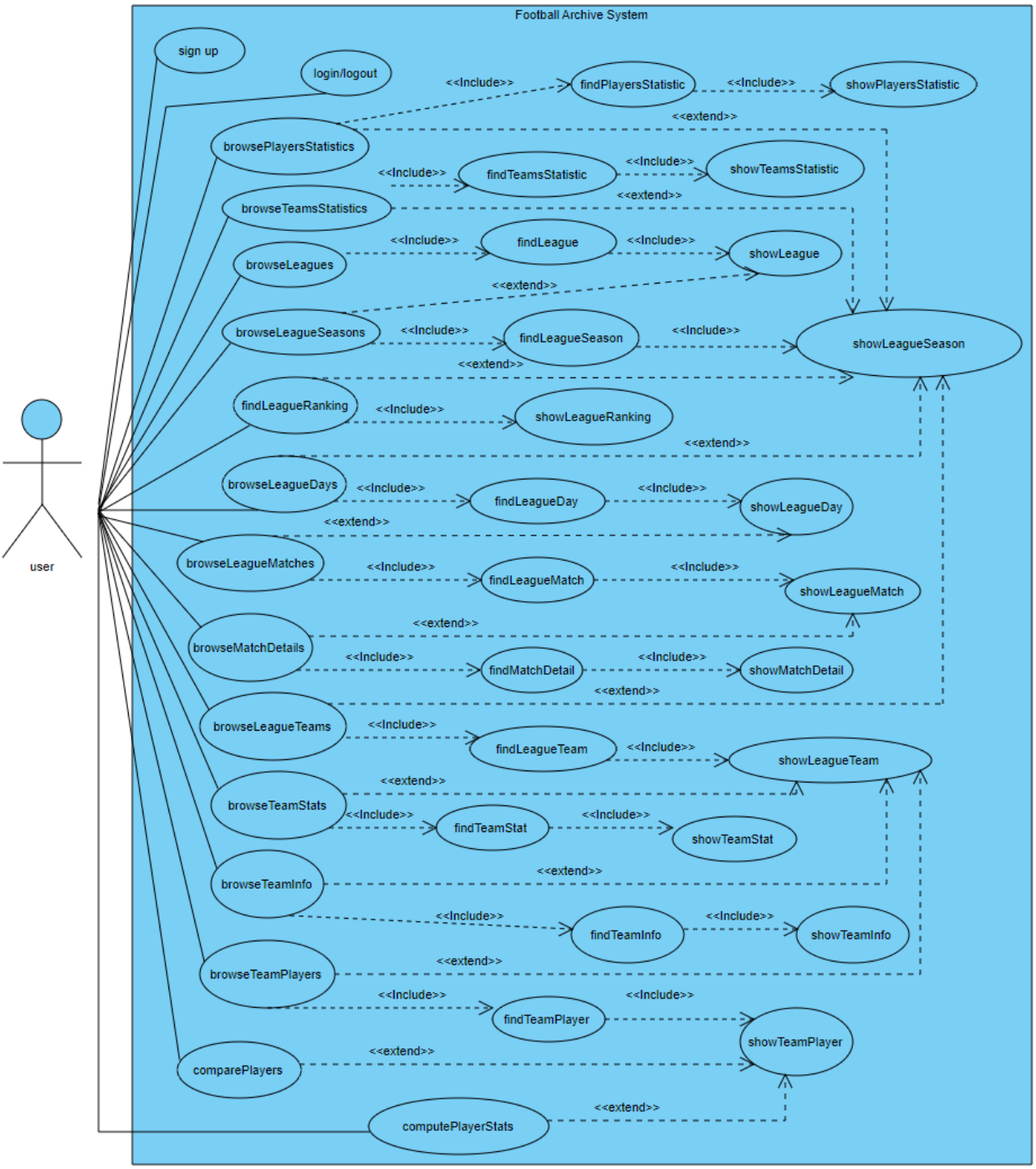
- The system shall allow users to compare two players using their statistics

- The system shall allow the administrator to log-in providing username and password
- The system shall allow the administrator to insert a new league season
- The system shall allow the administrator to insert a new league round
- The system shall allow the administrator to see the list of the top 10 most active users
- The system shall allow the administrator to see the list of the top 20 most searched players for a given season in a specific league
- The system shall allow the administrator to remove a user account

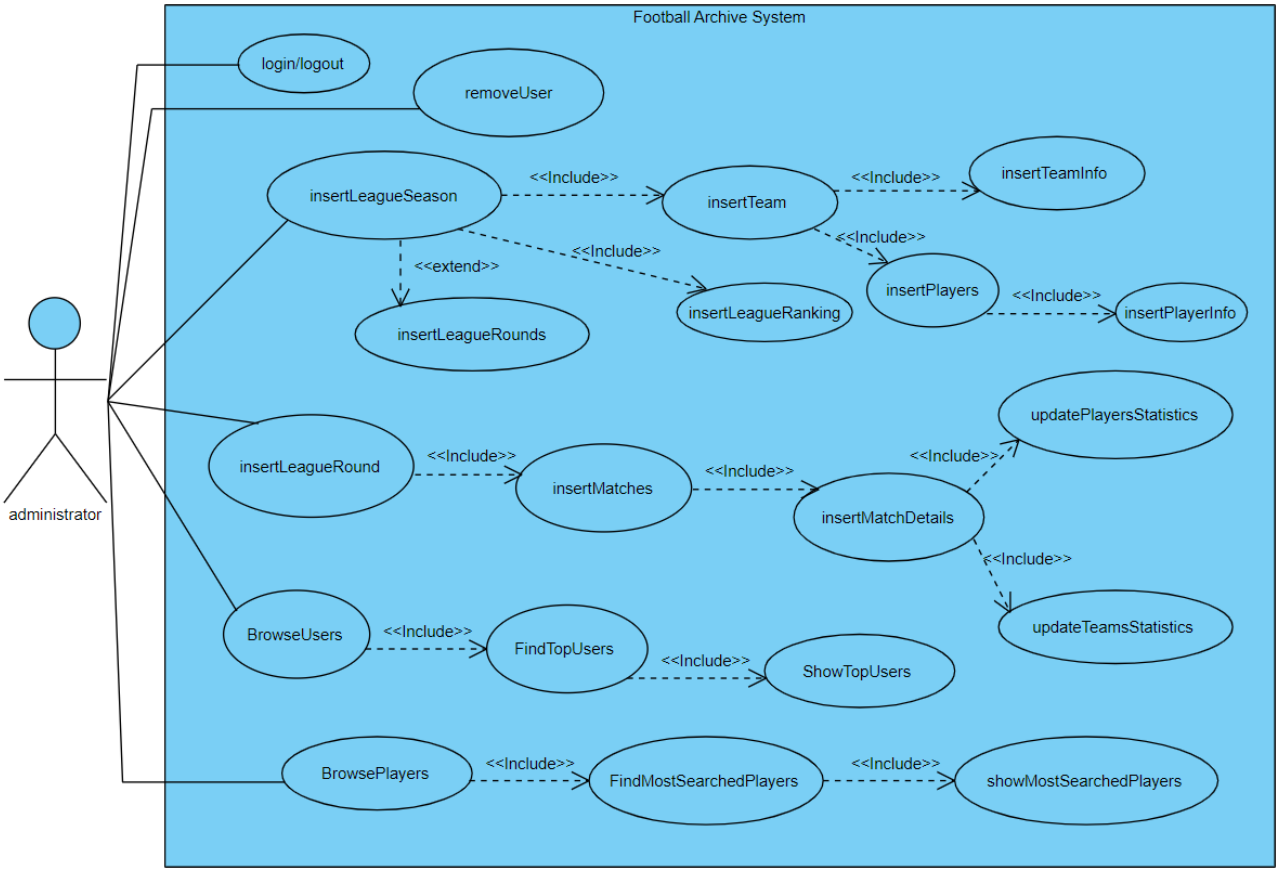
2.2 Non Functional Requirements

- The system shall allow short response time
- The system shall guarantee a high level of usability: users shall interact with the application using a graphical interface
- The system shall allow high availability: the service is always guaranteed using replicas
- Eventual consistency: for our application is important to providing a response to any query in any time. So, we decide to use an eventual consistency model

2.3 User Use Cases



2.4 Administrator Use Cases

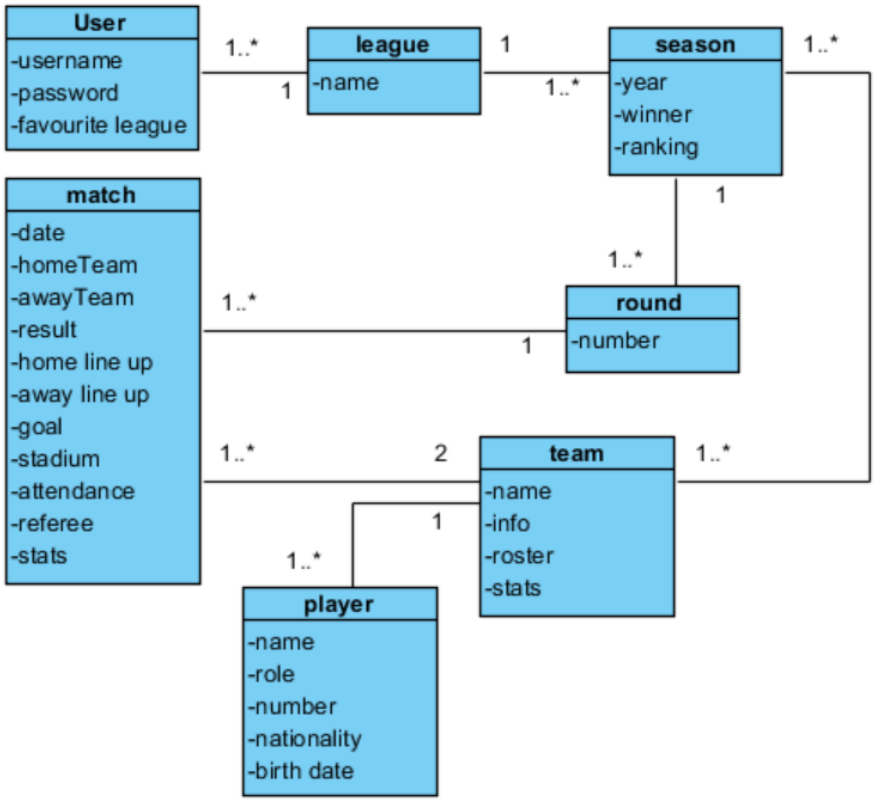


The *insertLeagueSeason* use case allows the administrator to insert:

- a past season, already ended, with all its information including all the matches played;
- a new season, before it starts, with all available information, such as which teams will participate, the players of each team etc. obviously excluding matches, as they have yet to be played.

The *insertLeagueRound* use case allows the administrator to insert a round of matches played. For this reason, the statistics of the teams and players must be updated.

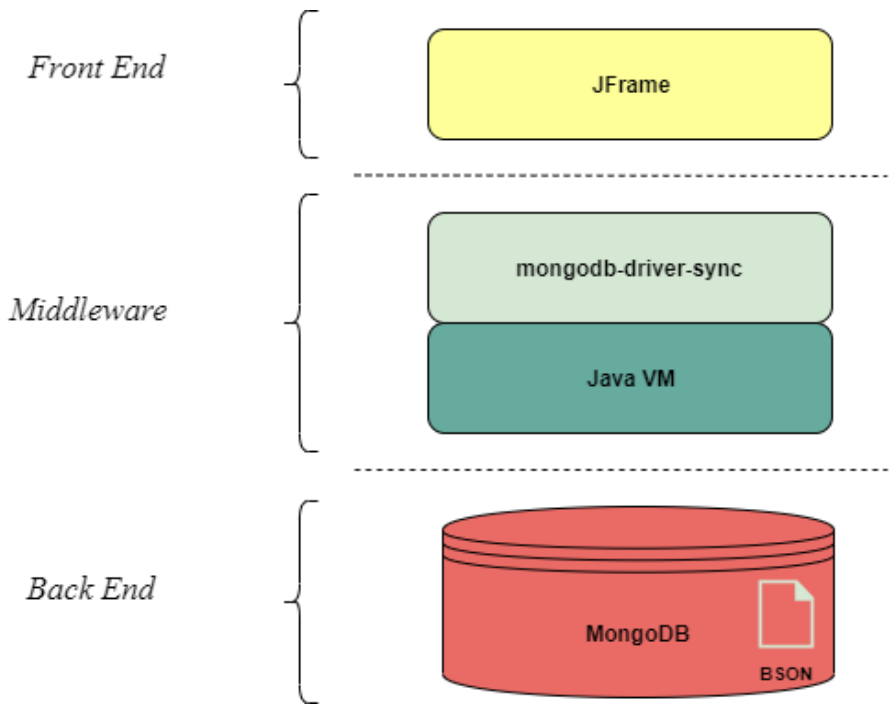
2.5 Analysis Classes



2.6 Software Architecture

The software is organized in a classical way: front end, middleware, back end.

In the front end, the system provides a GUI developed with JFrame for an easier interaction with users and administrators. In the middleware we handle all operation using Java and we use the MongoDB Java Driver to manipulate data stored in the MongoDB database. Finally, in the back end, mongoDB is in charge of handling our documents, stored in BSON format.



3 Web Scraping

3.1 Obtaining data to populate the database

The data to populate the database are retrieved from the sites mentioned above using a scraper written in Java that uses the JSOUP library and a scraper written in Python that uses the Selenium library. The Selenium library allows to manage the more complex web pages that use javascript and make AJAX requests.

In particular the website http://www.calcio.com/ was scraped using JSOUP to retrieve basic information about leagues, teams and matches.

The website https://www.sofascore.com/it/ which is much more complex, as it uses Javascript and AJAX requests to deliver its services, has been scraped using Selenium. From this website we retrieved match statistics.

Also for the italian league we added support for the so called "Fantacalcio" importing csv data from https://www.fantacalcio.it/ .

The scrapers can be found here:

1. https://github.com/gpellicci/calciocom-scraper
2. https://github.com/gpellicci/scraper-py

To obtain a usable json file, you should run in this order:

1. Run Java scraper for calcio.com

2. Run Python scraper for sofascore.com with scrapeV2.py
3. For “Serie A” you can also add the votes of the player from CSV using voteFromCSV.py
4. Do some post-processing to adjust the dataset, i.e change string types in integer. with changeType.py

3.2 Scrapers

3.2.1 Java scraper for *calcio.com*

This scraper is implemented exploiting JSOUP

```
public static void scrapeCampionato(String campionato, String filename) throws IOException{
    //PARAMETERS
    int MAX_GIORNATE = 38;
    final int STARTING_YEAR = 2015;
    final int ACTUAL_YEAR = 2019;
    String championship = campionato;
    System.out.println("Scraping "+championship);
    //String championship = "ita-serie-a";
    //String championship = "eng-premier-league";
    //END OF PARAMETERS


    List<String> months = new ArrayList<>(Arrays.asList("gennaio", "febbraio", "marzo", "aprile", "maggio", "giugno", "luglio", "agosto
    String base = "http://www.calcio.com";
    String year = "";
    JSONObject league = new JSONObject();
    for(int y = STARTING_YEAR; y < ACTUAL_YEAR; y++){
        year = y + "-" + (y+1) + "/";
        System.out.println("-- Getting season " + year);

        String connectTo = "http://www.calcio.com/giocatori/" + championship + "-" + year;
        if(connectTo.equals("http://www.calcio.com/giocatori/esp-primera-division-2016-2017/"))
            connectTo = "http://www.calcio.com/giocatori/esp-primera-division-2016-2017_2/";
        Document doc = Jsoup.connect(connectTo).get();
        Elements e = doc.getElementsByClass("standard_tabelle");
        Element el = e.get(0);
        Elements teams = el.getElementsByTag("tr");
        JSONObject season = new JSONObject();
        doc = Jsoup.connect("http://www.calcio.com/storia/" + championship).get();
        Elements seasonRows = doc.getElementsByClass("standard_tabelle").get(0).getElementsByTag("tr");
        for(Element s : seasonRows){
            if(s.text().contains(STARTING_YEAR + "/" + (STARTING_YEAR+1))){
                String winner = s.getElementsByTag("td").get(5).text();
                season.put("winner", normalizeText(winner));
                break;
            }
        }
        for (Element t : teams) {
            JSONObject team = new JSONObject();
            String link = t.getElementsByTag("td").get(3).getElementsByTag("a").attr("href");
            doc = Jsoup.connect(base + link).get();
            Elements infos = doc.getElementsByClass("standard_tabelle").get(0).getElementsByTag("tr");
            //GET INFO
            JSONObject info = new JSONObject();
            for (Element row : infos) {
                Elements tds = row.getElementsByTag("td");
                String key = camelCase(tds.get(0).text()).replace(":", "");
                if(key.equals("stadio")){
                    Elements els = tds.get(1).getAllElements();
```

```
            info.put(key, normalizeText(normalizeText(els.get(1).text())));
            info.put("capacity", normalizeText(tds.get(1).text().replace(els.get(1).text() + " ", "")));
        }
        else{
            info.put(key, normalizeText(tds.get(1).text().replace("\n", "")));
        }
    }

    //GET ROSA
    link = t.getElementsByTag("td").get(5).getElementsByTag("a").attr("href");
    doc = Jsoup.connect(base + link).get();
    Elements rows = doc.getElementsByClass("standard_tabelle").get(0).getElementsByTag("tr");
    String role = "";
    JSONObject roster = new JSONObject();
    for(Element row : rows){
        JSONObject player = new JSONObject();
        Elements tds = row.getElementsByTag("td");
        if(row.text().equals("Portiere") || row.text().equals("Difesa") || row.text().equals("Centrocampo") || row.text().equal
            role = row.text();
        }
        else{
            if(role.equals("Preparatore dei portieri"))
                break;
            if(role.equals("Allenatore in seconda"))
                role = "Allenatore";
            player.put("role", role);
            String number = tds.get(1).text();
            if(!number.equals(""))
                player.put("number", number);
            player.put("name", normalizeText(tds.get(2).text()));
            player.put("nationality", tds.get(4).text());
            String birthSplit[] = tds.get(5).text().split("\\.");
            if(birthSplit.length > 1){
                String birth = birthSplit[2] + "-" + birthSplit[1] + "-" + birthSplit[0] + "T" + "00:00:00:000+01:00";
                player.put("birth", birth);
            }
            roster.append("player", player);

            if(role.equals("Allenatore"))
                break;
            }

    }
```

```
    team.append("roster", roster);
    team.append("info", info);

    season.append("team", team);
}

//GET RISULTATI
String linkRisultati = base + "/calendario/" + championship + "-" + year.replace("/", "") + "-spieltag/";
if(linkRisultati.equals("http://www.calcio.com/calendario/esp-primera-division-2016-2017-spieltag/"))
    linkRisultati = "http://www.calcio.com/calendario/esp-primera-division-2016-2017-spieltag_2/";
for (int i = 1; i <= MAX_GIORNATE; i++) {
    JSONObject round = new JSONObject();
    System.out.println("   -- Risultati giornata " + i);
    doc = Jsoup.connect(linkRisultati + i).get();
    //ON FIRST ROUND -> GET NUMBER OF ROUNDS
    if(i == 1){
        Elements options = doc.getElementsByAttributeValueContaining("name", "runde").get(0).getAllElements();
        options.remove(0);
        MAX_GIORNATE = options.size();
    }
    //ON LAST ROUND -> GET RANKINGS
    if(i == MAX_GIORNATE){
        Elements colorExplainations = doc.getElementsByClass("data").get(4).getElementsByTag("table").get(1).getElementsByTag("
        List<JSONObject> colors = new ArrayList<>();
        for(Element row : colorExplainations){
            Elements tds = row.getElementsByTag("td");
            JSONObject colorExplaination = new JSONObject();
            colorExplaination.put("color", tds.get(0).attr("bgcolor"));
            colorExplaination.put("result", tds.get(1).text());
            colors.add(colorExplaination);
        }

        JSONObject rankings = new JSONObject();
        Elements ranks = doc.getElementsByClass("standard_tabelle").get(1).getElementsByTag("tr");
        ranks.remove(0);
        for(Element rank : ranks){
            JSONObject team = new JSONObject();
            Elements tds = rank.getElementsByTag("td");
            team.put("name", normalizeText(tds.get(2).text()));
            if(!tds.get(2).attr("bgcolor").equals("FFFFFF")){
                String target = tds.get(2).attr("bgcolor");
                for(JSONObject c : colors){
                    if(c.get("color").equals(target)){
                        team.put("award", c.get("result"));
                        break;
```

```
                }
            }
        }
        team.put("played", tds.get(3).text());
        team.put("won", tds.get(4).text());
        team.put("draw", tds.get(5).text());
        team.put("loss", tds.get(6).text());
        String[] goals = tds.get(7).text().split(":");
        team.put("goalScored", goals[0]);
        team.put("goalAllowed", goals[1]);
        team.put("points", tds.get(9).text());

        rankings.append("team", team);
    }
    season.put("ranking", rankings);
}
//DONE WITH RANKINGS, GET RISULTATI
Elements matches = doc.getElementsByClass("standard_tabelle").get(0).getElementsByTag("tr");
for (Element m : matches) {
    JSONObject match = new JSONObject();
    String linkMatch = m.getElementsByTag("td").get(5).getElementsByTag("a").get(0).attr("href");
    doc = Jsoup.connect(base + linkMatch).get();
    Elements tables = doc.getElementsByClass("standard_tabelle");
    //get match teams + date + result
    Elements headers = tables.get(0).getElementsByTag("th");
    match.put("homeTeam", normalizeText(headers.get(0).text()));
    match.put("awayTeam", normalizeText(headers.get(2).text()));
    String matchDate = headers.get(1).text();
    matchDate = matchDate.split(", ")[1];
    matchDate = matchDate.replace(".", "");
    String[] matchDates = matchDate.split(" ");
    matchDate = matchDates[2]+"-"+months.indexOf(matchDates[1])+"-"+matchDates[0]+"T"+matchDates[3]+":00:000+01:00";
    match.put("date", matchDate);
    String[] result = tables.get(0).getElementsByTag("td").get(1).text().split(":");
    match.put("homeResult", result[0]);
    match.put("awayResult", result[1]);
    //get match history (goals)
    Elements rows = tables.get(1).getElementsByTag("tr");
    rows.remove(0);
    for(Element row : rows){
        if(row.text().equals("Nessuno"))
            break;
        JSONObject goal = new JSONObject();
        Elements tds = row.getElementsByTag("td");
        String[] partial = tds.get(0).text().split(":");
        goal.put("homePartial", partial[0]);
        goal.put("awayPartial", partial[1]);
        String scorer = tds.get(1).text();
```

```
            String assist = "";
            String kind = "";
            String[] infos = scorer.split("\\s\\d+.");
            scorer = infos[0];
            if(infos.length > 1){
                infos[1] = infos[1].replace("\\s\\/\\s", "");
                infos = infos[1].split("\\s\\(");
                if(infos.length > 1)
                    assist = infos[1].replace(")", "");
                kind = infos[0].replace(" / ", "");
            }
            goal.put("scorer", normalizeText(scorer));
            goal.put("assist", normalizeText(assist));
            goal.put("kind", kind);
            String minute = tds.get(1).text().replace(scorer+" ", "");
            if(!kind.equals("")){
                minute = minute.replace(kind, "");
            }
            if(!assist.equals("")){
                assist = "(" + assist + ")";
                minute = minute.replace(assist, "");
            }
            minute = minute.replace(".", "").replace("/", "").replace(" ", "");
            goal.put("minute", minute);

            match.append("goal", goal);
        }
        int index = 2;
        if(tables.get(index).getElementsByTag("tr").get(0).text().equals("avvenimenti particolari"))
            index++;

        //get formazioni home
        rows = tables.get(index).getElementsByTag("tr");
        Boolean titolare = true;
        JSONObject roster = new JSONObject();
        for(Element row : rows){
            JSONObject player = new JSONObject();
            Elements tds = row.getElementsByTag("td");
            if(tds.size() == 1){
                if(tds.get(0).text().equals("Giocatori di riserva"))
                    titolare = false;
            }
            else{
                player.put("number", tds.get(0).text());
                Elements portions = tds.get(1).getAllElements();
                portions.remove(0);
                player.put("name", normalizeText(portions.get(0).text()));
                if(portions.size() > 1){
                    JSONObject event = new JSONObject();
```

```
                String what = portions.get(1).attr("alt");
                event.put("event", what);
                if(portions.size() == 3){
                    String when = portions.get(2).text().replace("'", "");
                    event.put("time", when);
                }
                player.put("event", event);
            }
            if(!tds.get(2).text().equals("")){
                if(titolare)
                    player.put("leaveTime", tds.get(2).text().replace("'", ""));
                else
                    player.put("enterTime", tds.get(2).text().replace("'", ""));
            }
            player.put("starter", titolare);    //titolare
            roster.append("player", player);
        }
    }
    match.put("homeLineup", roster);
    index++;
```

```
            //get coach
            rows = tables.get(index).getElementsByTag("th");
            match.put("homeCoach", normalizeText(rows.get(0).text().replace("Allenatore:\\s*", "")));
            match.put("awayCoach", normalizeText(rows.get(1).text().replace("Allenatore:\\s*", "")));
            index++;

            //get informations
            rows = tables.get(index).getElementsByTag("tr");
            //System.out.println(match.get("homeTeam")+ " - "+ match.get("awayTeam"));
            match.put("stadium", normalizeText(rows.get(0).getElementsByTag("td").get(2).text()));
            match.put("attendance", rows.get(1).getElementsByTag("td").get(2).text().replace(".", ""));
            match.put("referee", normalizeText(rows.get(2).getElementsByTag("td").get(2).text()));

            round.append("match", match);
        }
        season.append("round", round);
        season.put("year", year.replace("/", ""));
    }
    league.put("name", championship);
    league.append("season", season);
}

//PrintWriter writer = new PrintWriter("provafile.txt", "UTF-8");
PrintWriter writer = new PrintWriter(filename+".txt", "UTF-8");
writer.println(league.toString());
writer.close();
}
```

3.2.2 Python scraper for sofascore.com

This scraper is implemented in Python using chromedriver in Selenium

```
import json
import os
import time
import random
import unidecode
from selenium import webdriver
from selenium.webdriver.support.wait import WebDriverWait

def time_wait():
    x = random.uniform(0.4, 0.6)
    time.sleep(x)


def wait_for_ajax(driver):
    wait = WebDriverWait(driver, 15)
    try:
        wait.until(lambda driver: driver.execute_script('return jQuery.active') == 0)
        wait.until(lambda driver: driver.execute_script('return document.readyState') == 'complete')
    except Exception as e:
        pass

files = ["ita-serie-a", "ger-bundesliga", "eng-premier-league", "fra-ligue-1", "esp-primera-division"]
links = ["https://www.sofascore.com/it/torneo/calcio/italy/serie-a/23", "https://www.sofascore.com/it/torneo/calcio/germany/bundesliga/35",
months = ["Gen", "Feb", "Mar", "Apr", "Mag", "Giu", "Lug", "Ago", "Set", "Ott", "Nov", "Dic"]
months_num = ["01", "02", "03", "04", "05", "06", "07", "08", "09", "10", "11", "12"]
STARTING_YEAR = 15  #2015
ACTUAL_YEAR = 19    #2019


linkIndex = 1
url = links[linkIndex]          #pick links[i] to choose the season
filename = files[linkIndex]     #json filename

webdriverPath = "./chromedriver"
if os.name == 'nt':
    webdriverPath += ".exe"
browser = webdriver.Chrome(webdriverPath)
browser.maximize_window()
start_time = time.time()
browser.get(url)
```

```python
# get on the desired season
wait_for_ajax(browser)
dropdown = browser.find_element_by_class_name("styles__MenuWrapper-cdd802-1")
ul = dropdown.find_element_by_class_name("styles__Menu-cdd802-5")
li = ul.find_elements_by_tag_name("li")
dropdown.click()
#loop through the seasons
li_text = []
for l in li:
    li_text.append(l.text)
    print(l.text)

index = li_text.index(str(STARTING_YEAR) + "/" + str(STARTING_YEAR + 1))

for p in range(0, index):
    k = index - p       #4-0 = 4  ----> 4-3 = 1
    print("season " + li_text[k])

    # get on the desired season
    time_wait()
    browser.get(url)
    wait_for_ajax(browser)
    dropdown = browser.find_element_by_class_name("styles__MenuWrapper-cdd802-1")
    browser.execute_script("arguments[0].scrollIntoView(true);", dropdown)
    ul = dropdown.find_element_by_class_name("styles__Menu-cdd802-5")
    li = ul.find_elements_by_tag_name("li")

    dropdown.click()
    ##
    #li[i] which season?
    li[k].click()
    print("Clicked on season " + dropdown.text)

    #select per-turn view
    wait_for_ajax(browser)
    time_wait()
    radioButton = browser.find_elements_by_class_name("Tabs__Header-vifb7j-0")[1].find_elements_by_class_name("Label-sc-19k9vkh-0")[1]
    browser.execute_script("arguments[0].scrollIntoView(true);", radioButton)
    time_wait()
    radioButton.click()
    wait_for_ajax(browser)
```

```
    #select turn 1
    time_wait()
    resultDiv = browser.find_element_by_class_name("u-mV12")
    browser.execute_script("arguments[0].scrollIntoView(true);", resultDiv)
    wait_for_ajax(browser)
    roundButton = resultDiv.find_element_by_class_name("styles__MenuWrapper-cdd802-1")
    roundButton.click()
    wait_for_ajax(browser)
    time_wait()
    round_li = roundButton.find_elements_by_tag_name("li")
    round_li[0].click()
    time_wait()
    print("Composed of " + str(len(round_li)) + " rounds")
    season = {
        "round" : []
    }
    #loop through the rounds of the season
    for i in range(0, len(round_li)):
        wait_for_ajax(browser)
        f = open("data/" + filename + "_filled.txt", "r")
        json_championship = json.load(f)
        f.close()
        round = {
            "match" : []
        }

        resultDiv = browser.find_element_by_class_name("u-mV12")
        roundButton = resultDiv.find_element_by_class_name("styles__MenuWrapper-cdd802-1")
        roundButton.click()
        wait_for_ajax(browser)
        browser.execute_script("arguments[0].scrollIntoView(true);", resultDiv)
        wait_for_ajax(browser)
        round_li = roundButton.find_elements_by_tag_name("li")
        browser.execute_script("arguments[0].scrollIntoView(true);", round_li[i])
        round_li[i].click()
        wait_for_ajax(browser)
        time_wait()
        browser.execute_script("arguments[0].scrollIntoView(true);", resultDiv)     ##
        # loop through the matches to scrape data
        matchDiv = resultDiv.find_element_by_class_name("styles__EventListContent-b3g57w-2")
        match_li = matchDiv.find_elements_by_class_name("EventCellstyles__Link-sc-1m83enb-0")
        print("This round is composed of " + str(len(match_li)) + " matches")

        #loop through the matches of that week
        match_count = 0
        for j in range(0, len(match_li)):
            time_wait()
            # if j > 0:
            #     time.sleep(3)
```

```
            print("---MATCH #"+ str(j))
            matchStatus = match_li[j].find_element_by_class_name("Cell-decync-0").find_element_by_class_name("Section-sc-1a7xrsb-0").get_at

            #if match is not over (delayed, not yet played and so on)
            if "FIN" not in matchStatus:
                continue

            match_li[j].click()
            time_wait()
            wait_for_ajax(browser)
            homeTeamChecked = unidecode.unidecode(str(match_li[j].find_element_by_class_name("EventCellstyles__WinIndicator-ni00fg-3").text
```

```
            matchInfo = {
                "statisticHome" : [],
                "statisticAway": []
            }

            wait_for_ajax(browser)
            time_wait()
            time_wait()
            tmp = browser.find_element_by_class_name("u-mV12").find_elements_by_class_name("Label-sc-19k9vkh-0")
            # print(str(len(tmp)))
            for t in tmp:
                if t.text == "STATISTICHE":
                    browser.execute_script("arguments[0].scrollIntoView(true);", t)
                    t.click()
                    continue
            time_wait()
            wait_for_ajax(browser)
            statDiv = browser.find_element_by_xpath("/html/body/div[1]/main/div/div[2]/div/div[1]/div[4]/div/div[2]/div/div/div/div[2]/div/
            statss = statDiv.find_elements_by_class_name("Cell-decync-0")
            for s in statss:
                browser.execute_script("arguments[0].scrollIntoView(true);", s)
                portions = s.find_elements_by_class_name("Section-sc-1a7xrsb-0")
                # [0] home stat
                # [1] stat name
                # [2] away stat
                statName = portions[1].text
                homeStat = str(portions[0].text)
                stat = {
                    statName: homeStat
                }
                matchInfo['statisticHome'].append(stat)
                awayStat = str(portions[2].text)
                stat = {
                    statName: awayStat
                }
                matchInfo['statisticAway'].append(stat)

            browser.execute_script("arguments[0].scrollIntoView(true);", resultDiv)

            #add the match statistics to the json
            print("season "+str(p)+" round "+str(i)+" match "+str(match_count))
            json_championship["season"][p]["round"][i]["match"][index_match]["statisticHome"] = matchInfo["statisticHome"]
            json_championship["season"][p]["round"][i]["match"][index_match]["statisticAway"] = matchInfo["statisticAway"]
```

```
            time_wait()
            #scroll back to top of window
            browser.execute_script("arguments[0].scrollIntoView(true);", resultDiv)

            match_count = match_count + 1



        json_data = json.dumps(json_championship)
        f = open("data/" + filename + "_filled.txt", "w+")
        f.write(json_data)
        f.close()

print("------ %s seconds ------" % (time.time() - start_time))
exit(0)
```

4 Data Model

As result of the scraping phase, we obtained the documents that compose the league collection.

The approach is to have nested documents so that we can infer information about matches while traversing the nested documents.

The other collections are used for account and statistics handling.

4.1 League collection

One collection includes the League documents. Each document in this collection will represent a league and it contains all the seasons present for that league. Within each season there are the various matches played that year, the ranking, the teams and their players and other information shown below.

Here, we can see an example to understand the structure of the documents in this collection:

League

```
  _id: ObjectId("5ea213e2dd1316b4df25487c")
  name: "Serie A"
v season: Array
  > 0: Object
  > 1: Object
  > 2: Object
  > 3: Object
```

Season

```
v season: Array
  v 0: Object
      winner: "Juventus"
    > round: Array
      year: "2015-2016"
    > ranking: Object
    > team: Array
```

Round

It's an array where each element represents a round and contains all the matches played in that round

```
v round: Array
  v 0: Object
    > match: Array
```

Match

```
match: Array
  0: Object
    date: "2015-7-22T18:00:00:000+01:00"
    goal: Array
    awayTeam: "Roma"
    referee: "Marco Guida (Italia)"
    homeLineup: Object
    awayLineup: Object
    homeCoach: "Allenatore: Andrea Mandorlini"
    awayResult: 1
    homeResult: 1
    awayCoach: "Allenatore: Rudi Garcia"
    homeTeam: "Hellas Verona"
    stadium: "Marc Antonio Bentegodi (Verona / Italia)"
    attendance: 22075
    statisticHome: Array
    statisticAway: Array
```

Goal

```
goal: Array
  0: Object
    homePartial: 1
    scorer: "Bosko Jankovic"
    kind: "Tiro di destro"
    assist: "Emil Hallfresson"
    awayPartial: 0
    minute: 61
  1: Object
    homePartial: 1
    scorer: "Alessandro Florenzi"
    kind: "Tiro di destro"
    assist: "Edin Dzeko"
    awayPartial: 1
    minute: 66
```

Lineups

Both for the home team and the away team there is an array with the players on the field and on the bench. Events such as a card or a substitution are also reported for each player

```
homeLineup: Object
  player: Array
    0: Object
    1: Object
    2: Object
    3: Object
    4: Object
      leaveTime: 79
      number: 69
      starter: true
      name: "Samuel Souprayen"
      event: Object
        time: "33"
        event: "giallo"
      vote: 6.5
```

Stats

Both for the home team and the away team there is an array that contains all the game stats

```
statisticHome: Array
  0: Object
      PossessoPalla: 32
  1: Object
      TiriTotali: 12
  2: Object
      TiriInPorta: 7
  3: Object
  4: Object
```

Ranking

It contains an array of teams sorted by their ranking and for each team the following fields are shown:

```
ranking: Object
  team: Array
    0: Object
        goalScored: 75
        loss: 5
        award: "Champions League"
        won: 29
        name: "Juventus"
        goalAllowed: 20
        draw: 4
        played: 38
        points: 91
```

Team

Each element of this array represents a team that participates at that league during that season

```
team: Array
  0: Object
    roster: Array
    info: Array
  1: Object
  2: Object
```

Roster

It's an array with the players of that team and for each player the following fields are reported:

```
0: Object
    number: 30
    role: "Portiere"
    nationality: "Italia"
    name: "Davide Bassi"
    birth: "1985-04-12T00:00:00:000+01:00"
1: Object
```

Info

```
paese: "Italia"
squadra: "Atalanta"
stadio: "Gewiss Stadium"
colori: "blu-nero"
indirizzo: "Atalanta Bergamasca Calcio Zingonia - Corso Europa 24040 Bergamo"
e-mail: "info@atalanta.it"
telefono: "(+39) 35 / 41 86 211"
fax: "(+39) 35 / 41 86 247"
nomeCompleto: "Atalanta Bergamasca Calcio 1907"
fondato: "01.01.1907"
capacity: "26.562 Posti"
homepage: "http://www.atalanta.it"
```

4.2 Player statistics collection

In this collection we keep the computed player statistics, that are limited to avoid excessive load of the DB. For each league-year we have a limited number of statistics that are hold in the DB.
In particular, this collection has been designed to keep saved in the database the statistics of the 20 most requested players of each league season, without recalculating them every time.

This example shown the structure of the document:

```
_id: ObjectId("5ea31644041c9957229a9bdd")
played: 30
playerAsStarter: 30
minutesPlayed: 2689
yellowCard: 3
redCard: 0
highestVote: 7.5
lowestVote: 5
averageVote: 6.483870967741935
year: "2018-2019"
league: "Serie A"
player: "Cristiano Ronaldo"
goalCount: 21
assistCount: 7
access: 2
```

4.3 Team statistics collection

This collection contains the statistics of the teams searched by users. We have a document for each season of each league and it contains the statistics of the 18 or 20 teams.

Here, we can see an example to understand the data model of this documents. The document shown in this example is created when the user asks for the stats of the teams of the Italian league for the year 2018-2019:

```
_id: ObjectId("5ea583156c859775ac8142a5")
name: "Serie A"
year: "2018-2019"
team: Array
  0: Object
      name: "Atalanta"
      avgPossessoPalla: 57.78947368421053
      avgTiriTotali: 16.86842105263158
      avgTiriInPorta: 5.842105263157895
      avgTiriFuori: 6.973684210526316
      avgTiriBloccati: 4.052631578947368
      avgCalciDAngolo: 6.368421052631579
      avgFuorigioco: 1.9189189189189189
      avgFalli: 11.513513513513514
      avgCartelliniGialli: 1.5675675675675675
      avgTiriInArea: 6
      avgTiriDaFuoriArea: 4
      avgSalvataggiDelPortiere: null
      avgPassaggi: 490
      avgPrecisionePassaggi: null
      avgContrastiVinti: null
      avgContrastiAereiVinti: null
  1: Object
```


The field *team* is an array and each element contains the statistics of a team

4.4 Account collection

This collection includes the account documents. The structure of each document is shown in the following examples.

Normal user:


```
_id: ObjectId("5ea567cc60815c21c7245548")
username: "nicola.mota"
password: "nicola"
favoriteLeague: "Serie A"
access: 4
```


Admin:


```
_id: ObjectId("5ea2e98ea327aa62f1755abb")
username: "admin"
password: "admin"
admin: true
```

5 Implementation

5.1 MongoDB indexes

5.1.1 Index 1

User collection is indexed on the access counter in descending order. With this index it is possible to speed up the operation that shows the administrator to the most active users. In addition, it can also speed up the access operation on average, favoring the access of users who use the application the most.

Index: {access: -1}

The following are the performances obtained by running the query that shows the 10 most active users (with multiple accesses) without using the index:

```
"executionStats" : {
    "executionSuccess" : true,
    "nReturned" : 3004,
    "executionTimeMillis" : 5,
    "totalKeysExamined" : 0,
    "totalDocsExamined" : 3005
}
```

The following are the performances using the index:

```
"executionStats" : {
    "executionSuccess" : true,
    "nReturned" : 10,
    "executionTimeMillis" : 1,
    "totalKeysExamined" : 10,
    "totalDocsExamined" : 10
}
```

We can observe that the execution time decreases from 5 to 1 milliseconds and above all, previously 3005 documents were examined, while now only the first 10 are taken directly.

5.1.2 Index 2

The collection with the player stats is indexed in order to speed up the operations that show the statistics of a player, given a certain league and a certain season. We introduce a compound index for the fields “year” and “league”, which respectively represent the season year and the league to which that player's statistics refer.

Compound index: {year: -1, league: -1}

The following are the performances obtained by running the query that shows the stats of a player who played in a certain league in a certain season, without using the index:

```
"executionStats" : {
    "executionSuccess" : true,
    "nReturned" : 5,
    "executionTimeMillis" : 3,
    "totalKeysExamined" : 0,
    "totalDocsExamined" : 21
}
```

The following are the performances using the index:

```
"executionStats" : {
    "executionSuccess" : true,
    "nReturned" : 5,
    "executionTimeMillis" : 2,
    "totalKeysExamined" : 5,
    "totalDocsExamined" : 5
}
```

We can observe that the execution time decreases from 3 to 2 milliseconds and 5 documents were examined instead of 21. Of course the advantage of using this index grows as the number of players, for whom we have calculated the statistics over the years and in the various leagues, increases, since these documents are all put together in the *playerStat* collection.

5.1.3 Index 3

The collection with the team stats is indexed in order to speed up the operations that show team statistics given a certain league and a certain season. We use a compound index for the fields “year” and “name” which respectively represent the season year and the league to which that team statistics refer.

Compound index: {year: -1, name: -1}

The following are the performances obtained by running the query that shows the stats of a team of a certain league in a certain season, without using the index:

```
"executionStats" : {
      "executionSuccess" : true,
      "nReturned" : 1,
      "executionTimeMillis" : 1,
      "totalKeysExamined" : 0,
      "totalDocsExamined" : 6
}
```

The following are the performances using the compound index:

```
"executionStats" : {
      "executionSuccess" : true,
      "nReturned" : 1,
      "executionTimeMillis" : 0,
      "totalKeysExamined" : 1,
      "totalDocsExamined" : 1
}
```

We can observe that the execution time decreases from 1 to 0 milliseconds and only 1 document is examined instead of 6. We organized the statistics of the teams searched by users, in documents where each document represents a season of a certain league and within each of these documents there are the statistics of the 18 or 20 teams. So the advantage of this index is that it speeds up the search for the document of the season containing the statistics of the team chosen by the user. For now there are 4 seasons of 4 leagues in the database, so at most there can be 16 documents in the *teamStat* collection, but the idea for which this index was chosen is that with the passing of the seasons these documents will increase. In addition, they will increase even if the administrator decides to add the past seasons, not yet present, of the various championships. As the number of these documents grows, this index will become increasingly useful.

5.2 Replica setup

In order to guarantee a high level of availability, we give back the control to application after the write on the primary server, without waiting the update of all replicas. This choice guarantees a good performance in term of speed, but decrease the part related to consistency, it could happen that a user reads some "old" data from a replica, in case of primary server failure.

```
mongoClient = MongoClients.create(
    MongoClientSettings.builder()
        .applyToClusterSettings(builder ->
            builder.hosts(Arrays.asList(
                new ServerAddress(ip0, port0),
                new ServerAddress(ip1, port1),
                new ServerAddress(ip2, port2))))
            .writeConcern(new WriteConcern().ACKNOWLEDGED)
        .build());
```

The following code shows how it’s possible to build an architecture in which there are 3 mongo servers running in the backend:

```
#  PREPARAZIONE CARTELLE PER LOG E PER I DB
sudo mkdir -p /srv/mongodb/rs0-0 /srv/mongodb/rs0-1 /srv/mongodb/rs0-2
sudo mkdir -p /var/log/mongodb/rs0-0 /var/log/mongodb/rs0-1 /var/log/mongodb/rs0-2
#  RUN 3 INSTANCES OF MONGOD
sudo mongod --port 27017 --dbpath /srv/mongodb/rs0-0 --replSet rs0 --oplogSize 128
--logpath /var/log/mongodb/rs0-0/server.log --fork
sudo mongod --port 27018 --dbpath /srv/mongodb/rs0-1 --replSet rs0 --oplogSize 128
--logpath /var/log/mongodb/rs0-1/server.log --fork
sudo mongod --port 27019 --dbpath /srv/mongodb/rs0-2 --replSet rs0 --oplogSize 128
--logpath/var/log/mongodb/rs0-2/server.log --fork
#  CHECK LISTENING PROCESS
netstat -tulpn
#  CONNECT TO PRIMARY MONGOD ISTANCE
mongo --port 27017
var rsconf = {
      _id : "rs0" ,
      members : [
            {_id : 0 , host :"127.0.0.1:27017"},
            {_id : 1 , host :"127.0.0.1:27018"},
            {_id : 2 , host :"127.0.0.1:27019"}
      ]
} ;
rs.initiate(rsconf);
rs.conf();
rs.status();
#  KILL PROCESS USING PORT
sudo fuser -k 27017/ tcp
sudo fuser -k 27018/ tcp
sudo fuser -k 27019/ tcp
#  ELIMINA CARTELLE -
sudo rm -r /srv/mongodb/rs0-0/srv/mongodb/rs0-1/srv/mongodb/rs0-2
sudo rm -r /var/log/mongodb/rs0-0/var/log/mongodb/rs0-1/var/log/mongodb/rs0-2
```

5.3 Analytics and Statistics

5.3.1 Top 20 scorers

This analytic is performed to display the top 20 players with most goals in a given season of a certain league. To do so, we make an aggregation pipeline to recover from our database the information about the matches of the season. The first thing to do is to filter out the document so that only the required matches are used; done that, we aggregate on player(scorer) summing every goal made by that specific player; as a last thing to do, we order the resulting documents in descending order and limit to 20 results.

```
db.league.aggregate([
      {$project: {season:1, "name":1} },
      {$match:{"name":"ita-serie-a"}},
      {$unwind: "$season"},
      {$match: {"season.year":"2015-2016"}},
      {$unwind:"$season.round"},
      {$unwind:"$season.round.match"},
      {$unwind:"$season.round.match.goal"},
      {$group:
            {
                  _id: "$season.round.match.goal.scorer",
                  goalCount: { $sum: 1 }
            }
      },
      { $sort : { goalCount : -1, _id: 1}},
      { $limit : 10 }
])
```

5.3.2 Top 20 vote players

This analytic is about collecting the 20 players with the highest average vote. Traversing the matches of a season, we collect the vote for each player and make an average in the group stage of the aggregation pipeline. Once done, we sort in descending order and then limit to 20 results.

```
db.league.aggregate([
      {$project: {season:1, "name":1} },
      {$match:{"name":"Serie A"}},
      {$unwind: "$season"},
      {$match: {"season.year":"2015-2016"}},
      {$unwind:"$season.round"},
      {$unwind:"$season.round.match"},
      {$addFields: {player : {$concatArrays: [
                        "$season.round.match.homeLineup.player",
                        "$season.round.match.awayLineup.player"
                  ]
      }}},
      {$project: {
            "player" : 1,
            "season" : 1
            }
      },
      {$unwind: "$player"},
      {$project: {
            "player" : 1,
```

```
            "team" : {
                        $cond: {
                              if: {
                                    "$in": [
                                    "$player",
                                    "$season.round.match.homeLineup.player"
                              ]
                        },
                              then: "$season.round.match.homeTeam",
                              else: "$season.round.match.awayTeam"
                        }
                  }

            }
      },
      {$match: {
            "player.vote" : {
                "$exists": true,
                "$ne": null
            }
            }
    },
      {$group:
            {
                  _id: "$player.name",
                  averageVote: { $avg: "$player.vote" },
                  team : { $first: '$team' },
                  played: { $sum: 1 }
            }
      },
      { $sort : { averageVote : -1, _id: 1}},
      { $limit : 20 }
])
```

5.3.3 Player statistics (goals and assists)

This aggregation pipeline is used to produce a document that contains statistics for a player in a given season. Statistics contains a lot of interesting information but here, for simplicity, we only show how to compute the number of goals and assist for a specific player.

Given a championship, a season and a player, we group for that specific player and simply count assists and goal occurrence in the documents. This way we obtain those informations.

```
db.league.aggregate([
      {$project: {season:1, "name":1} },
```

```
        {$match:{"name":"ita-serie-a"}},
        {$unwind: "$season"},
        {$match: {"season.year":"2015-2016"}},
        {$unwind:"$season.round"},
        {$unwind:"$season.round.match"},
        {$unwind:"$season.round.match.goal"},
        {$match: {$or: [
            {"season.round.match.goal.assist":"Paul Pogba"},
            {"season.round.match.goal.scorer":"Paul Pogba"}
        ]
        }},
        {$group:
              {
                    _id: "Paul Pogba",
                    goalCount: {
                          $sum:  {
                                $cond: {
                                      if: { $eq:[
                              "$season.round.match.goal.scorer", "Paul Pogba"
                                       ] },
                                      then: 1,
                                      else: 0
                                }
                          }
                    },
                    assistCount: {
                          $sum:  {
                                $cond: {
                                      if: { $eq:[
                              "$season.round.match.goal.assist", "Paul Pogba"
                                       ] },
                                      then: 1,
                                      else: 0
                                }
                          }
                    }
              }
        }
])
```

5.3.4 Team statistics

This aggregation pipeline is used to produce a document that contains statistics for a team in a given season. Statistics contains a lot of interesting information but here, for simplicity, we only show how to compute the average ball possession and total shots for a specific team.

Given a championship, a season and a team, we group for that specific team and simply use $avg operator to compute the statistics discussed before.

```
var res = db.league.aggregate([
      {$project: {season:1, "name":1} },
      {$match:{"name":"Serie A"}},
      {$unwind: "$season"},
      {$match: {"season.year":"2015-2016"}},
      {$unwind:"$season.round"},
      {$unwind:"$season.round.match"},
      {$match: {"$or" : [
                  {"season.round.match.homeTeam" : "Roma"},
                  {"season.round.match.awayTeam" : "Roma"}
      ]}},
      {$project: {
            season:1,
            team : "Roma",
            PossessoPalla : {
                  $cond: {
                  if: {
                    $eq: ["$season.round.match.homeTeam" , "Roma"]
                  },
                  then: { $arrayElemAt: [
                        "$season.round.match.statisticHome", 0
                  ]},
                  else: { $arrayElemAt: [
                        "$season.round.match.statisticAway", 0
                  ]}
              }
            },
            TiriTotali : {
                  $cond: {
                  if: {
                    $eq: ["$season.round.match.homeTeam" , "Roma"]
                  },
                  then: { $arrayElemAt: [
                        "$season.round.match.statisticHome", 1
                  ] },
                  else: { $arrayElemAt: [
                        "$season.round.match.statisticAway", 1
                  ]}
              }
            }
      }},
      {$project: {
```

```
            team : 1,
            PossessoPalla : "$PossessoPalla.PossessoPalla",
            TiriTotali : "$TiriTotali.TiriTotali",
      }},
      {$group:
            {
                  _id: "$team",
                  "avgPossessoPalla" : {
                              $avg: "$PossessoPalla"
                  },
                  "avgTiriTotali" : {
                              $avg: "$TiriTotali"
                  }
            }
      }
])
```

5.4 Main software modules

The main class of our system, used to interact with the database is the *MongoManager* class. This class is responsible to make a connection to the running MongoDB instance, to create and to access our database, to create and to access our collections. It also contains: various methods for managing user accounts; methods for interacting with the database in order to implement the various functionalities to be provided to the user; methods for interacting with the database in order to implement administrator functionalities.

All the other classes implement the various functional requirements using the methods provided by the MongoManager class and integrating them with the GUI created with JFrame. The GUI is realized so that users and administrators can use the application in the simplest possible way.

5.5 Create

As an example of create operation, we have reported the following method that allows users to create a new account by providing username, password and favourite league as input

```
public boolean register(String username, String password, String favoriteLeague) {
    /* Allow registration of a user, not admin accounts, if not already existing*/
    Document doc = accountCollection.find(eq("username", username)).first();
    if (doc == null) {
        doc = new Document("username", username)
                .append("password", password)
                .append("favoriteLeague", favoriteLeague)
                .append("access", 0);

        accountCollection.insertOne(doc);

        System.out.println("Account '" + username + "' successfully registered!");
        return true;
    } else {
        System.out.println("Username '" + username + "' already registered!");
        return false;
    }

}
```

5.6 Read

The following method allows the administrator to see the list of the ten most active users

```
public List<Document> top10ActiveUsers() {

    //doc type: {"username": "n", "access": 2}

    MongoCursor<Document> cursor = accountCollection.find(ne("username", "admin"))
                                    .projection(Projections.exclude("_id", "password", "favoriteLeague"))
                                    .sort(Sorts.descending("access")).limit(10).iterator();
    List<Document> result = new ArrayList<>();
    try {
        while (cursor.hasNext()) {
            result.add(cursor.next());
        }
    } finally {
        cursor.close();
    }
    return result;
}
```

5.7 Update

This method allows the administrator to insert a new season for a particular league by updating the document for that league

```
public void insertNewSeason(String league, String year, Document seasonDoc){
    /*
        update the given league

        1- Remove if exists
        2- Add the new one
    */
    UpdateResult res = leagueCollection.updateMany(
        new Document("name", league),
        new Document("$pull", new Document("season", new Document("year", year)))
    );


    res = leagueCollection.updateMany(
        new Document("name", league),
        new Document("$push", new Document("season", seasonDoc))
    );
}
```

5.8 Delete

The following method allows the administrator to remove a user account by providing the username of that account and administrator credentials as input

```
public int removeAccount(String username, String password, String usernameToRemove){
    /* ONLY ADMIN CAN DO THIS!! -> insert admin credential plus target account
    Returns:
        -1: wrong credential
        0: no such username to remove
        1: account successfully removed
    */
    int result = -1;
    int loginResult = login(username, password);
    if(loginResult == 2){
        DeleteResult deleteResult = accountCollection.deleteOne(eq("username", usernameToRemove));
        result = (int) deleteResult.getDeletedCount();
        if(result != 0)
            System.out.println("Account '"+usernameToRemove+"' successfully removed");
        else
            System.out.println("No such username: " + usernameToRemove);
    }
    else{
        System.out.println("Wrong credentials. You need to be an admin to perform this operation.");
    }
    return result;
}
```

6 User manual

At the startup the application shows the login window.

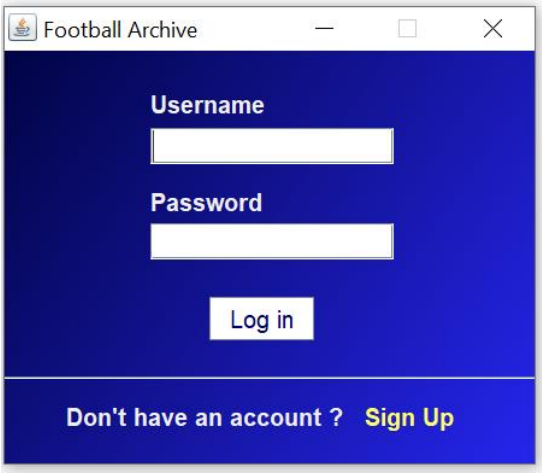


Fig 1: Login window

In case the user is already registered, he only has to insert his username and password and click on the "Log in" button. Otherwise press on "Sign Up" to create a new account.

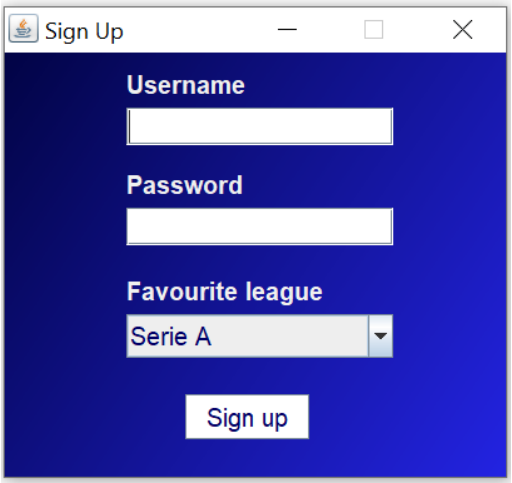


Fig 2: Sign up window

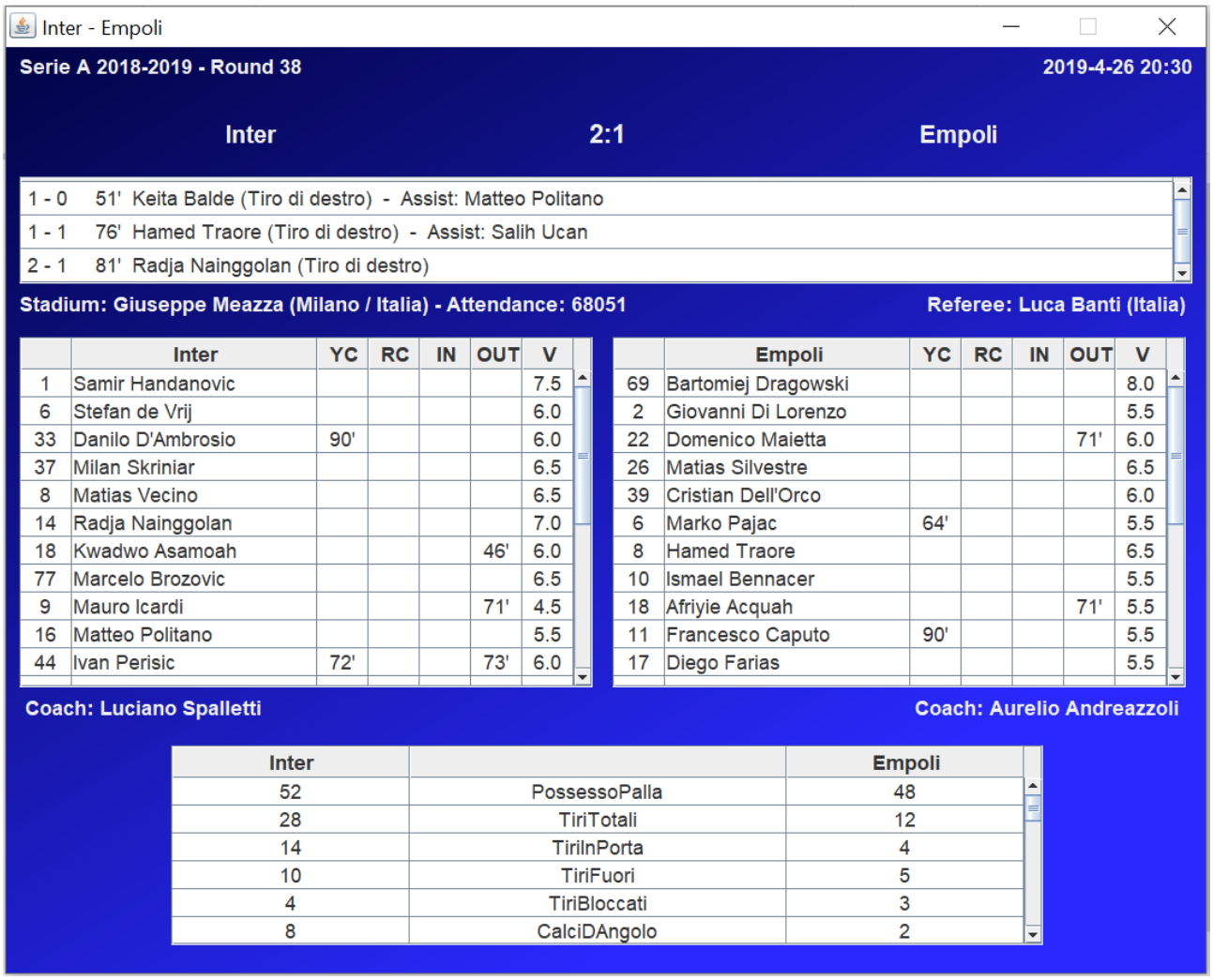
Insert username and password and choose your favorite championship among the four proposed: Serie A, Premier League, La Liga or Bundesliga.
After that, click on the “Sign up” button to complete the registration. If everything has been done correctly, a message will notify the success of the operation and you can access the system through the login window as explained above.

Once logged in, a window is shown with the results of the last day of the last season of the user's favorite championship

Fig 3: Main window with results

In this window the user can browse through the results of all the rounds played in the selected season. You can see the date, time and result of each match. It is also possible to change the league and season from the appropriate menus on the top left.

To see the details of a game, just click on the appropriate row and a window with the details of the match will appear.



	Inter	YC	RC	IN	OUT	V
1	Samir Handanovic					7.5
6	Stefan de Vrij					6.0
33	Danilo D'Ambrosio	90'				6.0
37	Milan Skriniar					6.5
8	Matias Vecino					6.5
14	Radja Nainggolan					7.0
18	Kwadwo Asamoah				46'	6.0
77	Marcelo Brozovic					6.5
9	Mauro Icardi				71'	4.5
16	Matteo Politano					5.5
44	Ivan Perisic	72'			73'	6.0

	Empoli	YC	RC	IN	OUT	V
69	Bartomiej Dragowski					8.0
2	Giovanni Di Lorenzo					5.5
22	Domenico Maietta				71'	6.0
26	Matias Silvestre					6.5
39	Cristian Dell'Orco					6.0
6	Marko Pajac	64'				5.5
8	Hamed Traore					6.5
10	Ismael Bennacer					5.5
18	Afriyie Acquah				71'	5.5
11	Francesco Caputo	90'				5.5
17	Diego Farias					5.5

Inter		Empoli
52	PossessoPalla	48
28	TiriTotali	12
14	TiriInPorta	4
10	TiriFuori	5
4	TiriBloccati	3
8	CalciDAngolo	2

Fig 4: Match details window

In the match details window you can see below the result, the scorers and the assist men. Also you can see the starting lineups, substitutes, coaches and the statistics of the match. There is also other information such as the stadium, the number of spectators and the referee.
In the table with the players, there are five columns:

- YC stands for “yellow card” and contains the possible minute in which a player has received a yellow card
- RC stands for “red card” and contains the possible minute in which a player has received a red card
- IN and OUT respectively represent the possible minute of entry and exit of a player
- V stands for “vote” and shows the player's vote (only for the Italian league)

If you return on the main window with the results and you click on the “Standings” button, you can see the standings of the selected season.

Standings

Serie A 2018-2019

Pos	Team	MP	W	D	L	GS	GA	Pts
1	Juventus	38	28	6	4	70	30	90
2	Napoli	38	24	7	7	74	36	79
3	Atalanta	38	20	9	9	77	46	69
4	Inter	38	20	9	9	57	33	69
5	Milan	38	19	11	8	55	36	68
6	Roma	38	18	12	8	66	48	66
7	Torino	38	16	15	7	52	37	63
8	Lazio	38	17	8	13	56	46	59
9	Sampdoria	38	15	8	15	60	51	53
10	Bologna	38	11	11	16	48	56	44
11	Sassuolo	38	9	16	13	53	60	43
12	Udinese	38	11	10	17	39	53	43
13	SPAL	38	11	9	18	44	56	42

Champion team and qualification for the Champions League group stage
Qualification for the Champions League group stage
Qualification for the Champions League preliminary round
Qualification for the Europa League group stage
Qualification for the Euopa League preliminary round
Relegation play-offs
Relegation

Fig 5: Standings window

In the standings window for each team you can see the position, matches played, wins, draws, losses, goals scored, goals allowed and the points. The colored rows represent the awards achieved by the teams, as shown in the legend.

If you return on the main window and you click on the “Players Stats” button, you can see the players stats for the chosen season.

Players Stats

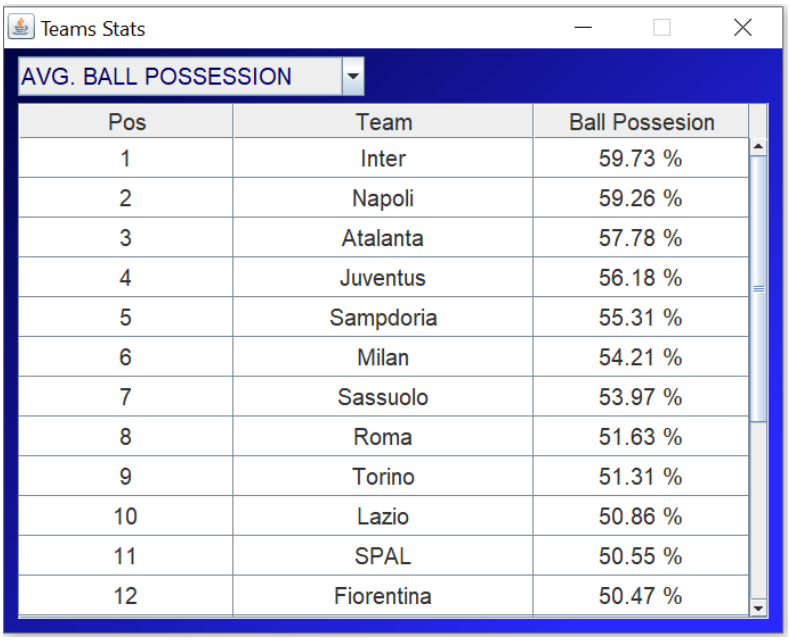
TOP 20 GOAL SCORERS

Pos	Player	Goals
1	Fabio Quagliarella	26
2	Duvan Zapata	23
3	Krzysztof Piatek	22
4	Cristiano Ronaldo	21
5	Arkadiusz Milik	17
6	Andrea Petagna	16
7	Dries Mertens	16
8	Francesco Caputo	16
9	Leonardo Pavoletti	16
10	Andrea Belotti	15
11	Ciro Immobile	15
12	Josip Ilicic	12

Fig 6: Players stats window

In the players stats window, the user can see: the ranking of the 20 players with most goals, assists, yellow cards, red cards. The ranking of the 20 players with the highest average vote and with the lowest average vote are available only for the Italian league.

If you return on the main window and you click on the “Teams Stats” button, you can see the teams stats for the selected season.



Pos	Team	Ball Possesion
1	Inter	59.73 %
2	Napoli	59.26 %
3	Atalanta	57.78 %
4	Juventus	56.18 %
5	Sampdoria	55.31 %
6	Milan	54.21 %
7	Sassuolo	53.97 %
8	Roma	51.63 %
9	Torino	51.31 %
10	Lazio	50.86 %
11	SPAL	50.55 %
12	Fiorentina	50.47 %

Fig 7: Teams stats window

In the teams stats window, the user can see the ranking of teams based on average ball possession per match, the average number of shots, the average number of shots on target or the average number of corners.

If you return on the main window and you click on the “Teams” button, you can see the teams participating in the selected season

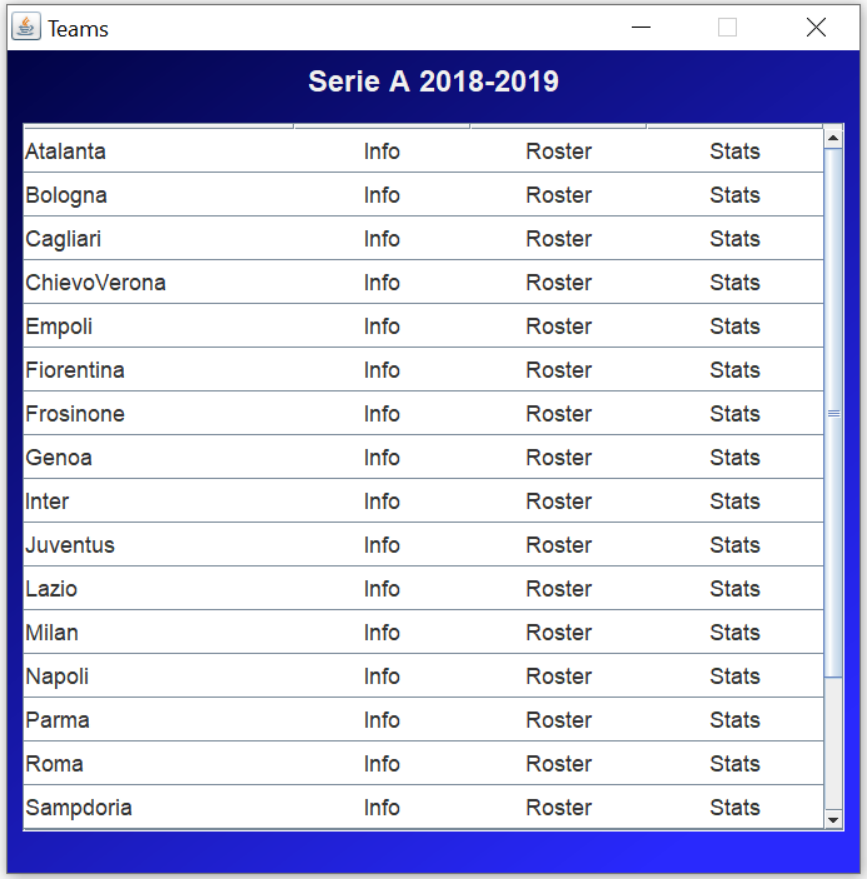


Fig 8: Teams window

In the teams window, if you click on "Info" or on "Roster" or on "stats", you can see respectively the general information, the list of players or the statistics of the team for the chosen season.

Fig 9: Info, roster and stats windows

If you return on the main window and you click on the “Player Stats” button, you can see the stats of a given player for the selected season.

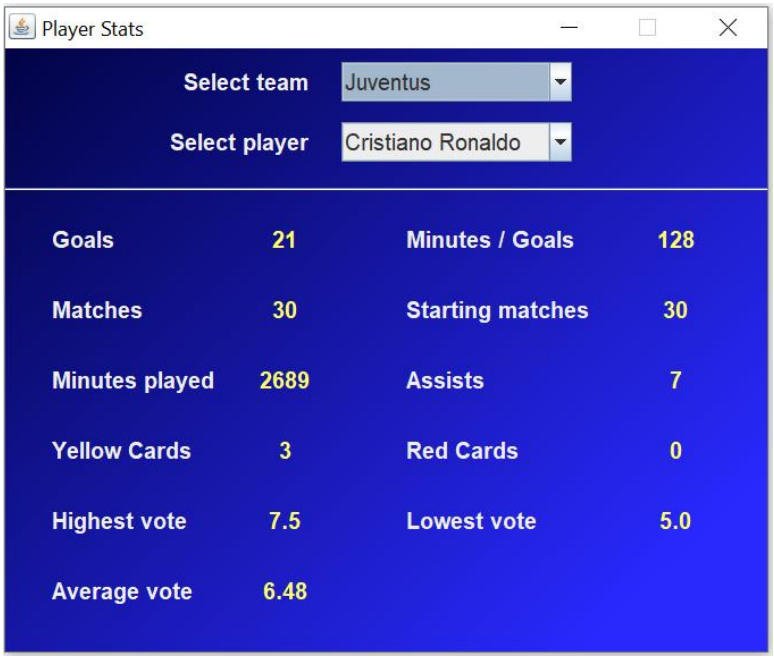


Fig 10: Player stats window

In the player stats window, you can first select the team and then select among the players of that team the one for which you want to see the stats. The stats shown for each player are the ones presented in Fig 9.

Finally, if you return on the main window and you click on the "Compare players" button, you can compare the stats of two given player for the selected season.

Fig 11: Compare players window

In the compare players window, you have to choose the two players to be compared, exactly as in the player stats window, and then the system will show the role of the chosen players and compare their statistics.

To log out, you simply need to close the main window, the one with the results. In this way all the windows opened will close and you will return back to the login window.

7 Admin manual

The administrator must log in by entering the administrator credentials. In this way the system will recognize it as administrator and open its window.

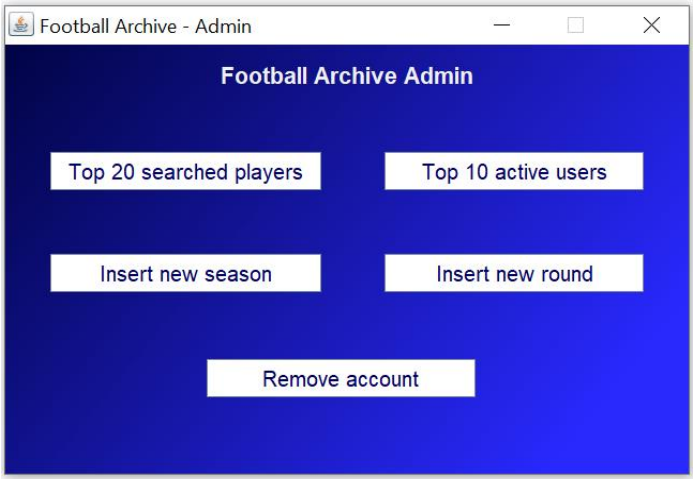


Fig 12: Admin window

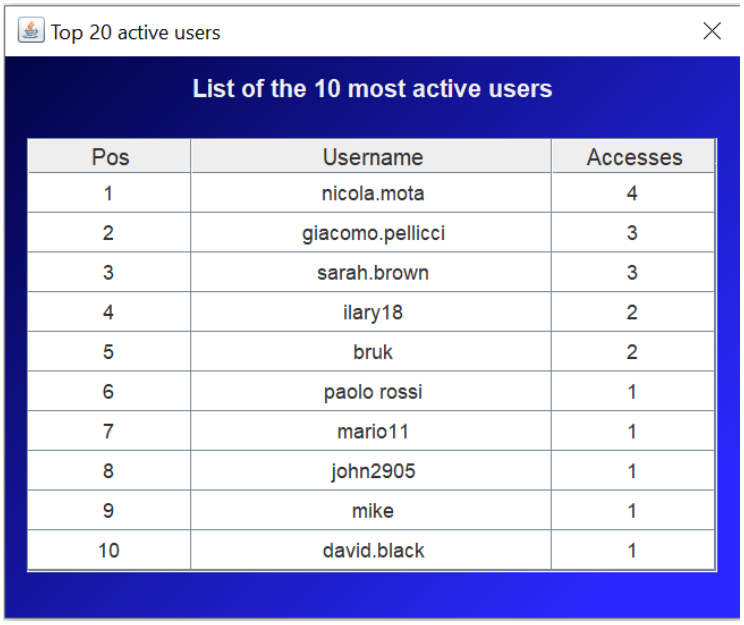
To see the list of the 20 most searched players, the administrator needs to click on the “Top 20 searched players” button.

Pos	Player	Team	Researches
1	Etrit Berisha	Atalanta	9
2	Angelo da Costa	Bologna	3
3	Mattia Del Favero	Juventus	3
4	Nicola Sansone	Bologna	2
5	Cristiano Ronaldo	Juventus	2
6	Timothy Castagne	Atalanta	2
7	Paulo Dybala	Juventus	1
8	Mauro Icardi	Inter	1
9	Tommaso Berni	Inter	1
10	Ciro Immobile	Lazio	1
11	Andrea Poli	Bologna	1
12	Adam Nagy	Bologna	1

Fig 13: Top 20 searched players window

In this window, the administrator must select the league and season for which he wants to see the list of the 20 most searched players.

If the administrator returns in the main window, he can click on the “Top 10 active users” button to see the list of the 10 most active users.



Pos	Username	Accesses
1	nicola.mota	4
2	giacomo.pellicci	3
3	sarah.brown	3
4	ilary18	2
5	bruk	2
6	paolo rossi	1
7	mario11	1
8	john2905	1
9	mike	1
10	david.black	1

Fig 14: Top 10 active users window

To remove an account, the administrator simply needs to click on the "Remove account" button in the main window.

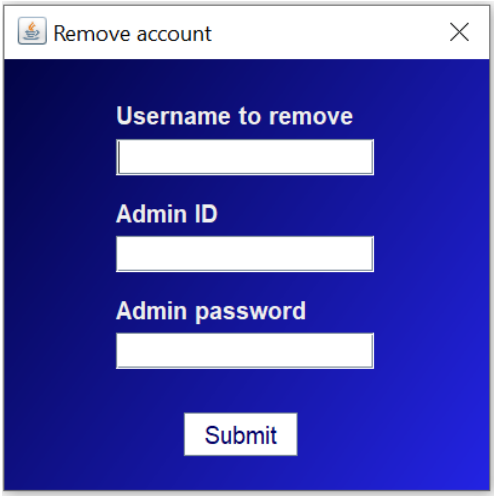


Fig 15: Remove account window

In this window, the administrator has to specify the username of the account to be removed and then the admin credentials. Finally, by clicking on the "Submit" button the specified account will be removed.

If you return to the administration window, you can click on the "Insert new season" or "Insert new round" button, respectively to insert a new season for a certain league or only the last round played for a specific season of a given league.

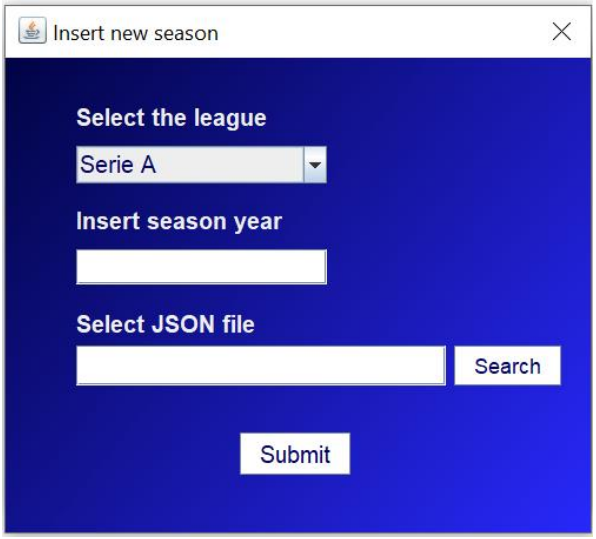


Fig 16: Insert new season window

In this window, the administrator must specify the league and year of the season that he wants to insert. After that, the administrator must select the related json file containing the season to be inserted from his computer.

To insert a new round played, the window that opens and the procedure to follow are exactly the same.

To log out, the administrator simply needs to close his main window. In this way all the windows opened will close and he will return back to the login window.