CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF INFORMATION TECHNOLOGY

ASSIGNMENT OF MASTER'S THESIS

Title:	Application for football league data collection and analysis
Student:	Bc. Artyom Trushin
Supervisor:	Ing. Jaroslav Kuchař, Ph.D.
Study Programme:	Informatics
Study Branch:	Web and Software Engineering
Department:	Department of Software Engineering
Validity:	Until the end of winter semester 2018/19

Instructions

The goal of the thesis is to design, implement, and test a web application for gathering football league statistic data and carrying out analysis of it. The server will collect the data, store them to a DB, and perform all computational tasks. GUI will present the collected and processed data including a specific interface for complex queries.
1) Analyse existing approaches, identify requirements and resources suitable for collecting of required data.
2) Design and implement a crawler for gathering:
    - statistical data of a football league for at least the last 10 years,
    - bookmaker odds for matches.
3) Design and implement a unit performing:
    - analysis of basic statistics of the league or separate football clubs,
    - prediction of upcoming matches based on all collected statistics.
4) Design and implement REST interface and web based GUI.
5) Test all parts of the project, perform an experiment for one selected league, and evaluate the quality of analysis and predictions.

References

Will be provided by the supervisor.

Ing. Michal Valenta, Ph.D.
Head of Department

prof. Ing. Pavel Tvrdík, CSc.
Dean

Prague March 5, 2017

Czech Technical University in Prague

Faculty of Information Technology

Department of Software Engineering

Master’s thesis

Application for football league data collection and analysis

Bc. Artyom Trushin

Supervisor: Ing. Jaroslav Kuchař, Ph.D.

30th June 2017

Acknowledgements

I would like to thank Ing. Jaroslav Kuchař, Ph.D for the suggestions and personal approach. Further I would like to thank my parents, girlfriend and friends for support throughout my studies.

Declaration

In Prague on 30th June 2017 . . . .

Citation of this thesis

Trushin, Artyom. *Application for football league data collection and analysis*. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2017.

Abstrakt

Hlavním cílem této diplomové práce je realizace webové aplikace pro shromažd'ování a analýzu statistik fotbalové ligy. Systém bude obsahovat webovou aplikaci a server pro poskytování dat. Webová aplikace bude zodpovědná za poskytování shromážděných dat a analýzu uživatelům. Server poskytující údaje bude shromažd'ovat data a provádět jejich analýzu. Kromě toho bude vytvořen REST API pro předání dát mezi dvěma servery.

Klíčová slova Web Crawling, Web Scrapping, REST, ASP.NET MVC, Statistika fotbalu, .NET

Abstract

The main goal of this thesis is to implement a web application for gathering and analyzing football league statistics. The system will comprise of a web application and data providing server. The web application will be responsible for providing collected data and analysis to users. The data providing server will collect data and carry out analysis of it. Furthermore, a REST API for data transmission between two servers will be created.

Keywords Web Crawling, Web Scrapping, REST, ASP.NET MVC, Football statistics, .NET

Contents

List of Figures

List of Tables

Introduction

Motivation

Football is one of the most popular games around the World. It is the most popular sport in terms of fans [2]. Nowadays, there are lots of websites, blogs, and other resources highlighting tons of football matches and tournaments. Especially, it concerns the best football leagues – World Cup, UEFA Champions League, FA Premier League and others.

They provide all possible statistics, but I couldn't find a website that covers all stats in one place. Sometimes it's complicated to find the stats you want if you are searching something special. For example, if the user intends to find the information about the longest streak without draws in the league in some particular season, he couldn't do that without a usage of some helper tools. That is why I decided to implement the system that will:

- gather all possible statistic data
- make new stats by analyzing collected info and provide it
- provide an interface for searching specific information that is interesting for particular user.

The goals

The main goal of the thesis is to gather all possible football statistics, provide analysis of it and show all data and result of the analysis to the user. The additional goal of this work is to provide the proper interface that could be used by the user for searching some specific statistics.

For these purposes, I should define requirements and use cases for the project. Then, I need to research existing approaches and resources suitable for data gathering. Finally, I have to implement a Web application that will show all gathered data, generated analysis results and provide the tool for searching specific statistics.

Chapter **1**

Analysis

The analysis part is the initial point in software development. In this chapter, I have defined all requirements - functional and non-functional. Furthermore, I have described use cases, a sequence diagram of the system and made a decision about target football league, which data I'm going to use on this project.

The entire system will be composed of several separate components: Crawler of statistical data of a football league, Bookmakers odds crawler, Data analyzer, GUI – website.

1.1 System functional requirements

All system functional requirements could be divided into several groups by system components.

1.1.1 Crawler of statistical data – finished matches

F1. The system should gather general statistical data of all finished games:

- Match participants - home and away team names
- Date and time
- Season and the round of the season
- Result score
- Available betting odds (optional – not from bookmakers web sites)

F2. The system should gather additional statistics of all games (all points are optional - if required information is available):

- Players who scored and goal time
- Assists
- Goal attempts – shots on/off target and blocked shots
- Ball possession
- Yellow/red cards
- Penalties, fouls, offsides
- Free/Corner kicks

1.1.2 Bookmaker odds crawler – upcoming matches

F3. The system should gather next betting odds for all upcoming games:

- Match Result – home/draw/away
- Double Chance
- Both Teams to Score – Yes/No
- Total Match Goals - Over/Under 2.5 Goals
- First goal/ last goal (optional)
- Correct score (optional)
- Half Time/Full Time (optional)
- Total Goals O/U - Team 1/Team 2 (optional)
- Total Match Goals Over/Under X.X Goals (optional)

F4. The system must allow the administrator to specify a crawl schedule (default crawler frequency will be defined in the Design part).

1.1.3 Analysis and prediction algorithm

F5. The system must provide the entire league statistical analysis by season:

- Number and percentage of draws and home/away team wins
- Standard table with current season results:
  - Rows – names of all teams
  - Columns – matches played, home wins, draws, away wins, number of goals scored, amount of goals conceded, goals difference, points, games with total under/over 2.5 (optional), games when both teams score/and opposite (optional)
- Statistics by rounds:
  - Average number of draws and home/away team wins
  - Average number of goals in round
  - Average number of matches with total goals under/over 2.5
  - Average number of matches when both teams score and opposite number
  - Average number of corners, yellow cards/red cards, penalties (optional)

F6. The system must separately provide statistical analysis of football clubs:

- Team form – last 5/10 games:
  - Number of draws, wins and losses
  - Original and average number of goals scored and conceded.
  - Number of points in the period and average points per game
  - Average numbers of corners, yellow/red cards, penalty kicks, received penalty (optional)
  - Number and percentage of goals in 1st/2nd halftimes (or in concrete time periods: 1-15, 16-30, 31-45 .., 75-90 minutes) (optional)
- Match preview statistical analysis:
  - Teams confrontation statistic – all matches of teams, and standard statistics

  - Both teams season statistics

F7. The system must provide simple prediction algorithm based on collected data.

1.1.4 Web-based GUI and REST interface

F8. The system must provide web-based GUI for displaying all gathering data and statistical analysis in a user-friendly form.

F9. The system must implement REST interface for communication between computing server and GUI.

F10. The GUI must implement specific interface for complex users queries. The query will contain next parameters:

- Specific period – could be set as range of dates, season rounds or range of seasons
- Condition(-s) – defined what matches are interested for the user. Types of condition:
  - Team name – could be selected only Home games, Away games or all games
  - Result of game – Home/Away win or draw
  - Total goals – over/under X.X goals, where X.X could be = {0.5, 1.5, 2.5, 3.5, 4.5, 6.5, 7.5}
  - Both teams score – Yes/No
  - Selected/Opposite team total of goals
  - Total goals at 1st/ 2nd halftime (optional)
  - Both teams score at 1st/ 2nd halftime (optional)
  - Games on which the specific player has participated (optional)
- Result – user could select one of predefined result form:
  - All suitable matches
  - Only count of all suitable matches
  - Maximal/minimal streak of suitable matches
  - Number of streaks with count of matches over/under selected number

1.2 System non-functional requirements

NF1. Legal resources – all used data should be free to use, or data usage permission should be received (prior authorization).

NF2. The entire project would be implemented in C# (using .Net Framework version 4.5 or higher).

NF3. Store gathered data – all data gathered by crawler should be stored to local database.

NF4. Target data – for simplicity only one league data will be gathered, the covered league will be defined later.

NF5. The amount of gathered data – should be collected data of at least 10 last seasons.

NF6. Failure handling - all bugs, problems with gathering and errors should be logged.

NF7. Reusability – the application should be designed and implemented that will allow reuse as many as possible components of the application for extending functionality.

NF8. Extensibility – all components of the project should be created using technologies that will allow to extend and create new features in the future easily.

1.3 System use cases

1.3.1 User views main league statistics

1. The user opens the web application.
2. The user goes to "Football stats" page and chooses a season that is interesting for him. (By default, a current season will be opened).
3. The user can now see the main statistics of the chosen season – standing table and the last tour matches results.

1.3.2 User views list of season games

1. The user opens the web application and go to Football stats.

2. The user chooses season and go to Results tab.
3. Now the user can see all played matches in the selected season.

1.3.3 User views a played game details

1. The user opens the list of season games.
2. The user finds game that is interesting for him – all games ordered by date.
3. The user clicks on game he has found.
4. Now, the user can see match details: match summary, statistics, head-to-head stats and bookmaker odds (if available).

1.3.4 Admin sets up schedule for bookmaker odds crawler

1. The admin connects to the server where the crawler deployed.
2. The admin finds the configuration file – path and other related info will be documented.
3. The admin changes in configuration file the necessary parameters.
4. The admin restarts the crawler to apply new settings.

1.3.5 User views league aggregated stats

1. The user opens the web application and goes to "Football stats" page.
2. The user opens Aggregated stats tab and chooses a season if needed.
3. The user now can see all provided aggregated stats.

1.3.6 User views a particular team statistics

1. The user opens the web application and goes to the "Football stats" page.
2. The user chooses season and goes to Teams tab.
3. The user clicks on a team name which stats he wants to see.
4. The user now can view all stats of the team in the selected season.

1.3.7 User searches specific stats by using provided tool

1. The user opens the web application and goes to the "Statistics search" page.
2. The user chooses the period he is interested in.
3. The user defines match conditions that he is interested in. Maximal number of conditions will be defined in design part.
4. The user defines result format.
5. The user taps on Start searching to view required stats.
6. After search request is processed the user can see the result of the search.

1.3.8 User views predictions on upcoming matches

1. The user opens the web application and goes to the "Predictions" page.
2. The user chooses a match from shown list of matches.
3. The user now can see a prediction with odds on the chosen game.

1.3.9 User views upcoming game information

1. The user opens the web application and goes to the "Football stats" page.
2. The user taps on Fixtures tab.
3. The user now can see the list of upcoming matches on the next 1-2 tours.
4. The user chooses a concrete match that is interesting for him.
5. The user now can see the match details.

The table 1.1 below demonstrates the mapping of all use cases to functional requirements. This table helps to verify that all functional requirements are covered by use cases.

Table 1.1: Mapping of Use Cases to Functional Requirements

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
UC1	✓				✓			✓	✓	
UC2	✓							✓	✓	
UC3	✓	✓	✓					✓	✓	
UC4				✓						
UC5	✓	+/-			✓			✓	✓	
UC6	✓	✓				✓		✓	✓	✓
UC7	✓	✓			+/-	+/-		✓	✓	
UC8	✓	?	✓			✓	✓	✓	✓	
UC9			✓			✓		✓	✓	

1.4 Target football league

Before starting research of football data resources and all other preparation, I need to choose what league will be covered in this project.
There are five most popular football leagues in the Europe: French Ligue 1, Italian Serie A, German Bundesliga, Spanish La Liga and English Premier League. Therefore, I decided to choose one of them, because they have better media coverage and more web resources. Also, I took into account that two most popular teams in the world – Barcelona and Real Madrid are playing in Spanish La Liga. Therefore, Spanish La Liga was chosen as target league that will be covered in this Diploma.

1.5 System sequence diagram

A sequence diagram is an interaction diagram that depicts program object interactions arranged in time sequence. It helps better understanding of the entire system workflow.

As can be seen from Figure 1.1 particular components of the whole system work independently. When the user opens some webpage with stats, the web server sends a request to REST API for every required data. REST API gets data from Database and return them to the web server, while data collectors work independently on their schedule. Analyzer checks availability of the new data. If new data comes, it will start to analyze them and produce out-coming stats. The result of data analysis finally sent to database and work of analyzer is done for that moment.

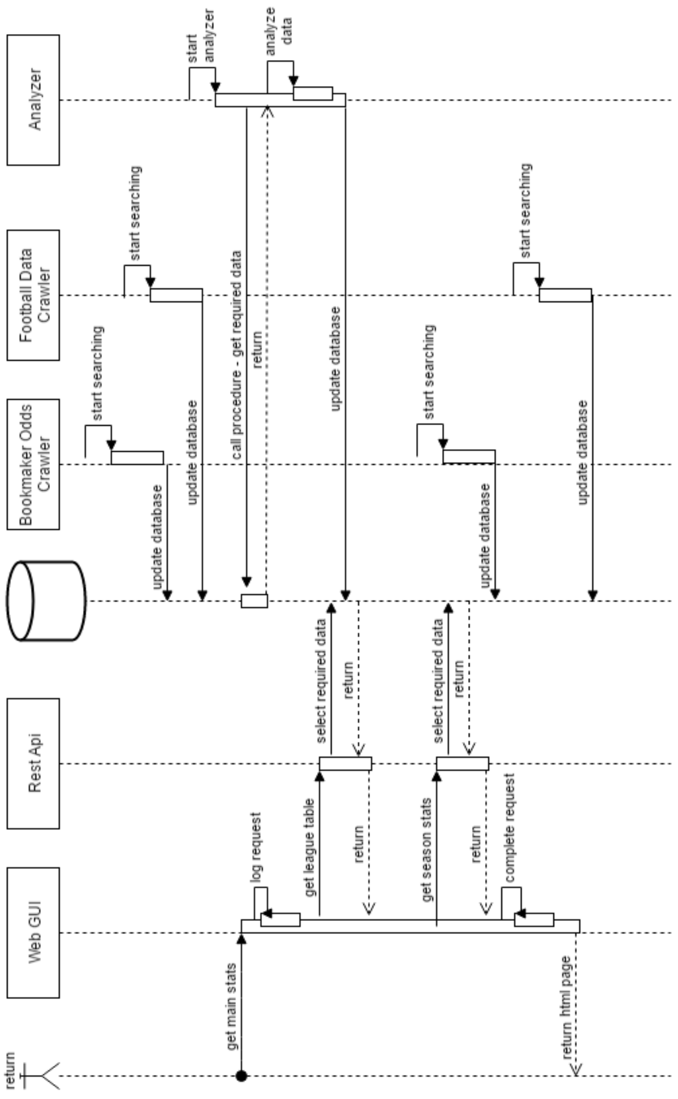


Figure 1.1: Sequence diagram of entire system

Chapter **2**

Research of resources and existing solutions

2.1 Resources – web sites

In this section I am going to research all available resources that contain required data.

2.1.1 Research criteria

At the beginning I need to define most important criteria of the research:

1. Legality of the usage – this criterion defines if data from analyzed resource are free to use or if it is under copyright law and prior authorization from the Provider is required for using these data
2. Data amount – how many last seasons of chosen league are covered in the resource
3. Variety of data – how many different statistical data are covered in the resource
4. Bookmaker odds – variety of bookmaker odds contained in the resource
5. Original data source – define the original data resource if data are taken from an another resource
6. Data structure – defines how difficult it is to parse data from the resource. Will be estimated by 3 levels – *good, mediocre* and *bad* structured
7. Addition criteria (optional)

2.1.2 List of resources

1. Flashscore.com or analogues There are lots of sites that have almost or exactly the same design and data structure as Flashscore (Livesports.cz, Soccer24.com, Myscore.ru, Myscore.com.ua, . . .). Those websites are user-friendly, provide live-scores and historical data for lots of sports. This resource covers almost all professional football leagues. The following criteria are applied:

1. Under copyright law, prior authorization is required
2. Covered all seasons from 1998-1999, 19 last seasons + current season
3. Provided a lot of different stats. All required statistical data are covered
4. The resource contains a lot of bookmaker odds
5. Used data from Enetpulse.com(link below)
6. Good data structure – easy to parse

Result: If prior authorization will be acquired, then it will be a good choice for the project.

2. Football-Data (www.football-data.co.uk) This website looks more like a big commercial for gambling than a website on football statistics, but it also contains a lot of statistical information and live-scores. A user could find all football data on livescore.football-data.co.uk. There are all historical data of Spanish La Liga from the season 1999-2000. This site is not suitable for web scrapping (crawling), because all moves on the site are realized by POST requests. But on the other side, all data are provided in .csv format, so web-scrapping is not required. The following research criteria are applied:

1. Free to use
2. 20+ last seasons + current season
3. For all seasons basic stats are covered and starting from season 2005-2006 additional stats are also covered
4. All general bookmaker odds are covered
5. Not linked to another resources – own data
6. All data in CSV format – the best structure for gathering data
7. It doesn’t guarantee the correctness of provided information

Result: Good choice for the project.

3. SoccerSTATS.com This resource provides statistics and results for most of the world leagues. It contains general game stats and a lot of accumulated statistics in unusual format. It's very complicated to parse. The following research criteria are applied:

1. Under copyright law
2. Only 6 last seasons + current season
3. Only general stats and some specific accumulated stats are available
4. Doesn't contain bookmaker odds
5. Not mentioned another resources – own data
6. Bad data structure – hard to parse

Result: Bad choice for gathering data, but a good example of specific statistics.

4. StatBunker Statbunker offers many statistics on Spanish La Liga and the stats cover the last 9 seasons. It is a very user-friendly website, but doesn't contain all needed stats. Also, it contains a lot of specific aggregated statistics, so it's a good example of analyzing data. Probably the most interesting feature of this website is the possibility to restrict league tables to a specific range of minutes. For example, its possible to get the table of the Premier League considering only the points for the first 10 or 15 minutes of play.

1. Under copyright law, however, doesn't clearly define what data are forbidden to use
2. Covers 9 last seasons + current season
3. Contains general stats and some additional stats as cards and penalties, but doesn't contain another required data
4. Doesn't contain bookmaker odds
5. Not mentioned another resources – own data
6. Mediocre data structure

Result: Not the best choice, but could be used as alternative.

5. Footstats Footstats provides classic football statistics only on the top European leagues like fouls, corners, cards, goals, etc. It covers 15 last seasons + current season. All data are taken from Football-Data web resource, essentially this site is representation of gathered data from another website with basic searching in the data. But it lacks bookmaker odds.

1. Free to use
2. Covers 15 last seasons + current season
3. All required statistics
4. Doesn't contain bookmaker odds
5. Football-Data [2.1.2]
6. Mediocre data structure

Result: Not the best choice, because it's better to use original data source.

6. Soccerway Soccerway is very popular football website that covers over 1000 football leagues and cups from 134+ countries. It is the world's largest football database and is owned and powered by digital sports media business PERFORM. It's very user-friendly and contains good statistical graphics.

1. Under copyright law.
2. Covers 23 last seasons + current season.
3. It contains all required statistics for last 6 and current seasons. Only general stats are covered for later seasons.
4. This resource doesn't focus on any bookmaker odds.
5. Data provider – Opta.
6. Data structure is not very convenient for parsing – normal level.

Result: Because of criteria 1,3,4 this resource is not suitable for the project.

7. 24score.com This is common sport score website that is available in Russian and English. It contains specific aggregated statistics. The nice feature of this resource – it contains statistics about all referees worked during the season.

1. Doesn't contain any information about copyrights – free to use.
2. Provides data for last 8 and current seasons.
3. Covers all required statistics and provides some specific stats.
4. Contains some bookmaker odds, not all required data is covered.
5. Not mentioned another resources – own data.
6. Good data structure – easy to parse.

Result: Despite of the fact that not all required data are covered, it's still a good choice for the project.

8. Football-Lineups Football-Lineups.com is a collaborative Database where football fans can review and post teams tactics and formations[link]. Contains all needed information except bookmaker odds. Good structured data for web scrapping.

1. Under copyright law, but this website grants a limited license to download the material on it solely for personal, noncommercial use.
2. Covers last 16 and current seasons.
3. Provides all required stats for all seasons and some additional statistics are provided for the last 5 seasons.
4. Doesn't contain any bookmaker odds.
5. Own data.
6. Very good structure for parsing and web-crawling.

Result: Alternative choice for the project, the only negative side of it, that it doesn't cover bookmaker odds.

9. SoccerVista It's a good free-to-use resource, any way of usage web site's content are not forbidden. The website contains a lot of additional information for matches, but doesn't contain all required data for this diploma project.

1. Copyrights info is not mentioned – free to use.
2. Provides only 2 last and current seasons for Spanish La Liga.
3. For target league only general stats are provided.
4. Contains only basic bookmaker odds.
5. Not mentioned another resources – own data.
6. Normal data structure.

Result: Bad choice because of lack of required data.

10. Betstudy Betstudy provides only the most common statistics, but it does so for most of the world leagues and it is also possible to order all the tables by any field. The website is simple and clear and it also features predictions on future fixtures.

1. Under copyright law – but doesn't clearly define what data are forbidden to use.

2. Covers 23 last and current seasons.
3. Only common stats.
4. Provides bookmaker odds only for upcoming matches, doesn't provide historical data.
5. Own data.
6. Normal data structure.

Result: Not the best choice, it doesn't contain all required data.

Summary
I have analyzed 20+ web resources, not all of them are mentioned in this part (see the Table 2.1). Almost all suitable resources, which covers all required data for this project, use data from 2 biggest football data providers – Opta and EnetPulse. And those websites are under copyrights low. Only one suitable free-to-use resource has been detected - Football-Data. Also, I really liked the data structure and coverage of sites that use data from Enetpulse, so I will try to get prior authorization.

Table 2.1: List of web resources

web recource	Research criteria						Result
Flashscore.com	-	✓	✓	✓	-	✓	alternative
SoccerSTATS.com	-	-	-	-	✓	-	
Football-Data	✓	✓	✓	✓	✓	✓	to use
STATBUNKER	+/-	✓	+/-	-	✓	✓	
Footstats	-	✓	✓	-	-	✓	
Soccerway	-	✓	✓	-	-	✓	
24score.com	✓	+/-	✓	+/-	✓	✓	
Football-Lineups	+/-	✓	+/-	-	✓	✓	
SoccerVista	-	-	+/-	+/-	✓	✓	
Betstudy	+/-	✓	-	-	✓	✓	
livefutbol.com	-	✓	+/-	-	✓	-	
worldfootball.com	-	✓	✓	-	✓	+/-	
vitibet.com	✓	-	-	+/-	✓	✓	
futbol24	✓	✓	-	-	✓	✓	
scibet.com	-	✓	+/-	-	✓	+/-	
annabet.com	✓	✓	✓	✓	✓	✓	
soccercenter.com	✓	-	-	✓	✓	✓	
betexplorer	-	✓	+/-	-	✓	✓	
betvirus.com	-	✓	✓	✓	✓	-	
scorecenter.com	✓	✓	-	✓	✓	+/-	

2.2 Resources – web services

2.2.1 Research criteria

Firstly, I have to define research criteria as I did in previous subsection:

1. Data amount
2. Variety of data
3. Data structure
4. Price

2.2.2 List of resources

1. Opta Opta is the world's leading sports data provider. It provides a lot of services with a wide variety of data. They collect, package, analyze and distribute more live data, in more detail, than anyone else. This is global company that works with well-known bookmakers (Sky Bet, Paddy Power, William Hill, betfair, etc.) , TV channels (Eurosport, Fox sports, Sky sports, etc.), famous sport clubs (Arsenal, Roma, Bayern Munich, Barcelona, etc.) and other world-known brands (Nike, Adidas, Bloomberg, etc.) (The whole list could be found on official Opta web page). This resource provides all required for my project data, but it is very expensive choice. Criteria:

1. Provides probably all possible data for all seasons.
2. Covers all data required for the project.
3. Doesn't define all possible ways of delivering data, but possibility of XML format is mentioned.
4. Price depends on services you want to use. Talking about football feeds, Opta offer around 40 feeds - from play by play stats to more detailed summaries. Price range starts at $600 runs to a couple of thousand per month.

Result: It's a very good choice for commercial profitable project, but too expensive for this diploma work.

2. EnetPulse Enetpulse covers more than 30 sports around the clock in various degrees of detail [1]. As the previous resource, this is global company from Denmark that works with a lot of well-known firms – sports media (tennis.com, livescore.com, UOL Brazil, etc.), sport betting (Betfair, 10Bet, SportingBet, etc.), broadcasters (TV2 Denmark, Viasat, TVP.pl, etc.) and applications (WhoScored, Sportflash,

etc.). The company provides several solutions: Historical Sports Data, Odds Comparison, Live Stats, fixtures and others. This data provider covers all required data in this project, but it also very expensive choice.
Criteria:

1. Doesn't mention number of covered seasons. But as we know Flashscore use this data provider, therefore we could guess that it provides more than 20 seasons.
2. Covers all required stats for last 5 seasons, for later season only general statistics are covered.
3. All data delivered via XML Push or by using Sports Data API.
4. All required data for last 5 seasons (for 1 league) costs $2500. If consumer want to use more leagues, he should buy a monthly subscription to the service. For example the subscription to top-5 leagues costs $1200 per month.

Result: As well as Opta it's a very good choice for commercial projects. But couldn't be used in this project because of the high cost.

3. XMLSoccer This resource is specialized in providing a cheap and stable machine-readable data-feed from their always up to date database. It doesn't provide any graphical stats as unlike previous resources, therefore it's a lot of cheaper than others. However, it covers all data I need for this work. Another interesting feature of this resource is providing parsing libraries in Java, .Net, PHP.
Criteria:

1. Provides data for 17 last seasons.
2. Covers all data for the project.
3. As the name of service implies, it provides data in XML format.
4. 10$ for 1 month full access, or 90$ for one year.

Result: Very good and cheap choice for commercial purpose.

Summary
I have looked at 3 data providers, 2 of them are global companies that provide a wide variety of data including some graphical stats for TV and other media. Those 2 are very expensive for the project. Last one is cheap and simple solution, that provides enough data for the project. It could be used if the project was for commerce.

2.3 Bookmaker odds for upcoming matches

Because of the fact, that I want to track and gather only common bookmaker odds, I could use any bookmakers for data gathering. I have analyzed a lot of bookmakers and all of them have almost the same data structure, data coverage, etc. Therefore, I could randomly choose one of them for my project. There are bookmakers that provide data in English: Bwin, Paddy Power, bet365, William Hill, Unibet. Also, Czech bookmakers could be used: Tipsport, Chance, Fortuna.
Result: I have decided to use Bwin and as alternative Chance bookmakers.

2.4 Existing solution

In this section, I will briefly describe a lot of statistics that existing solutions provide. In addition to almost all descriptions, there is an attached example that shows how it usually looks like. First of all, I want to mention common statistics that appear almost at all web resources focused on football stats.
Note: all pictures in this section are related to La Liga season 2015/2016.

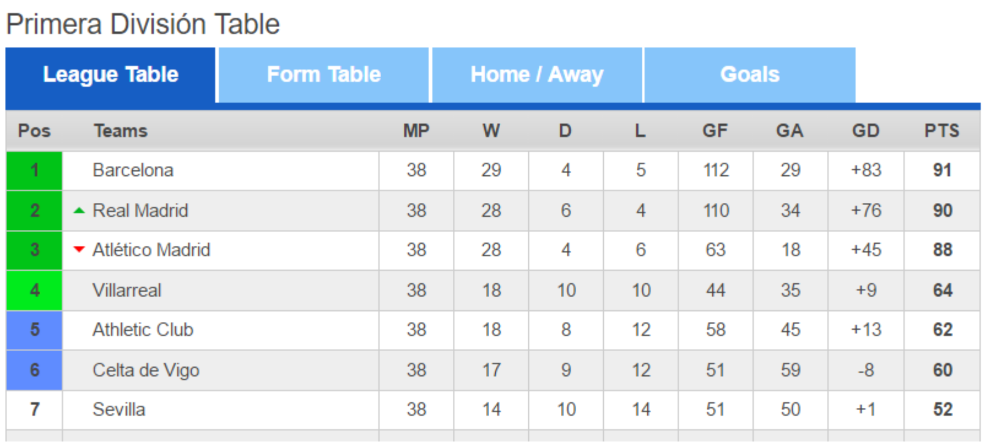
2.4.1 Common statistics

Standing table – commonly rows represent team names, and columns represent next indicators:

- Matches played (MP)
- Wins (W), Draws (D) and Losses(L)
- Goals (G) – in format *goals-scored:goals-conceded.* Sometimes it represents as 2 columns: Goal For (GF) and Goals Against
- Goal Difference (GD or dif or +/-) - optional column, it's calculated from previous indicator
- Points (Pts) – count of earned points, could be calculated form W (3 points for win), D (1 point for draw) and L(0 pts)

The table 2.1 usually could show all played matches or only Home/Away matches.
Results – list of all played matches grouped by rounds or date. As a rule each matched row has a link of game details and stats. Usually it contains the following indicators:

- Date and time of match

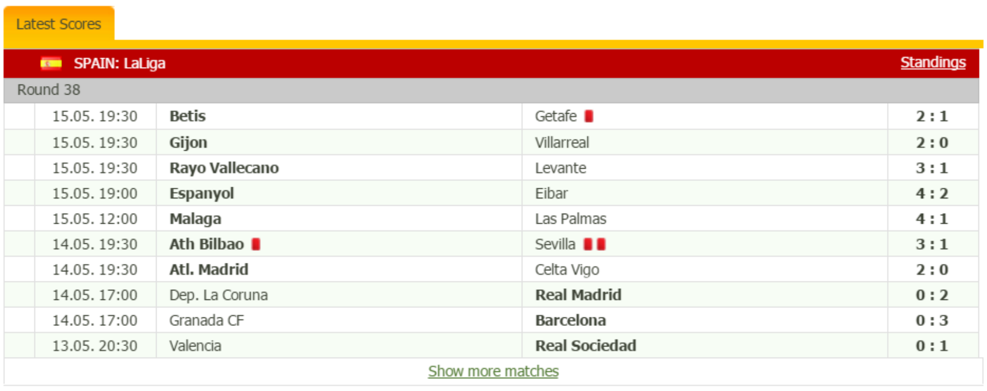

Primera División Table

League Table
Form Table
Home / Away
Goals

Pos	Teams	MP	W	D	L	GF	GA	GD	PTS
1	Barcelona	38	29	4	5	112	29	+83	**91**
2	▲ Real Madrid	38	28	6	4	110	34	+76	**90**
3	▼ Atlético Madrid	38	28	4	6	63	18	+45	**88**
4	Villarreal	38	18	10	10	44	35	+9	**64**
5	Athletic Club	38	18	8	12	58	45	+13	**62**
6	Celta de Vigo	38	17	9	12	51	59	-8	**60**
7	Sevilla	38	14	10	14	51	50	+1	**52**

Figure 2.1: Example of standing table

- Round of league season
- Competitors – names of football clubs
- Result score
- Red cards (optional)


Latest Scores
SPAIN: LaLiga
Standings

Round 38

15.05. 19:30	**Betis**	Getafe 🟥	**2 : 1**
15.05. 19:30	**Gijon**	Villarreal	**2 : 0**
15.05. 19:30	**Rayo Vallecano**	Levante	**3 : 1**
15.05. 19:00	**Espanyol**	Eibar	**4 : 2**
15.05. 12:00	**Malaga**	Las Palmas	**4 : 1**
14.05. 19:30	**Ath Bilbao** 🟥	Sevilla 🟥🟥	**3 : 1**
14.05. 19:30	**Atl. Madrid**	Celta Vigo	**2 : 0**
14.05. 17:00	Dep. La Coruna	**Real Madrid**	**0 : 2**
14.05. 17:00	Granada CF	**Barcelona**	**0 : 3**
13.05. 20:30	Valencia	**Real Sociedad**	**0 : 1**

Show more matches

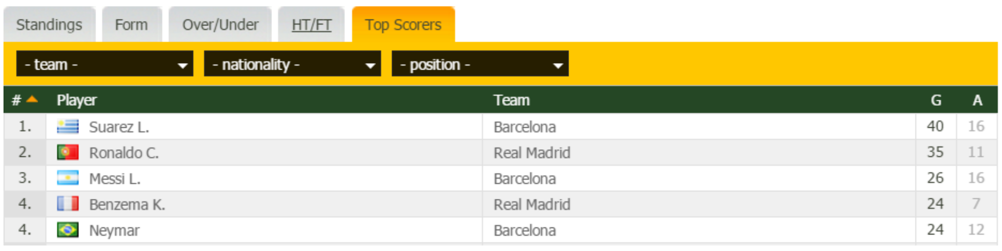
Figure 2.2: Results of 38. round

List of seasons – list of covered in resource seasons, usually with name of a season champion.

Top Scorers – a list of players who scored the most goals (see Figure 2.3). Usually it is represented as a table with the next columns:

- Position in table – rank

- Player and team name
- Number of goals
- Less common – minutes played and minutes-per-goal
- Less common – assists and very rare – *assists+goals*
- Very rare – number of penalty goals



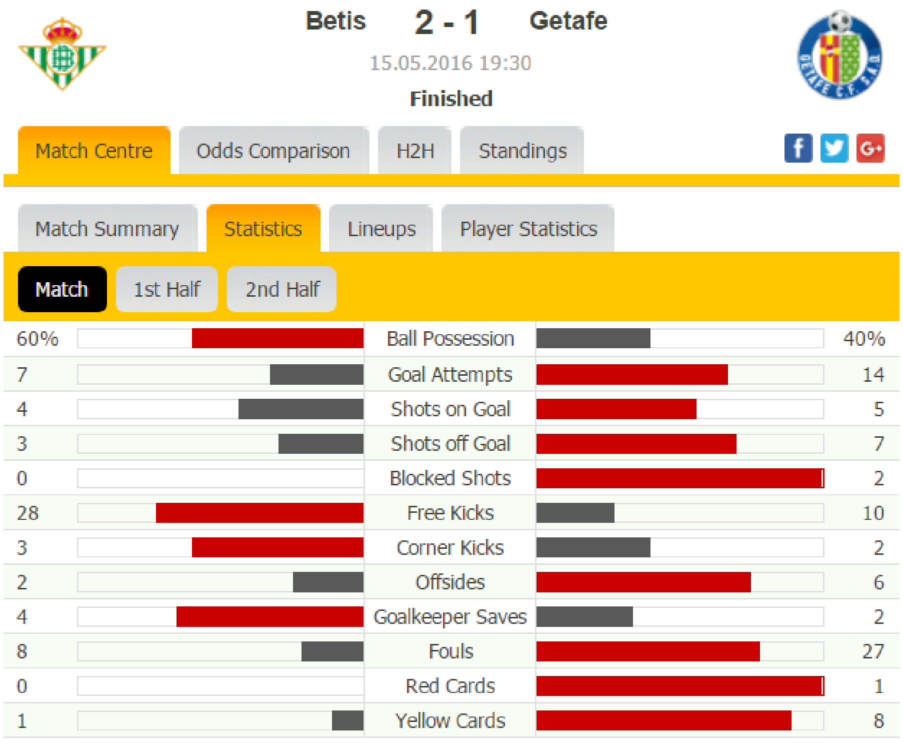
#	Player	Team	G	A
1.	Suarez L.	Barcelona	40	16
2.	Ronaldo C.	Real Madrid	35	11
3.	Messi L.	Barcelona	26	16
4.	Benzema K.	Real Madrid	24	7
4.	Neymar	Barcelona	24	12

Figure 2.3: List of top strikers

Match details and stats – different web resources have different coverage of match statistics and details. The most popular details are :

- Teams starting lineups and substitutes
- Goals – name of scorer and optional – name of the assisted player. Also could be indicated as penalty
- Substitutions – team name and names of substituted players, who is In and who is Out
- Yellow/red cards – player name and team, and minute of game.

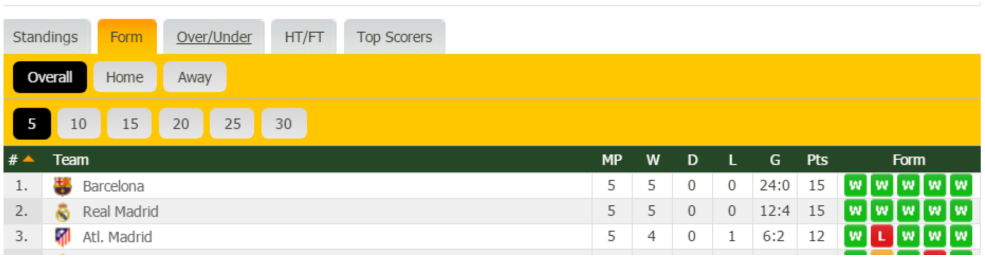
The common stats of match are: ball possession, goal attempts, shots on/off goal, blocked shots, free and corner kicks, offsides, fouls, red and yellow cards (see below on Figure 2.4).



Betis		Getafe
60%	Ball Possession	40%
7	Goal Attempts	14
4	Shots on Goal	5
3	Shots off Goal	7
0	Blocked Shots	2
28	Free Kicks	10
3	Corner Kicks	2
2	Offsides	6
4	Goalkeeper Saves	2
8	Fouls	27
0	Red Cards	1
1	Yellow Cards	8

Figure 2.4: Sample of match stats: Betis vs Getafe

Form – commonly shows last 5 or more matches of each team. On a modern website, it is usually shown as an additional column to the standing table with icons that represent last matches (all icons are links to match details).



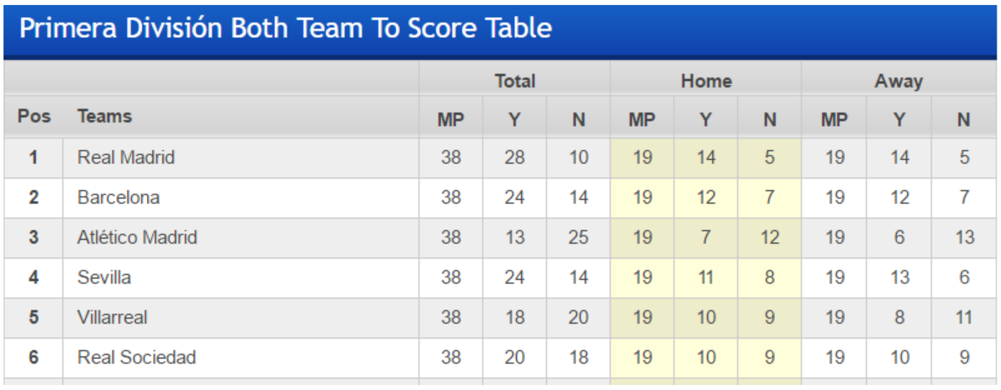
#	Team	MP	W	D	L	G	Pts	Form
1.	Barcelona	5	5	0	0	24:0	15	W W W W W
2.	Real Madrid	5	5	0	0	12:4	15	W W W W W
3.	Atl. Madrid	5	4	0	1	6:2	12	W L W W W

Figure 2.5: Form table for last 5 matches

2.4.2 More specific statistics

In this section I have described rarely occurring statistics that seems interesting for me. Focus falls on a statistical indicator or format of bunch of stats, but a source(-s) also have been mentioned.

Both Teams To Score (BTTS) Table – shows for each team numbers of games where both teams score (Y) and not score (N). Sometimes, the table additionally provides percentage of each result. Not very common statistical table though.
Resources: betstudy, soccerstats, 24score.com

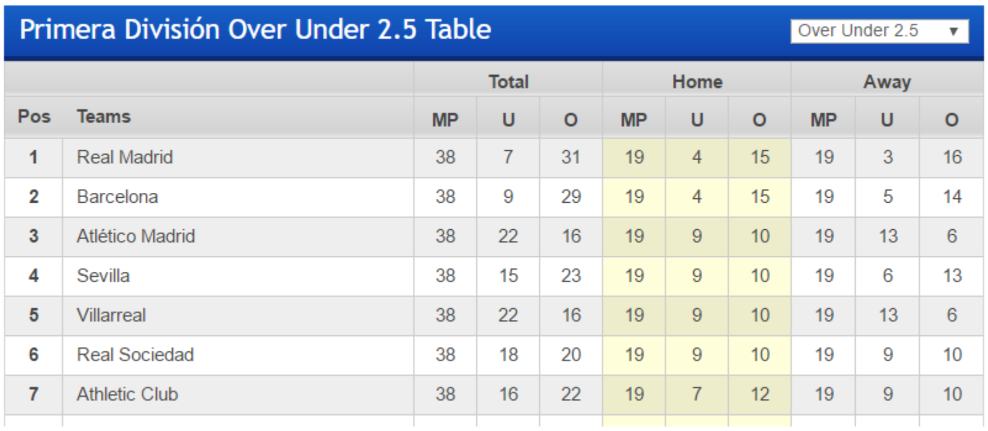


Primera División Both Team To Score Table

		Total			Home			Away		
Pos	Teams	MP	Y	N	MP	Y	N	MP	Y	N
1	Real Madrid	38	28	10	19	14	5	19	14	5
2	Barcelona	38	24	14	19	12	7	19	12	7
3	Atlético Madrid	38	13	25	19	7	12	19	6	13
4	Sevilla	38	24	14	19	11	8	19	13	6
5	Villarreal	38	18	20	19	10	9	19	8	11
6	Real Sociedad	38	20	18	19	10	9	19	10	9

Figure 2.6: Sample of BTTS table

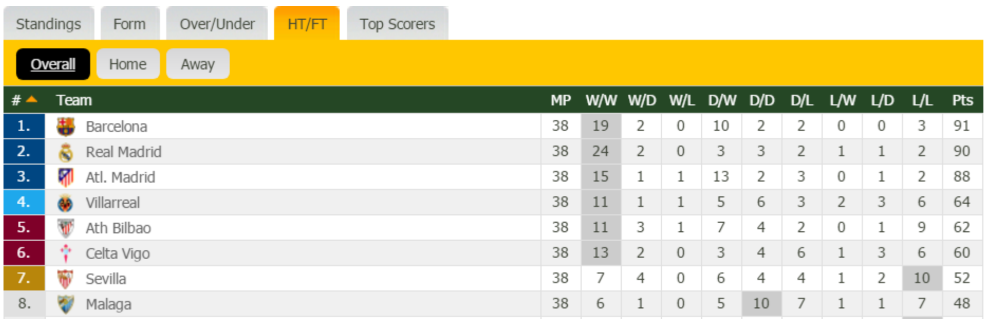
Under/Over table – contains a number of games with total goal under or over some special measure for each team in the league. Usually, the user is able to choose some specific total and see how many games each team has played under/over selected number. Often a table has three showing modes – all matches, home/away matches – same as the standing table. In some resources(all websites that use EnetPulse data provider) this table additionally has icons that shows results for the last five games.
Resources: EnetPulse and Opta clients, betstudy



		Total			Home			Away		
Pos	Teams	MP	U	O	MP	U	O	MP	U	O
1	Real Madrid	38	7	31	19	4	15	19	3	16
2	Barcelona	38	9	29	19	4	15	19	5	14
3	Atlético Madrid	38	22	16	19	9	10	19	13	6
4	Sevilla	38	15	23	19	9	10	19	6	13
5	Villarreal	38	22	16	19	9	10	19	13	6
6	Real Sociedad	38	18	20	19	9	10	19	9	10
7	Athletic Club	38	16	22	19	7	12	19	9	10

Figure 2.7: Sample of Under/Over table

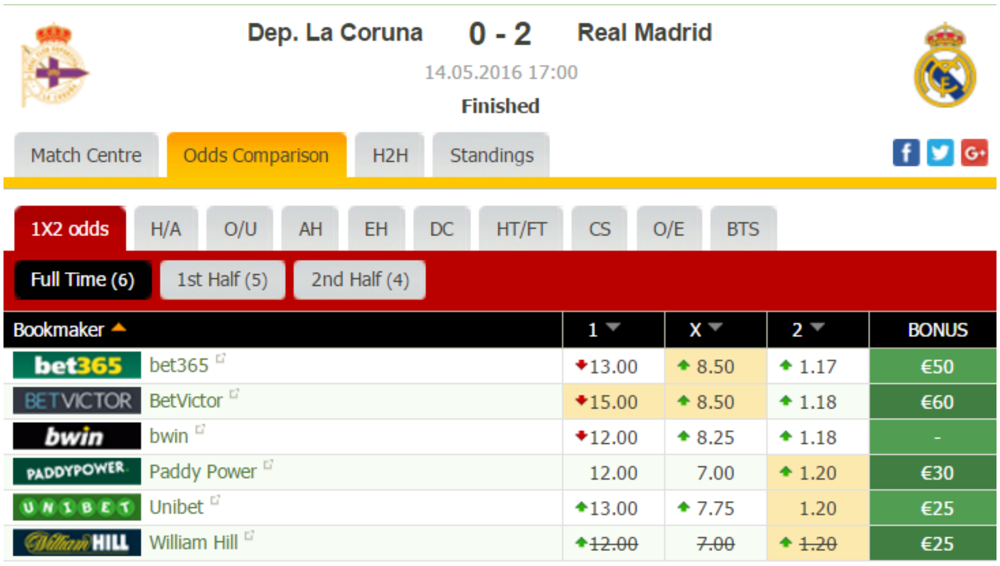
HT/FT table – Half Time / Full Time – one of the popular bet types which consists of betting at the same time on the half time and full time score of a match. Therefore, there are 9 possible results for football. This type of bet is also called Half Time / Final score, Half Time / Correct Score. This table contains amount of each result for each team in the league. It's very useful statistics for gamblers.
Resources: EnetPulse and Opta clients, soccerStats, betstudy



#	Team	MP	W/W	W/D	W/L	D/W	D/D	D/L	L/W	L/D	L/L	Pts
1.	Barcelona	38	19	2	0	10	2	2	0	0	3	91
2.	Real Madrid	38	24	2	0	3	3	2	1	1	2	90
3.	Atl. Madrid	38	15	1	1	13	2	3	0	1	2	88
4.	Villarreal	38	11	1	1	5	6	3	2	3	6	64
5.	Ath Bilbao	38	11	3	1	7	4	2	0	1	9	62
6.	Celta Vigo	38	13	2	0	3	4	6	1	3	6	60
7.	Sevilla	38	7	4	0	6	4	4	1	2	10	52
8.	Malaga	38	6	1	0	5	10	7	1	1	7	48

Figure 2.8: Top of the HT/FT table

Odds Comparison – another good feature for gamblers. A lot of bookmaker odds are gathered in one place. If someone wants to start betting on football, this information could be useful to identify most profitable bookmakers. Also it could be used for detecting betting forks[1].
Resources: OddsPortal, EnetPulse consumers

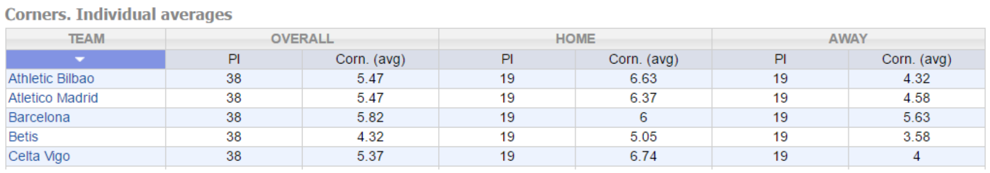


Bookmaker ▲		1 ▼	X ▼	2 ▼	BONUS
bet365	bet365	▼13.00	▲8.50	▲1.17	€50
BETVICTOR	BetVictor	▼15.00	▲8.50	▲1.18	€60
bwin	bwin	▼12.00	▲8.25	▲1.18	-
PADDYPOWER	Paddy Power	12.00	7.00	▲1.20	€30
UNIBET	Unibet	▲13.00	▲7.75	1.20	€25
William HILL	William Hill	▲~~12.00~~	~~7.00~~	▲~~1.20~~	€25

Figure 2.9: Odds Comparison: Deportivo La Coruña – Real Madrid

Tables covered specific stats
The resource (24score.org) provides next bunch of stats covered in the same way: three types of accumulated tables. Those stats are corners, fouls, cards, offsides and ball possession.

First type of provided table – individual averages. It contains all league teams, number of played matches and average number of particular statistical indicator.



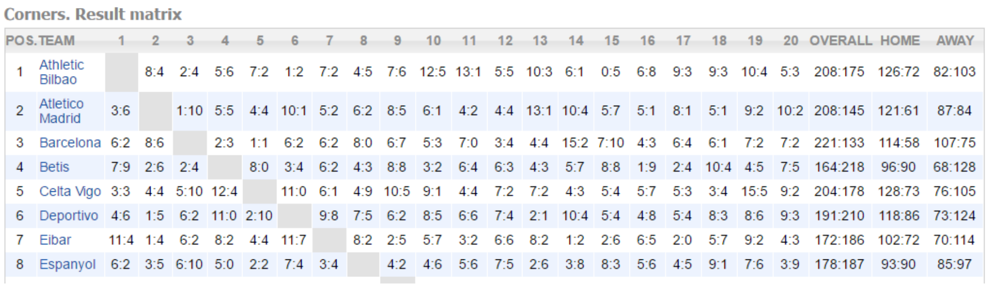
Corners. Individual averages

TEAM	OVERALL		HOME		AWAY	
▼	Pl	Corn. (avg)	Pl	Corn. (avg)	Pl	Corn. (avg)
Athletic Bilbao	38	5.47	19	6.63	19	4.32
Atletico Madrid	38	5.47	19	6.37	19	4.58
Barcelona	38	5.82	19	6	19	5.63
Betis	38	4.32	19	5.05	19	3.58
Celta Vigo	38	5.37	19	6.74	19	4

Figure 2.10: Example of individual averages table – corners

[1]betting forks - specific bet type, when player receives a guaranteed profit

Second one – result matrix. This matrix contains particular statistic scores of each played matches, in matrix format.

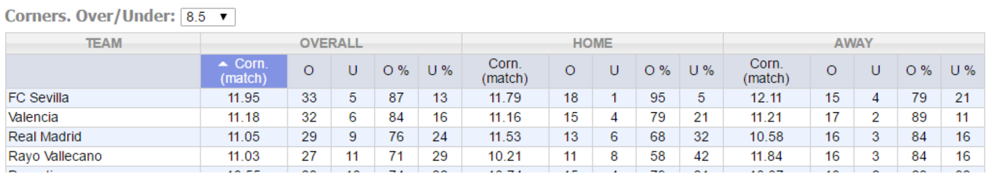


Corners. Result matrix

POS.	TEAM	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	OVERALL	HOME	AWAY
1	Athletic Bilbao		8:4	2:4	5:6	7:2	1:2	7:2	4:5	7:6	12:5	13:1	5:5	10:3	6:1	0:5	6:8	9:3	9:3	10:4	5:3	208:175	126:72	82:103
2	Atletico Madrid	3:6		1:10	5:5	4:4	10:1	5:2	6:2	8:5	6:1	4:2	4:4	13:1	10:4	5:7	5:1	8:1	5:1	9:2	10:2	208:145	121:61	87:84
3	Barcelona	6:2	8:6		2:3	1:1	6:2	6:2	8:0	6:7	5:3	7:0	3:4	4:4	15:2	7:10	4:3	6:4	6:1	7:2	7:2	221:133	114:58	107:75
4	Betis	7:9	2:6	2:4		8:0	3:4	6:2	4:3	8:8	3:2	6:4	6:3	4:3	5:7	8:8	1:9	2:4	10:4	4:5	7:5	164:218	96:90	68:128
5	Celta Vigo	3:3	4:4	5:10	12:4		11:0	6:1	4:9	10:5	9:1	4:4	7:2	7:2	4:3	5:4	5:7	5:3	3:4	15:5	9:2	204:178	128:73	76:105
6	Deportivo	4:6	1:5	6:2	11:0	2:10		9:8	7:5	6:2	8:5	6:6	7:4	2:1	10:4	5:4	4:8	5:4	8:3	8:6	9:3	191:210	118:86	73:124
7	Eibar	11:4	1:4	6:2	8:2	4:4	11:7		8:2	2:5	5:7	3:2	6:6	8:2	1:2	2:6	6:5	2:0	5:7	9:2	4:3	172:186	102:72	70:114
8	Espanyol	6:2	3:5	6:10	5:0	2:2	7:4	3:4		4:2	4:6	5:6	7:5	2:6	3:8	8:3	5:6	4:5	9:1	7:6	3:9	178:187	93:90	85:97

Figure 2.11: Top fragment of result matrix – corners

The third statistic type – Over/Under table. It's available only for corner and yellow card stats. The same table as for goals, but corner-/yellow card totals are considered instead of goal totals.

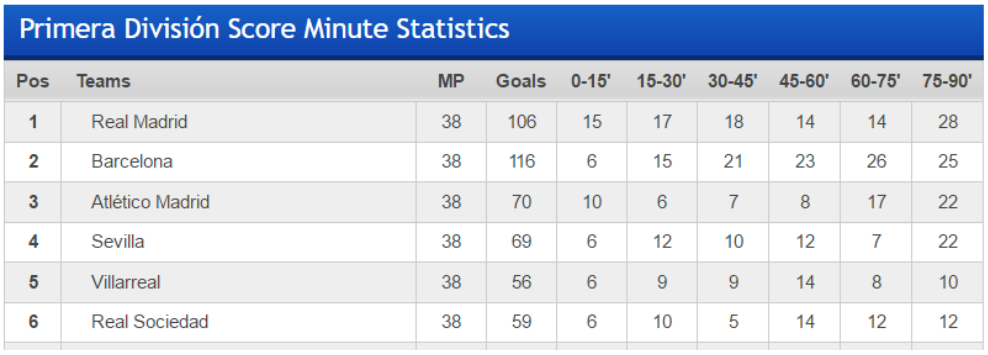


Corners. Over/Under: 8.5

TEAM	OVERALL					HOME					AWAY				
	Corn. (match)	O	U	O %	U %	Corn. (match)	O	U	O %	U %	Corn. (match)	O	U	O %	U %
FC Sevilla	11.95	33	5	87	13	11.79	18	1	95	5	12.11	15	4	79	21
Valencia	11.18	32	6	84	16	11.16	15	4	79	21	11.21	17	2	89	11
Real Madrid	11.05	29	9	76	24	11.53	13	6	68	32	10.58	16	3	84	16
Rayo Vallecano	11.03	27	11	71	29	10.21	11	8	58	42	11.84	16	3	84	16

Figure 2.12: Over/Under corner table

Scoring minutes table – for each team it shows the number of goals for every specific time range of the game. Commonly, whole match time is divided into 15 minutes ranges: 0-15, 15-30,. . . , 75-90.
Resources: betstudy, soccerStats



Primera División Score Minute Statistics

Pos	Teams	MP	Goals	0-15'	15-30'	30-45'	45-60'	60-75'	75-90'
1	Real Madrid	38	106	15	17	18	14	14	28
2	Barcelona	38	116	6	15	21	23	26	25
3	Atlético Madrid	38	70	10	6	7	8	17	22
4	Sevilla	38	69	6	12	10	12	7	22
5	Villarreal	38	56	6	9	9	14	8	10
6	Real Sociedad	38	59	6	10	5	14	12	12

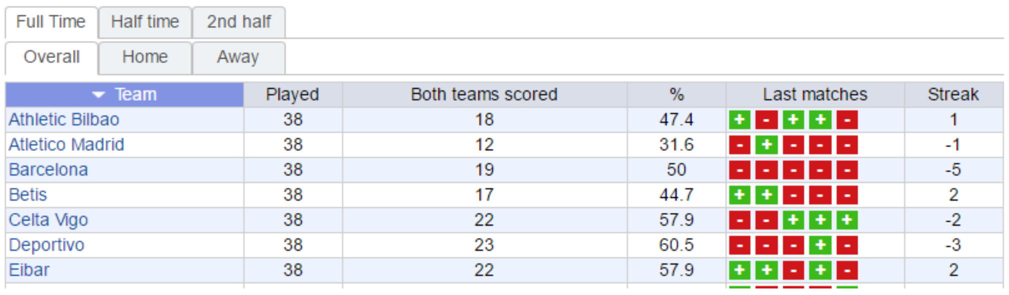
Figure 2.13: Sample of scoring minutes table

Tables with scoring numbers

In 24score.org the specific group of the stats is covered the same way – the table with all teams and with the following indicators:

- Number of played matches
- Number of games for which particular statistic is true
- Percentage of those games
- Stat results of last 5 matches
- Current streak

There are three types of stats in the group: both teams scored, failed to score, clean sheets.



Team	Played	Both teams scored	%	Last matches	Streak
Athletic Bilbao	38	18	47.4	+ - + + -	1
Atletico Madrid	38	12	31.6	- + - - -	-1
Barcelona	38	19	50	- - - - -	-5
Betis	38	17	44.7	+ + - - -	2
Celta Vigo	38	22	57.9	- - + + +	-2
Deportivo	38	23	60.5	- - - + -	-3
Eibar	38	22	57.9	+ + - + -	2

Figure 2.14: Example of the table – both teams scored.

Head To Head (H2H) stats – comparison of two teams. I have found only 2 groups of sites providing this stats – Opta and EnetPulse clients. Opta provides much more information in such comparison. EnetPulse provides solely form of two teams and history of their confrontation. Opta also provides those stats and in addition covers next stats:

- Table with calculated stats of all matches between two selected teams – number of wins/draws/losses, goals, points
- Player stats – topscorers, assists leaders, first goal scorers, most undisciplined players of each team
- Scoring minutes information
- Trophies records

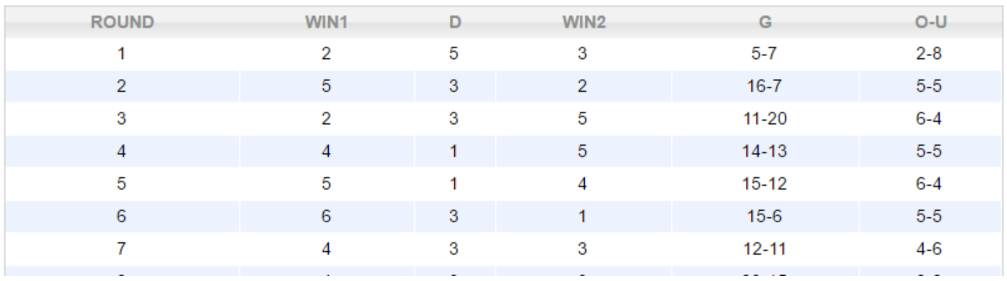
Resources: EnetPulse and Opta clients

Random statistical facts pop-up – I have found it only on one website – soccerStats. It’s available only for current season, for current situation in the league. Interesting facts randomly pop-up in some frequencies (every 5-7 seconds). Those facts are about winning or loosing streaks, streaks of match with/without goals, percentage of home/away wins, etc. Every fact is related to some concrete team. Examples:

82% of Real Madrid’s matches had over 2.5 goals scored in total.
Osasuna have lost 53% of their home matches.
Atletico Madrid are undefeated in their last 11 away matches.

Those facts could be useful for gamblers and football commentators. The latest can use it in boring moments of the match to keep alive the interest of the public.
Resources: soccerStats

Results by rounds – shows aggregated stats by rounds. Provided number of wins, draws and losses for each tour. In addition amount of goals (home-away) and number of games with total under/over 2.5 are provided.
Resources: 24score.com



ROUND	WIN1	D	WIN2	G	O-U
1	2	5	3	5-7	2-8
2	5	3	2	16-7	5-5
3	2	3	5	11-20	6-4
4	4	1	5	14-13	5-5
5	5	1	4	15-12	6-4
6	6	3	1	15-6	5-5
7	4	3	3	12-11	4-6

Figure 2.15: Results of first 7 rounds

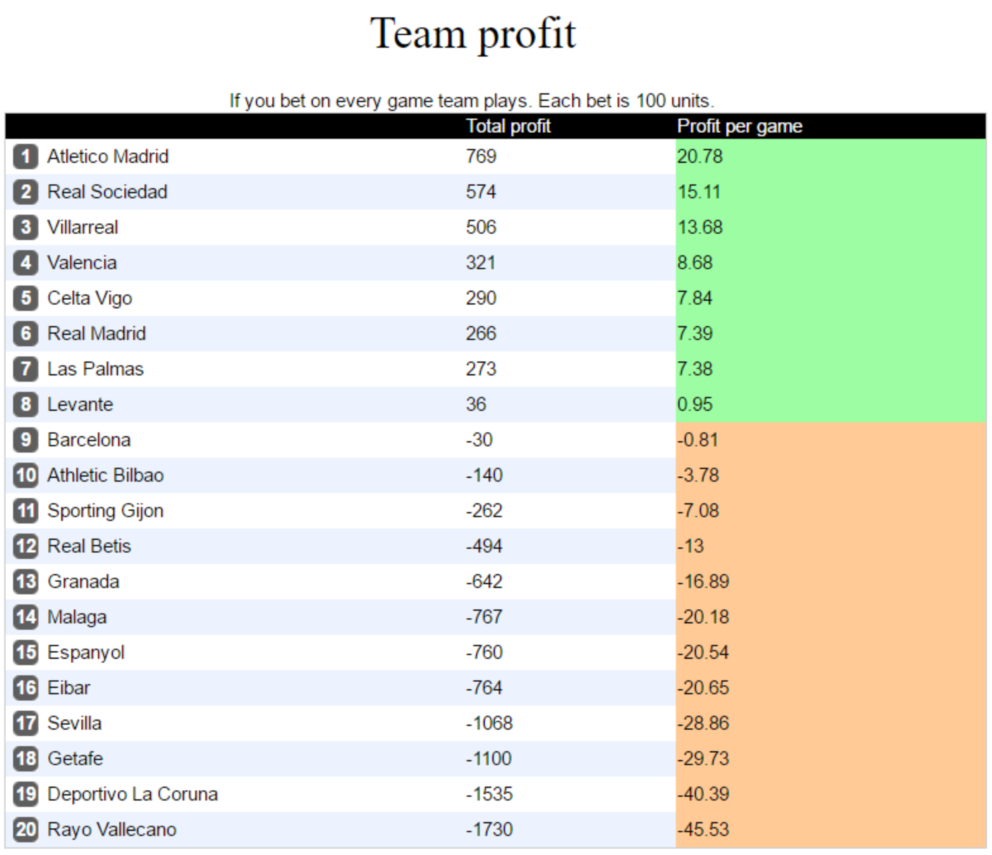
Referees table – aggregated information about referees. The table consists of next columns:

- Referee name
- Number of games he worked on
- Numbers of home team wins, draws and away team wins
- Number of all yellow cards and average number per game. Shows number of all cards, and separately home/away team cards.
- Red cards – same as yellow cards, only without average numbers.

Resources: 24score.com

Correct scores – aggregated statistic of all match scores. There are 2 ways how to consider no draw scores – to take or not to take into account which team won (home or away). If it is taken into account, scores 1-0 and 0-1 are different and should be calculated separately. In addition to numbers of games there could be provided a percentage of each score to number of all matches.
Resources: soccerStats, betstudy, futbol24.com

Team profit (betting) – one of the most interesting stats for gamblers. It's very specific and rare info, only one resource providing this data was found. This table shows how much you would win or lose if you bet for the win the same amount of money on your favorite team every match. It's very surprising that most winning teams do not bring much profit, because their wins are often estimated with small coefficients.
Resources: soccervista.com


Team profit

If you bet on every game team plays. Each bet is 100 units.

		Total profit	Profit per game
1	Atletico Madrid	769	20.78
2	Real Sociedad	574	15.11
3	Villarreal	506	13.68
4	Valencia	321	8.68
5	Celta Vigo	290	7.84
6	Real Madrid	266	7.39
7	Las Palmas	273	7.38
8	Levante	36	0.95
9	Barcelona	-30	-0.81
10	Athletic Bilbao	-140	-3.78
11	Sporting Gijon	-262	-7.08
12	Real Betis	-494	-13
13	Granada	-642	-16.89
14	Malaga	-767	-20.18
15	Espanyol	-760	-20.54
16	Eibar	-764	-20.65
17	Sevilla	-1068	-28.86
18	Getafe	-1100	-29.73
19	Deportivo La Coruna	-1535	-40.39
20	Rayo Vallecano	-1730	-45.53

Figure 2.16: Most profitable teams

Summary

In this section I have described a lot of interesting statistics that is provided by most popular resources. But of course I have not covered all features because of the huge variability of stats. Only most the common and interesting statistics have been covered. But still, I want to mention soccer-Stats as the site providing the most diverse statistics.

Chapter **3**

Design

In this chapter I am going to describe the architecture of the entire project, database structure and integration of all components integrated in the project. Therefore, the decision on the following aspects has to be made:

- Define general structure of a component
- Define what frameworks to use
- Define what design pattern could be used in particula cases
- Define how to integrate all components
- Define the way of data storage.

Development tools
Firstly, the technology and programming tools that will be used for the implementation of the program has to be defined. As was already mentioned in the previous chapter as non-functional requirement – the project would be implemented using programming language C# and .Net framework version >=4.5. In this case obvious option of IDE (Integrated Development Environment) is Visual Studio. I have decided to use last version of it - VS2017. As database for easier integration I decided to use also Microsoft product – MS SQL database and SQL Server Management Studio (SSMS) tool.

General architecture
Before designing each part of the system I want to describe general view of project structuer. The next Figure 3.1 illustrates physical separation of components on 2 servers.

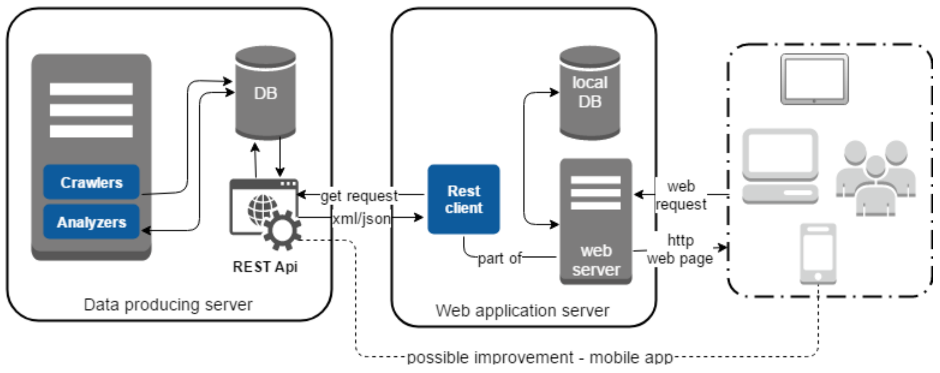


Figure 3.1: General architecture of the system

Web application server contains the web based GUI and database for storing logs and another information required for web app. As was described earlier in the project, GUI has to provide only data collected by crawlers and produced by analyzers. Additionally, special tool for searching specific stats will be implemented as part of the project. However, all business logic will be implemented on the server side. In case of further development of the project there are lots of possibilities for further improvements for web GUI – user registrations, chat for users, comments of predictions, etc. The future improvements will be described in more details in Conclusion.

In comparison to web application, *Data producing* server contains much more components then previous one. It is divided into the next logical parts: bookmaker odds and football historical data collectors, several analyzers and other helper modules. Previously mentioned components will be described in the chapters to come.

3.1 Data producing server

3.1.1 Data collectors

All data collectors and crawlers are structured in a similar way; the only difference is their implementation.
The following simple class diagram depicts that all data collectors implement IDataCollector interface, which provides only a few methods: Start(), Stop() and GetTimeout(). All data collectors are managed by TaskManager as a result, all manipulation (start or stop) is performed by this class. GetTimeout() method is closely connected with Stop() method, because it is used for restarting data collector in case it hangs out.

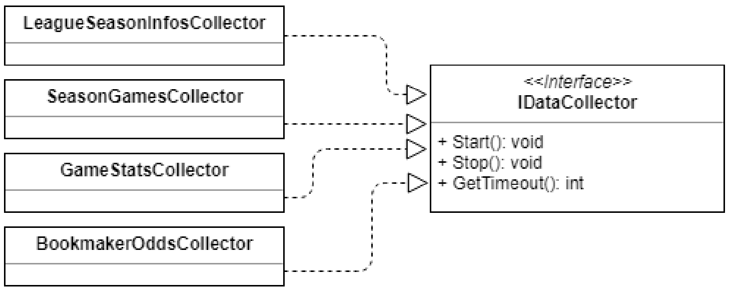


Figure 3.2: Class diagram of data collectors

Gathering of historical data
In Analysis part I have decided to use 2 sources of data – *Football-Data* and *Flashscore.com or analogues* [2.1.2]. In Football-Data source all provided data already structured and saved in CSV format, so data collectors just need to download all required files, parse them and store all data to local database. It's quite simple implementation. Its implementation doesn't required external frameworks and 3rd services because .Net framework has own libraries that allow to download file from internet (System.Web.WebClient) and to work with CSV files (System.IO).
In next source all data are placed on different pages, so I need to implement several data collectors for each page type. Its implementation could be divided to next steps: open a page and download its content, parse it, store all data to database.

1. In first step could appear one problem – sometimes page with data shows only a certain amount of all data, and for loading the rest of data user should do some action: click on a link, scroll to the bottom of page, etc. For solving this type of problem could be used headless browsers. It's a web browser without a graphical user interface, controlled programmatically. Mostly used for automation, testing, and web scrapping.

2. In second step program need to parse html page to some appropriate data structure. For this purpose we could use some helper tools that makes it easier. All possible tools and other frameworks are described in Frameworks section [3.3].

3. In third step program just store all parsed data to database. For interaction with database an object-relational mapping (ORM) could be used. But firstly I need to design database model based on analysis of gathered data varieties.

Figure 3.3 demonstrates the database model of historical data. There are tables used by LeagueSeasonInfo, SeasonGames and GameStats collectors:

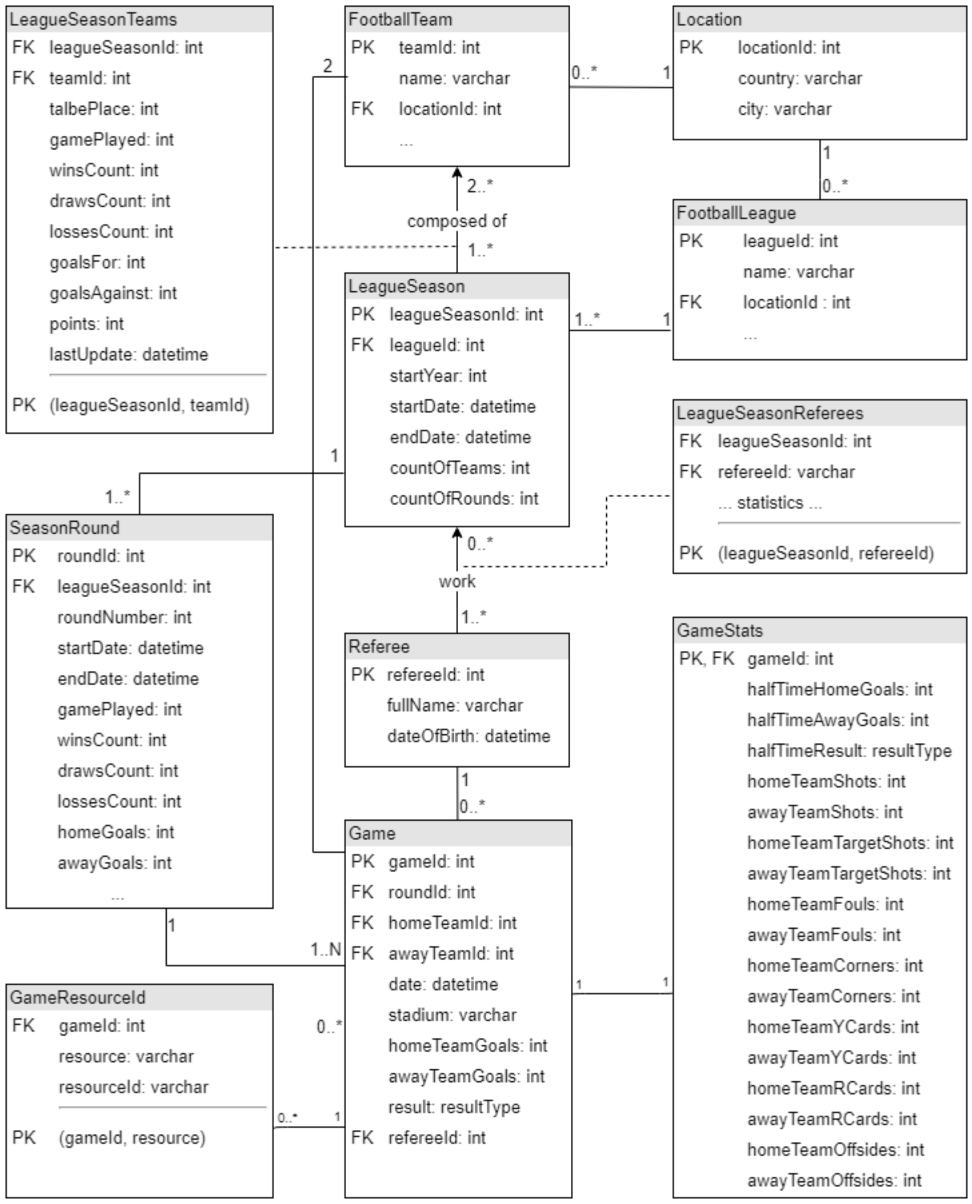


Figure 3.3: Class diagram of data collectors

LeagueSeasonTeams is known as associative table for resolving many-to-many relationships. In this case it represents all teams that played in the league in particular season. Besides of main associative columns – teamId and leagueSeasonId – it contains some statistics of the club in concrete season. Those columns will be used by analyzers to store their results.

Referee stands for referee, contains main info: full name, date of birth. As future improvement it could be expanded with next data – nationality, gender, photo, career start, etc.

LeagueSeasonReferees same as LeagueSeasonTeams table, only represents all referees that worked in particular season.

SeasonRound as the name implies it contains all season rounds with some stats, that will be generated by analyzers. Main indicators of each round are seasonId and round number.

Game the main information in whole database. Represents a game and contains next information: game date, competitors, round of a season, score, result and referee. Also could be gathered additional info, such as stadium name, game weather, attendance, etc.

GameStats this table has one-to-one relationship with Game table, so it just expands information about game. There are stats that gathered by data collectors. All those stats were mentioned in functional requirements section.

GameResourceId simple table that contains all game IDs in different resources. For example resource Flashscore.com in all games html page contains id for each game that could be used for opening page with statistics of a corresponding game.

Gathering of bookmaker odds

The process of bookmakers odds gathering is exactly the same as for gathering historical data from websites. All steps are identical, so I only need to defined database model. All bookmaker odds have been already described in functional requirements subsection. Therefore, I just put all of them to one table BookmakerOdds and additionally added table that represents a bookmaker:

- <u>Bookmaker</u> table columns: bookmakerId (int, PK), bookmakerName (varchar). Some additional info columns could be added.

- BookmakerOdds table columns: recordId (int, PK), gameId (int, FK), bookmakerId (int, FK), createdTime (datetime), match results = homeWin (float), draw(float), awayWin(float), double chances = 1X(float), 12(float), X2(float), BTTS-YES(float), BTTS-NO(float), Over2.5(float), Under2.5(float), half-time results = homeWin (float), draw(float).

Used abbreviations
database:
PK – primary key FK – foreign key
bookmaker odds:
1X – home team wins or draw
X2 – away team wins or draw
12 – home or away team wins
BTTS-YES(NO) – both teams to score - yes (no)
Over/Under2.5 – total of game over/under 2.5

3.1.2 Analyzers

All analyzers would implement the same interface, differences are only in their implementation. The main interface (IDataAnalyzer) provides only 2 methods: RunAnalyzing and GetTargetDataTable. First method is used for starting analyzing process and second one returns target database table that stores particular analyzer's results. There are all analyzer types that will be implemented in this project:

StandingTable analyzer – will fill columns of LeagueSeasonTeams table, that already have been presented in database model for historical data [3.1.1]. Next data for each team will be provided:

- game played (int)
- number of wins (int), draws (int) and losses (int)
- scored goals (goalsFor: int) and conceded goals (goalsAgainst: int)
- points (int) and position in table (int)

Target data table – LeagueSeasonTeams (see Figure 3.3)

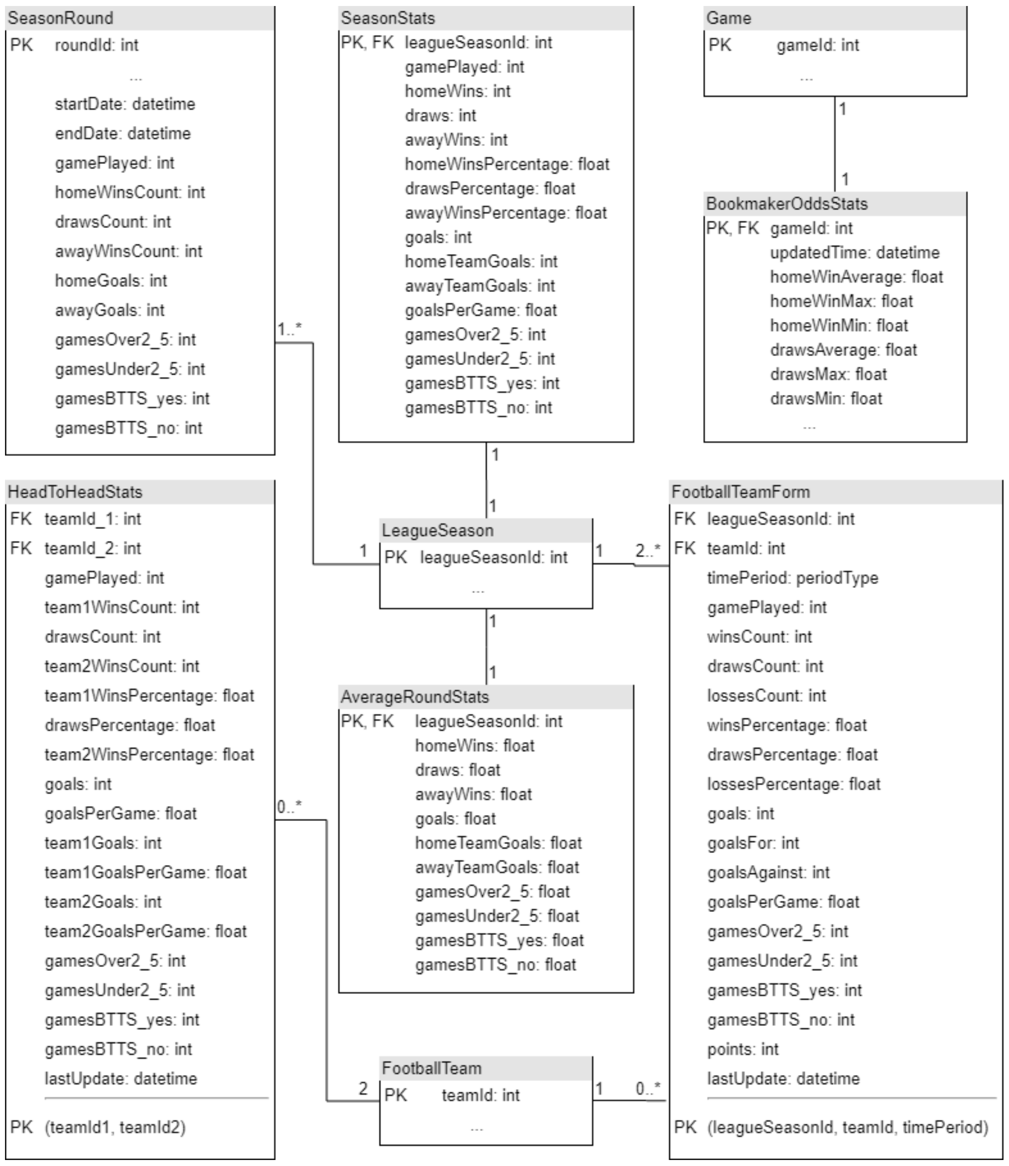


Figure 3.4: Class diagram of data collectors

SeasonStats analyzer – is responsible for main season stats. It will calculate next data:

- number of each type of result (home/away team win, draw) and their percentage
- count of all home/away team goals and the total number
- percentage and number of games where both teams scored

- percentage and number of games where game total more then 2.5 (and may be another totals too)

Target data table – SeasonStats (see Figure 3.4)

RoundsStats analyzer – will provide data for columns of SeasonRounds table, that also have been presented in database model for historical data [3.1.1]. Next data for each round will be calculated:

- game played (int) and number of wins (int), draws (int) and losses (int)
- home team goals (homeGoals: int) and away team goals (awayGoals: int)
- number of games where both teams scored (btts-yes: int) and opposite (btts-no: int)
- number of games where goals total more then 2.5 (over2_5: int) and opposite (under2_5: int)

Target data table – SeasonRounds (see Figure 3.4)

AverageRoundStats analyzer – shows stats of average round. Actually it provides same stats as SeasonStats analyzer but applied to number of round games.
[For example, if each round composed from 8 games and season stats percentages are: home wins – 50%, draws and away wins – 25% each, then round stats average numbers are: home wins = 4, draws = 2, away wins = 2.]
Target data table – AverageRoundStats (see Figure 3.4)

HeadToHeadStats analyzer – is responsible for stats of two teams comparison. Will be calculated for each two teams in league. Next statistics will be covered:

- number of each results (first/second team wins and draws) and its percentage
- number of games with total more then 2.5 and opposite
- number of games with both teams scored and opposite
- average game total and average totals of each team goals

Target data table – HeadToHeadStats (see Figure 3.4)

FootballTeamForm analyzer – provides stats of each football team form in the season. It contains same data as SeasonStats but for one particular team in selected time period. Next time periods will be covered: all season, last 10 matches and last 5 matches. Therefore, for each team in one season target data table will be contains 3 records for each time period.
Target data table – FootballTeamForm (see Figure 3.4)

BookmakerOddsStats analyzer – provides bookmaker odds stats. It just will calculate average and maximal/minimum values of each gathered odd types. This type of analysis will be conducted every time new bookmaker odds are gathered.
Target data table – BookmakerOddsStats (see Figure 3.4)

3.1.3 Game prediction

It is not an easy task to implement game prediction algorithm. There are a lot of information that should be taken into account, such as:

- Each match staring lineup of both team – players participated in the match
- Analysis of starting lineup for upcoming match and analysis of each player
- Information about missing match players – injuries, red cards, number of yellow cards and another causes
- Transfers and transfer rumors
- Another less relevant data which can affect the motivation of players:
  - Team manager dismissal of rumors about that
  - Personal problems of players
  - Significant dates for club: club foundation date, birthday of manager or owner and much more.

Most likely almost all of those facts are taken into account by bookmakers for analyzing and determining odds. It basically means that a very good prediction algorithm will provide a practically identical estimate of match as bookies.
A separate diploma thesis or research is needed to fully understand all caveats and issues that can arise during its implementation. Because of the fact, that this thesis lacks of this type of collected information, this

problem is out of the scope of this project. Nevertheless, I implemented simplified version of this algorithm that will use the following information:

- bookmaker odds as a probability of all outcomes
- rounds statistics
- league season streaks of results
- two teams head-to-head statistics and their streaks

3.1.4 Infrastructure components

The main logical units of the system – data collectors and analyzers – already have been designed and described. But there are should be another components for organizing and managing all those units. Next structure have been created: the main running application, will be run all time and decide when each data collector and analyzer to start or stop. For decision making will be used next helper class – *Scheduler* – that will provide information about units that should be started. And the last helper class is *StatsSearchingEngine*. It will provide methods for searching required stats from the entire database.

3.1.4.1 Main windows service

It should be long-running applications that run in the background repetitively without the need of any user interface or user interaction like Windows Forms Application, WPF Application, Console Application. So, the obvious decision in this case to implement this component as Windows service. There are several frameworks for Windows service implementation and one of them – Topshelf – have been chosen (see [3.3.2]).

3.1.4.2 Task scheduling

The data server should collect data from different resources and then analyze it, and for these purposes, crawlers and analyzers have been designed. But then the question arises: how often should crawlers and analyzers be started? The simplest approach is to define some frequencies for each crawler and run them strictly on the schedule. But this approach has weaknesses, which is clear after analyzing the schedule of games.

Let's take the case of La Liga. Normally a season round is played for four days: one match on Friday and Monday, and four games Saturday and Sunday. But some tours, few in a season, can take place on Tuesday and Wednesday. An usually two last rounds are played in one day all matches at the same time. Therefore, when determining the frequency for crawlers there appear next problems:

- If it is very high (f.ex. every hour), then 99% of the runs will do nothing, because new data appears only after the game.
- Otherwise, when frequency is low (f.ex. few times per day), then also a lot of tasks is unnecessary. And on days when many matches are played, the server could not show actual information for some time period.

Another obvious approach is to run the crawler after each match. For this approach the system need firstly to collect data about the schedule of season games, and crawlers will be launched 2 hours after each game. For gathering of season schedule I design additional type of crawler, that is similar as main games stats crawler. Also next database model was created (see Figure 3.5):

TaskExecutor stands for crawler or analyzer. An executor identified by name, some description, name of class in code and running parameters (maximal time for run and number of retries, if run failed).

Task represents one run of one crawler or analyzer. It contains next information: id number, status of run, time spent, result of run and executor that is responsible for the task.

LeagueSeasonReferees defines what task should be created after some another task is done. It includes information about two executors: one that done his task, and another one that should be run after. Also it specifies the time over which it’s necessary to run a new task.

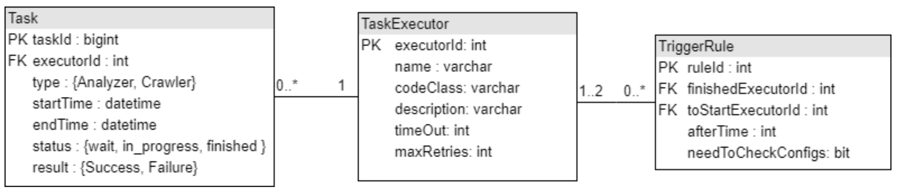


Figure 3.5: Scheduling database model

3.1.4.3 Statistics search engine

The last functional requirement of the project is specific interface for complex users queries. For its implementation I decide to design so-called Search Engine class that will be responsible for all complex stats searches. Also it could be used by analyzers.

Firstly, I need to define what methods the Search Engine should implement. According to mentioned functional requirement 4 methods have been defined:

- *GetGames(conditions)* – returns list of game objects (List<Game>) which fulfill conditions
- *GetGamesNumber(conditions)* – return number of all corresponding games. In essence this method does the same search as previous but returns less information. Therefore, should be faster then previous.
- *GetMaximalStreak / GetMinimalStreak (conditions)* – returns a series of consecutive games that fulfill conditions. Since there can be many results the method will return only first streak. Actually very rarely someone interested in minimal streaks, so this method probably will be removed in the future.
- *GetNumberOfStreaks (conditions)* – returns the number of suitable streaks. Besides standard conditions (described below) this method have additional input parameter – number of games in streak. This parameter composed of two values: comparing symbol (more,less,equal) and interesting number (1,2,..).

Secondly, all input parameters should be defined. All methods have similar set of input parameters – conditions. This set contains:

- *TeamId* – identification of interesting team.
- *Game place* – Home, Away or All. Used only when TeamId specified.
- *Result of game* – Team1, Draw or Team2. If TeamId is set the result will be interpreted as Win, Draw or Loss. Otherwise as HomeTeamWin, Draw or AwayTeamWin.
- *Game total of goals* – composed from 2 indicators: Over or Under and interesting number.
- *Both teams to score* – yes or no, simple boolean variable.

- *One team total of goals* – same as Game total but additionally has 3rd indicator: Team1 or Team2. If TeamId specified, Team1 = goals scored by selected team and Team2 = goals conceded. Otherwise, Team 1 = goals scored by home team and Team2 = guest team.

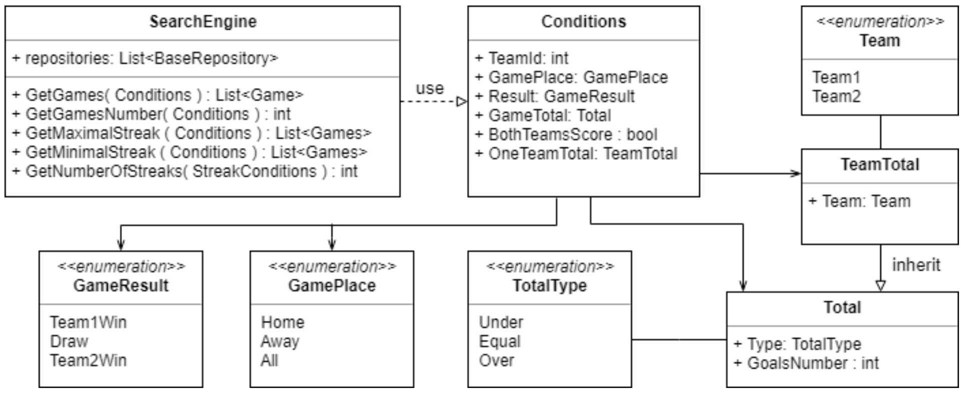

SearchEngine
+ repositories: List<BaseRepository>
+ GetGames(Conditions) : List<Game>
+ GetGamesNumber(Conditions) : int
+ GetMaximalStreak (Conditions) : List<Games>
+ GetMinimalStreak (Conditions) : List<Games>
+ GetNumberOfStreaks(StreakConditions) : int
use
Conditions
+ TeamId: int
+ GamePlace: GamePlace
+ Result: GameResult
+ GameTotal: Total
+ BothTeamsScore : bool
+ OneTeamTotal: TeamTotal
<<enumeration>> Team
Team1
Team2
TeamTotal
+ Team: Team
inherit
<<enumeration>> GameResult
Team1Win
Draw
Team2Win
<<enumeration>> GamePlace
Home
Away
All
<<enumeration>> TotalType
Under
Equal
Over
Total
+ Type: TotalType
+ GoalsNumber : int

Figure 3.6: Search engine class diagram

3.1.5 Logging

Why to implement

Logging is very important process, that is useful mainly for maintain the whole system. But actually there are 2 reasons for performing them: diagnostic and audit.

Diagnostic logging shows what your code is doing: what methods are called, define caller method, what parameters are used, and most important information about errors – code stack trace, error message, error type, etc. So, if an error occurred, developer can investigate the problem through logs and quickly define root of issue and fix it. That's why it is so important.

Audit logging is a business requirement. It captures significant events in the system, that is interesting for management or marketing. This is things like what request is more popular or from which location comes the greater number of requests, etc. For IT guys, who support the system, it's probably not very useful data. But for business purposes this can play an important role.

How to implement

For the implementation of logging there are several effective frameworks can be used. But at this stage of development I decide to implement logging by my own. Therefore, one database table for logs and simple Logger class have been designed. Figure 3.7 shows all columns of Log database table and methods that Logger will provide. As you can understand from the figure, there will be 3 levels of severity: Verbose, Information and Error. Verbose level used for every program activity while Information level is used for more important events. The data column could be used for storing any appropriate information for the log, so it will be depends on application or other columns.

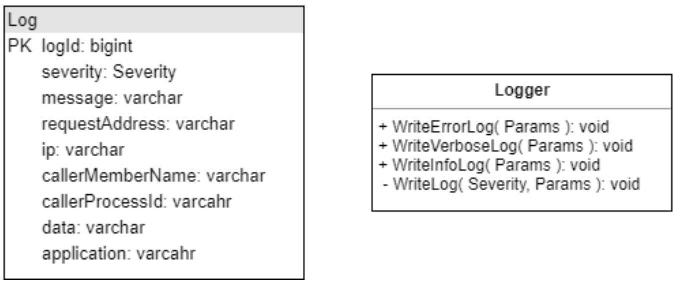

Log
PK logId: bigint
severity: Severity
message: varchar
requestAddress: varchar
ip: varchar
callerMemberName: varchar
callerProcessId: varcahr
data: varchar
application: varcahr
Logger
+ WriteErrorLog(Params): void
+ WriteVerboseLog(Params): void
+ WriteInfoLog(Params): void
- WriteLog(Severity, Params): void

Figure 3.7: Database Log table and Logger class

3.1.6 REST Api

The whole system is composed from 2 servers, where Web application server just represents data from Data producing server. So the project architecture could be interpreted as client-server application, where web application is a thin client. Therefore, I need to design how servers will communicate and interchange data between each other. I decide to use REST architecture, because it suitable for these purpose and easy implemented.

3.1.6.1 REST

REST is the abbreviation for Representational State Transfer. Basically it is a a design concept or architecture for managing state information. It defines several constraints:

- *Client-Server* – it is well-known network architecture. The main principle: all network units are servers or clients. A client component, desiring that a service be performed, sends a request to the server via a connector. The server either rejects or performs the request and sends a response back to the client [4].

- *Statelessness* – it means that the server doesn't store any information about client's previous requests. Each request is treated as an independent. That approach improve server scalability.

- *Caching* – it basically means that a client can save trips over the network by reusing previous responses from a cache.
- *Uniform Interface* – this term unites next 4 interface constraints:
  - *Resource-Based* – each individual resources is identified in requests using URIs as resource identifiers.
  - *Manipulation of resources through representations* – the server describes resource state in response so a client knows a representation of a resource, including any metadata attached. So he has enough information to modify or delete the resource on the server, provided it has permission to do so.
  - *Self-descriptive messages* – it means that each message includes all necessary information to describe how to process the message.
  - *The hypermedia constraint* – clients deliver state via body contents, query-string parameters, request headers and the requested URI. Services provide state to clients via body content, response codes, and response headers. This is technically referred-to as hypermedia.
- *Layered System* – it means that a client cannot usually detect whether it is connected directly to the end server, or to an intermediary along the way. Intermediary servers may improve system scalability by enabling load-balancing and by providing shared caches. Layers may also enforce security policies.
- *Code on Demand* (optional) – the server can send executable code in addition to data. This code is automatically deployed when the client requests it, and will be automatically redeployed if it changes.

Access to resources on a server should be provided through Uniform Resource Identifier (URI) and permit four basic operations: Create, Read, Update, Delete. These operations could be completely mapped to HTTP methods: POST, GET, PUT, DELETE. Server responses will contain a standard code of state, type of content and body with results.

3.1.6.2 List of endpoints

For the current requirements web application need only get data from data server, so all endpoints implement as GET operation. But for future development it could be expanded, and users would be able to add information to data server.
There are all methods provided by REST Api:

- **Leagues information**
  - */leagues/info/all* Get list of all available leagues with short info
  - */leagues/info/{leagueId}* Get more detailed information about requested league
  - */leagues/{leagueId}/allseasons* Get short info about all available seasons of required league
- **Seasons info and game stats**
  - */seasons/{seasonId}/standingtable* Get current standing table for required season
  - */seasons/{seasonId}/allgames* Get all finished games of the season with main results
  - */seasons/{seasonId}/matchstats/{id}/* Get all known stats of particular game
  - */seasons/{seasonId}/totalstats/table* Get table of game totals in the season (over/under 2.5)
  - */seasons/{seasonId}/btts/table* Get season table of both-team-to-score game stats (yes/no)
- **Season rounds information**
  - */seasons/{seasonId}/allrounds* Get current and all finished season rounds
  - */seasons/{seasonId}/round/{roundId}/games* Get all games of particular round
  - */seasons/{seasonId}/round/games* Get all games of current round
  - */seasons/{seasonId}/round/table* Get table of main stats for each rounds
  - */seasons/{seasonId}/round/stats* Get average stats of all season rounds
- **Teams information**
  - */seasons/{seasonId}/team/all* Get list of all teams in the season
  - */seasons/{seasonId}/team/{teamId}* Get season stats of particular team
  - */seasons/{seasonId}/team/{teamId}/form/{type}* Get list of all season games of one team [Possible types: all, home or away]

  - */seasons/{seasonId}/team/{teamId}/form5/{type}* Get list of last 5 season games of one team
  - */seasons/{seasonId}/team/{teamId}/form10/{type}* Get list of last 10 season games of one team team
  - */seasons/{seasonId}/h2h/{team1Id}/{team2Id}* Get head-to-head stats of two teams

- **Predictions**

  - */seasons/{seasonId}/predictions* Get all current predictions of upcoming games
  - */seasons/{seasonId}/predictions/all* Get all predictions of all season games
  - */seasons/{seasonId}/predictions/finished* Get all evaluated predictions and their stats

- **Searching tool**

  - */searching/games/{params}* Get all corresponding games
  - */searching/games/number/{params}* Get number of appropriate games
  - */searching/streak/maximal/{params}* Get one maximal streak of suitable games
  - */searching/streak/minimal/{params}* Get one minimal streak of corresponding games
  - */searching/streak/number/{params}* Get number of appropriate streaks

3.2 Web application server

3.2.1 Architecture

As was mentioned before, for development of the entire project I choose to use C# and .Net framework. For web application development .Net framework has ASP.NET platform. ASP.NET is an open source web framework for building modern web apps and services with .Net [2]. It provides 2 main approaches for web application development: Web Forms and MVC.

Web Forms Microsoft first brought out ASP.NET Web Forms. It provides an abstraction over the HTTP protocol to make development closer to traditional desktop development and managing state information easier. The main feature of this approach is ViewState, an information that are transferring at header of requests and responses. This makes possible to create statefulness web application. But also it is main problem for web application performance, because ViewState is stored in the page itself resulting increased page size.

ASP.NET MVC ASP.NET MVC is a Microsofts one more Web application framework designed with separation of concerns and testability in mind. It is built on CLR[2] and completely based on MVC architecture and so we think in terms of controllers and Views. It has in most cases better performance, more understandable project structure, full control over HTML and several other features compared with Web Forms. For better understanding the idea and workflow of ASP.NET MVC please see Figure 3.8.

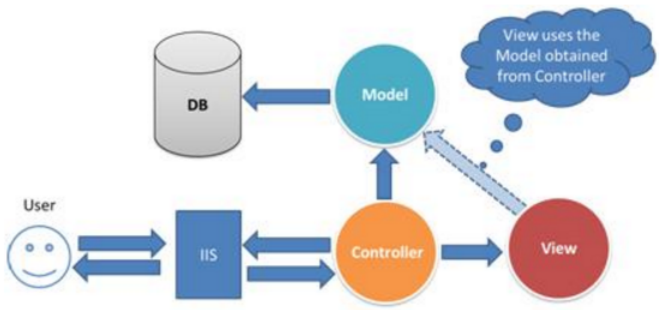


Figure 3.8: How ASP.NET MVC works. Source: [7]

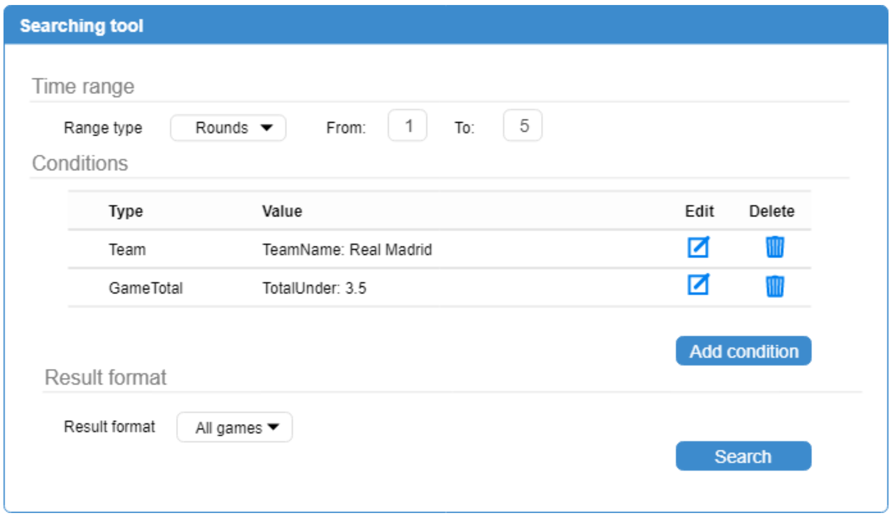
ASP.NET MVC is more modern and in most aspects better framework, so I decided to use this framework for web application development.

3.2.2 Searching tool

All required functionalities have been already described in Analysis chapter. I just need to define the way how it could be implemented. The main aspect that should be taken into account is a user-friendly interface. Figure 3.9 shows the design of the page with this searching tool. As you could see, There are three parts with set parameters: time range, conditions and result form. When the user opens this page, Time range part looks like Result format: only selection of range type is available. After the user chooses it, the page automatically adds next UI components depended on the choice. Conditions part should consist from 1 to 6 conditions. For adding new condition user have to click to Add condition button and then

[2]CLR – the virtual machine component of Microsoft’s .NET framework

fulfill pop-up form. The pop-up form for new conditions works in the same way as Time range. User firstly has to choose a type of condition, and then a form will be supplemented by next required input fields. When all fields are filled out the user can click to Save button and a new condition is added to the table on the page. Condition editing works, in the same way, using pop-up form.



Type	Value	Edit	Delete
Team	TeamName: Real Madrid		
GameTotal	TotalUnder: 3.5		

Figure 3.9: Searching tool mock-up

3.2.3 Database

The web application server need to store their own data. It should contains logging information and history of request to Data Producing server. For current goals there are not lots of data have to stored to DB, but for future improvements the database model will be significant expended.

The current database model will be contain only 4 tables: Log, RequestHistory, ResponseHistory and RequestParams. The Log table structure will be described in next section. The other three tables are shown on Figure3.10

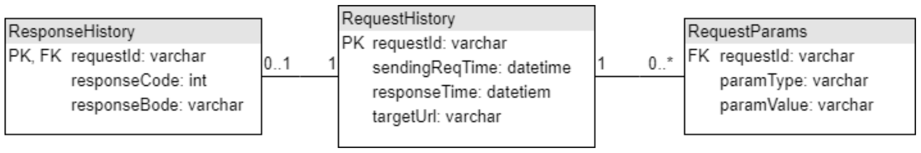


Figure 3.10: Database model for web application

3.3 Frameworks

In this subsection, I am going to briefly describe all frameworks that have been chosen for the project implementation. I have already mentioned some of them in previous sections in this chapter.

3.3.1 Nancy

Nancy is a lightweight, low-ceremony, framework for building HTTP based services on .NET and Mono [10]. It is inspired by the Sinatra framework for Ruby. Therefore, Nancy was named after the daughter of Frank Sinatra. Nancy is designed to handle all types of requests: DELETE, GET, HEAD, OPTIONS, POST, PUT and PATCH. This framework takes care about all web service specifics, so a developer could focus on more important things: a logic of his application. It's built, by the community, as an open-source framework. So every developer gets full access to the source code. Also, it is licensed under the MIT license.

3.3.2 Topshelf

Topshelf is a Windows service framework for the .NET platform. It greatly simplifies the creation of a Windows service, especially testing, debugging processes and installation into the Windows Service Control Manager (SCM). This is a useful framework for developers, so they can focus on service logic instead of the details of interacting with the built-in service support in the .NET framework. Developers dont need to understand the complex details of service classes, perform installation via InstallUtil, or learn how to attach the debugger to services for troubleshooting issues [11]. It' very flexible framework that supports most of the commonly used service installation options. Additionally, it is a lightweight tool (around 200KB) and it's also an open-source project. Also, it works with Mono, so it possible to deploy services to Linux.

3.3.3 PhantomJS

PhantomJS is a headless WebKit scriptable with a JavaScript API. It has fast and native support for various web standards: DOM handling, CSS selector, JSON, Canvas, and SVG [12]. This framework allows developers to access the browsers DOM API. It's helpful utility for Web scrapping. Using PhantomJS crawlers could get data, which is available only after some user's actions on the page. For example, for getting some information on some page user should click on next button. Then it sends AJAX request to the server, and after getting response required data will appears on the page. For automating this process, this framework could be used.

3.3.4 Entity Framework

Entity Framework (EF) is an Object Relational Mapping (ORM) framework that enables .NET developers to work with relational data using domain-specific objects. It eliminates the need for most of the data-access code that developers usually need to write [13]. This framework provides three approaches for different scenarios:

- Database first – if database already have been created or the developer wants to design database ahead of other parts of the application
- Code first – if the developer wants to focus on domain classes and then generate the database from created domain classes
- Model first – if the developer wants to design the database schema on the visual designer and then creates the database and classes.

There are many other ORM frameworks for .NET in the market, but the fact, that EF is an open-source project and it is provided by Microsoft, distinguishes it from alternatives.

Chapter **4**

Implementation and testing

Guided by the tasks of this thesis, I had to implement the application for gathering football statistical data and analyzing it. In previous chapters, the analysis and design of the application have been provided. Based on results of the design part, two parts of the project have been realized. Also, the REST API was created. During the whole implementation process, I followed the architecture designed in the previous chapter. There are too many classes, so I'm going to describe only most interesting of them:

- REST implementation
- Prediction algorithm
- Statistics searching

4.1 REST implementation

In this section, I am going to describe the implementation of REST API. It divides into two parts: server and client. The server part will be implemented on the data providing server and a client will be used by web application for querying data.

4.1.1 Server

In design part, I already have defined the list of endpoints for REST API. Therefore, I just need to implement all those methods which are basically selections of data from the database. Because I decided to use Nancy framework for its implementation, it very simplifies this task.

The next listing 4.1 depicts how easy it seems to implement one of the API methods.

Listing 4.1: An example of REST method implementation

```
LegueModule.cs
...
using System;
using FIT.Diploma.Server.DataAccess;
using FIT.Diploma.Server.DataAccess;
using FIT.Diploma.Shared.Service;
using Nancy;

public class LeagueModule : NancyModule
{
    public LeagueModule() : base("/leagues")
    {
        // "/leagues/info/all" endpoint implementation
        Get["/info/all"] = parameters => {
            var leagueRepo = new LeagueRepository();
            var dbLeagues = leagueRepo.GetAll();
            List<LeagueObject> result = ObjectMapper.Convert( dbLeagues );
            return Response.AsJson( result );
        };

        // "/leagues/{leagueId}/allseasons" endpoint implementation
        Get["/{leagueId}/allseasons"] = parameters => {
           ...
        };
        ...
     }
}
```

Then I just need to implement other methods the same way.

4.1.2 Client

There are few options how to implement client:

- *HttpWebRequest* – the .Net pioneer class from for maximal control of request
- *HttpWebRequest* – a higher-level abstraction built on top of HttpWebRequest. It has less control of request. However, it is easier to use and requires less code
- *HttpWebRequest* – The modern version of WebClient with some new features. For example, there is availability to send requests asynchronously, that could improve the performance of the application. It is available only for the latest .Net framework versions (4.5+).
- *HttpWebRequest* – The alternative that is developed by the community. It is one of the only options for a portable, multi-platform,

unencumbered, fully open-source HTTP client that you can use in all of your applications [14].

First three classes, which are provided by Microsoft, have a dependency on .Net framework and could be run only on Windows OS, whereas RestSharp is an open-source project created for use in other platforms with .Net Core.

This project is fully implemented in C# and .Net framework. Therefore, it doesn't require multi-platforming features. Therefore, I have decided to use HttpClient for better performance of the application. The listing below 4.2 shows an example of a REST call.

Listing 4.2: An example of REST client call

```
RestClient.cs
...
public List<League> GetLeagues()
{
    HttpClient client = new HttpClient();
    client.BaseAddress = new Uri("http://localhost:9000/");
    client.DefaultRequestHeaders.Accept.Add(
                new MediaTypeWithQualityHeaderValue("application/json"));
    HttpResponseMessage resp = await client.GetAsync("leagues/info/all");
    if (resp.IsSuccessStatusCode)
    {
        // response parsing and mapping to the reqired format
        ...
    }
}
...
```

4.2 Prediction algorithm

As was mentioned in Design part, for predictions of game result next parameters were used:

- bookmaker odds
- rounds statistics
- league season streaks of results
- head-to-head statistics of competitors

4.2.1 Bookmaker odds

From that parameter, I could get the probabilities of each result by the opinion of bookmaker analytics. All bookmaker odds in the project are collected in decimal format. For calculating the implied probability from it, the next equation could be applied:

$$(1/decimalOdds) * 100 = impliedProbability \tag{4.1}$$

Therefore, the first step of algorithm is to calculate all results implied probabilities.

4.2.2 Rounds statistics

This parameter shows the season rounds statistics, and compares it with the similar stats of previous seasons. It means that algorithm takes into account numbers of home wins, draws and away wins per round in current season. Then it calculates the same numbers for previous seasons and makes predictions on whole round based on this stats.

4.2.3 League season streaks

This parameter indicates the current league situation in the season. Basically, it calculate the season games streaks with one of the next conditions:

- Games without draw
- Matches where only home teams win or no guest wins
- Matches where only away teams win or no home team wins

Therefore, if there is some streak, the algorithm will increase the probability of the result that could end the streak. This strategy is based on the fact, that all streaks come to an end at some point.

4.2.4 Head-to-head statistics

This parameter is based on similar principles as previous two, but only games between 2 teams are taken into account.

4.3 Statistics searching

For searching specific statistics, I have designed and implemented the web interface. A web application part is just a form on webpage in which the user specifies the search parameters. Then web application sends all parameters as a JSON file to the data providing server by the Rest client. Rest server on the data server side receives the request and delegates the search task to Search Engine.
The Search Engine class has to implement ISearchEngine (see on the listening 4.3).

Listing 4.3: ISearchEngine.cs

```
public interface ISearchEngine
{
    List<GameObject> GetGames(GameParameters conditions);
    int GetGamesNumber(GameParameters conditions);
    List<GameObject> GetMaximalStreak(GameParameters conditions);
    List<GameObject> GetMinimalStreak(GameParameters conditions);
    int GetNumberOfStreaks(GameParameters condition, StreakParameters
        streakCond);
}
```

In fact, the implementation of each method then looks like a generation of SQL SELECT query. But because I use Entity Framework for accessing the database, it is not needed to create actual SQL query. The next listening 4.4 shows the fragment of the *GetGames()* method implementation.

Listing 4.4: SearchEngine.cs

```
public class SearchEngine : ISearchEngine
{
    List<GameObject> GetGames(GameParameters conditions){
        var gameRepo = new GamesRepository();
        var gameStatsRepo = new GameStatsRepository();
        var timePeriod = ParametersMapper.GetTimePeriod(
                                            conditions.TimeRange);
        var result = gameRepo.GetAll( timePeriod );
        if(!string.IsNullOrEmpty(conditions.TotalCondition)) {
            result = result.Where( ... );
        }
        ...
        return result;
    }
    ...
}
```

After the SearchEngine returns some results, Rest server converts it to JSON format and send the response to the Rest client.

4.4 Testing

Software testing is an investigation conducted to provide stakeholders with information about the quality of the software product or service under test [9]. Essentially, it can verify if the product meets all defined requirements. Moreover, testing could detect bugs and unexpected program behavior. Additionally, the proper level of test coverage allows to provide refactoring of the project without fear to break some functional parts.

4.4.1 Unit tests

Unit testing is one of the testing types that is focused on verifying the correctness of separate units functionalities. All tested components should be tested in isolation from other system units. Unit tests are very close to the code itself, and they test mainly a pure functionality of units without understanding any business logic. They can be created one time and run every time that source code is changed to make sure that no bugs are introduced [15].

For testing purposes, I have chosen the following parts of the project: crawlers, analyzers, search engine and repositories. For testing REST components, integration tests could be created in further development. For the unit test implementation, the NUnit framework has been chosen.

4.4.1.1 NUnit framework

NUnit is a unit-testing framework for all .Net languages. Initially ported from JUnit, the last production release has been completely rewritten with many new features and support for a wide range of .NET platforms. It is open-source software and NUnit 3.0 (last version) is released under the MIT license. Earlier releases used the NUnit license Both of these licenses allow the use of NUnit in free and commercial applications and libraries without restrictions [16].

4.4.1.2 Summary

All main parts of the project have been covered by unit tests. Its implementation was written in accordance with the guide from that resource [17].

4.4.2 Prediction evaluation

For evaluation the quality of prediction algorithm I have applied it to 10 season rounds (=100 games). In result, there were 57 success and 43 fail predictions. But those numbers don't show the quality of algorithm. It's better to consider the profitability of predictions. Because a lot of success predictions had a small coefficient, the profitability of testing cases was a negative number. If I bet for each prediction the 100 currency units, in the result, I would lose 857 units.
In this project, I have tested only one variant of the algorithm, with one set of parameters. But it's a good topic for future improvement.

This table shows how much you would win or lose if you bet for the win the same amount of money on your favorite team every match. It's very surprising that most winning teams do not bring much profit, because their wins are often estimated with small coefficients.

Conclusion

The primary task of this thesis was to design and implement the system for gathering football statistics and carrying out analysis of it. The secondary task was to provide collected data and analysis results to the user. Both tasks were implemented and there are some strong sides and as well some flaws in implementation and design that occurred.

In the Analysis chapter, use cases and requirements for the program were defined, and as well some other analysis results, that can be found [1]. After determining requirements and target football league, the research of all suitable resources and existing approaches was provided [2]. This study revealed the broad range of statistics that could be afforded by this kind of project. Also, it helped to choose in which format this system should provide statistics.

In the next chapter [3], the design of the project was described. The whole system architecture was separated into the web application server and the data server. For communication between those parts, REST API was created. Also, the main structural components and database models were designed. Additionally, I briefly described all main frameworks that were used for implementation.

Based on the analysis and design parts, I have implemented the target application. The most interesting implementation parts have been described in that chapter [4]. Also, all main parts of the project were covered by tests. As a result, the functional system was created. Both non-functional and functional requirements have been fulfilled. But because of the fact, that the main problem of this project is broadening, the quality of each implemented components can be improved in the future development.

To conclude I would like to say that this thesis was a great benefit for me in which I have learned and tried in practice several interesting technologies. Personally, I see a promising potential of this project, the future of which I have briefly described in the next section.

Future work

The current thesis has a lot of work that can be implemented in the future. The most important features are described below.

Optional requirements not all optional requirements were implemented, because of the lack of data. Therefore, it could be taken as future work for searching new data sources and implementing new data collectors for them.

Additional statistics due to the research of existing solutions [2], lots of statistical types was defined. And all of them or the most popular of them should be implemented in this project. Thereby, I could create the first web application that contains all possible statistics in one place.

An attractive GUI in the project, the web application was created as a user interface. But there was the minimal amount of works with styles and JavaScript. Therefore, the output of the project doesn't look as attractive as I wanted it to be.

Mobile application in the future analysis of users and their devices should be provided. And if a significant number of users comes to the webpage through mobile phones, mobile version of the web application has to be implemented.

Bibliography

[1] EnetPulse. *THE MOST RELIABLE PROVIDER OF SPORTS DATA [online]*. [Accessed: 2017-06-30]. Available from: `http://www.enetpulse.com/`

[2] Microsoft Corporation. *ASP.NET [online]*. [Accessed: 2017-06-29]. Available from: `https://www.asp.net/`

[3] totalSPORTEK.com. *25 Worlds Most Popular Sports (Ranked by 13 factors) [online]*. [Accessed: 2017-06-29]. Available from: `http://www.totalsportek.com/most-popular-sports/`

[4] Leonard Richardson, S. R., Mike Amundsen. *RESTful Web APIs Services for a Changing World*. O'Reilly Media, 2013, ISBN 978-1-449-35806-8.

[5] Sitefinity.com. *ASP.NET MVC or Web forms [online]*. [Accessed: 2017-06-28]. Available from: `http://docs.sitefinity.com/for-developers-asp-net-mvc-or-web-forms`

[6] Martin Fowler, M. F., David Rice. *Patterns of Enterprise Application Architecture*. Addison Wesley, 2002, ISBN 0-321-12742-0.

[7] CodeProjects. *WebForms vs. MVC by Marla Sukesh[online]*. September 2014, [Accessed: 2017-06-29]. Available from: `https://www.codeproject.com/Articles/528117/WebForms-vs-MVC`

[8] RestApiTutorial.com. *What Is REST? [online]*. [Accessed: 2017-06-24]. Available from: `http://www.restapitutorial.com/lessons/whatisrest.html`

[9] Erik Hazzard's Blog. *How Logging Made me a Better Developer [online]*. [Accessed: 2017-06-20]. Available from: `http://vasir.net/blog/development/how-logging-made-me-a-better-developer`

[10] NancyFx. *Nancy documentation [online]*. [Accessed: 2017-06-29]. Available from: `https://github.com/NancyFx/Nancy`

[11] Topshelf. *Topshelf Key Concepts*. [Accessed: 2017-06-29]. Available from: `http://docs.topshelf-project.com/en/latest/overview/faq.html`

[12] PhantomJS. *PhantomJS overview [online]*. [Accessed: 2017-06-29]. Available from: `http://phantomjs.org/`

[13] EntityFrameworkTutorial.net. *What is Entity Framework? [online]*. [Accessed: 2017-06-28]. Available from: `http://www.entityframeworktutorial.net/what-is-entityframework.aspx`

[14] The Geeky Gecko. *WebClient vs HttpClient vs HttpWebRequest [online]*. [Accessed: 2017-06-30]. Available from: `http://www.diogonunes.com/blog/webclient-vs-httpclient-vs-httpwebrequest/`

[15] Microsoft MSDN. *Working with Unit Tests [online]*. [Accessed: 2017-06-29]. Available from: `https://msdn.microsoft.com/en-us/library/ms182515(v=vs.90).aspx`

[16] NUnit.org. *What is NUnit? [online]*. [Accessed: 2017-06-30]. Available from: `http://www.nunit.org/`

[17] Dimecasts.Net. *Creating tests with NUnit [online]*. [Accessed: 2017-06-29]. Available from: `http://dimecasts.net/Casts/CastDetails/`

Appendix **A**

Acronyms

GUI	Graphical user interface
XML	Extensible markup language
REST	Representational State Transfer
HTML	Hypertext Markup Language
JSON	JavaScript Object Notation
API	Application Programming Interface
ORM	Object Relational Mapping
JS	Javascript
SQL	Structured Query Language
OS	Operating System
CSV	Comma-separated values
DOM	Extensible markup language

Appendix B

Contents of enclosed CD

```
 readme.txt ........................ the file with CD contents description
 publish ............ the published web application prepared for deploy
 src ......................................... the directory of source codes
   server ................................ data producing server sources
   web app ..................................... web application sources
   shared ..................... shared sources that used in both servers
   thesis ............. the directory of LaTeX source codes of the thesis
 text ............................................. the thesis text directory
   thesis.pdf ............................ the thesis text in PDF format
   thesis.ps ................................ the thesis text in PS format
 attachments ................. the directory with electronic attachments
```