HoopStats
Design Description

Version 0.1

Revision History

Date	Version	Description	Author
2012-11-09	0.1	Initial Draft	Andreas Köhle, Bal Krishna Nyaupane, Predrag Filipovikj, Igor Šarić, Dino Blažeka, Armindo Simões

Table of Contents

1. Introduction

The HoopStats project is intended as a reimaging of the web application databasebasketball.com. This application provides basketball statistics which are free to use to anybody. Unfortunately it does not provide any features for visualization of statistics except tables and plain text. Furthermore, there is limited number of choices of usage in terms of which data the user wants to display.
The goal of the HoopStats project is to take the free-to-use data from the web and create a new system, which would provide more dynamic and visually appealing approach. The application provides mechanism to users for creating queries. This is done by giving them an option in which users can choose from different templates for querying the application. Here the users can choose e.g. a player or team and select the data that they are interested in. After that they can combine this template with conditions to filter out the data that is not of interest.
Furthermore the data output will be replaced with a more modern visual style. This style will be more users friendly and will allow the users to easily interpret what is presented to them.

In addition to the visual changes a mobile application for Android phones will be developed, as mobile devices become more and more important for users and especially for the avid basketball fan for which the application aims at.

The following list summarizes the main features of HoopStats again:

- Users can make their own basketball related queries
- Visuals of application are improved by using modern visual output
- Mobile application to use HoopStats on-the-go

1.1 Purpose and Scope of this document

The purpose of this document is to provide information about the detailed design of HoopStats project, project team and all of the respective stakeholders. The document is created in the initial design phase of the project and is revised and updated after each development cycle and during the cycles if it is necessary.

1.2 Intended Audience

The document is intended for the following stakeholders:

- Project members
- Project supervisor
- Customer
- All stakeholders who constantly monitor the project
- Potential future developers
- SCORE competition judges

The project members take references from this document while implementing HoopStats. The customer and supervisor can familiarize themselves about design choices made by the project team and get more detailed picture about the underlying architecture. For potential future developers this document can be used as a reference to make them familiar with the application and the design choices.

1.3 Scope

The main scope of this document is to provide insight of architecture and the detailed design of HoopStats. It mainly includes software architecture, detailed software design and external interfaces. It explains the technology and architecture which are going to be used. It also explains the UI, detailed database design, system backend and web application architecture.

1.3.1 *Acronyms and abbreviations*

Acronym or abbreviation	Definitions
MDH	Mälardalen University
FER	Faculty of electrical engineering and computing, Zagreb
POLIMI	Politecnico di Milano
GUI	Graphical user interface
IIS	Internet Information Services
HTTP	Hyper Text Transfer Protocol
DOM	Document Object Model
AJAX	Asynchronous Java and XML
REST	Representational State Transfer
ORM	Object Relational Mapper
UI	User Interface
MVC	Model View Controller

1.4 References

- Project Home
  - http://www.fer.unizg.hr/rasip/dsd/projects/basketball
- Project plan
  - http://www.fer.unizg.hr/_download/repository/Project_Plan%5B3%5D.pdf
- Requirements Definition
  - http://www.fer.unizg.hr/_download/repository/Requirements_Definition%5B4%5D.pdf
- DatabaseBasketball Home
  - http://www.databasebasketball.com/

2. Software architecture

In the following paragraph the architecture of the HoopStats application will be explained. Also there will be detailed description on how different components work together to deliver the features.

2.1 Conceptual design

The system consists of three basic components which also can be seen in figure 2.1:

- Microsoft SQL Server database
- ASP.NET MVC 4 web application
- Android native mobile application

For deployment, the Window Server 2008 operating system will be used. It will run Internet Information Services (IIS) which work as a web server, providing the platform for the ASP.NET web application, and Microsoft SQL Server 2008, providing the platform for the database.

2.1.1 Microsoft SQL Server database

For data persistence Microsoft SQL Server Express Edition 2008 will be used. If possible, complete databases provided by other basketball statistics websites will be included. If such databases would be impossible to obtain, the data will possibly be scrapped off the basketball statistics websites.

2.1.2 ASP.NET MVC 4 web application

ASP.NET web application will be used as a core part of the system. It is going to access the database and provide the functionalities to create flexible queries from the user choices in the UI. After retrieving the queried data, it will update the view presented to the user using a controller (MVC). The view will also receive updates in JSON, which means it will be aided by technologies like JavaScript and Ext JS to enable AJAX and therefore a more dynamic frontend. The application that will be used by both the Android application and the web application to access and the data is shown in figure 2.1. The client will access the web application using a modern browser like Google Chrome.

2.1.3 Android native mobile application

The mobile application will be a native Android application. This approach has been opted in the light of the requirement, to provide an Android client which should offer the best usability and user experience. For development of this application, JAVA programming language will be used along with the Android SDK. Application will need a minimum requirement of any modern Android phone with resolution higher than or equal to 320x480 and an Internet connection, and with OS version higher than Android 2.2.

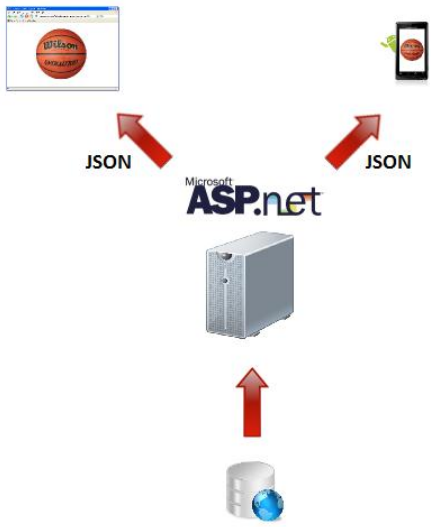


Figure 2.1 – Components and interaction between them

3. External interfaces

3.1 Hardware Interfaces

HoopStats application will be delivered in two forms: a Web application and an Android client. The HoopStats web application will be optimized primarily for Google Chrome browser, but other browsers will be supported too. The web application will be implemented as a thin client, which means that all the processing will be done on the server side. The web application will be optimized to display the obtained data and for processing and sending the user input to the server.
The Android client will be implemented as a native application. It will be also be implemented as a thin client and won't use much of the resources of the device. The application will be optimized for processing and sending the user input, sending it to the server and displaying the response to the user.

3.2 Software Interfaces

In order to use our web application, user must have a web browser, preferably Google Chrome. The web application will be optimized for this browser, but others will be supported too. The client and server will communicate using the HTTP protocol. Since we do not have any sensitive information, there will not be any

encryption imposed. The client will send a HTTP GET request to the server. In the most of the cases response will be in JSON format, containing only data. This will be for asynchronous communication between the user and the client. In other cases it will be ordinary http response message from the server.
The requirement for the Android application is to have a device (phone) that runs Android OS. The version of the OS should be Android 2.2 or newer. The Android application will follow the service oriented architecture and will communicate and obtain data from the server by using services exposed by the service layer of the web application. The Android client and server will communicate using the JSON format.
JSON is the preferred format for REST communication because of the small overhead and also because it is very easy to process by both the web and the Android client.

3.3 Communication Interfaces

In order to use the application the client must have an internet connection. Any hardware interface that supports connection to the Internet can be used. Android phones can use the GSM network, WiFi. The personal computer (most likely to be used for the web application) must have at least NIC or a dial-up modem.

3.4 User Interfaces

3.4.1 Web client

These two figures illustrate an initial draft of how the web client will look like, regarding the players search and comparison.

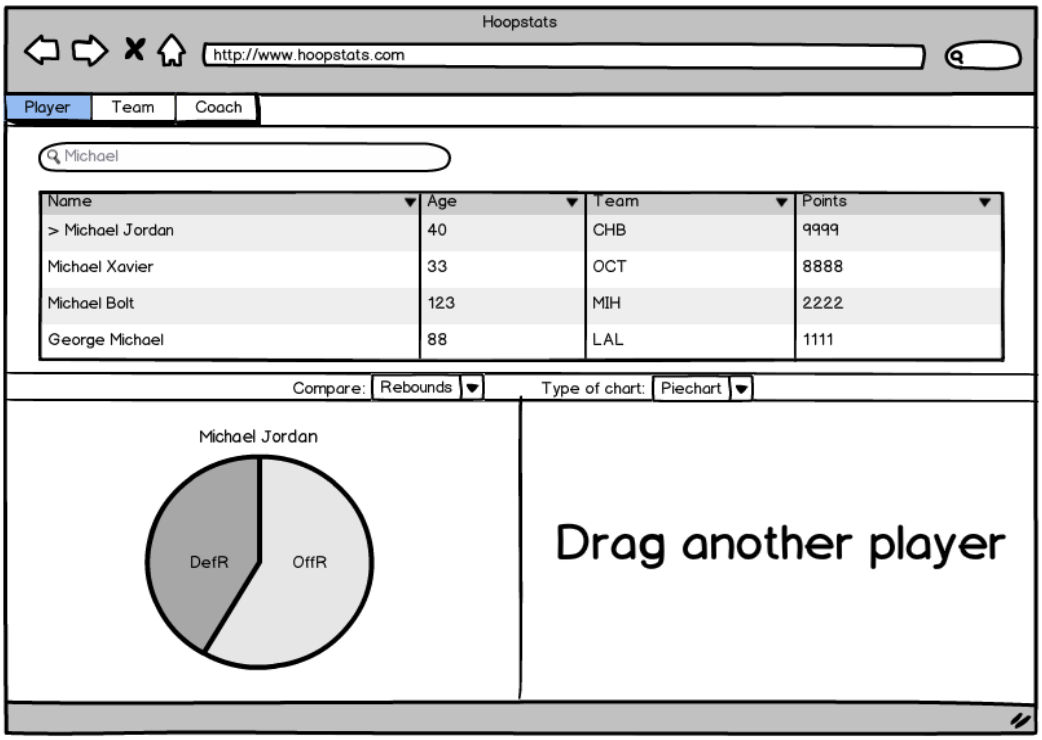


Figure 3.1 – Web client interface for visual representation of property.

Figure 3.1, after searching by a player's name, a table with the results is produced, and after dragging a particular row to the bottom-left corner (in this case, Michael Jordan) a single property is shown in a form of a chart.

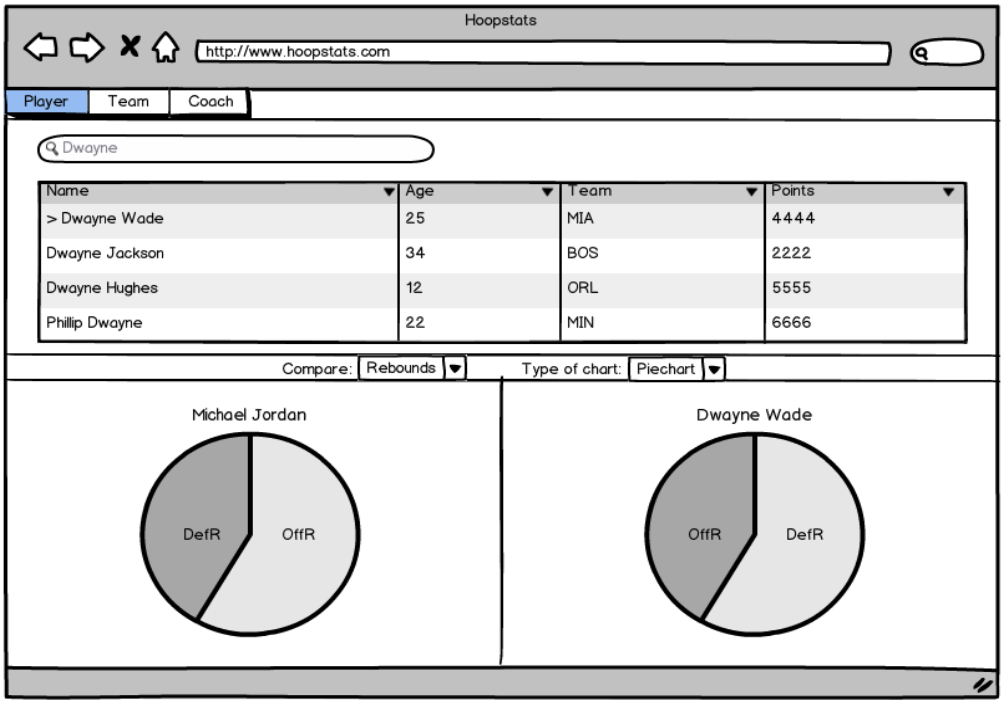


Figure 3.2 – Web client interface for comparison between two players.

Figure 3.2, the player comparison is done after performing another search and dragging another player to the bottom-right corner. A comparison is shown regarding the property and the chart type chosen.

3.5 Android client

Figure 3.3 illustrates the home screen of our Android application. It consists of a title bar and four big buttons: Players, Teams, Coaches, and About. Each of these buttons takes the user to the new screen.

Figure 3.3 – Android select template screen

One such screen is the Players screen which can be seen in Figure 3.3. It is a simple screen that consists of a search text field, which is used to search the database for players by name, results of the search and a keyboard that disappears when the search button is pressed or the search text field loses focus in any other way.

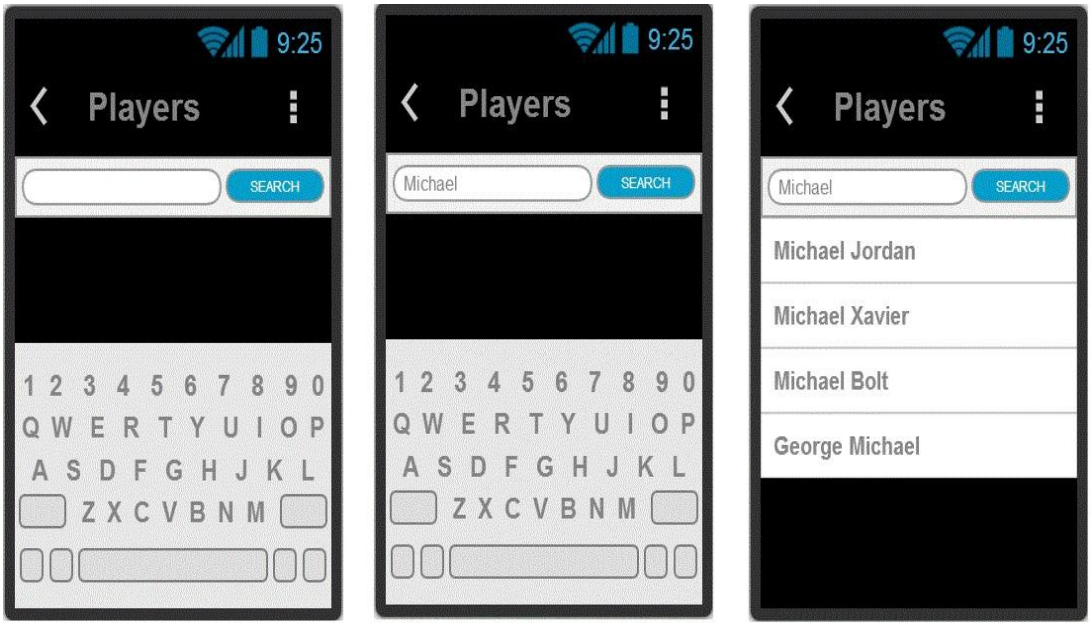


Figure 3.4 – Android search player by name

When user taps on the desired row of the results table, the application takes him to a screen similar to the one shown in Figure 3.4 which shows player's picture and his details.

Figure 3.5 – Android player details screen

4. Detailed software design

In this section, the detailed software design for the HoopStats project will be explained. Furthermore the different entities in the persistence, business and user interface layer will be explained in detail.

4.1 Implementation modules / components

The HoopStats application will be implemented as a typical client – server application. The implementation of the server side of the application will be in .NET. Client applications will be delivered as Web application and native Android application. The client applications are going to be implemented as thin clients, which mean that they will not use much resource on the client device.

All of the processing and fetching of data from the database will be performed by an independent .NET component. This component will later be referenced and used by the ASP.NET MVC application to build the business layer application. The component will use the Entity Framework 5.0 ORM component, which means that all of the classes in this component will be identical to the database tables in the MS SQL database.

The Web application will be implemented as an ASP.NET MVC 4 application. The MVC model imposes that the application will follow the basic template and will be implemented as a three tier application. For better communication with the client side of the application view model classes will be introduced. For translating data from data models into view model classes, mapper classes with respective functions for data mapping will be provided. The exact structure of the view model classes will be known upon the implementation. They will be presented in the next revision of the document.

The service layer provided by the MVC application will be used by both the Web and the Android client.

As previously stated in the text, the Android client will be implemented as thin client. The application will be native Android application build in JAVA using the Android SDK. It will use exactly the same view models as the web application, which will be provided by the service layer in JSON format.

4.2 Data flow / Interactions / Dependencies

The three main components of the HoopStats application exchange data and interact through defined communications channels. Figure 4.1 shows an overview of the communication channels between the different system parts.

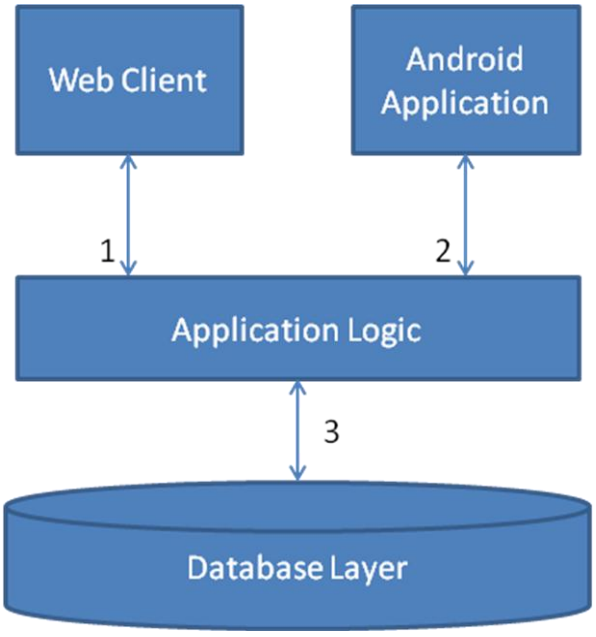


Figure 4.1 – Communication Overview

1. The web client communicates with the application logic via HTTP GET / POST requests. When a query is sent from the client to the application logic, it returns the basketball data in JSON format and HTML for the views. The returned format depends on the type of GET-Request.
2. The Android application communicates with the Application Logic in a similar way the web client does via HTTP and JSON to deliver the basketball statistics data.
3. To query the database for the relevant statistics the Application Logic is coupled with the database through the ET. The application sends SQL statements to the DBMS which interprets them and delivers the results which are transformed by the ET into objects.

The overall style of the communication is synchronous, which means that all components of the system have to be available ready for communication in order for the system to work. The components communicate with each other through predefined interfaces with standardized data formats. This allows a decoupling of the internal dynamic implementation of the components. Therefore, changes in the workflow of a component do not affect the other components.

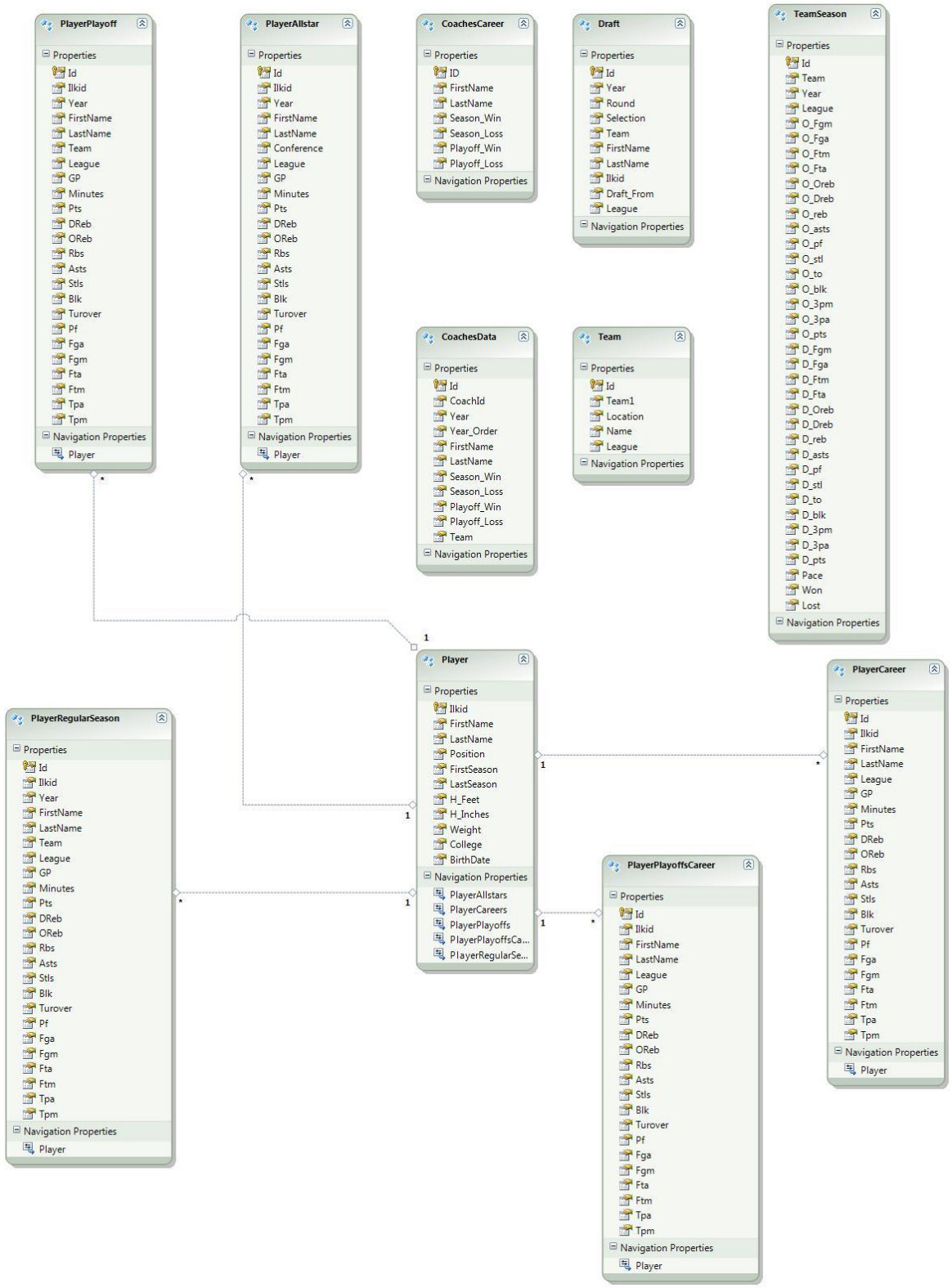
All components of the system rely on the same database, which is represented through static classes in the whole system. If the structure or meanings of parts of the data are changed, other static system parts which use the data have to be changed as well. Otherwise most parts of the system wouldn't be able to work correctly as the data has to be interpreted differently.

4.3 Database Model

The HoopStats database is predefined since it can be data obtained directly from

http://www.databasesports.com/. The tables and their columns have already been defined by the provider, and for the alpha prototype will be used as they are. If there is a serious performance issue with the database schema, changes are considered in order to boost the persistency layer performance. The data was obtained in CSV format. HoopStats application for data persistency uses a relational database management system – the MS SQL Server 2008. To avoid license issues, the Express edition of the application along with the corresponding management studio is used.

On the figure below, the initial tables along with their columns are shown. Since the data layer is not strictly specified, it is very likely that new tables and data is introduced into the system. This will be in Beta prototype if necessary.



In the HoopStats application, we are going to use the Entity Framework 5.0 ORM, and for that purpose in the initial version of the database there will not be any stored procedures. The data will be persistent through time, so insert and modify functions will not be exploited a lot. Indexes of the tables will be created if there will be strong need for that – if the performance drops significantly.

4.4 Web site organization

The HoopStats web application will follow the ASP.NET MVC 4 pattern. All of the pages will have dynamic content, except for the start page. In the initial version there will be controllers for each logical part of the application. There will be separate controllers with respective actions for the players and teams. In the player's controller, beside the standard function, there will be an option for comparing two players. Comparing teams is not a feature for the alpha prototype. As responsiveness and user experience are critical requirements for the application it will extensively use JavaScript and some of the most famous libraries for asynchronous client – e.g. DOM manipulation and visual representation of the data.

4.5 Interfaces to External Systems

The application will expose web services which will be consumed by the web client and the Android application. The web services will be REST and the data will be delivered in JSON format. The exposed services will be free for use by any external system.