Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Факультет «Высшая школа электроники и компьютерных наук»
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой
_____ К. А. Домбровский
«___»_____2017г.

Приложение для организации и проведения футбольных турниров

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ – 09.03.01.2017.238 ПЗ ВКР

Руководитель работы,
доцент каф. «Электронные
Вычислительные машины»
_____ Е. С. Ярош
«___»_____2016г.

Автор работы
студент группы КЭ-445
_____ А.В. Сивоплясов
«_____»_____2017г.

Нормоконтролер, ст. преп. каф.
«Электронные вычислительные
машины»
_____ В. В. Лурье
«_____» _____2017г

Челябинск 2017

Оглавление

ВВЕДЕНИЕ

В современном мире веб-приложения играют колоссальную роль. Практически каждая организация имеет собственный сайт. В условиях использования современных информационных технологий – это необходимый фактор, предоставляющий возможность расширить сферу рекламной деятельности и позволяющий привлечь дополнительных клиентов.

В настоящий момент российский футбол находится на рекордно низком уровне. Неудачи на крупных международных турнирах вкупе с плохим качеством игры на внутренней арене приводят к неконкурентоспособности нашей сборной, а, следовательно, и к снижению интереса к ней со стороны зрителей. Для решения этой проблемы нужно заниматься развитием детского футбола.

Компания «Kidsfootball» занимается организацией и проведением детских футбольных турниров в городе Челябинске. Такие турниры проводятся для того, чтобы помочь начинающим лучше понять футбол, получить новые знания и применить их знания на практике.

Информатизация деятельности компании позволит привлечь к ней внимание, поднять ее работу на новый уровень, повысить ее рейтинг.

1 ПОСТАНОВКА И АНАЛИЗ ЗАДАЧИ

Целью данной дипломной работы является создание приложения, позволяющего просматривать информацию о турнирах, организованных компанией «Kidsfootball», и записываться на них.

Общее представление о предметной области можно получить из контекстной диаграммы в нотации IDEF0, показанной на рисунке 1.1. Входное воздействие на систему могут оказывать следующие категории пользователей:

- зарегистрированный пользователь;
- незарегистрированный пользователь;
- тренер;
- организатор.

Работа системы управляется следующими воздействиями:

- нормативные документы.

Результатом работы системы являются:

- регистрация пользователя;
- регистрация команды на турнир;
- отзыв о турнире;
- запись на турнир;
- ответ на сообщение пользователя.

Для работы системы необходимы следующие механизмы:

- веб-приложение для пользователей и организатора.

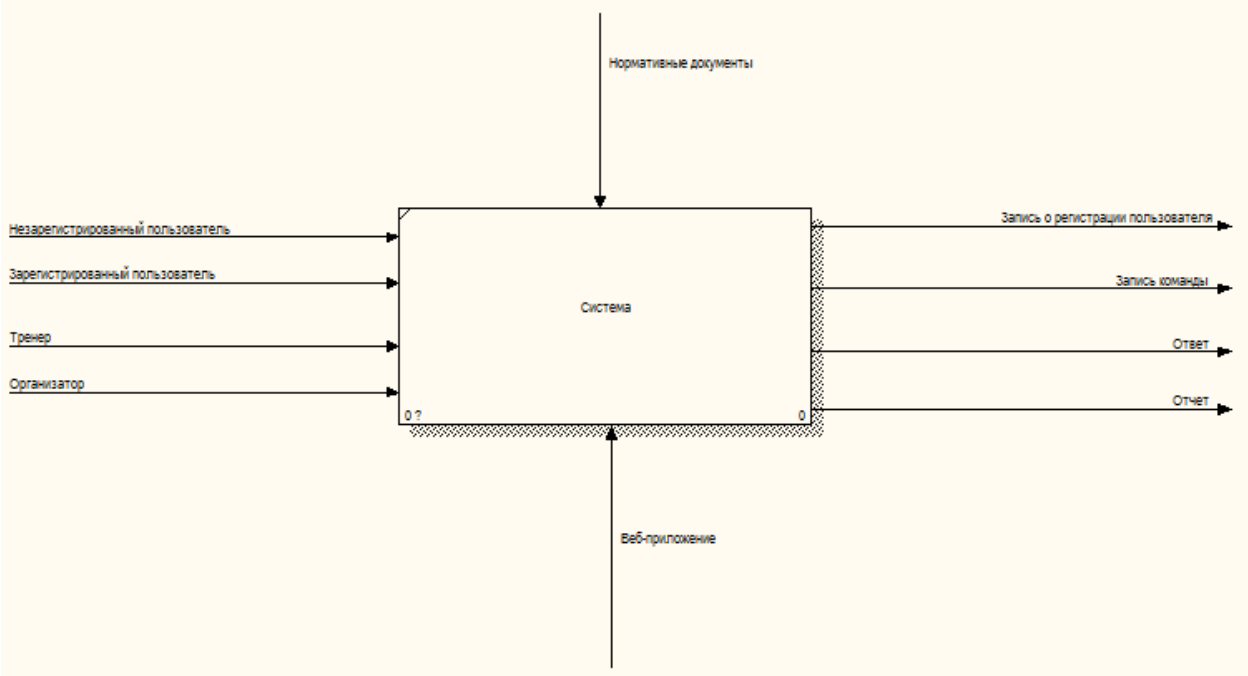


Рисунок 1.1 – Первый уровень диаграммы IDEF0

Укрупненно систему можно представить состоящей из двух частей: пользовательской и административной (рисунок 1.2).

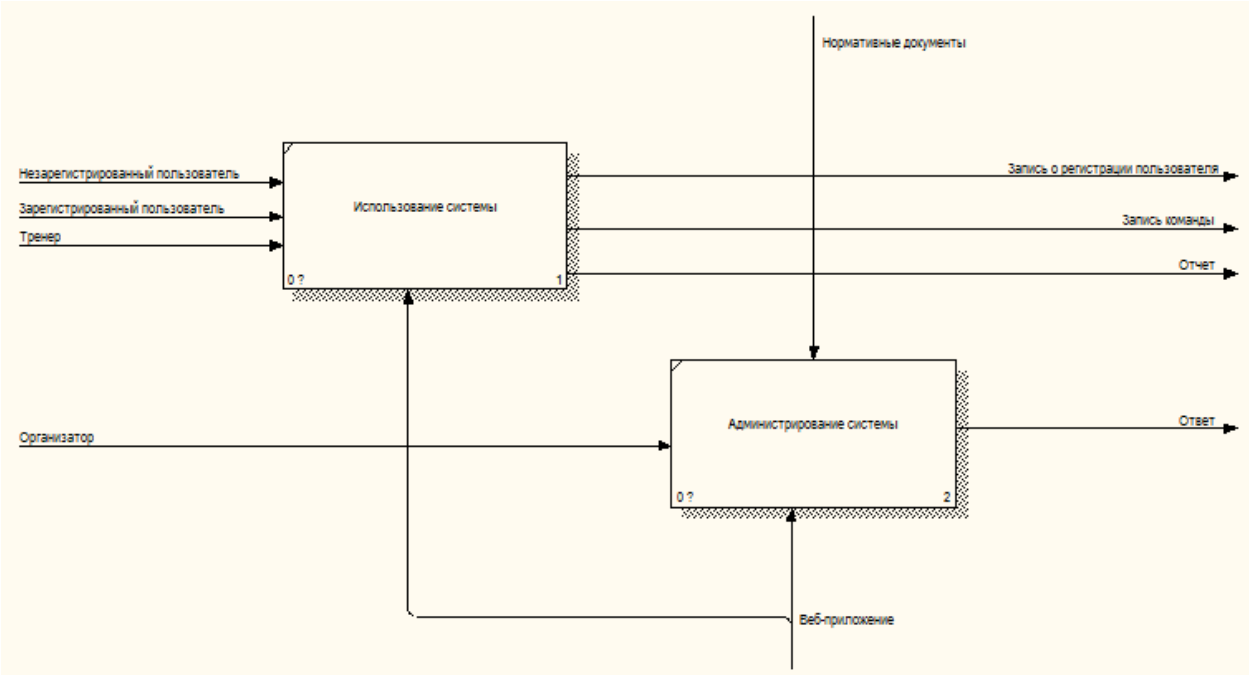


Рисунок 1.2 – Второй уровень диаграммы IDEF0

Клиентская часть должна обладать следующими возможностями.

Для пользователя (рисунок 1.3):

- просмотр списка турниров;
- регистрация в системе;

- связь с организаторами (отправка сообщения непосредственно из приложения);
- просмотр статистики турнира;
- получение email рассылки (если есть подписка);
- просмотр новостей;

Для тренера:

- то же, что и для пользователей;
- запись на турнир;
- получение сообщения от организатора на email о включении команды на турнир или отказе;
- просмотр сведений о турнире, о матче, о судейской бригаде;
- использование ранее введенной информации о команде в последующих турнирах.

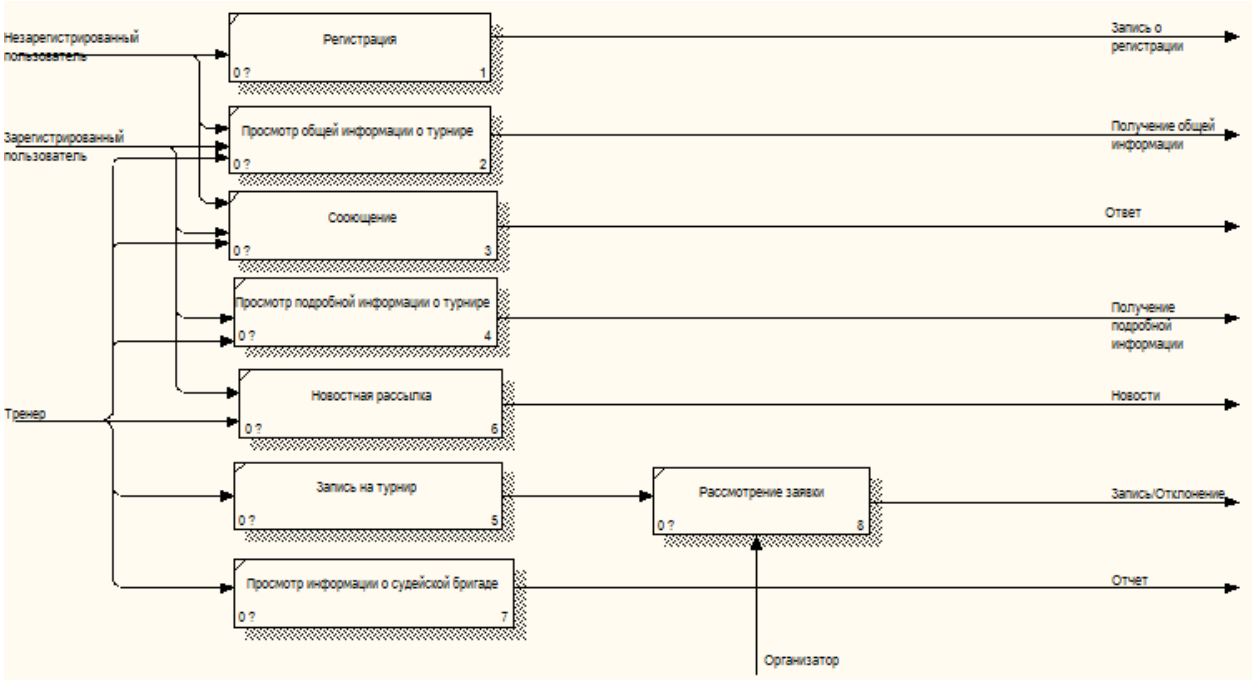


Рисунок 1.3 – Третий уровень диаграммы IDEF0 – использование системы

Для организатора должно быть создано веб-приложение, имеющее следующие функции (рисунок 1.4):

- формирование всех таблиц;
- прием заявки от тренера;
- отклонение заявки от тренера;
- формирование расписания матчей;
- формирование ленты новостей;
- автоматическая рассылка новостей всем подписавшимся;
- просмотр обращений по email.

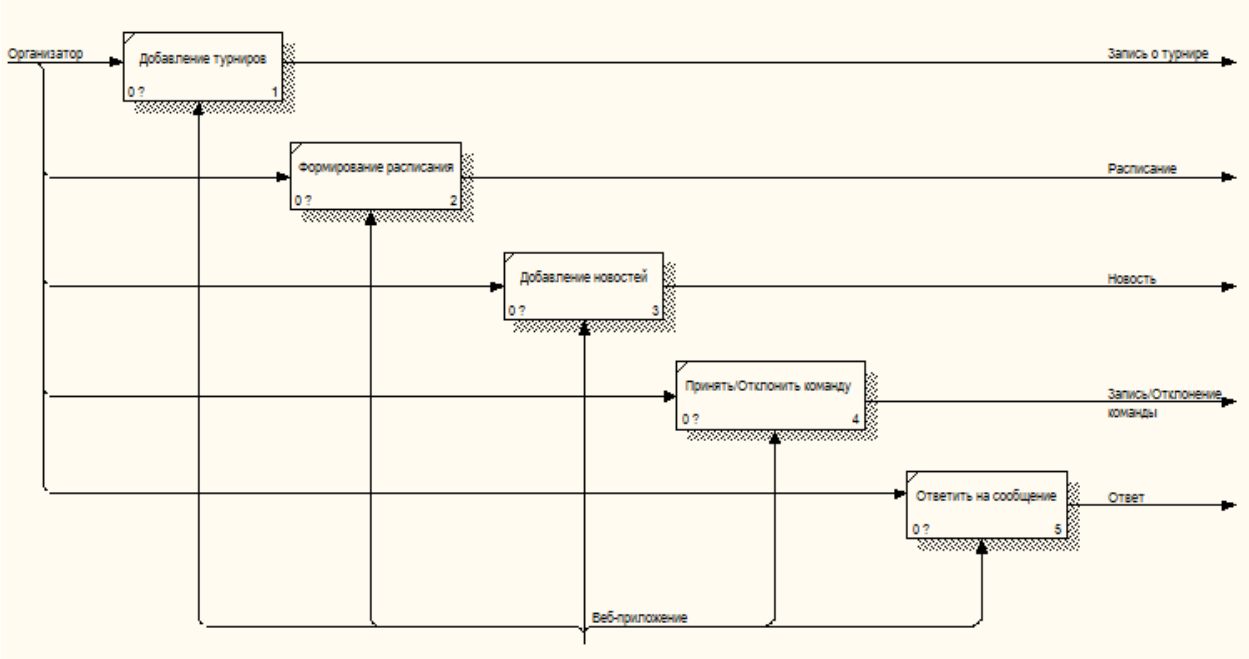


Рисунок 1.4 – Третий уровень диаграммы IDEF0 – администрирование системы

Для выполнения указанных требований необходимо решение следующих задач:

- разработка архитектуры системы;
- выбор средств реализации;
- организация данных и информационного обмена;
- создание базы данных на сервере;
- создание серверной части;
- реализация административного веб-сервиса;
- размещение сервера на хостинге;
- разработка пользовательского веб-сервиса.

Краткие выводы по разделу один

В данном разделе рассмотрено общее представление о предметной области. Представлены контекстные диаграммы в нотации IDEF0. Поставлены цель и задачи для разработки.

2 ОБЗОР РОДСТВЕННЫХ РАЗРАБОТОК

На данный момент существует определенное количество приложений для организации футбольных турниров. Большинство из них обладает крайне ограниченным функционалом. Но также есть небольшое количество приложений, функционал которых приближен к требуемому.

Одно из таких приложений **Tourney Master 3.** У него есть ряд достоинств: приложение позволяет выбрать разные виды спорта и добавить значительное количество настроек для них. На рисунке 2.1 показана главная страница приложения.

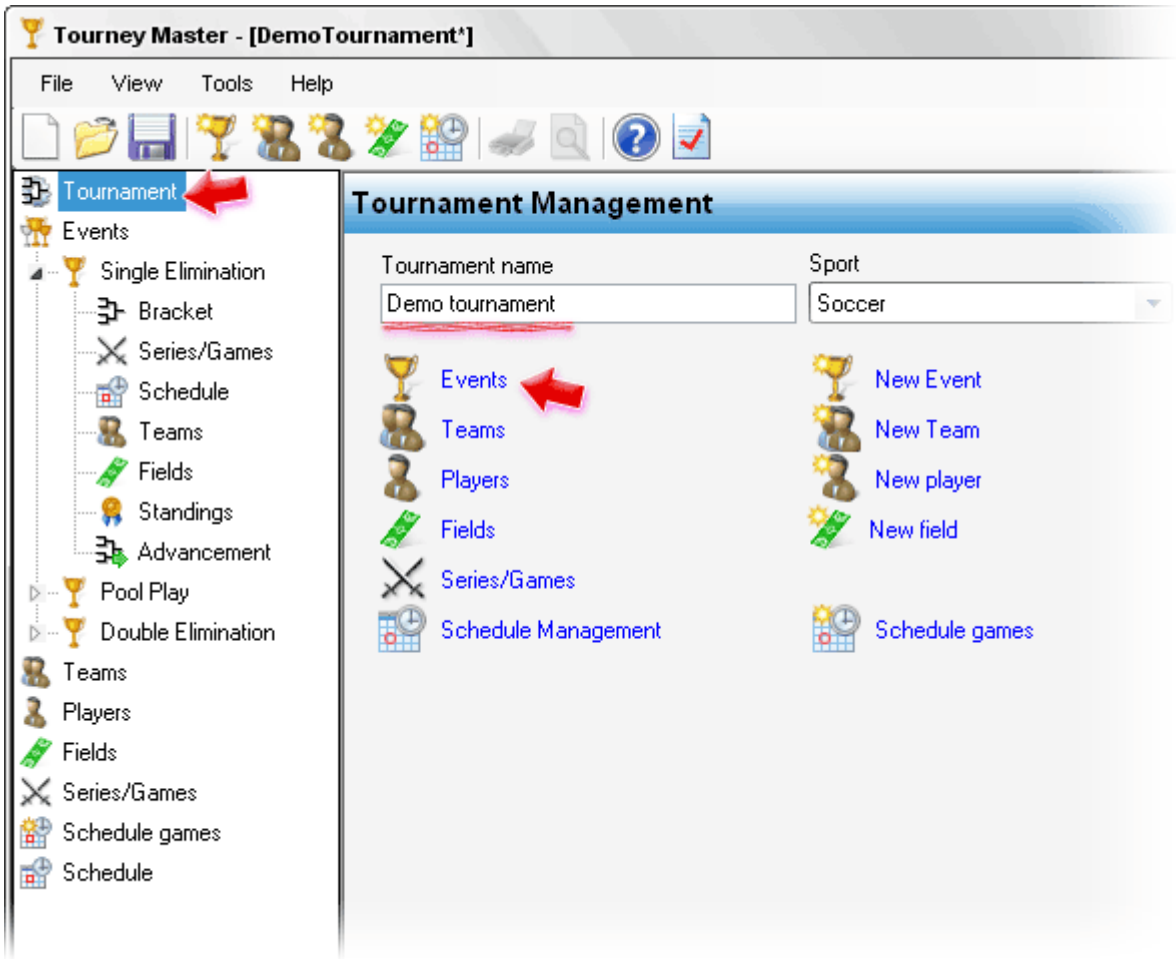


Рисунок 2.1 – Главная страница Tourney Master

В разделе Events можно задать основные настройки турнира. Разрешается выбрать различную сетку. На рисунке 2.2 изображена сетка вида плей-офф.

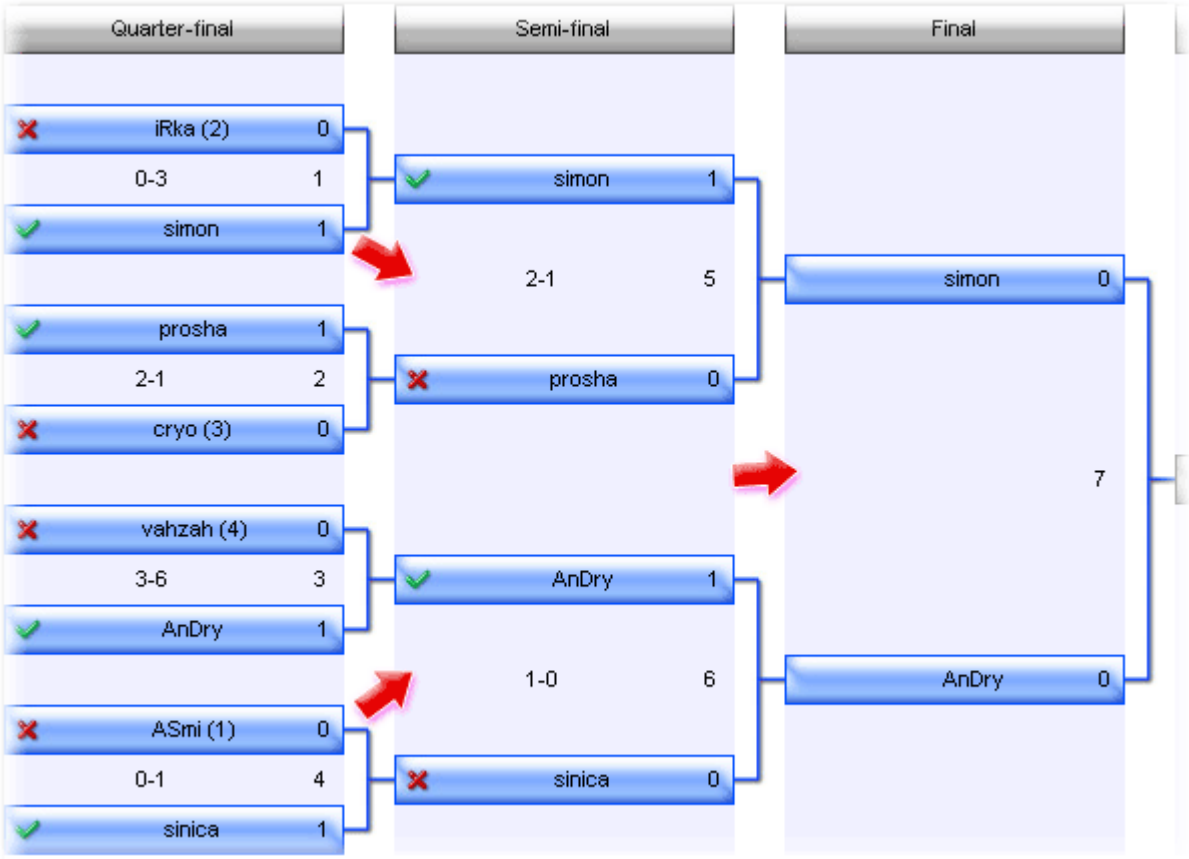


Рисунок 2.2 – Сетка турнира

В разделе Players отображается список игроков. Для игроков можно указать следующую информацию: Ф.И.О., ник, адрес, телефон, город, индекс, страна, e-mail, телефон, пол, вес, уровень подготовки, день рождения, вес, иконка участника.

В разделе **Standings** имеется возможность посмотреть основную статистику турнира. На рисунке 2.3 показан рейтинг бомбардиров.

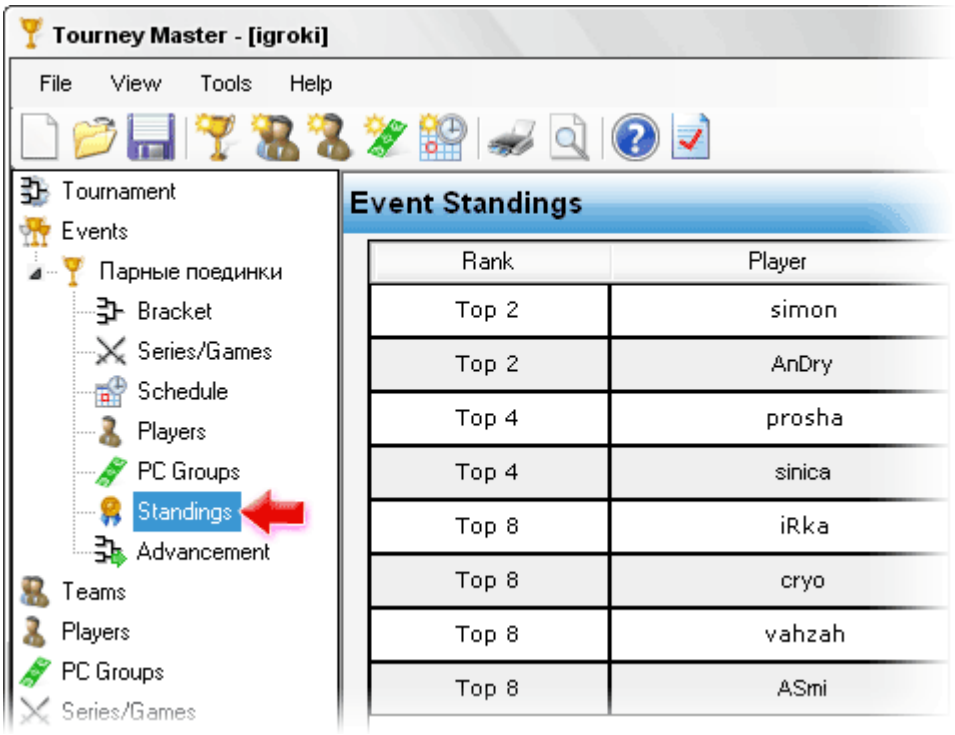


Рисунок 2.3 – Статистика турнира

Изучив приложение Tourney Master 3, можно выделить основные достоинства и недостатки.

Достоинства:

- поддержка значительного количества видов спорта;
- имеется возможность добавить много параметров для игроков и команд.

Недостатки:

- отсутствие клиентской части;
- отсутствие поддержки русского языка;
- отсутствие поддержки ОС старше Windows7.

Также можно выделить сайт http://www.konkuri.com (рисунок 2.4).

К преимуществам можно отнести удобство, простоту

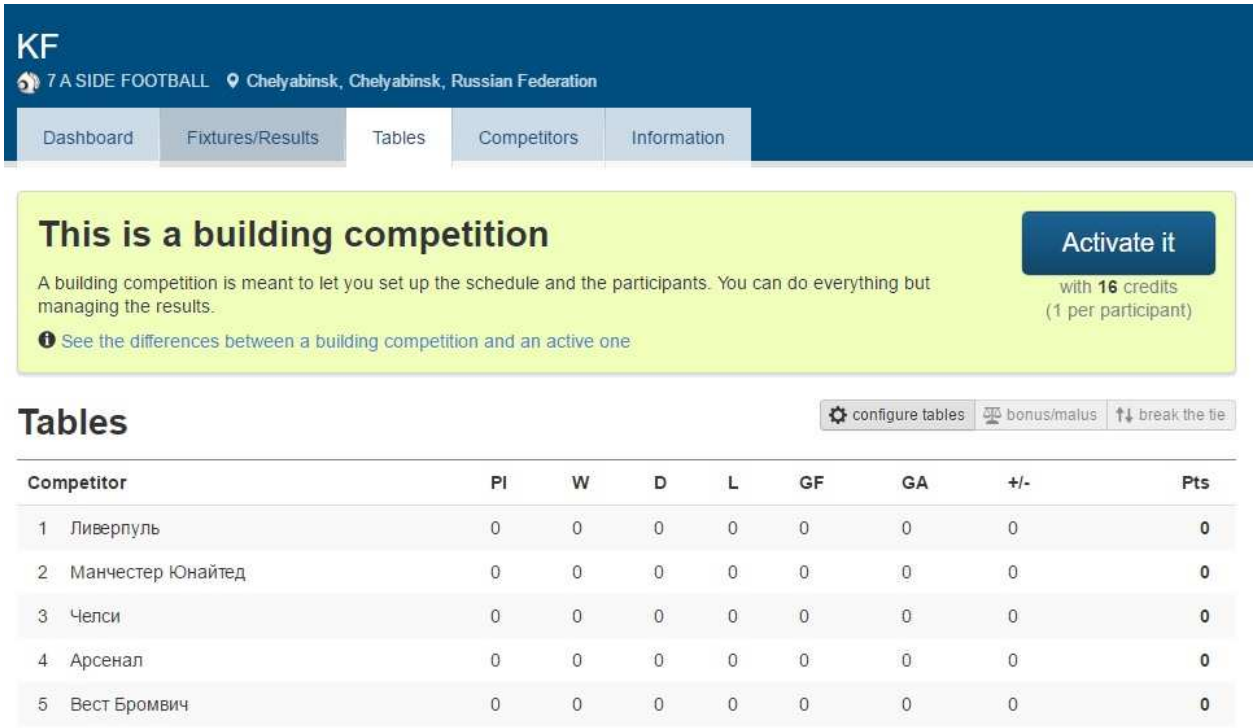


Рисунок 2.4 – Турнирная таблица на сайте konkuri.com

Также, особенностью этого сайта является автоматическое составление расписания турнира (рисунок 2.5). Это упрощает задачу формирования сетки турнира, но при этом исключает возможность составления организаторами собственного расписания.

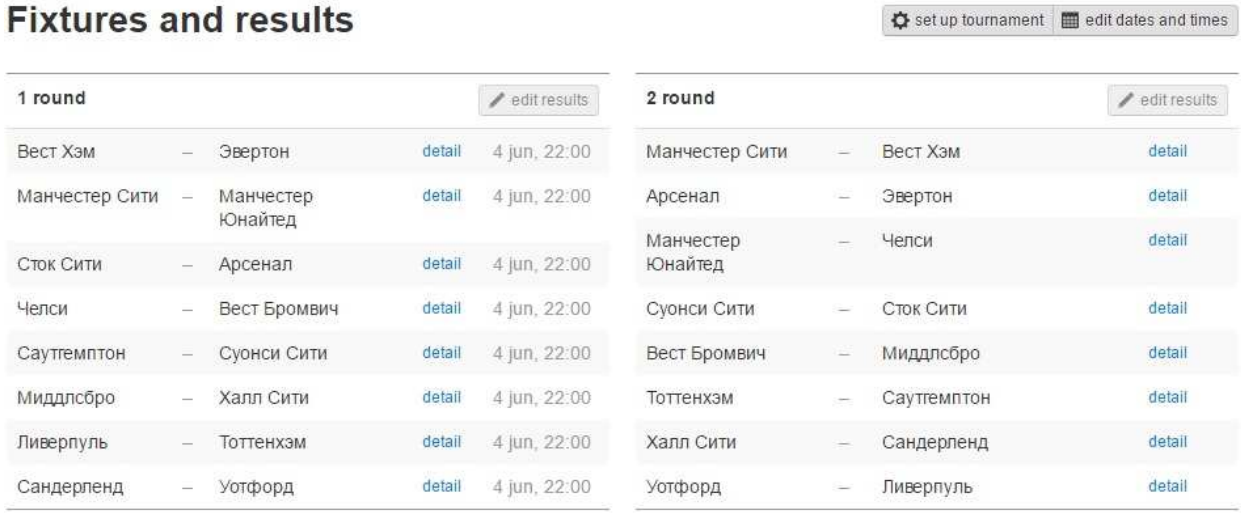


Рисунок 2.5 – Расписание турнира на сайте konkuri.com

Изучив приложение konkuri, можно выделить основные достоинства и недостатки.

Достоинства:

- удобный интерфейс;
- имеется возможность добавить много параметров для игроков и команд.

Недостатки:

- отсутствует статистика игроков;
- автоматическое формирование расписания;
- отсутствие русскоязычного интерфейса;

Краткие выводы по разделу два

Исходя из вышеизложенного можно сделать вывод, что разработка собственного веб-приложения целесообразна и необходима.

3 СРЕДА РЕАЛИЗАЦИИ

Разрабатываемое приложение должно являться кроссплатформенным, доступ к нему должен осуществляться с устройств, имеющих подключение к интернету.

Один из плюсов веб приложений – отсутствие необходимости установки на устройство клиента объемного программного обеспечения, возможность работать под любыми операционными системами.

Для разработки клиентской части приложения используется связка языков HTML, CSS и JavaScript.

HTML — стандартизированный язык разметки документов в сети Internet. Большинство веб-страниц содержат описание разметки на языке HTML. Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

CSS — формальный язык описания внешнего вида документа, написанного с использованием языка разметки. Преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML и XHTML.

JavaScript — мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией языка ECMAScript (стандарт ECMA-262).

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Для разработки серверной и административной частей используется язык PHP, один из популярнейших языков для написания веб-приложений и серверов. Также рассматривались языки программирования Python и Ruby on Rails.

PHP – скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов.

Преимущества PHP:

- является свободным программным обеспечением, распространяемым под особой лицензией (PHP license);
- легок в освоении на всех этапах;
- поддерживается большим сообществом пользователей и разработчиков;
- имеет развитую поддержку баз данных;
- имеется огромное количество библиотек и расширений языка;
- может использоваться в изолированной среде;
- может быть развёрнут на большинстве серверов;
- портирован под большое количество аппаратных платформ и операционных систем.

Недостатки PHP:

- не подходит для создания десктопных приложений или системных компонентов;
- имеет слабые средства для работы с исключениями;
- глобальные параметры конфигурации влияют на базовый синтаксис языка, что затрудняет настройку сервера и разворачивание приложений.

Ruby – является мощным объектно-ориентированным языком сценариев для разработки веб-сайтов. Он используется для формирования

или программирования мобильных приложений и веб-сайтов. Этот язык является эффективно сбалансированным и масштабируемым с помощью функционального программирования. Ruby используется многими программистами из-за его простых и эффективных методов записи.

Преимущества Ruby:

-открытая разработка;

-работает на многих платформах;

- может внедряться в HTML-разметку;

- относится к языкам программирования сверхвысокого уровня (VHLL), то есть обладает высоким уровнем абстракции и предметным подходом в реализации алгоритмов;

- реализует концептуально чистую объектно-ориентированную парадигму;

- предоставляет продвинутые методы манипуляции строками и текстом;

- легко интегрирует в свои программы высокопроизводительные серверы баз данных (DB2, MySQL, Oracle и Sybase);

- благодаря VHLL, программы на Ruby хорошо масштабируются и легко сопровождаются;

- имеется простой программный интерфейс для создания многопоточных приложений;

- имеет продвинутые средства для работы с массивами;

- возможности языка можно расширить при помощи библиотек, написанных на C или Ruby;

- дополнительные возможности для обеспечения безопасности;

- встроенный отладчик.

Недостатки Ruby:

- небольшое количество информационных ресурсов, посвященных Ruby;

- менее производителен по сравнению со многими другими языками, применяемыми в веб-разработке;

- относительно медленно разрабатывается и развивается.

Python – это высокоуровневый скриптовый язык, используемый для выполнения сценариев на стороне сервера для сайтов и мобильных приложений. Он выполняет резервное копирование многих парадигм программирования. В веб-разработке этот язык используется благодаря его гибкости и широкому спектру применения. Python может легко работать на серверах LINUX и Windows.

Преимущества Python:

- открытая разработка;

- удобный синтаксис, обеспечивающий хорошую читаемость кода;

- механизмы модульности хорошо продуманы и могут быть легко использованы;

- абсолютно всё в Python является объектами в смысле ООП, но при этом объектный подход не навязывается программисту.

Недостатки Python:

- плохая поддержка многопоточности;

- отсутствие коммерческой поддержки средств разработки;

- изначальная ограниченность средств для работы с базами данных;

Основываясь на этих данных, был выбран язык PHP, благодаря его распространённости, связке с HTML и удобному интерфейсу для работы с системой управления базами данных MySQL.

Для создания API (интерфейс программирования приложений) используется фреймворк Yii2, позволяющий достаточно просто организовать структуру API. Также рассматривались фреймворки Slim и Laravel.

Slim — микрофреймворк, идеально подходящий для небольших проектов или приложений, где полноценный фреймворк покажется лишним. Его используют многие PHP-разработчики для создания RESTful API и сервисов. Среди функций Slim — кэширование HTTP на стороне клиента, URL-маршрутизация, шифрование сессий и cookie, а также мгновенные сообщения по HTTP-запросам. Документация полная и сделана качественно.

Yii2 – фреймворк, где упор делается на производительность сайта. Он быстрее всех остальных PHP-фреймворков, так как использует технологию загрузки по требованию lazy loading. Yii2 полностью объектно-ориентированный и основан на принципе Don't-Repeat-Yourself («не повторяйся»), так что основа для кода будет чистая и логичная.

Также в [3] указано, что Yii2 интегрирован с jQuery и поставляется с набором AJAX функций. Здесь также есть мощный генератор исходного кода — Gii, который способствует объектно-ориентированному программированию и быстрому прототипированию, а также предоставляет веб-интерфейс, в котором можно интерактивно генерировать нужный код.

Laravel – довольно новый фреймворк, однако уже считается одним из лучших, с отличной документацией. У Laravel множество функций, обеспечивающих быструю разработку приложений. Имеется свой движок для шаблонов Blade и элегантный синтаксис, который позволяет облегчить выполнение частых задач: аутентификация, сессии, анализ очередей.

Основываясь на этих данных, был выбран язык фреймворк Yii2, благодаря его быстродействию, а также наличию генератора кода Gii, который упрощает процесс разработки.

Для реализации API выбран архитектурный стиль взаимодействия компонентов распределённого приложения в сети – REST (Representational State Transfer — «передача состояния представления»).

Преимущества REST API:

- надёжность;
- портативность компонентов;
- масштабируемость;
- производительность;
- простота интерфейсов;лёгкость внесения изменений.

Разработка приложения будет производиться в PHPStorm.

JetBrains PhpStorm — кросс-платформенная интегрированная среда разработки для PHP. Разрабатывается компанией JetBrains на основе платформы IntelliJ IDEA.

PhpStorm представляет собой интеллектуальный редактор для PHP, HTML и JavaScript с возможностями анализа кода на лету, предотвращения ошибок в коде и автоматизированными средствами рефакторинга для PHP и JavaScript.

На рисунке 3.3 приведен итоговый перечень средств, с помощью которых реализована система.

PHP 5.6

PHPStorm

Python

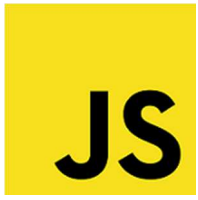
Ruby

Yii2

Laravel

Slim

JavaScript



HTML

CSS

Рисунок 3.3 – Перечень используемых средств

Краткие выводы по разделу три

В данном разделе были представлены средства, используемые для разработки системы.

4 ОБЩАЯ АРХИТЕКТУРА СИСТЕМЫ

Трехуровневая архитектура – архитектурная модель, которая отличается наличием в ней трех составных уровней (рисунок 6.1): клиентское приложение, сервер приложений, который обеспечивает связь с клиентским приложением и с сервером баз данных.

Клиентское приложение – первый уровень архитектуры, который представляет собой графический интерфейс для конечного пользователя. Обычно на этот уровень выносятся простые операции, интерфейс авторизации, проверка на валидность и различные способы шифрования данных. По соображениям безопасности клиентский уровень не имеет прямых связей непосредственно с базой данных.

Сервер приложений – второй уровень архитектуры. На нем происходят основные операции, логические решения и вычисления. Этот уровень координирует работу всей программы и реализует связь между двумя другими уровнями, перемещая и обрабатывая данные между ними.

Сервер баз данных – третий уровень приложений, который выносится на самостоятельный уровень и обеспечивает хранение данных. Доступ к нему осуществляется через сервер приложений.

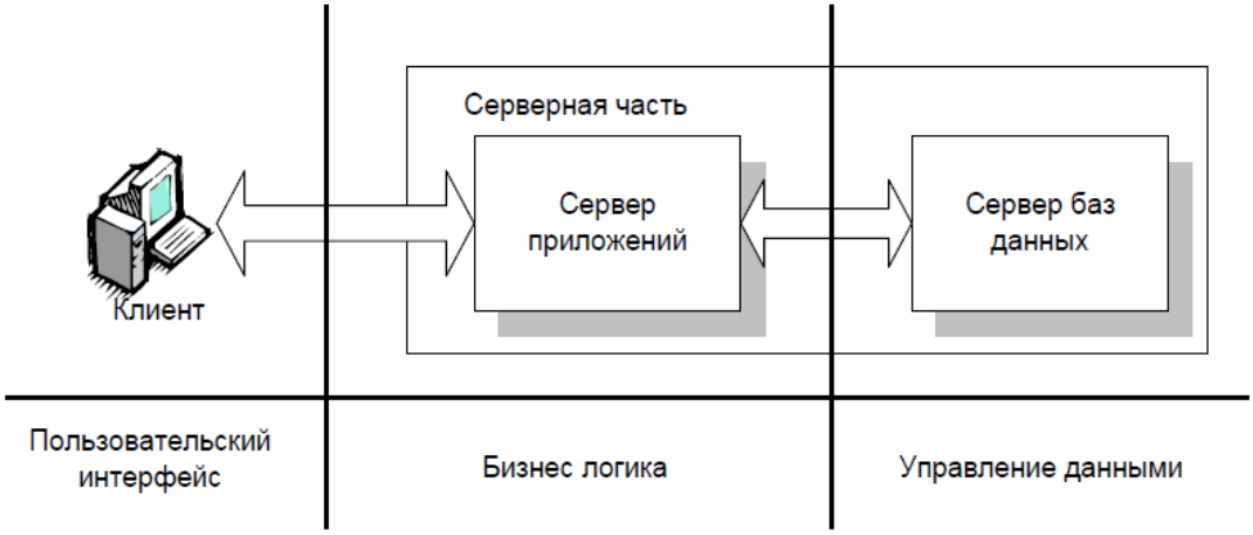


Рисунок 4.1 – Общая структура трехуровневой архитектуры

Достоинства трёхуровневой архитектуры:

- масштабируемость;
- конфигурируемость;
- высокая безопасность;
- высокая надёжность;
- низкие требования к скорости канала (сети) между терминалами и сервером приложений.

На рисунке 4.3 показана общая схема взаимодействия компонентов системы.

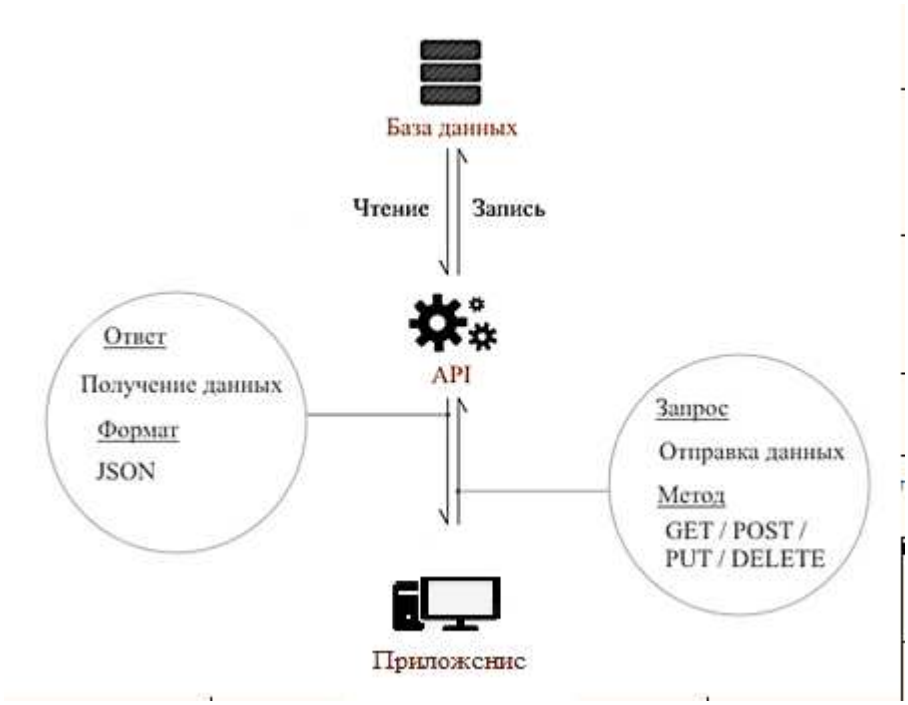


Рисунок 4.2 – Работа системы в общем виде

Необходимый функционал показан на UML – диаграммах Use Case. На рисунке 4.3 диаграмма для пользователей, на рисунке 4.4 для администрирования.

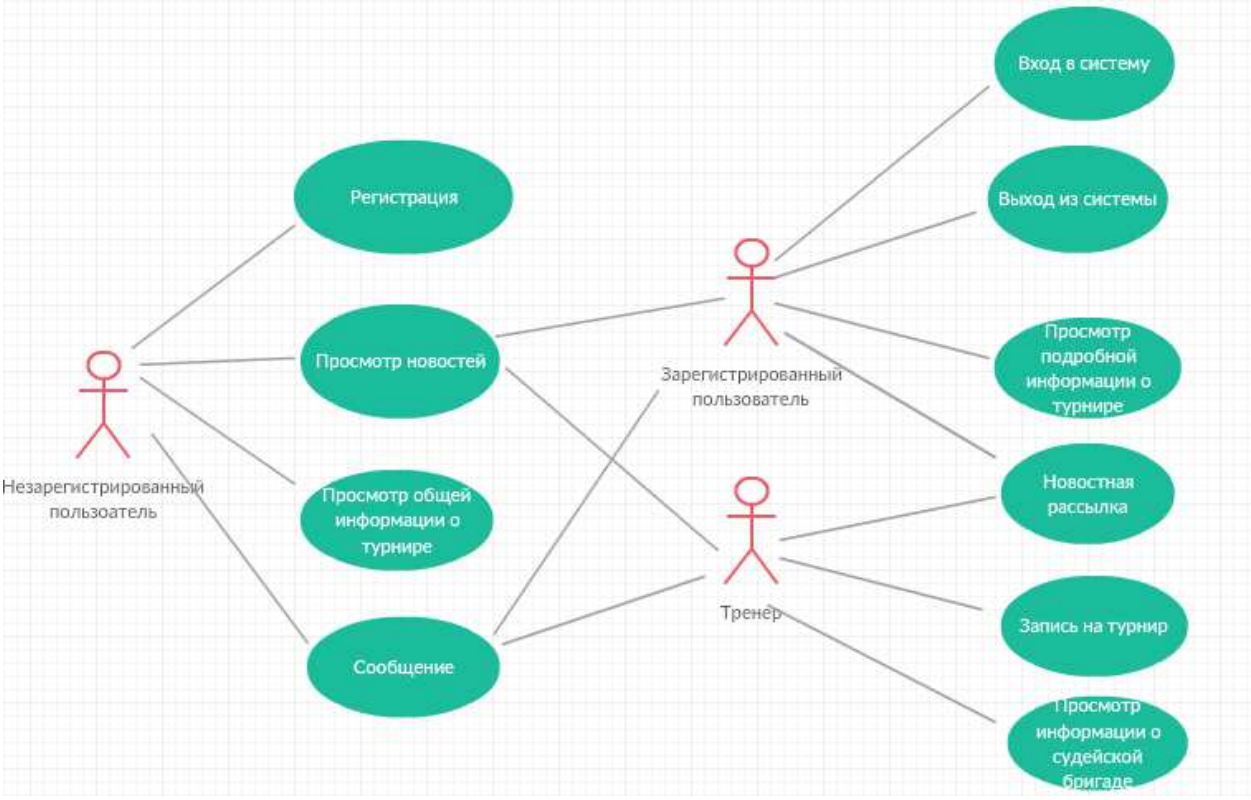
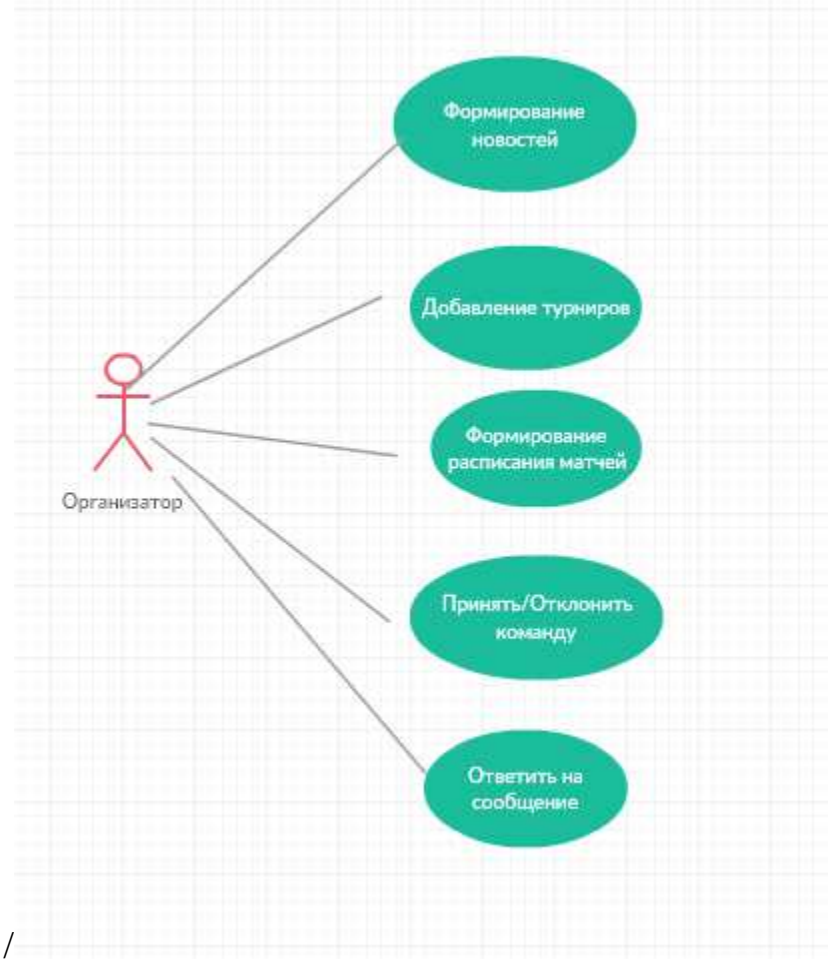


Рисунок 4.3 – Use Case диаграмма для пользователей



/

Рисунок 4.4 – Use Case диаграмма для управления системой

Краткие выводы по разделу четыре

В данном разделе показана архитектура системы, ее достоинства. Представлены компоненты, описана их взаимосвязь и работа в системе.

5 ОРГАНИЗАЦИЯ БАЗЫ ДАННЫХ

База данных системы хранится на сервере. Доступ к базе данных осуществляется через phpMyAdmin, потому что это одно из наиболее известных и качественных средств для работы с MySql – базами на сервере.

Схема базы данных представлена на рисунке 5.1.

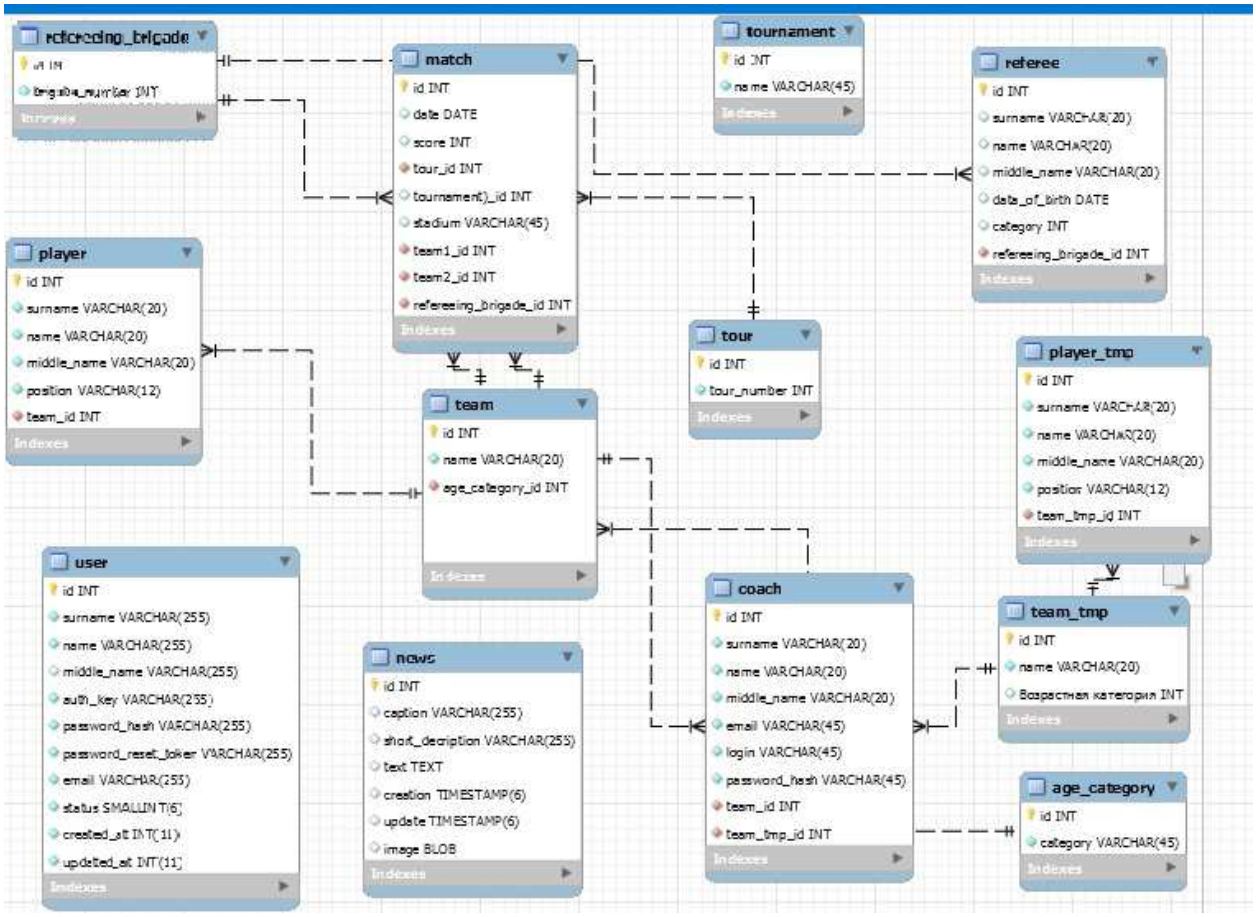


Рисунок 5.1 – Схема базы данных

В таблице 5.1 представлено краткое описание таблиц базы данных на сервере. Подробное описание каждой таблицы находится ниже.

Таблица 5.1. Назначение таблиц базы данных

Наименование таблицы	**Назначение**
User	Хранение данных о пользователе
Team	Хранение данных о команде
team_tmp	Временное хранение новой команды
Player	Хранение данных об игроках
player_tmp	Временное хранение данных об игроках
Coach	Хранение данных о тренере
Match	Хранение данных о матче
News	Хранение новостей
Referee	Хранение данных о судьях
referee_brigade	Хранение данных о судейской бригаде
tour	Хранение данных о туре
Tournament	Хранение данных о турнире
age_category	Хранение данных о возрастной категории

5.1 Описание таблиц

Таблица «user» (рисунок 5.2) хранит данные о всех зарегистрированных пользователях:

1. id – первичный ключ;
2. surname – фамиля пользователя;
3. name – имя пользователя;
4. middle_name – отчество пользователя;
5. email – почта;

6. login – логин пользователя;
7. password_hash – хэш пароля;

8. password_reset_token –токен;
9. auth_key – ключ авторизации;
10. receive_news – подписка на новости.

SQL-скрипт создания таблицы «user»:

```
CREATE TABLE IF NOT EXISTS `users` (
  `id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `surname` varchar(45) NOT NULL,
  `name` varchar(45) NOT NULL,
  `middle_name` varchar(45) NOT NULL,
  `email` varchar(45) NOT NULL,
  `login` varchar(45) NOT NULL,
  `password_hash` varchar(45) NOT NULL,
  `password_reset_token` varchar(45) NOT NULL,
  `auth_key` varchar(45) NOT NULL,
  `receive_news` tinyint(1) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;
```

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
1	id	int(11)			Нет	*Нет*	AUTO_INCREMENT
2	surname	varchar(45)	utf8_general_ci		Нет	*Нет*	
3	name	varchar(45)	utf8_general_ci		Нет	*Нет*	
4	middle_name	varchar(45)	utf8_general_ci		Нет	*Нет*	
5	email	varchar(45)	utf8_general_ci		Нет	*Нет*	
6	login	varchar(45)	utf8_general_ci		Нет	*Нет*	
7	password_hash	varchar(45)	utf8_general_ci		Нет	*Нет*	
8	password_reset_token	varchar(45)	utf8_general_ci		Нет	*Нет*	
9	auth_key	varchar(45)	utf8_general_ci		Нет	*Нет*	
10	receive_news	tinyint(1)			Нет	*Нет*	

Рисунок 5.2 – Проект таблицы user

Таблица «team» (рисунок 5.3) хранит данные о всех командах, принятых на турнир:

1. id – первичный ключ;
2. name – название команды;
3. age_category_id – возрастная категория;
4. tournament_id – турнир.

SQL-скрипт создания таблицы «team»:

```
CREATE TABLE IF NOT EXISTS `team` (
  `id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `name` char(20) NOT NULL,
  `age_category_id` int(11) NOT NULL,
  `tournament_id` int(11) NOT NULL,
    foreign key (age_category_id) references age_category (id) on delete cascade on update cascade,
    foreign key (tournament_id) references tournament (id) on delete cascade on update cascade
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;
```

	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
☐	1	id	int(11)			Нет	*Нет*	AUTO_INCREMENT
☐	2	name	char(20)	utf8_general_ci		Нет	*Нет*	
☐	3	age_category_id	int(11)			Нет	*Нет*	
☐	4	tournament_id	int(11)			Нет	*Нет*	

Рисунок 5.3 – Проект таблицы team

Таблица «team_tmp» (рисунок 5.4) временная, хранит данные о всех командах, подавших заявку на участие в турнире:

1. id – первичный ключ;
2. name – название команды;
3. age_category_id – возрастная категория;
4. tournament_id – турнир.

SQL-скрипт создания таблицы «team_tmp»:

```
CREATE TABLE IF NOT EXISTS `team_tmp` (
  `id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `name` char(20) NOT NULL,
  `age_category_id` int(11) NOT NULL,
  `tournament_id` int(11) NOT NULL,
```

foreign key (age_category_id) references age_category (id) on delete cascade on update cascade,

foreign key (tournament_id) references tournament (id) on delete cascade on update cascade

) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;

	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
☐	1	id	int(11)			Нет	*Нет*	AUTO_INCREMENT
☐	2	name	char(20)	utf8_general_ci		Нет	*Нет*	
☐	3	age_category_id	int(11)			Нет	*Нет*	
☐	4	tournament_id	int(11)			Нет	*Нет*	

Рисунок 5.4 – Проект таблицы team_tmp

Таблица «player» (рисунок 5.5) хранит данные обо всех игроках, заявленных на турнир:

1. id – первичный ключ;
2. surname – фамилия игрока;
3. name – имя игрока;
4. middle_name – отчество игркоа;
5. age – возраст игрока;
6. position – позиция игрока;
7. team_id – команда.

SQL-скрипт создания таблицы «player»:

```
CREATE TABLE IF NOT EXISTS `player` (
  `id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `surname` char(45) NOT NULL,
  `name` char(45) NOT NULL,
  `middle_name` char(45) NOT NULL,
  `age` int(11) NOT NULL,
  `position` char(15) NOT NULL,
```

`team_id` int(11) NOT NULL,

foreign key (team_id) references team (id) on delete cascade on update cascade

) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;

	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
☐	1	id	int(11)			Нет	*Нет*	AUTO_INCREMENT
☐	2	surname	char(45)	utf8_general_ci		Нет	*Нет*	
☐	3	name	char(45)	utf8_general_ci		Нет	*Нет*	
☐	4	middle_name	char(45)	utf8_general_ci		Нет	*Нет*	
☐	5	age	int(11)			Нет	*Нет*	
☐	6	position	char(15)	utf8_general_ci		Нет	*Нет*	
☐	7	team_id	int(11)			Нет	*Нет*	

Рисунок 5.5 – Проект таблицы player

Таблица «player_tmp» (рисунок 5.6) временная, хранит данные обо всех игроках из команд, только подавших заявку на участие :

1. id – первичный ключ;
2. surname – фамилия игрока;
3. name – имя игрока;
4. middle_name – отчество игркоа;
5. age – возраст игрока;
6. position – позиция игрока;
7. team_id – команда.

SQL-скрипт создания таблицы «player»:

CREATE TABLE IF NOT EXISTS `player` (

`id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,

`surname` char(45) NOT NULL,

`name` char(45) NOT NULL,

`middle_name` char(45) NOT NULL,

`age` int(11) NOT NULL,

`position` char(15) NOT NULL,

`team_id` int(11) NOT NULL,

foreign key (team_id) references team (id) on delete cascade on update cascade
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;

	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
☐	1	id	int(11)			Нет	Нет	AUTO_INCREMENT
☐	2	surname	char(45)	utf8_general_ci		Нет	Нет	
☐	3	name	char(45)	utf8_general_ci		Нет	Нет	
☐	4	middle_name	char(45)	utf8_general_ci		Нет	Нет	
☐	5	age	int(11)			Нет	Нет	
☐	6	position	char(15)	utf8_general_ci		Нет	Нет	
☐	7	team_id	int(11)			Нет	Нет	

Рисунок 5.6 – Проект таблицы player_tmp

Таблица «coach» (рисунок 5.7) хранит данные о тренерах, зарегистрировавших свои команды:

1. id – первичный ключ;
2. surname – фамилия;
3. name – имя;
4. middle_name – отчество;
5. age – возраст;
6. mail – почта;
7. login – логин;
8. password_hash – хэш пароля;
9. password_reset_token – токен;
10. auth_key – ключ авторизации;
11. team_id – команда;
12. team_tmp_id – временная команды.

SQL-скрипт создания таблицы «coach»:

```
CREATE TABLE IF NOT EXISTS `coach` (
 `id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
 `surname` char(20) NOT NULL,
 `name` char(20) NOT NULL,
 `middle_name` char(20) NOT NULL,
 `age` int(11) NOT NULL,
```

`mail` char(45) DEFAULT NULL,

`login` char(45) DEFAULT NULL,

`password_hash` char(255) DEFAULT NULL,

`password_reset_token` char(255) DEFAULT NULL,

`auth_key` char(255) DEFAULT NULL,

`team_id` int(11) DEFAULT NULL,

`team_tmp_id` int(11) DEFAULT NULL,

foreign key (team_id) references team (id) on delete cascade on update cascade,

foreign key (team_tmp_id) references team_tmp (id) on delete cascade on update cascade

) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
1	id	int(11)			Нет	*Нет*	AUTO_INCREMENT
2	surname	char(20)	utf8_general_ci		Нет	*Нет*	
3	name	char(20)	utf8_general_ci		Нет	*Нет*	
4	middle_name	char(20)	utf8_general_ci		Нет	*Нет*	
5	age	int(11)			Нет	*Нет*	
6	mail	char(45)	utf8_general_ci		Да	*NULL*	
7	login	char(45)	utf8_general_ci		Да	*NULL*	
8	password_hash	char(255)	utf8_general_ci		Да	*NULL*	
9	password_reset_token	char(255)	utf8_general_ci		Да	*NULL*	
10	auth_key	char(255)	utf8_general_ci		Да	*NULL*	
11	team_id	int(11)			Да	*NULL*	
12	team_tmp_id	int(11)			Да	*NULL*	

Рисунок 5.7 – Проект таблицы coach

Таблица «match» (рисунок 5.8) хранит данные о матчах:

1. id – первичный ключ;
2. date – дата матча;
3. team1_id – команда 1;
4. team2_id – команда 2;
5. refereeing_brigabe – судейская бригада;
6. tournament_id – турнир;
7. tour_id – тур;
8. stadium – стадион.

SQL-скрипт создания таблицы «match»:

```
CREATE TABLE IF NOT EXISTS `match` (
  `id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `date` date DEFAULT NULL,
  `team1_id` int(11) DEFAULT NULL,
  `team2_id` int(11) DEFAULT NULL,
  `refereeing_brigade_id` int(11) DEFAULT NULL,
  `score` int(11) DEFAULT NULL,
  `tournament_id` int(11) DEFAULT NULL,
  `tour_id` int(11) DEFAULT NULL,
  `stadium` char(45) DEFAULT NULL,
  foreign key (tournament_id) references tournament (id) on delete cascade on update cascade,
  foreign key (tour_id) references tour (id) on delete cascade on update cascade,
  foreign key (team1_id) references team (id) on delete cascade on update cascade,
  foreign key (team2_id) references team (id) on delete cascade on update cascade
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;
```

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
1	id	int(11)			Нет	*Нет*	AUTO_INCREMENT
2	date	date			Да	*NULL*	
3	team1_id	int(11)			Да	*NULL*	
4	team2_id	int(11)			Да	*NULL*	
5	refereeing_brigade_id	int(11)			Да	*NULL*	
6	tournament_id	int(11)			Да	*NULL*	
7	tour_id	int(11)			Да	*NULL*	
8	stadium	char(45)	utf8_general_ci		Да	*NULL*	

Рисунок 5.8 – Проект таблицы match

Таблица «news» (рисунок 5.9) хранит новости турнира:

1. id – первичный ключ;
2. caption – заголовок новости;
3. short_description – краткое описание;

4. text – текст новости;
5. creation – дата создания;
6. update – дата изменения;
7. image – изображение.

SQL-скрипт создания таблицы «news»:

```
CREATE TABLE IF NOT EXISTS `news` (
  `id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `caption` varchar(255) NOT NULL,
  `short_decription` varchar(255) NOT NULL,
  `text` text NOT NULL,
  `creation` date NOT NULL,
  `update` date DEFAULT NULL,
  `image` varchar(45) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;
```

	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
☐	1	id	int(11)			Нет	*Нет*	AUTO_INCREMENT
☐	2	caption	varchar(255)	utf8_general_ci		Нет	*Нет*	
☐	3	short_decription	varchar(255)	utf8_general_ci		Нет	*Нет*	
☐	4	text	text	utf8_general_ci		Нет	*Нет*	
☐	5	creation	date			Нет	*Нет*	
☐	6	update	date			Да	*NULL*	
☐	7	image	blob			Нет	*Нет*	

Рисунок 5.9 – Проект таблицы news

Таблица «referee» (рисунок 5.10) хранит данные о судьях турнира:

1. id – первичный ключ;
2. surname – фамилия судьи;
3. name – имя судьи;
4. middle_name – отчество судьи;
5. date_of_birth – дата рождения;
6. category – судейская категория;
7. refereeing_brigade_id – судейская бригада.

SQL-скрипт создания таблицы «referee»:

```
CREATE TABLE IF NOT EXISTS `referee` (
  `id` int(11) NOT NULL,
  `surname` char(45) NOT NULL,
  `name` char(45) NOT NULL,
  `middle_name` char(45) NOT NULL,
  `date_of_birth` date DEFAULT NULL,
  `category` int(11) DEFAULT NULL,
  `refereeing_brigade_id` int(11) DEFAULT NULL,
  foreign key (refereeing_brigade_id) references refereeing_brigade (id) on delete cascade on update cascade
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;
```

	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
☐	1	id	int(11)			Нет	*Нет*	AUTO_INCREMENT
☐	2	surname	char(45)	utf8_general_ci		Нет	*Нет*	
☐	3	name	char(45)	utf8_general_ci		Нет	*Нет*	
☐	4	middle_name	char(45)	utf8_general_ci		Нет	*Нет*	
☐	5	date_of_birth	date			Да	*NULL*	
☐	6	category	int(11)			Да	*NULL*	
☐	7	refereeing_brigade_id	int(11)			Да	*NULL*	

Рисунок 5.10 – Проект таблицы referee

Таблица «referee_brigade» (рисунок 5.11) хранит данные о судейских бригадах:

1. id – первичный ключ;
2. brigade_number – номер бригады.

SQL-скрипт создания таблицы «refereeing_brigade»:

```
CREATE TABLE IF NOT EXISTS `refereeing_brigade` (
  `id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `brigabe_number` int(11) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;
```

	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
☐	1	id	int(11)			Нет	*Нет*	AUTO_INCREMENT
☐	2	brigabe_number	int(11)			Нет	*Нет*	

Рисунок 5.11 – Проект таблицы referee_brigade

Таблица «tour» (рисунок 5.12) хранит данные о туре:

1. id – первичный ключ;
2. tour_number – номер тура.

SQL-скрипт создания таблицы «tour»:

```
CREATE TABLE IF NOT EXISTS `tour` (
  `id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `tour_number` int(11) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;
```

	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
☐	1	id	int(11)			Нет	*Нет*	AUTO_INCREMENT
☐	2	tour_number	int(11)			Нет	*Нет*	

Рисунок 5.12 – Проект таблицы tour

Таблица «tournament» (рисунок 5.13) хранит данные турнирах:

1. id – первичный ключ;
2. name – название турнира.

SQL-скрипт создания таблицы «tournament»:

```
CREATE TABLE IF NOT EXISTS `tournament` (
  `id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `name` char(45) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;
```

	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
☐	1	id	int(11)			Нет	*Нет*	AUTO_INCREMENT
☐	2	name	char(45)	utf8_general_ci		Нет	*Нет*	

Рисунок 5.13 – Проект таблицы tournament

Таблица «age_category» (рисунок 5.14) хранит данные о возрастных категориях:

1. id – первичный ключ;
2. category – категория.

SQL-скрипт создания таблицы «age_category»:

```
CREATE TABLE IF NOT EXISTS `age_category` (
  `id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `category` varchar(45) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;
```

	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
☐	1	id	int(11)			Нет	*Нет*	AUTO_INCREMENT
☐	2	category	varchar(45)	utf8_general_ci		Нет	*Нет*	

Рисунок 5.14 – Проект таблицы age_category

Таблица «action» (рисунок 5.14) хранит данные о действиях игроков:

1. id – первичный ключ;
2. action – действия;
3. player_id – игрок;
4. team_id – команда;
5. match_id – матч.

SQL-скрипт создания таблицы «action»:

```
CREATE TABLE IF NOT EXISTS `action` (
  `id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `action` varchar(11) DEFAULT NULL,
  `player_id` int(11) DEFAULT NULL,
  `team_id` int(11) DEFAULT NULL,
  `match_id` int(11) DEFAULT NULL,
   foreign key (player_id) references player (id) on delete cascade on update
cascade,
   foreign key (match_id) references `match` (id) on delete cascade on update
cascade,
   foreign key (team_id) references team (id) on delete cascade on update cascade
```

) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
1	id	int(11)			Нет	*Нет*	AUTO_INCREMENT
2	action	varchar(11)	utf8_general_ci		Да	*NULL*	
3	player_id	int(11)			Да	*NULL*	
4	team_id	int(11)			Да	*NULL*	
5	match_id	int(11)			Да	*NULL*	

Рисунок 5.15 – Проект таблицы action

Краткие выводы по разделу пять

В данном разделе расположена общая схема базы данных, описана ее структура. Представлено подробное описание таблиц с рассмотрением всех полей.

6 РАЗРАБОТКА СОСТАВНЫХ ЧАСТЕЙ

6.1 Серверная часть

API получает и обрабатывает HTTP – запросы от клиента и возвращает результат в формате JSON с определенным кодом ответа HTTP, соответствующим результату.

JSON – текстовый формат обмена данными, основанный на JavaScript и обычно используемый именно с этим языком. Как и многие другие текстовые форматы, JSON легко читается людьми.

Несмотря на происхождение от JavaScript (точнее, от подмножества языка стандарта ECMA-262 1999 года), формат считается независимым от языка и может использоваться практически с любым языком программирования. Для многих языков существует готовый код для создания и обработки данных в формате JSON.

Метод HTTP – последовательность из любых символов, кроме управляющих и разделителей, указывающая на основную операцию над ресурсом. Обычно метод представляет собой короткое английское слово, записанное заглавными буквами. Название метода чувствительно к регистру.

Каждый сервер обязан поддерживать как минимум методы GET и HEAD

Кроме методов GET и HEAD, часто применяется метод POST.

В данной работе также используются методы PUT и DELETE.

Метод GET используется для запроса содержимого указанного ресурса. С помощью метода GET можно также начать какой-либо процесс. В этом случае в тело ответного сообщения следует включить информацию о ходе выполнения процесса.

Метод HEAD аналогичен методу GET, за исключением того, что в ответе сервера отсутствует тело. Запрос HEAD обычно применяется для извлечения метаданных, проверки наличия ресурса (валидация URL) и чтобы узнать, не изменился ли он с момента последнего обращения.

Заголовки ответа могут кэшироваться. При несовпадении метаданных ресурса с соответствующей информацией в кэше копия ресурса помечается как устаревшая.

Метод POST применяется для передачи пользовательских данных заданному ресурсу. При этом передаваемые данные включаются в тело запроса. Аналогично с помощью метода POST обычно загружаются файлы на сервер.

При результате выполнения 200 (Ok) в тело ответа следует включить сообщение об итоге выполнения запроса. Если был создан ресурс, то серверу следует вернуть ответ 201 (Created) с указанием URI нового ресурса.

Метод PUT применяется для загрузки содержимого запроса на указанный в запросе URI. Если по заданному URI не существует ресурс, то сервер создаёт его и возвращает статус 201 (Created). Если же был изменён ресурс, то сервер возвращает 200 (Ok) или 204 (No Content).

Метод DELETE удаляет указанный ресурс.

Используемые в работе коды состояния HTTP:

- 2xx Success («Успех»).

  Сообщения данного класса информируют о случаях успешного принятия и обработки запроса клиента. В зависимости от статуса сервер может ещё передать заголовки и тело сообщения.
- 4xx Client Error («Ошибка клиента»).

  Класс кодов 4xx предназначен для указания ошибок со стороны клиента.
- 5xx Server Error («Ошибка сервера»).

  Коды 5xx выделены под случаи неудачного выполнения операции по вине сервера.

Все ресурсы API с кратким описанием представлены в таблице 6.1.

Таблица 6.1. Ресурсы RestAPI

URL	**Метод**	**Параметры**	**Таблица**	**Описание**	**Роль**
/signup	POST	surname, name, middle_name, email, password, receive_news	User	Регистрация	Все
/login	POST	email, password	User	Вход	Все
/news	GET	title, short_description, text, image	News	Просмотр новостей	Все
/news/:id	GET	Title, Text, Image	News	Просмотр конкретной новости	Все
/news/:id	POST	Title, Short_description Text, Image	News	Добавление новости	Организатор
/news/:id	PUT	Title, Short_description	News	Обновление новости	Организатор

		Text, Image			
/news/:id	DELETE	Title, Short_description Text, Image	News	Удаление новости	Организатор
/teams/	POST	Name, Age_category	Teams	Просмотр команд	Все
/teams/:id	POST	Name Age_category	Teams	Просмотр конкретной команды	Зарегистрировнный пользователь, тренер
		Surname, Name, Position	players		
/teams/	POST	Name Age_category	Teams	Добавление команды	Организатор
/teams/	PUT	Name Age_category	Teams	Изменение команд	Организатор
/teams/	DELETE	Name Age_category	Teams	Удаление команд	Организатор
/teams/:id	POST	Name, Age_category	Teams	Добавление игроков в команду	Организатор, Тренер
		Surname, Name, Middle_name Age, Position	players		

/teams/:id	PUT	Name, Age_category	Teams	Обновление информации об игроке	Организатор, Тренер
		Surname, Name, Middle_name Age, Position	players		
/teams/:id	DELETE	Name, Age_category	teams	Удаление игроков из команды	Организатор, Тренер
		Surname, Name, Middle_name Age, Position	Players		
/table/	POST	Name Age_category	Teams	Просмотр турнирной таблицы	Все
/statistic/	POST	Surrname, Name	players	Просмотр статистки игроков	Зарегистрированный пользователь, тренер
		Name	Teams		
/calendar/	POST	Name,	match	Просмотр	Все

		Team1, Team2, Tour, Tournament, Date		расписани я матчей	
/calendar/: id	POST	Name, Team1, Team2, Tour, Tournament, Date	Match	Просмотр конкретно го матча	Все
/calendar/: id	POST	Name, Team1, Team2, Tour, Tournament, Date	Match	Добавлени е матча	Организатор
/calendar/: id	GET	Name, Team1, Team2, Tour, Tournament, Date	Match	Изменени е матча	Организатор
/calendar/: id	DELET E	Name, Team1, Team2, Tour, Tournament, Date	Match	Удаление матча	Организатор

На рисунке 6.1 показа диаграмма классов приложения.

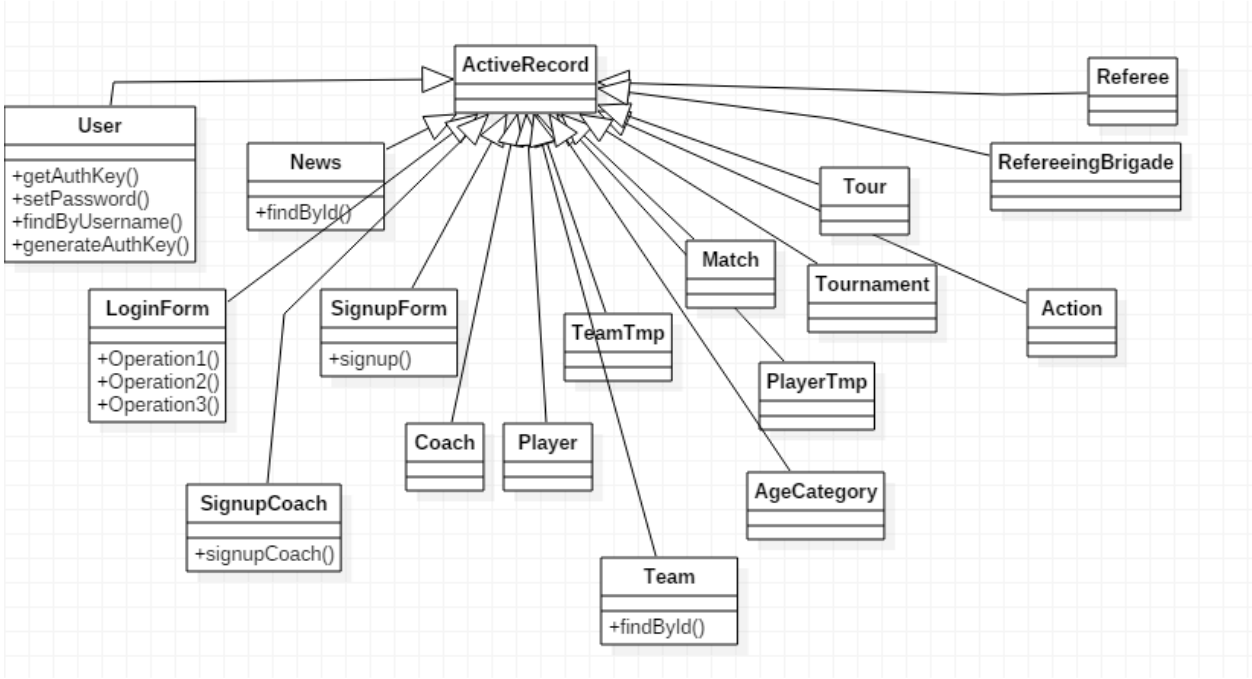


Рисунок 6.1 – Диаграмма классов

В фреймворке реализован интерфейс ActiveRecord представляющий собой шаблон проектирования приложений, описанный Мартином Фаулером. AR является популярным способом доступа к данным реляционных баз данных в объектно-ориентированном программировании. Каждый класс, наследованный от ActiveRecord, представляет собой таблицу из базы данных, объектом класса будет являться запись из этой таблицы.

В PHPStorm все классы расположены в разделе models (рисунок 6.2). Также в разделе modules/admin реализован интерфейс для организатора.

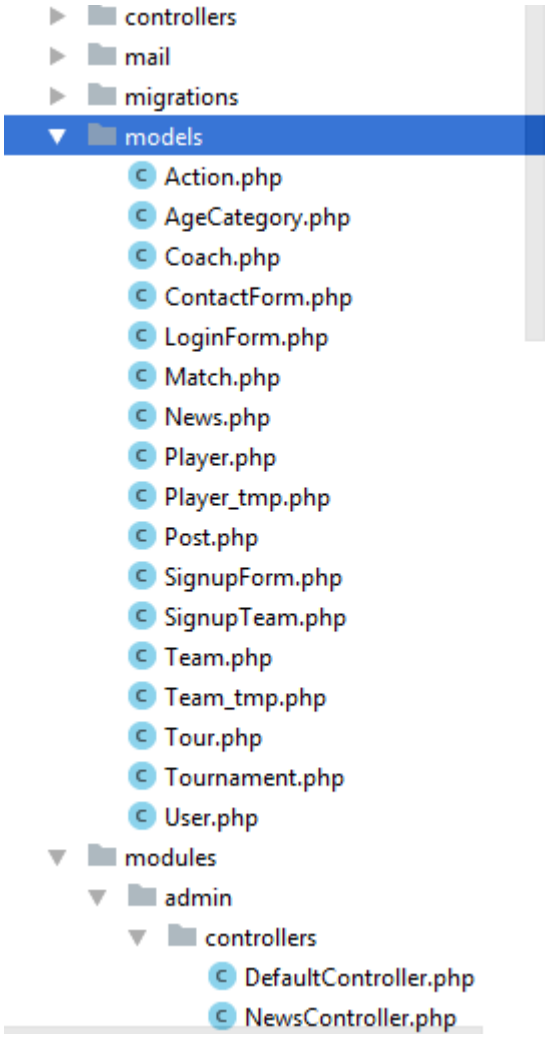


Рисунок 6.2 – Структура проекта в PHPStorm

В таблице 6.2 описаны основные функции из диаграммы классов.

Таблица 6.2

Класс	**Функция**	**Описание**
Team	findById	Поиск команды по ID
SignupCoach	signupCoach	Регистрация тренера
News	findById	Поиск новости по ее ID
SignupForm	Signup	Регистрация пользователя
User	getAuthKey	Получение ключа авторизации
	setPassword	Установка пароля
	findByUsername	Поиск по логину

		пользователя
	generateAuthKey	Создание ключа авторизации
LoginForm	getUser	Получение пользователя из класса User
	Login	Авторизация
	validatePassword	Проверка пароля

Ниже представлены диаграммы для добавления игроков в команду (рисунок 6.3) и их изменения (рисунок 6.4), диаграмма для отображения страницы с новостью (рисунок 6.5) и для отображения статистики (рисунок 6.6).

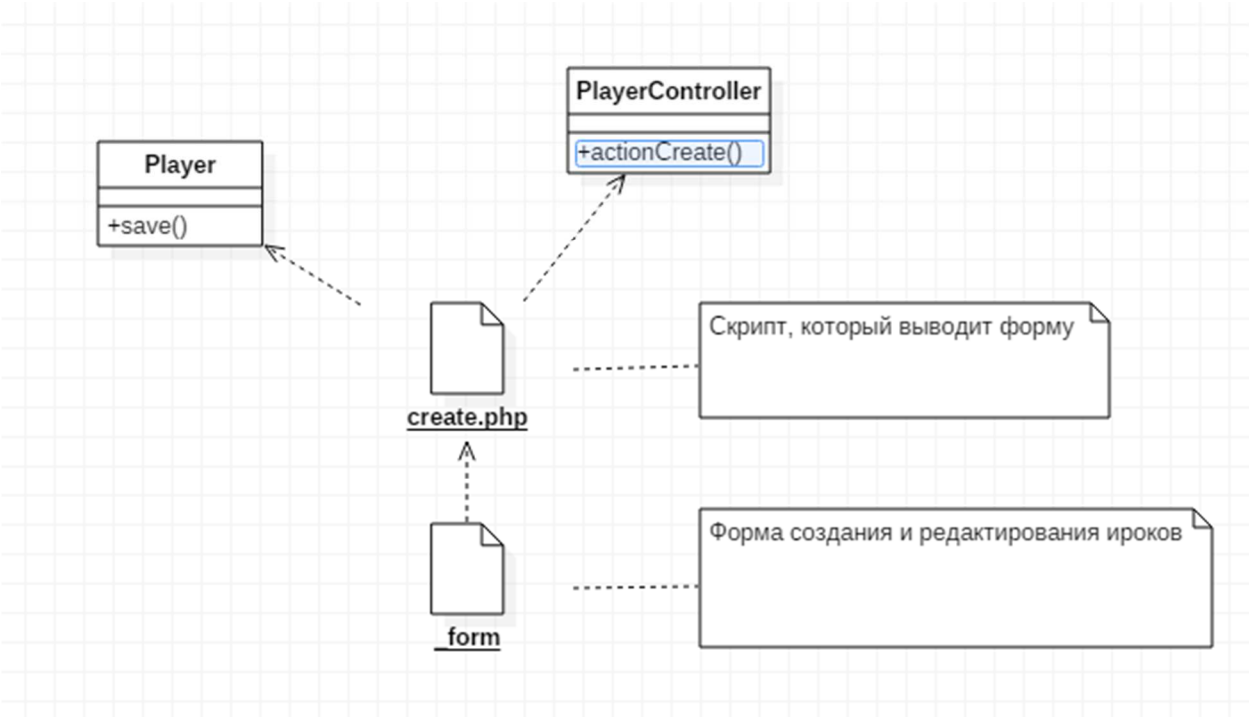


Рисунок 6.3 – Добавление игрока

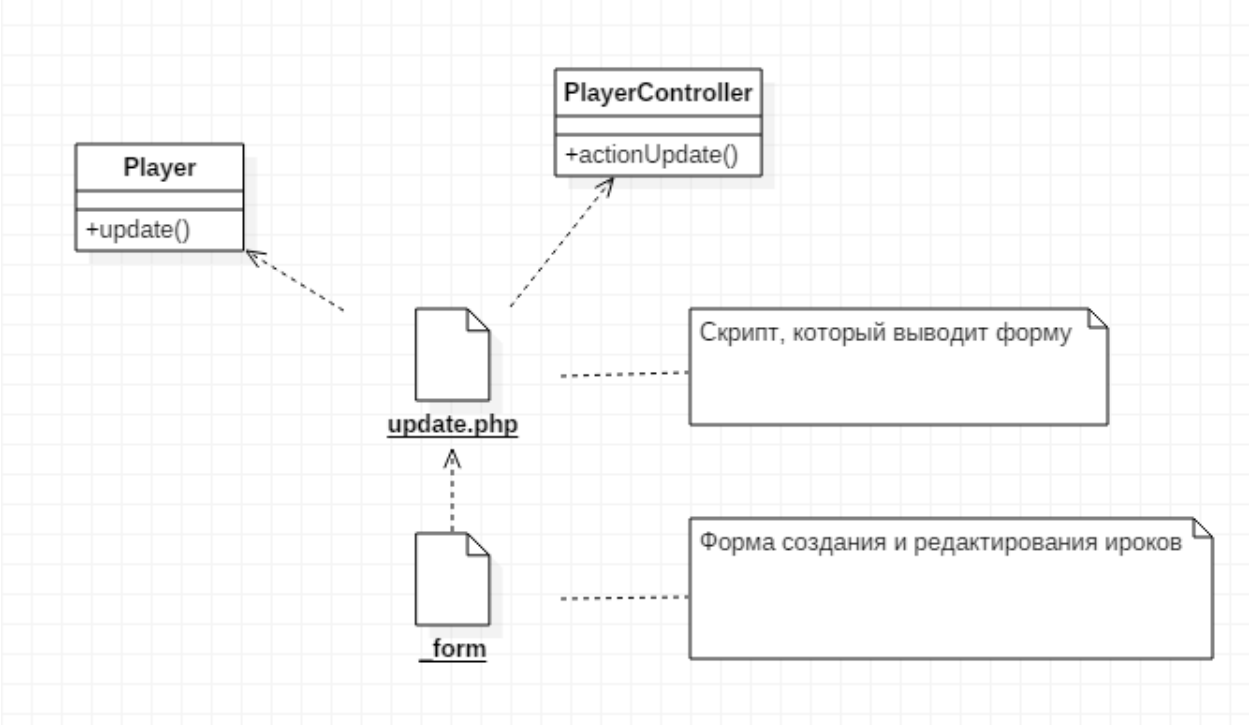


Рисунок 6.4 – Изменения игрока

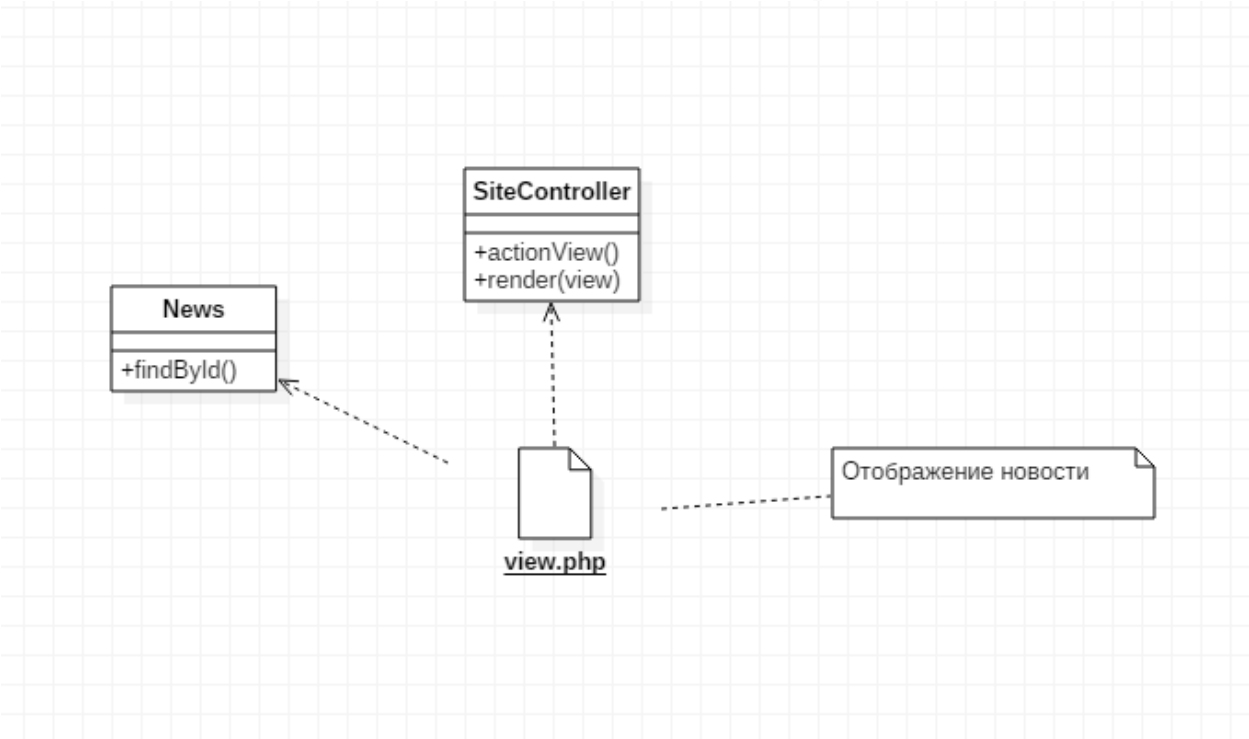


Рисунок 6.5 – Страница с новостью

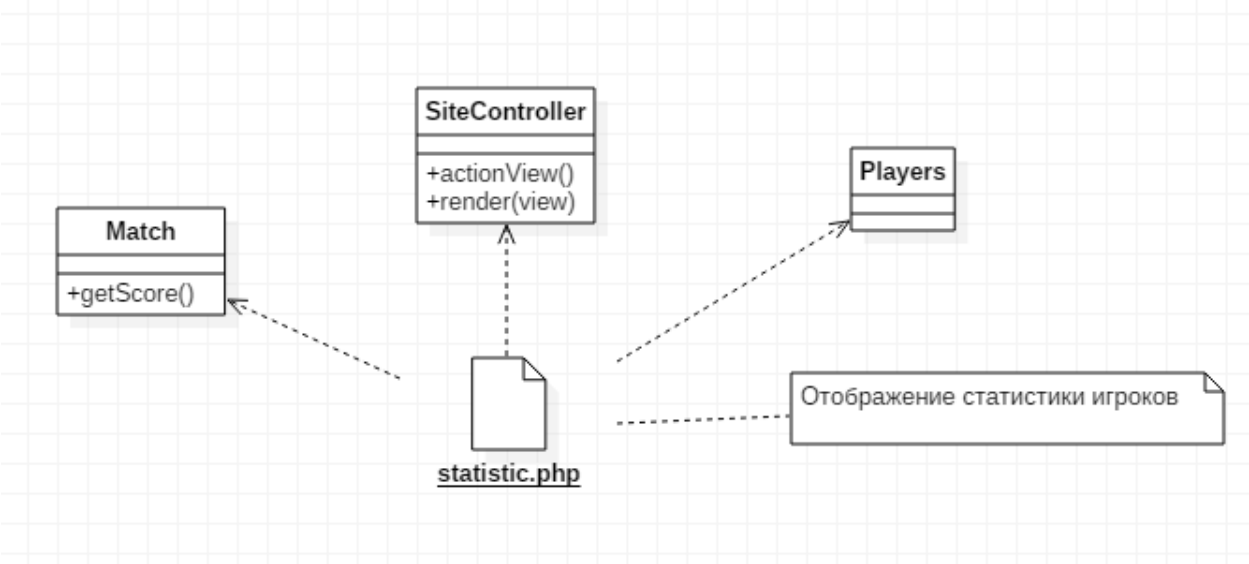


Рисунок 6.6 – Статистика игроков

6.2 Клиентская часть

При запуске приложения открывается представление «home» (рисунок 6.7), отвечающее за отображение главной страницы. Нажав кнопку «Принять участие», пользователь переходит на страницу регистрации для тренера.

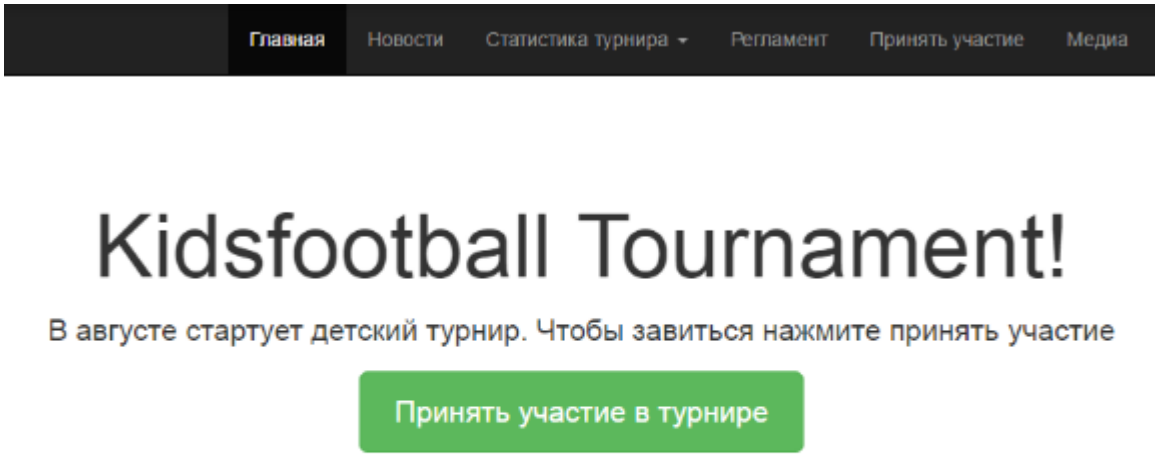


Рисунок 6.7 – Стартовая страница

В пункте «Принять участие» (рисунок 6.8) тренер заполняет поля регистрации и нажав кнопку «Отправить» переходит в пункт «Регистрация команды».

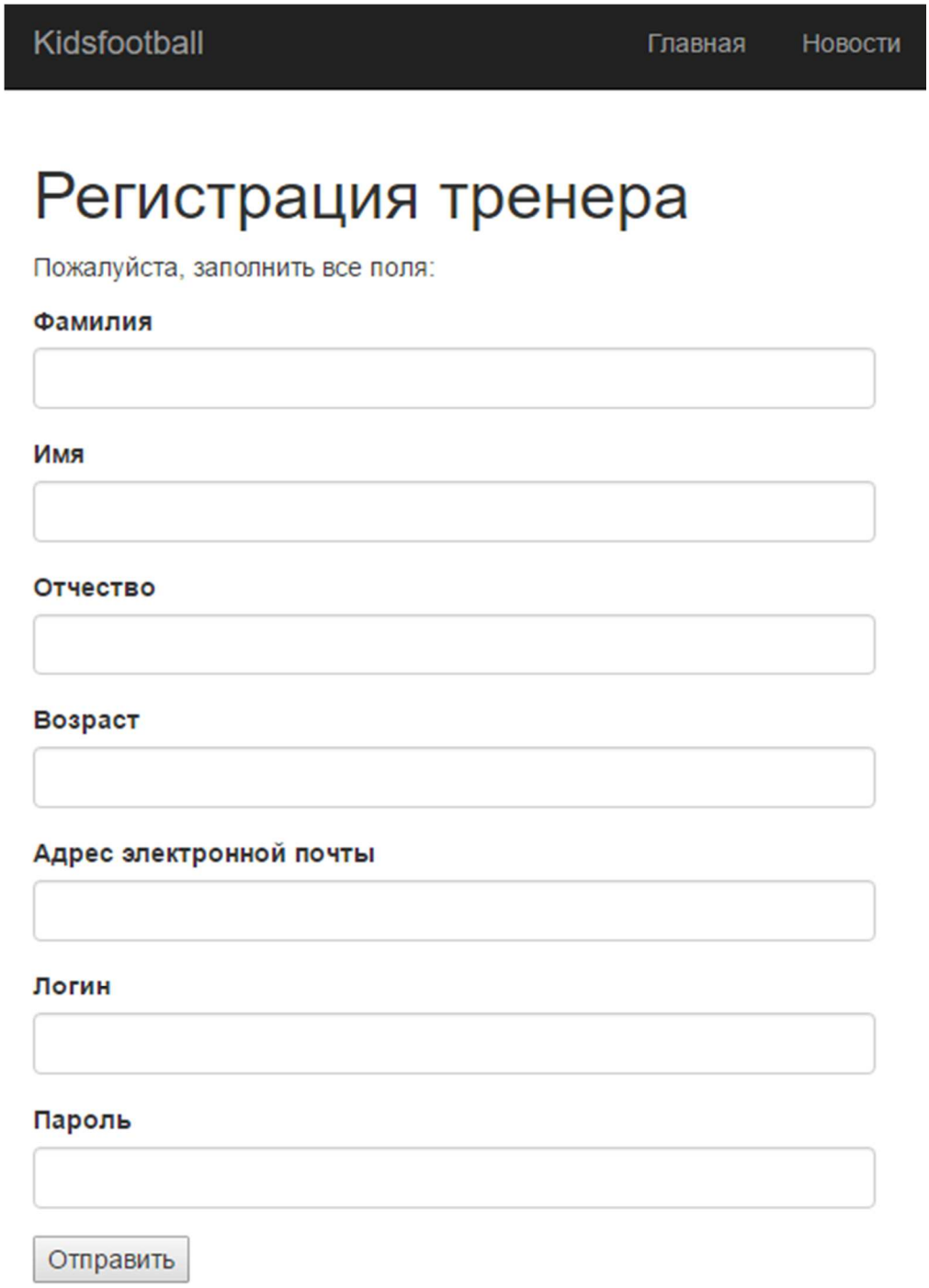


Рисунок 6.8 – Регистрация тренера

Также новый пользователь может зарегистрироваться. При выборе пункта регистрации загружается новое представление, где необходимо заполнить все текстовые поля, нажать на кнопку, и регистрация будет завершена (рисунок 6.9).

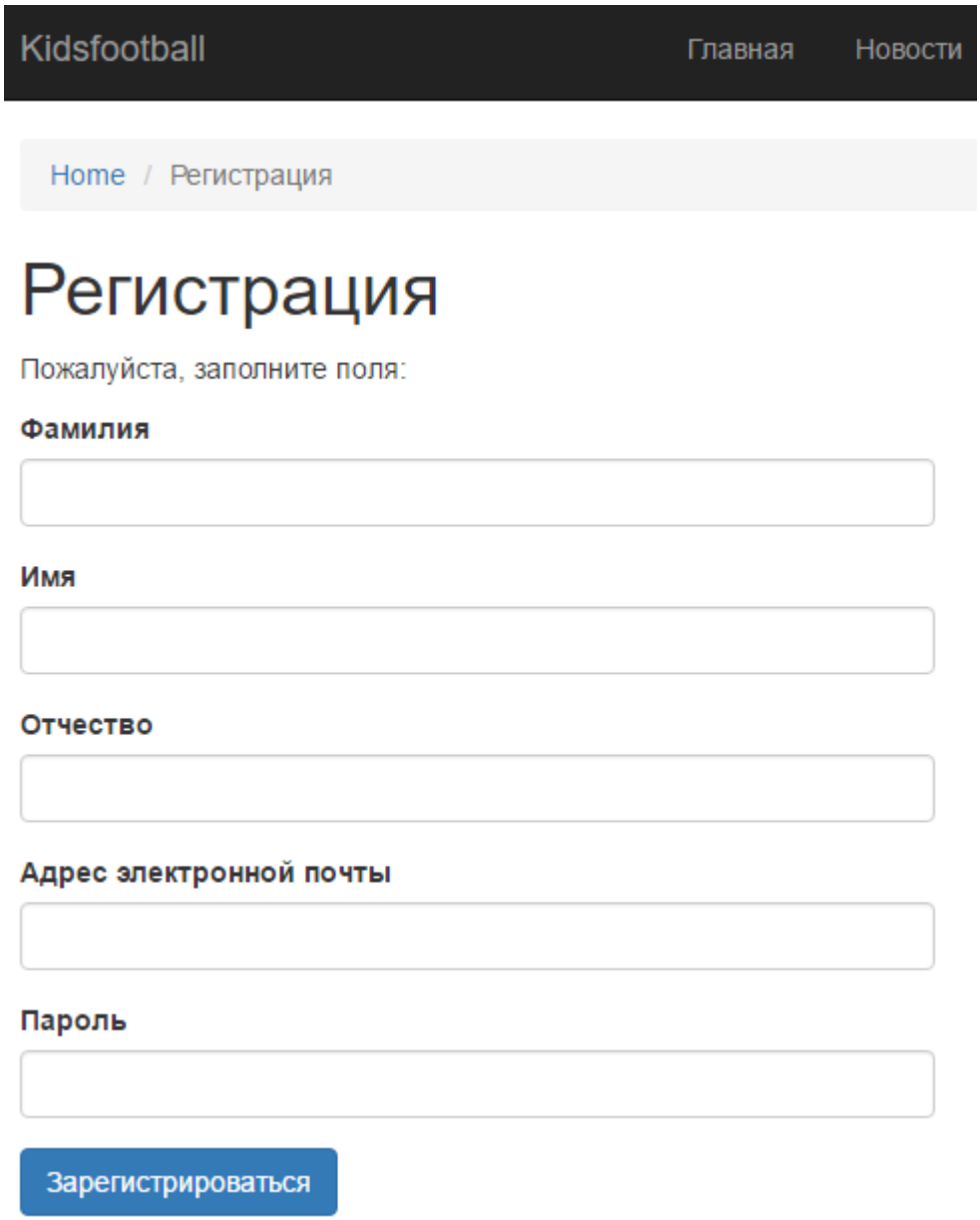


Рисунок 6.9 – Регистрация

Для входа в систему существует фрагмент «Войти» на котором расположены текстовые поля для ввода логина и пароля, и кнопка для входа (рисунок 6.10).

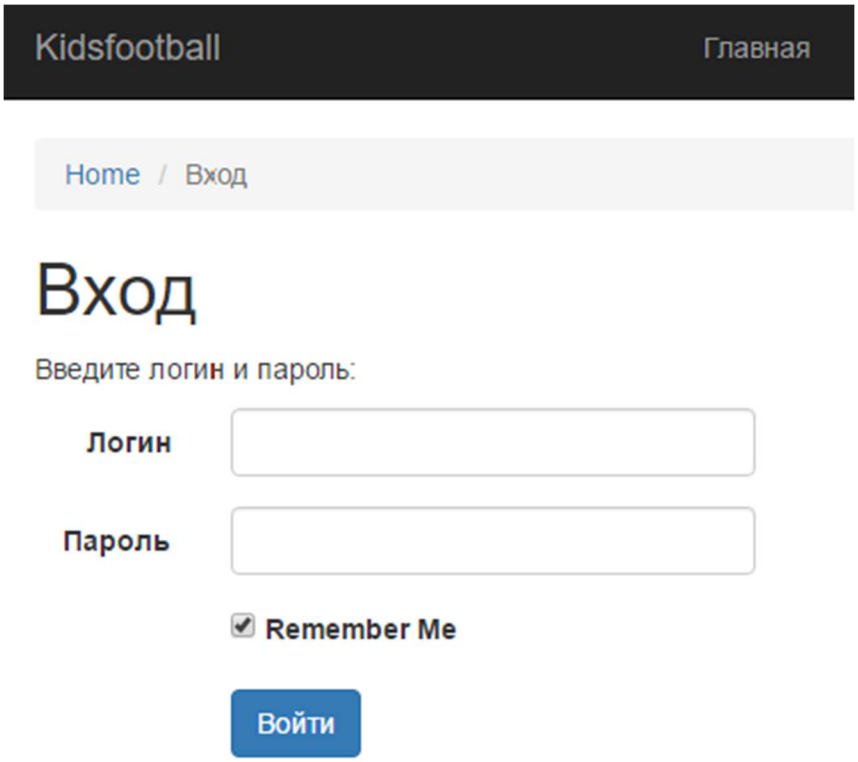


Рисунок 6.10 – Страница входа

В представлении «News» отображается список новостей. Также пользователь может выбрать конкретную новость (рисунок 6.11) с полным текстом.

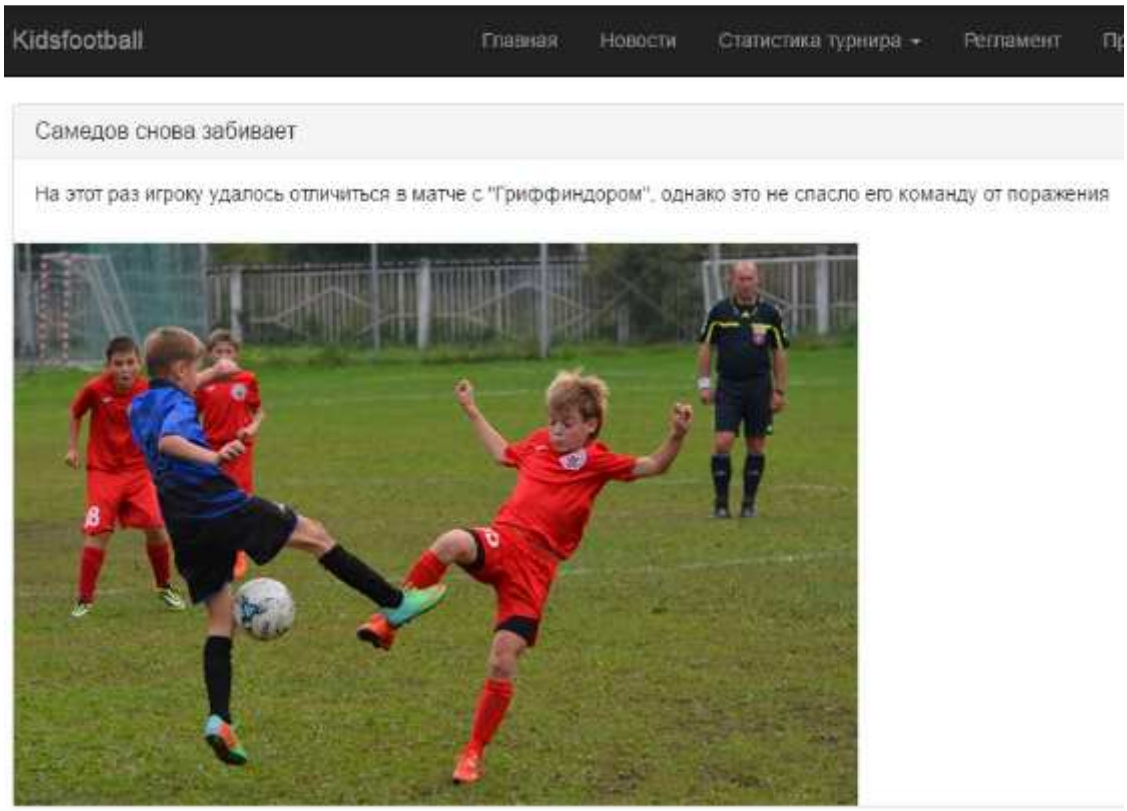


Рисунок 6.11 – Страница «Новости»

В пункте меню «Статистика турнира» пользователь может выбрать конкретный турнир и посмотреть по нему подробную статистику. В нее входит: календарь (рисунок 6.12), турнирная таблица (рисунок 6.13), статистика игроков, список команд .

Команды | Календарь | Турнирная таблица | Статистика

Тур	Дата	Команды	Счет
1	11.06.2017,18:00	Локомотив - Сигнал	1:0
1	11.06.2017,19:00	Сатурн - Гриффиндор	4:0
1	11.06.2017,20:00	Тигры - Ливерпуль	1:1
1	11.06.2017,21:00	Ювентус - Челябинск	2:1

Рисунок 6.12 – Расписание турнира

Команды | Календарь | Турнирная таблица | Статистика

Позиция	Команла	Игры	Победы	Ничьи	Пораж.	Мячи	Очки
1	Локомотив	7	7	0	0	20-3	21
2	Сатурн	7	6	0	1	18-4	18
3	Сигнал	7	3	1	3	12-7	10
4	Ливерпуль	7	2	3	2	9-9	9
5	Челябинск	7	3	0	4	8-11	9
6	Ювентус	7	2	0	5	5-13	6
7	Тигры	7	1	1	5	2-14	4
8	Гриффиндор	7	0	1	6	2-21	1

Рисунок 6.13 – Турнирная таблица

Организатор в разделе «News» (рисунок 6.14) может добавлять, редактировать и удалять новости.

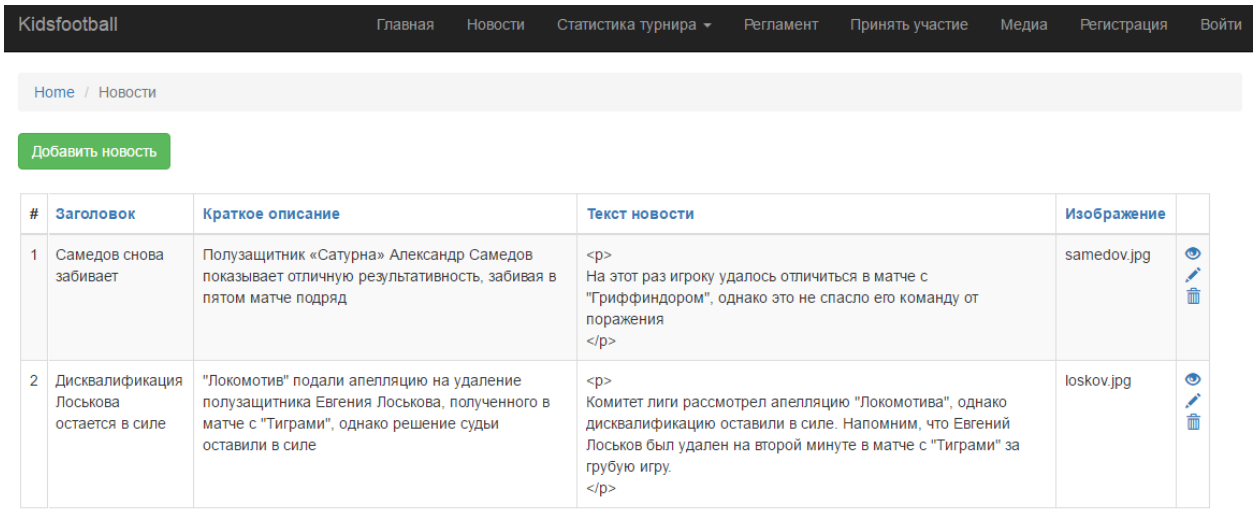


Рисунок 6.14 – Раздел редактирования новостей для организатора

ЗАКЛЮЧЕНИЕ

В рамках дипломной работы было выполнено следующее:

1. изучены бизнес-процессы предметной области, сформулирован набор требований к приложению;
2. рассмотрены родственные разработки, обоснована необходимость создания приложения;
3. определена среда реализации приложения;
4. разработана архитектура системы;
5. спроектирована база данных;
6. реализована серверная и клиентская части, произведено тестирование приложения.

Возможными путями развития приложения являются:

1. разработка мобильного приложения;
2. улучшение интерфейса;
3. добавление нового функционала.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Лутц М. Изучаем Python/ М.Лутц. – М.: Символ, 2011. – 1280с.

2 Сравнения PHP фреймворков. – https://habrahabr.ru/post/305626/

3 Устройство и характеристика фреймворка Yii2. – https://github.com/yiisoft/yii2/tree/master/docs/guide-ru

4 Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования/Э.Гамма, Р.Хелм, Р.Ддонсон, Дж.Влиссидес. – СПб.: Питер, 2001. – 368 с.

5 Вайсфельд М. PHP и MySQL. Объекто-ориентированное мышление/ М.Вайсфельд. – СПб.: Питер, 2010 – 304 с. электронный вариант.

6 Суэринг С., Конверс Т. PHP и MySQL. Библия программиста/С. Суэринг, Т.Конверс. – М.: Вильямс, 2010. - 912 с. электронный вариант.

7 Метц С. Ruby. Объектно-ориентированное проектирование/ С. Метц. – СПб.: Питер, 2017. – 304с.

8 MySQL Documentation – https://dev.mysql.com/doc/

9 Справочник по JavaScript – http://javascript.ru/manual

10 Фаулер М. Архитектура корпоративных программных приложений / М. Фаулер, Д. Райс, М. Фоммел, Э. Хайет. – 1-е изд.; Вильямс, 2010. – 544 с.: