**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**



Федеральное государственное автономное образовательное учреждение

высшего профессионального образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт кибернетики

Кафедра Автоматики и компьютерных систем

Специальность «230400 Информационные системы и технологии»

Пояснительная записка к курсовому проекту по дисциплине

«Интернет-технологии»

на тему «Волейбольная статистика матчей

Студент группы 8И11 \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_ А.Г. Кулемеев

Проверил Ассистент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ И.И. Савенко

**Задание**

Разработать веб-приложение, удовлетворяющее следующим требованиям:

* соответствие стандартам HTML5, CSS2/3;
* современный дизайн;
* язык программирования на стороне сервера – PHP, либо любой другой;
* язык программирования на стороне клиента – JavaScript;
* СУБД MySQL, либо любая другая;
* наличие не менее пяти веб-страниц с различным содержимым;
* возможность входа для редактора (может редактировать элементы каталога), менеджера (может просматривать и удалять заказы) и администратора (может выполнять действия редактора и менеджера, а также загружать с сервера содержимое каталога и заказы в формате XML);
* отсутствие уязвимостей SQL-injection, XSS и др.

# **Реферат**

Отчет содержит 22 с., 9 рисунков.

Данный пояснительная записка содержит теоретический материал, описывающий и .NET Framework[4], Entity Framework [5] и краткое описание алгоритмов работы и создания веб-приложения.

# **Список используемых сокращений**

БД – База данных

EF – Entity Framework

VS 2013 – Microsoft Visual Studio 2013

MVC –Model View Controller

# **Содержание**

С.

[**Задание** 2](#_Toc406591531)

[**Реферат** 3](#_Toc406591532)

[**Список используемых сокращений** 4](#_Toc406591533)

[**Содержание** 5](#_Toc406591534)

[**Введение** 6](#_Toc406591535)

[**1** **Теоритическая часть** 8](#_Toc406591536)

[**1.1** **Entity Framework** 8](#_Toc406591537)

[**1.2** **.NET Framework** 10](#_Toc406591538)

[**1.3** **MVC** 11](#_Toc406591539)

[**2** **Практическая часть** 14](#_Toc406591540)

[**2.1** **Проектирование базы данных** 14](#_Toc406591541)

[**2.2 Проектирование приложения** 16](#_Toc406591542)

[**Заключение** 21](#_Toc406591543)

[**Список используемой литературы** 22](#_Toc406591544)

# **Введение**

В наше время Web-технологии набирают все большую популярность. Практически все крупные предприятия сейчас уже имеют собственные сайты. В связи с большим спросом на web-разработки, сами технологии постоянно развиваются. Сегодня все чаще и чаще мы слышим разговоры о веб-приложениях. И хотя эти разговоры на слуху не так давно, веб-приложениями мы с вами пользуемся уже не год и не два. Многие из них прочно вошли в нашу жизнь, и мы над ними даже не задумываемся, когда используем. Некоторые считают, что создание веб-приложений сегодня – это просто дань моде, другие – что за ними будущее. Где же истина, и что на самом деле представляют собой эти загадочные для многих веб-приложения?

Веб-приложение представляет собой пользовательскую программу, основная часть которой содержится на удаленном сервере, а пользовательский интерфейс (UI) отображается в браузере в виде веб-страниц.

По своей сути запуск веб-приложения ничем не отличается от загрузки обычной веб-страницы: вводим ссылку в браузер - и оно перед нами, точнее, верхняя часть айсберга, которой является интерфейс пользователя. В этом есть несколько плюсов. Первый из них – это то, что само по себе приложение абсолютно не зависит от того, какая операционная система установлена на компьютере пользователя, то есть оно, по сути, является кросс-платформенным.

Вторым плюсом веб-приложений можно назвать то, что сам факт их существования полностью меняет способ распространения продукта. Здесь создатели отходят от традиционных способов распространения программных продуктов путем продажи копий и установки их на каждый компьютер пользователей. Теперь все гораздо проще: единственная версия приложения расположена на сервере, а все пользователи имеют доступ к ней, вернее, к ее пользовательскому интерфейсу из любого места в мире.

Третьим плюсом для пользователя мы назовем то, что ему нет необходимости устанавливать и настраивать программное обеспечение – все уже установлено на серверах и настроено разработчиками. Все, что требуется от пользователя, это изменить на свой вкус несколько ключевых параметров.

Четвертым плюсом можно назвать то, что для работы с приложением от пользователя, по большому счету, ничего и не нужно. Разве что компьютер и установленный браузер.

Следующий положительный момент веб-приложений касается их разработчиков. Ввиду того, что основная часть веб-приложения сконцентрирована на сервере в одном месте, куда проще заниматься его настройкой, не нужно содержать огромные команды специалистов технической поддержки, занимающихся консультациями пользователей и настройкой приложения на компьютерах во всем мире. Это гораздо менее затратно в финансовом плане и куда более эффективно. При этом пользователю невидима архитектура приложения, в любой момент можно добавить любое количество серверов, на которых установлена основная составляющая приложения, добавить вычислительные мощности, и пользователь этого даже не заметит.

Таким образом, мы видим, что веб-приложения имеют большое количество плюсов при отсутствии видимых минусов, самым большим и очевидным из которых является невозможность использования приложений при отсутствии доступа к Всемирной паутине.

# **Теоритическая часть**

## **Entity Framework**

**Entity Framework** представляет специальную объектно-ориентированную технологию на базе фреймворка .NET для работы с данными. Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работает с объектами.

Первая версия Entity Framework - 1.0 вышла еще в 2008 году и представляла очень ограниченную функциональность, базовую поддержку ORM (object-relational mapping - отображения данных на реальные объекты) и один единственный подход к взаимодействию с бд - Database First. С выходом версии 4.0 в 2010 году многое изменилось - с этого времени Entity Framework стал рекомендуемой технологией для доступа к данным, а в сам фреймворк были введены новые возможности взаимодействия с бд - подходы Model First и Code First.

Дополнительные улучшения функционала последовали с выходом версии 5.0 в 2012 году. И наконец, в 2013 году был выпущен Entity Framework 6.0, обладающий возможностью асинхронного доступа к данным.

Центральной концепцией Entity Framework является понятие сущности или entity. Сущность представляет набор данных, ассоциированных с определенным объектом. Поэтому данная технология предполагает работу не с таблицами, а с объектами и их наборами.

Отличительной чертой Entity Framework является использование запросов LINQ для выборки данных из БД. С помощью LINQ мы можем не только извлекать определенные строки, хранящие объекты, из бд, но и получать объекты, связанные различными ассоциативными связями.

Другим ключевым понятием является Entity Data Model. Эта модель сопоставляет классы сущностей с реальными таблицами в БД.

Entity Data Model состоит из трех уровней: концептуального, уровень хранилища и уровень сопоставления (маппинга).

На концептуальном уровне происходит определение классов сущностей, используемых в приложении.

Уровень хранилища определяет таблицы, столбцы, отношения между таблицами и типы данных, с которыми сопоставляется используемая база данных.

Уровень сопоставления (маппинга) служит посредником между предыдущими двумя, определяя сопоставление между свойствами класса сущности и столбцами таблиц.

Таким образом, мы можем через классы, определенные в приложении, взаимодействовать с таблицами из базы данных.

**Способы взаимодействия с БД:**

Entity Framework предполагает три возможных способа взаимодействия с базой данных:

**Database first**: Entity Framework создает набор классов, которые отражают модель конкретной базы данных

**Model first**: сначала разработчик создает модель базы данных, по которой затем Entity Framework создает реальную базу данных на сервере.

**Code first**: разработчик создает класс модели данных, которые будут храниться в бд, а затем Entity Framework по этому модели генерирует базу данных и ее таблицы

## **.NET Framework**

.NET Framework — программная платформа, выпущенная компанией Microsoft в 2002 году. Основой платформы является исполняющая среда Common Language Runtime (CLR), способная выполнять как обычные программы, так и серверные веб-приложения. NET Framework поддерживает создание программ, написанных на разных языках программирования [2].

Считается, что платформа .NET Framework явилась ответом компании Microsoft на набравшую к тому времени большую популярность платформу Java компании Sun Microsystems (ныне принадлежит Oracle).

Хотя .NET является патентованной технологией корпорации Microsoft и официально рассчитана на работу под операционными системами семейства Microsoft Windows, но существуют независимые проекты (прежде всего это Mono и Portable.NET), позволяющие запускать программы .NET на многих других операционных системах.

Основной идеей при разработке .NET Framework являлось обеспечение свободы разработчика за счёт предоставления ему возможности создавать приложения различных типов, способные выполняться на различных типах устройств и в различных средах.

Вторым принципом стало ориентация на системы, работающие под управлением семейства операционных систем Microsoft Windows.

Программа для .NET Framework, написанная на любом поддерживаемом языке программирования, сначала переводится компилятором в единый для .NET промежуточный байт-код Common Intermediate Language (ранее назывался Microsoft Intermediate Language, MSIL). В терминах .NET получается *сборка*, англ.*assembly*. Затем код либо исполняется виртуальной машиной Common Language Runtime (CLR), либо транслируется утилитой NGen.exe в исполняемый код для конкретного целевого процессора. Использование виртуальной машины предпочтительно, так как избавляет разработчиков от необходимости заботиться об особенностях аппаратной части. В случае использования виртуальной машины CLR, встроенный в неё JIT-компилятор «на лету» (just in time) преобразует промежуточный байт-код в машинные коды нужного процессора. Современная технология динамической компиляции позволяет достигнуть высокого уровня быстродействия. Виртуальная машина CLR также сама заботится о базовой безопасности, управлении памятью и системе исключений, избавляя разработчика от части работы.

Архитектура .NET Framework описана и опубликована в спецификации Common Language Infrastructure (CLI), разработанной Microsoft и утверждённой ISO и ECMA. В CLI описаны типы данных .NET, формат метаданных о структуре программы, система исполнения байт-кода и многое другое.

Объектные классы .NET, доступные для всех поддерживаемых языков программирования, содержатся в библиотеке Framework Class Library (FCL). В FCL входят классыWindows Forms, ADO.NET, ASP.NET, Language Integrated Query, Windows Presentation Foundation, Windows Communication Foundation и другие. Ядро FCL называется Base Class Library (BCL).

На данный момент последней версией является .NET Framework 4.5.1.

## **MVC**

Платформа ASP.NET MVC представляет собой фреймворк для создания сайтов и веб-приложений с помощью реализации паттерна MVC. Концепция паттерна (шаблона) MVC (model - view - controller) предполагает разделение приложения на три компонента:

Контроллер (controller) представляет класс, обеспечивающий связь между пользователем и системой, представлением и хранилищем данных. Он получает вводимые пользователем данные и обрабатывает их. И в зависимости от результатов обработки отправляет пользователю определенный вывод, например, в виде представления.

Представление (view) - это собственно визуальная часть или пользовательский интерфейс приложения. Как правило, html-страница, которую пользователь видит, зайдя на сайт.

Модель (model) представляет класс, описывающий логику используемых данных.

На рисунке 1 представлена общая схема взаимодействия этих компонентов:

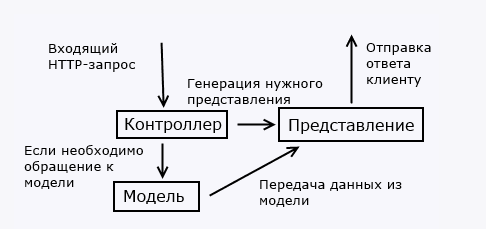


Рисунок 1 – Общая схема взаимодействия компонентов MVC

В этой схеме модель является независимым компонентом - любые изменения контроллера или представления не затрагивают модель. Контроллер и представление являются относительно независимыми компонентами, и нередко их можно изменять независимо друг от друга. Конкретные реализации и определения данного паттерна могут отличаться, но в силу своей гибкости и простоты он стал очень популярным в последнее время, особенно в сфере веб-разработки.

# **Практическая часть**

## **2.1 Проектирование базы данных**

База данных приложения содержит 19 таблиц:

* «Championships» – таблица для описания турниров, которая содержит информацию о названии турнира, количестве команд, участвующих в турнире, дате начала турнира, дате окончания турнира.
* «Teams» – описывает команды. Содержит информацию о названии команды и городе.
* «Coaching staff» – содержит информацию о тренерском штабе команды. Имеет атрибут – ФИО тренера.
* «Matches» – таблица предназначена для описания матчей, которая содержит информацию о продолжительности матча, счете матча, количестве партий и дате матча.
* «Players» – содержит информацию об игроках. Содержит информацию о ФИО игрока, игровом номере, дате рождения, росте, звании игрока.
* «Set» – содержит информацию о партиях.
* «Violations» – содержит информацию о нарушениях.
* «Referees» – содержит информацию о судьях.
* «AspNetUsers» – содержит информацию о пользователях.
* «AspNetUserRoles» – содержит информацию о соответствии роли для пользователя.
* «AspNetUserLogins» – содержит информацию о внешних именах входа.
* «AspNetUserClaims» – содержит дополнительную информацию, связанную с текущим пользователем.

Созданы таблицы-справочники, которые необходимы, для того, чтобы хранить сведения о каких-то данных, которые могут быть в дальнейшем использоваться для организации процедур проверки правильности данных или поиску по этим данным:

* «Stage» – для хранения информации о стадиях (групповой, четвертьфинал и т.д.).
* «Series» – таблица предназначена для описания результатов нескольких встреч.
* «Position referee» – в сущности хранится информация о должностях судьи.
* «Position coach» – в сущности хранится информация о должностях тренера.
* «Specialization» – в сущности хранится информация об амплуа игроков
* «AspNetRoles» – в сущности хранится информация о ролях пользователей

На рисунке 2 представлена диаграмма базы данных.

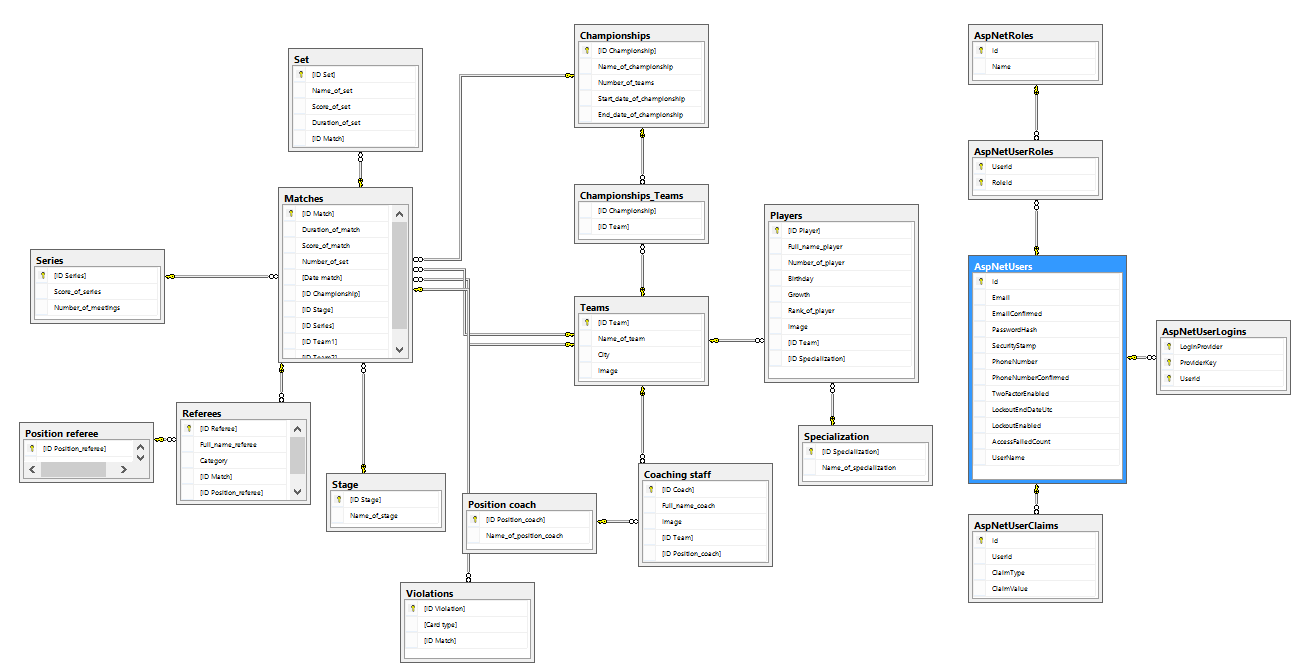


Рисунок 2 – Диаграмма базы данных

## **2.2 Проектирование приложения**

Для работы приложения при помощи подхода Database first EF была получена модель VolleyStatisticsMatches. В классе AccountViewModels содержится модель для авторизации и регистрации. Модель ManageViewModels отвечает за управление профилем. Так же были добавлены классы Metadata и PartialClasses для отображения полей таблиц БД на русском языке. Класс CaptchaImage позволяет получить Captcha, используемую при регистрации. Класс smsc\_api предназначен для отправки СМС-сообщений.

AccountController – отвечает за вывод и получение информации во время регистрации, авторизации и управлением профилем. ChampionshipsController – контроллер для вывода и получения информации о турнирах. CoachingStaffController – контроллер для вывода и получения информации о тренерском штабе. MatchesController – контроллер для вывода и получения информации о матчах. PlayersController – контроллер для вывода и получения информации об игроках. TeamsController – контроллер для вывода и получения информации о команде. UsersController – контроллер для вывода и получения информации о пользователях.

На рисунке 3 представлена главная страница приложения.



Рисунок 3 - главная страница

Страница авторизации пользователей состоит из формы для аутентификации (логин и пароль), а так ссылку для регистрации.

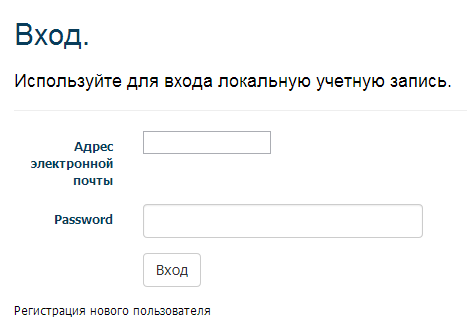


Рисунок 4 - форма аутентификации

Зайдя под пользователем с ролью «администратор», в меню приложения появляются дополнительная ссылка «Пользователи».



Рисунок 4 – меню Администратора

Зайдя под обычным пользователем, в меню приложения появляются меню для пользователя.



Рисунок 5 - меню Пользователя

Страница «Турниры», рисунке 6, позволяет просмотреть информацию о турнирах.



Рисунок 6 – страница Турниры

Страница «Матчи», рисунке 7, позволяет просмотреть информацию о сыгранных матчах.

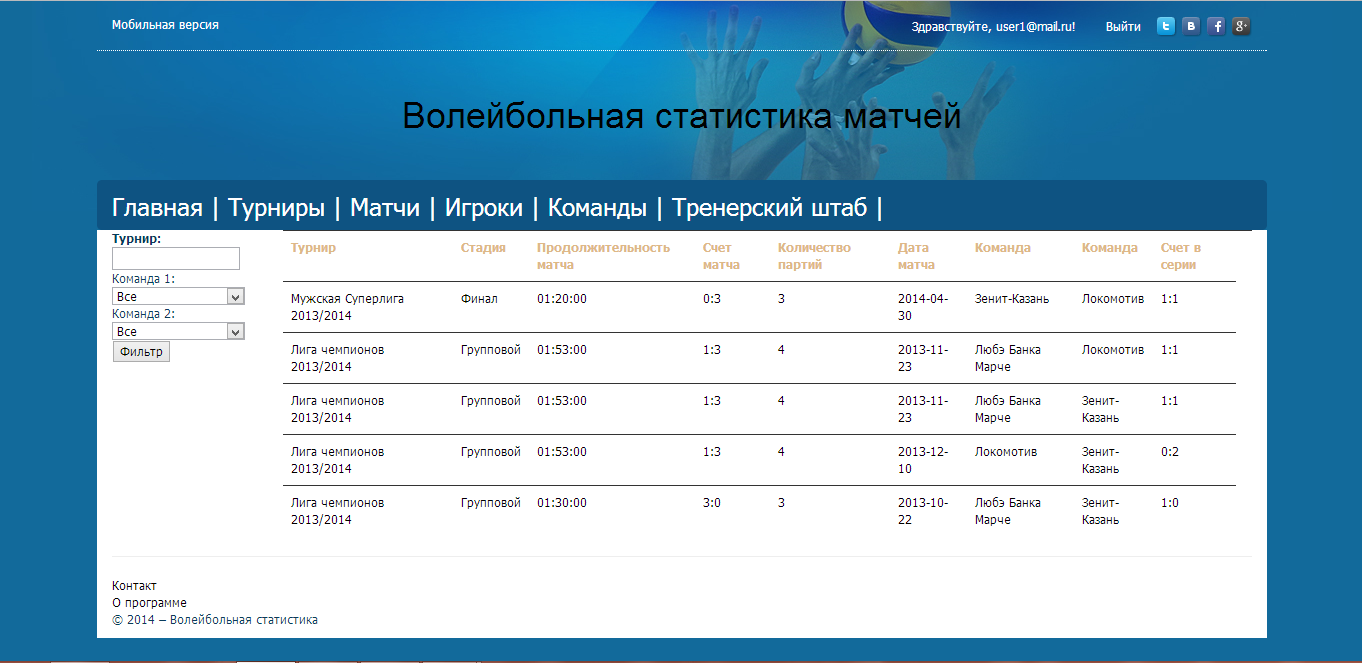


Рисунок 7 – страница Матчи

Страница «Игроки», рисунке 8, позволяет просмотреть информацию об игроках.



Рисунок 8 – страница Игроки

Страница «Тренерский штаб», рисунке 9, позволяет просмотреть информацию о тренерском штабе команды.



Рисунок 9 – страница Тренерский штаб

# **Заключение**

Целью проекта, было создание веб-приложения.

Данная задача была разделена на несколько этапов:

1. Аналитический этап – обзор возможных технологий для реализации, языков программирования.
2. Выбор решения для реализации.
3. Реализация веб-приложения.

В результате выполнения данной работы был проанализирован порядок организации и особенности данного процесса. Так же была спроектирована БД, необходимая для работы веб-приложения. Результаты работы отвечают поставленным задачам и служат достижению поставленной задачи.

В результате работы было создано веб-приложение, для просмотра статистики волейбольных матчей. Данное веб-приложение отвечает всем требованиям, указанным в задании. Так же в приложении добавлены такие функции, как восстановление пароля, возможность входа через другие учетные записи, такие как учетные записи facebook и google, поддержка двухфакторной аутентификации.

Исходный код приложения представлен на bitbucket.org[3].

# **Список используемой литературы**

1. Сайт о создании сайтов [электронный ресурс]. Режим доступа: свободный. http://htmlbook.ru (09.10.14).
2. Сайт о программировании, про создание сайтов и IT-технологии [электронный ресурс]. Режим доступа: свободный. http://metanit.com (09.10.14).
3. Ссылка на проект Режим доступа: свободный. [https://bitbucket.org/darkphoenix/mvc-project](http://msdn.microsoft.com/ru-RU/aa496123) (13.12.14).
4. Платформа .Net Framework [электронный ресурс]. Режим доступа: свободный. [http://msdn.microsoft.com](http://msdn.microsoft.com/ru-RU/aa496123) (13.09.14).
5. Learning ASP.NET [электронный ресурс]. Режим доступа: свободный. <http://www.asp.net> (09.09.14)
6. Visual Studio 2013 [электронный ресурс]. Режим доступа: свободный. [http://msdn.microsoft.com](http://msdn.microsoft.com/ru-RU/aa496123) (09.09.14).