+

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

###### ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ФГБО ВО НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«МОСКОВСКИЙ ЭНЕРГЕТИЧЕСКИЙ ИНСТИТУТ

(ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ)»

ИЖЕНЕРНО-ЭКОНОМИЧЕСКИЙ ИНСТИТУТ

Кафедра «Прикладная и бизнес-информатика»

**ОТЧЕТ ПО КУРСОВОМУ ПРОЕКТУ**

по дисциплине «Управление разработкой информационных систем»

Тема: «Разработка информационной системы футбольной секции»

Студент(ка) группы \_\_\_ИЭс-161-16\_\_\_\_Киш А.В.\_\_\_\_\_

(Ф.И.О.)

Руководитель к.т.н., Петров С.А. \_\_\_\_\_\_\_

(уч. степень, звание, Ф.И.О.)

|  |  |  |
| --- | --- | --- |
| Сдана на проверку |  |  |
| Возвращена на доработку |  |  |
| Допущена к защите |  |  |
| Оценка |  |  |
|  |

Москва 2018

Оглавление

[1. Описание предметной области 2](#_Toc503472895)

[2. Обоснование выбора объекта автоматизации 4](#_Toc503472896)

[2.1. Назначение, цели создания ИС. 4](#_Toc503472897)

[2.2. Общее описание 5](#_Toc503472898)

[2.3. Требования к системе. 5](#_Toc503472899)

[2.4. Требования к функциональным характеристикам. 6](#_Toc503472900)

[2.5. Функциональное назначение системы. 6](#_Toc503472901)

[3. Описание выбранных для реализации технологий 9](#_Toc503472902)

[4. Проектирование информационной системы 14](#_Toc503472903)

[5. Реализация информационной системы 18](#_Toc503472904)

[6. Обзор решения 24](#_Toc503472905)

[Заключение 29](#_Toc503472906)

[Литература 30](#_Toc503472907)

## 1. Описание предметной области

В качестве предметной области для информационной системы (ИС) была выбрана деятельность футбольной секции. Пользователь выбирает тип футбольного поля, который ему интересен, выбирает тренера и производит оплату.

Основные задачи информационных систем:

* ***Интерпретация данных.*** Под интерпретацией понимается процесс определения смысла данных, результаты которого должны быть согласованными и корректными. Обычно предусматривается многовариантный анализ данных.
* ***Диагностика.*** Под диагностикой понимается процесс соотношения объекта с некоторым классом объектов и/или обнаружение неисправности в некоторой системе. Неисправность - это отклонение от нормы. Такая трактовка позволяет с единых теоретических позиций рассматривать и неисправность оборудования в технических системах, и заболевания живых организмов, и всевозможные природные аномалии.
* ***Мониторинг.*** Основная задача мониторинга - непрерывная интерпретация данных в реальном времени и сигнализация о выходе тех или иных параметров за допустимые пределы.
* ***Проектирование.*** Проектирование состоит в подготовке спецификаций на создание «объектов» с заранее определёнными свойствами. Под спецификацией понимается весь набор необходимых документов - чертёж, пояснительная записка и т.д. Основные проблемы здесь - получение чёткого структурного описания знаний об объекте и проблема «следа».
* ***Прогнозирование.*** Прогнозирование позволяет предсказывать последствия некоторых событий или явлений на основании анализа имеющихся данных. Прогнозирующие системы логически выводят вероятные следствия из заданных ситуаций.
* ***Планирование.*** Под планированием понимается нахождение планов действий, относящихся к объектам, способным выполнять некоторые функции. В таких ЭС используются модели поведения реальных объектов с тем, чтобы логически вывести последствия планируемой деятельности.
* ***Обучение.*** Под обучением понимается использование компьютера для обучения какой-то дисциплине или предмету. Системы обучения диагностируют ошибки при изучении какой-либо дисциплины с помощью ЭВМ и подсказывают правильные решения.
* ***Управление.*** Под управлением понимается функция организованной системы, поддерживающая определенный режим деятельности. Такого рода ЭС осуществляют управление поведением сложных систем в соответствии с заданными спецификациями.
* ***Поддержка принятия решений.*** Поддержка принятия решения - это совокупность процедур, обеспечивающая лицо, принимающее решения, необходимой информацией и рекомендациями, облегчающие процесс принятия решения. Эти ЭС помогают специалистам выбрать и/или сформировать нужную альтернативу среди множества выборов при принятии ответственных решений.

Информационная система футбольной секции должна обеспечить:

* Регистрацию занятий, полей, тренеров
* Регистрацию персональных данных футболистов, включающую в себя фамилию, имя, отчество, телефон
* Регистрация оплаты занятий

Разрабатываемую систему следует считать прототипом более функциональной информационной системы.

## 2. Обоснование выбора объекта автоматизации

## 2.1. Назначение, цели создания ИС.

Назначение ИС:  
Информационная система предназначена для использования в футбольной секции. Эта информационная система должна сделать работу администратора секции удобной и эффективной.

Цели создания ИС:  
Целью является разработка программного обеспечения для автоматизации деятельности администратора футбольной секции. Она автоматизирует оформление занятий и регистрацию новых пользователей (фамилия, имя, отчество, телефон), ведет учет платежей за занятия.

Характеристика объектов автоматизации:  
В данном курсовом проекте в качестве исследуемой организации рассматривается футбольная секция, которая предоставляет занятия по футболу пользователям.

## 2.2. Общее описание

Информационная система представляет собой хранилище данных, которое обеспечивает сохранность, упорядоченность, поиск и полный доступ к сведениям о футбольной секции.

## 2.3. Требования к системе.

Основное требование к системе – дружественный интерфейс пользователя. Кроме того, следует учесть, что система ориентирована на конечного пользователя, который не должен знать структуру программного кода, внутренние представление данных, средства программирования, которые использовались при разработке системы.  
Кроме того, работа в приложении должна быть построена наиболее эффективно, то есть должны быть реализованы самые необходимые пользователю функции. Пользователь должен иметь доступ ко всем данным в БД, а также иметь возможность их добавления, корректировки, удаления. Ему также должна быть доступна такая функция, как поиск в БД.  
С информационной системой «Футбольная секция» должен работать квалифицированный сотрудник, который имеет опыт работы с программными средствами, используемыми в данной работе, и может модернизировать ее.

## 2.4. Требования к функциональным характеристикам.

Система должна обеспечить возможность выполнения перечисленных ниже свойств:

* **Гибкость** – настраивается на любую структуру, для широкого круга деятельности;
* **Безопасность** – ограничение общего доступа к заранее определенным ресурсам БД футбольной секции;
* **Целостность и доступность** – документы в базе футбольной секции должны быть представлены в общепринятых в организации форматах, защищенных от изменений;
* **Возможность поиска** – фильтрация по базе данных событий;
* **Возможность подключения независимых модулей** для импорта данных из внешних источников в текущую базу данных.

## 2.5. Функциональное назначение системы.

Программное изделие должно выполнять следующие функции:

* регистрация занятий футбольной секции;
* выбор пользователем занятий;
* поиск занятий в базе данных;
* проверка наличия занятий;
* регистрация пользователей.

Разработка модели использования. Диаграмма вариантов использования (диаграмма прецедентов).

Субъект (actor) — это некто или нечто (человек, машина и т.д.) взаимодействующее с системой. Субъект взаимодействует с прецедентом, ожидая получить некий полезный результат.

Типичным графическим изображением субъекта является «штриховой человечек». В общем случае субъект может быть показан в виде прямоугольного символа класса. Подобно обычному классу субъект может обладать атрибутами и операциями (связанными с событиями, сообщения о которых он отправляет и получает). На рисунке 1 показаны три субъекта: пользователь, администратор и футбольная секция.



Рис. 1 Субъекты системы «Футбольная секция»

Прецеденты удовлетворяют функциональные требования за счет предоставления субъекту полезного результата. При этом не имеет значения, в какой последовательности решает бизнес-аналитик свои задачи: сначала обозначает субъектов, а затем прецеденты, или наоборот. Прецедент наглядно показывает варианты использования системы. На рисунке 2 показаны прецеденты моделируемой системы.

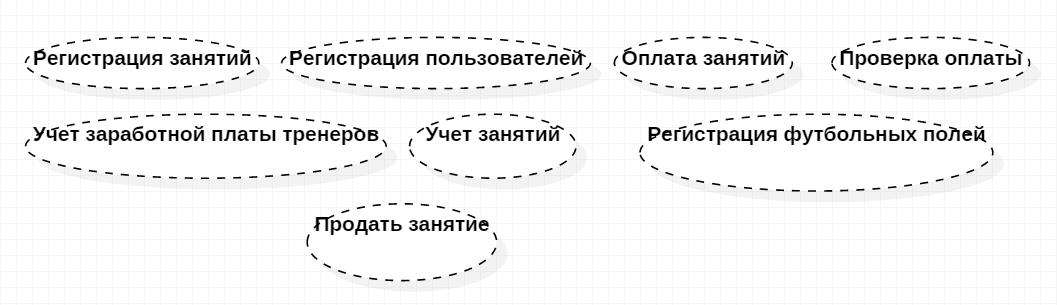


Рис. 2 Прецеденты системы «Футбольная секция»

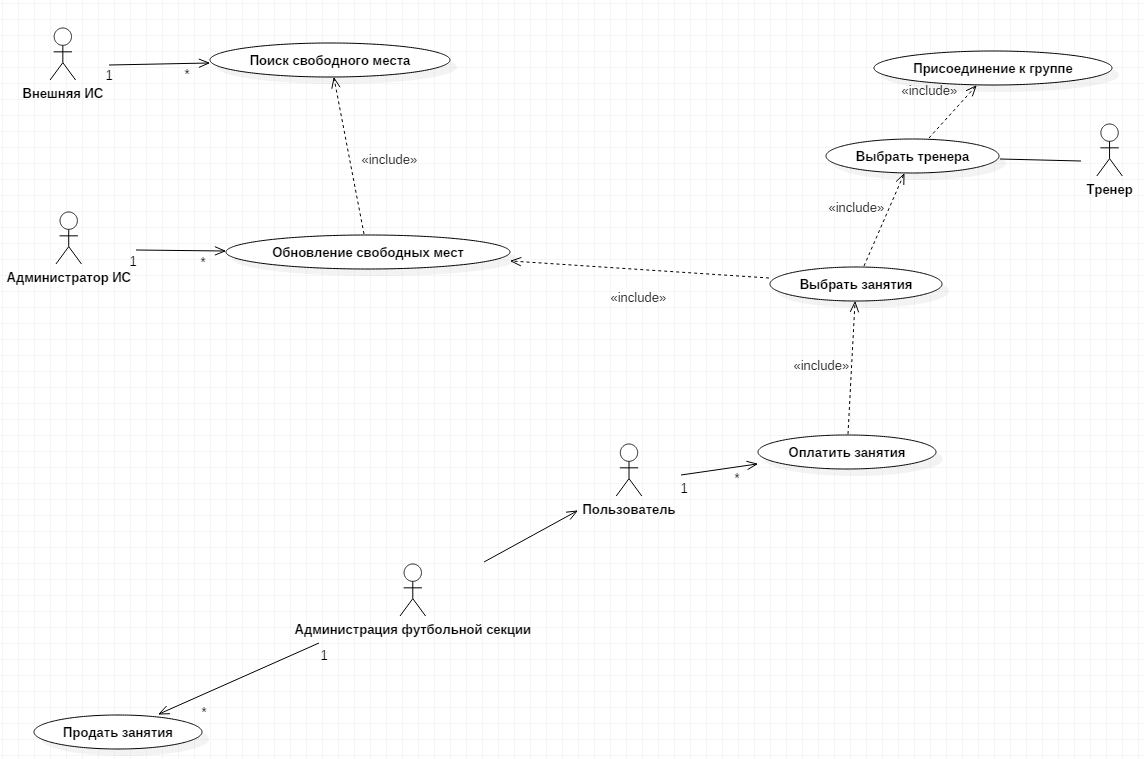
Диаграмма прецедентов – это наглядное графическое представление субъектов и прецедентов и их взаимодействий в системе вместе с любыми дополнительными определениями и спецификациями. Она представляет собой не просто некую схему, а является полностью документированной моделью предполагаемого поведения системы. Диаграмма прецедентов для моделируемой системы представлена на рисунке 3.

Рис. 3 Диаграмма прецедентов для системы «Футбольная секция»

## 3. Описание выбранных для реализации технологий

В качестве технологий для разработки системы выступают:

* .NET Framework;
* Language Integrated Query (LINQ).

Весь код программы написан в среде Microsoft Visual Studio 2015 на языке программирования C#.

Microsoft Visual Studio 2015 — это набор инструментов для создания программного обеспечения: от планирования до разработки пользовательского интерфейса, написания кода, тестирования, отладки, анализа качества кода и производительности, развертывания в средах клиентов и сбора данных телеметрии по использованию. Эти инструменты предназначены для максимально эффективной совместной работы; все они доступны в интегрированной среде разработки (IDE) Visual Studio.

Visual Studio можно использовать для создания различных типов приложений, от простых приложений для магазина и игр для мобильных клиентов до больших и сложных систем, обслуживающих предприятия и центры обработки данных. Можно создавать:

1. приложения и игры, которые выполняются не только на платформе Windows, но и на Android и iOS;
2. веб-сайты и веб-службы на основе ASP.NET, JQuery, AngularJS и других популярных платформ;
3. приложения для самых разных платформ и устройств, включая, но не ограничиваясь: Office, Sharepoint, Hololens, Kinect и "Интернета вещей";
4. игры и графические приложения для разных устройств Windows, включая Xbox, с поддержкой DirectX.

По умолчанию Visual Studio обеспечивает поддержку C#, C и C++, JavaScript, F# и Visual Basic. Visual Studio хорошо работает и интегрируется со сторонними приложениями, например Unity и Apache Cordova.

C# (произносится си шарп) – объектно-ориентированный язык программирования. Разработанный в 1998 – 2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как язык разработки приложений для платформы Microsoft .Net Framework и впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

C# относится к семье языков с С-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщенные типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Переняв многое от своих предшественников – языков C++, Pascal, Модула, Smalltalk и, в особенности, Java – C#, опираясь на практику использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, в C# в отличии от C++, не поддерживает множественное наследование классов (между тем допускается множественное наследование интерфейсов).

C# разрабатывался как язык программирования прикладного уровня для CLR и, как таковой, зависит, прежде всего, от возможностей самой CLR. Это касается, прежде всего, системы типов C#, которая отражает BCL. Присутствие или отсутствие тех или иных выразительных особенностей языка диктуется тем, может ли конкретная языковая особенность быть транслирована в соответствующие конструкции CLR. Так, с развитием CLR от версии 1.1 к 2.0 значительно обогатился и сам C#, как и всем другим .NET – ориентированным языкам, многие возможности, которых лишены «классические» языки программирования.

.NET Framework – программная платформа, выпущенная компанией Microsoft в 2002 году. Основной платформой является общеязыковая среда исполнения Common Language Runtime (CLR), которая подходит для разных языков программирования. Функциональные возможности CLR доступны в любых языках программирования, использующих эту среду.

Считается, что платформа .NET Framework явилась ответом компании Microsoft на набравшую к тому времени большую популярность платформу Java компании Sun Microsystems. Хотя, .NET является патентованной технологией корпорации Microsoft и официально рассчитана на работу под ОС семейств Microsoft Windows, существуют независимые проекты, позволяющие запускать программы .NET на некоторых других ОС.

Основной идеей при разработке .NET Framework являлось обеспечение свободы разработчиков за счет предоставления ему возможности создавать приложения различных типов, способные выполняться на различных типах устройств и в различных средах. Вторым принципом стала ориентация на системы, работающие под управлением семейства ОС Microsoft Windows.

Программа для .NET Framework, написанная на любом поддерживаемом языке программирования, сначала переводится компилятором в единый для .NET промежуточный байт-код CIL (Common Intermediate Language). Затем код либо исполняется виртуальной машиной 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 нее 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 заботиться о базовой безопасности, управлении памятью и системе исключений, избавляя разработчика от части работы.

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

Language Integrated Query (LINQ) — это технология, позволяющая выполнять запросы к источникам данных, используя языки программирования платформы .NET Framework и синтаксис, похожий на язык запросов SQL.

Впервые LINQ появился в Visual Studio 2008 и, постепенно развиваясь, поддерживал всё новые источники данных. На текущий момент, в зависимости от вида источника данных, выделяют несколько разновидностей LINQ:

* LINQ to Objects: запросы к коллекциям;
* LINQ to Entities: запросы к БД через Entity Framework;
* LINQ to Sql: запросы к MS SQL Server;
* LINQ to XML: запросы к XML файлам;
* LINQ to DataSet: запросы к объекту DataSet (не путать с DbSet, используемый EF);
* Parallel LINQ (PLINQ): выполнение параллельных запросов.

Используя, некоторые новые особенности языка, позволяет использовать SQL-подобный синтаксис непосредственно в коде программы, написанной, например, на языке C#:

* Анонимная типизация
* Методы расширения
* Лямбда-исчисления
* Дерево выражений
* Стандартные операторы языка запросов

Общие сведения о выражениях запросов:

* Выражения запросов можно использовать для запроса и преобразования данных, полученных из любого источника данных, поддерживающего LINQ. Например, один и тот же запрос может извлекать данные из базы данных SQL и на выходе создавать поток XML.
* Выражения запросов достаточно легко изучить, поскольку они во многом схожи с элементами языка C#.
* Все переменные в выражении запросов имеют строгую типизацию, хотя во многих случаях не требуется явным образом указывать тип, поскольку компилятор его вычисляет.
* Выполнение запроса не происходит до использования переменной запроса в операторе foreach.
* При компиляции выражения запросов преобразуется в вызовы методов стандартных операторов запроса согласно правилам, указанным в спецификациях языка C#. Любой запрос, которые может быть выражен с помощью синтаксиса запросов, также может быть выражен с помощью синтаксиса методов. Однако синтаксис запросов в большинстве случаев понятнее и лаконичнее.
* При создании запросов LINQ рекомендуется использовать синтаксис запросов везде, где это возможно, и синтаксис методов, если это необходимо. Между этими двумя видами не существует различий ни в семантике, ни в производительности. Выражения запросов просто более понятны, чем соответствующие им выражения, написанные с помощью синтаксиса методов.
* Для некоторых операций запросов, таких как Count<TSource> иди Max, нет соответствующих предложений в выражениях запросов, поэтому их необходимо создавать с помощью вызова методов. Синтаксис запросов и синтаксиса методов можно сочетать друг с другом различными способами.
* В зависимости от типа, к которому применяется запрос, выражения запросов можно компилировать в деревья выражений или в делегаты. Запросы IEnumerable<T> компилируются в делегаты. Запросы IQueryable и IQueryable<T> компилируются в деревья выражений.

## 4. Проектирование информационной системы

Проектирование информационных систем всегда начинается с определения цели проекта. Основная задача любого успешного проекта заключается в том, чтобы на момент запуска системы и в течение всего времени ее эксплуатации можно было обеспечить:

* требуемую функциональность системы и степень адаптации к изменяющимся условиям ее функционирования;
* требуемую пропускную способность системы;
* требуемое время реакции системы на запрос;
* безотказную работу системы в требуемом режиме, иными словами - готовность и доступность системы для обработки запросов пользователей;
* простоту эксплуатации и поддержки системы;
* необходимую безопасность.

Производительность является главным фактором, определяющим эффективность системы. Хорошее проектное решение служит основой высокопроизводительной системы.

Проектирование информационных систем охватывает три основные области:

1. проектирование объектов данных, которые будут реализованы в базе данных;
2. проектирование программ, экранных форм, отчетов, которые будут обеспечивать выполнение запросов к данным;
3. учет конкретной среды или технологии, а именно: топологии сети, конфигурации аппаратных средств, используемой архитектуры (файл-сервер или клиент-сервер), параллельной обработки, распределенной обработки данных и т.п.

В реальных условиях проектирование - это поиск способа, который удовлетворяет требованиям функциональности системы средствами имеющихся технологий с учетом заданных ограничений.

К любому проекту предъявляется ряд абсолютных требований, например максимальное время разработки проекта, максимальные денежные вложения в проект и т.д. Одна из сложностей проектирования состоит в том, что оно не является такой структурированной задачей, как анализ требований к проекту или реализация того или иного проектного решения.

Считается, что сложную систему невозможно описать в принципе. Это, в частности, касается систем управления предприятием. Одним из основных аргументов является изменение условий функционирования системы, например директивное изменение тех или иных потоков информации новым руководством. Еще один аргумент - объемы технического задания, которые для крупного проекта могут составлять сотни страниц, в то время как технический проект может содержать ошибки. Возникает вопрос: а может, лучше вообще не проводить обследования и не делать никакого технического проекта, а писать систему «с чистого листа» в надежде на то, что произойдет некое чудесное совпадение желания заказчика с тем, что написали программисты, а также на то, что все это будет стабильно работать?

Жизненный цикл программного обеспечения представляет собой модель его создания и использования. Модель отражает его различные состояния, начиная с момента возникновения необходимости в данном ПО и заканчивая моментом его полного выхода из употребления у всех пользователей.

Известны следующие модели жизненного цикла:

* Каскадная модель. Переход на следующий этап означает полное завершение работ на предыдущем этапе.
* Поэтапная модель с промежуточным контролем. Разработка ПО ведется итерациями с циклами обратной связи между этапами. Межэтапные корректировки позволяют уменьшить трудоемкость процесса разработки по сравнению с каскадной моделью; время жизни каждого из этапов растягивается на весь период разработки.
* Спиральная модель. Особое внимание уделяется начальным этапам разработки - выработке стратегии, анализу и проектированию, где реализуемость тех или иных технических решений проверяется и обосновывается посредством создания прототипов (макетирования). Каждый виток спирали предполагает создание некой версии продукта или какого-либо его компонента, при этом уточняются характеристики и цели проекта, определяется его качество и планируются работы следующего витка спирали.

Целью построения логической модели является получение графического представления логической структуры исследуемой предметной области.

Логическая модель предметной области иллюстрирует сущности, а также их взаимоотношения между собой.

Сущности описывают объекты, являющиеся предметом деятельности предметной области, и субъекты, осуществляющие деятельность в рамках предметной области. Свойства объектов и субъектов реального мира описываются с помощью атрибутов.

Взаимоотношения между сущностями иллюстрируются с помощью связей. Правила и ограничения взаимоотношений описываются с помощью свойств связей. Обычно связи определяют либо зависимости между сущностями, либо влияние одной сущности на другую.

Ниже представлена логическая модель информационной системы футбольной секции. На ней изображено пять сущностей.   
В сущности Kurs(занятия) имеет следующие атрибуты: id, Specification, Cost, KabinetId, сущность соединена связью один ко многим с Kurs(занятия).   
Сущность Uchet(учет) имеет id, описание и цену. Эта сущность соединена связью один ко многим с Kurs(занятия).   
Сущность Learner(футболист) имеет id, имя, отчество, фамилию и телефон. Она соединена связью один ко многим с сущностью Teacher(тренер).   
Сущность Teacher(тренер) имеет id, ФИО и заработную плату. Эта сущность соединена связью один ко многим с сущностью Learner(футболист).   
Сущность Kabinet(футбольное поле) имеет следующие атрибуты: id, Name. Она связана с сущностью Kurs(занятия) связью один ко многим.

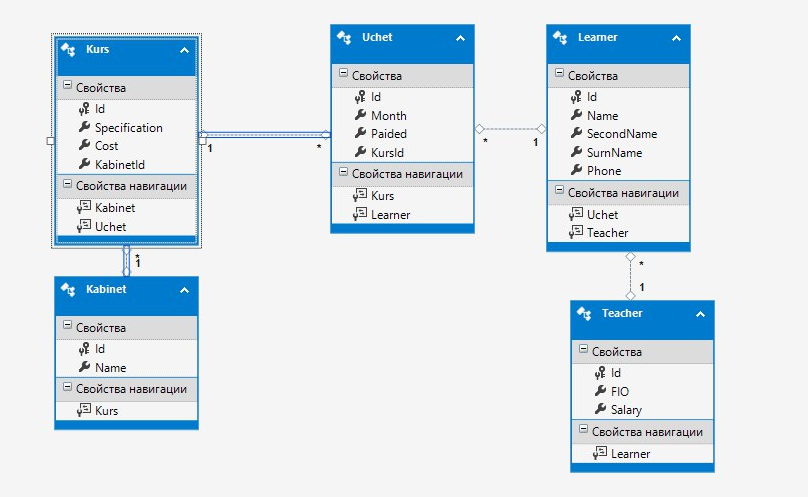


Рис. 4 Логическая модель футбольной секции.

## 5. Реализация информационной системы

Для взаимодействия пользователя с системой был разработан графический интерфейс. Перед нами не стоит задача освоения реализации систем с богатыми графическими возможностями, поэтому остановимся на варианте простого консольного меню.

Пусть меню будет поддерживать иерархическую вложенность разделов, чтобы не «пугать» пользователя отображением всех возможностей сразу, а предоставлять ему возможность постепенно углубляться и находить нужный функционал.

Примерный результат должен выглядеть следующим образом:

1 - Пункт меню 1

2 - Пункт меню 2

3 - Пункт меню 3

0 - Выход (на уровень вверх)

Для каждого из пунктов меню должно быть назначено действие для запуска, либо описаны вложенные разделы.

Наше меню будет иметь следующие разделы: тип футбольного поля, занятия, тренеры, футболисты, учет, отчеты. Ниже предоставлена форма, как будет оно выглядеть в программе.

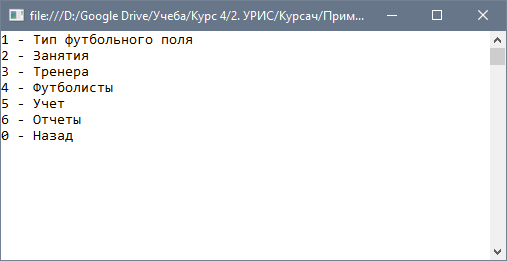


Рис. 5 Консольное меню футбольной секции.

Далее представлен код меню, где описывается каждый раздел и вложенными в них подразделами.

using System;

using Krugok.Report;

using Krugok.Repositories;

namespace Krugok

{

public static class MenuGenerator

{

public static HierarchicalMenuItem GenerateKabinetMenu(KrugokContainer db, string position)

{

KabinetRepository kabinetRepository = new KabinetRepository(db);

HierarchicalMenuItem kabinetHierarchicalMenuItem = new HierarchicalMenuItem("Тип футбольного поля", position);

kabinetHierarchicalMenuItem.MenuItems.Add(new ReadMenuItem("Список футбольных полей", "1", kabinetRepository));

kabinetHierarchicalMenuItem.MenuItems.Add(new CreateMenuItem("Создать футбольное поле", "2", kabinetRepository));

kabinetHierarchicalMenuItem.MenuItems.Add(new UpdateMenuItem("Обновить футбольное поле", "3", kabinetRepository));

kabinetHierarchicalMenuItem.MenuItems.Add(new DeleteMenuItem("Удалить футбольное поле", "4", kabinetRepository));

return kabinetHierarchicalMenuItem;

}

public static HierarchicalMenuItem GenerateTeacherMenu(KrugokContainer db, string position)

{

TeacherRepository teacherRepository = new TeacherRepository(db);

HierarchicalMenuItem teacherHierarchicalMenuItem = new HierarchicalMenuItem("Тренера", position);

teacherHierarchicalMenuItem.MenuItems.Add(new ReadMenuItem("Список тренеров", "1", teacherRepository));

teacherHierarchicalMenuItem.MenuItems.Add(new CreateMenuItem("Создать тренера", "2", teacherRepository));

teacherHierarchicalMenuItem.MenuItems.Add(new UpdateMenuItem("Обновить тренера", "3", teacherRepository));

teacherHierarchicalMenuItem.MenuItems.Add(new DeleteMenuItem("Удалить тренера", "4", teacherRepository));

return teacherHierarchicalMenuItem;

}

public static HierarchicalMenuItem GenerateKursMenu(KrugokContainer db, string position)

{

KursRepository kursRepository = new KursRepository(db);

HierarchicalMenuItem kursHierarchicalMenuItem = new HierarchicalMenuItem("Занятия", position);

kursHierarchicalMenuItem.MenuItems.Add(new ReadMenuItem("Список занятий", "1", kursRepository));

kursHierarchicalMenuItem.MenuItems.Add(new CreateMenuItem("Создать занятие", "2", kursRepository));

kursHierarchicalMenuItem.MenuItems.Add(new UpdateMenuItem("Обновить занятие", "3", kursRepository));

kursHierarchicalMenuItem.MenuItems.Add(new DeleteMenuItem("Удалить занятие", "4", kursRepository));

return kursHierarchicalMenuItem;

}

public static HierarchicalMenuItem GenerateLearnerMenu(KrugokContainer db, string position)

{

LearnerRepository learnerRepository = new LearnerRepository(db);

HierarchicalMenuItem learnerTypeHierarchicalMenuItem = new HierarchicalMenuItem("Футболисты", position);

learnerTypeHierarchicalMenuItem.MenuItems.Add(new ReadMenuItem("Список футболистов", "1", learnerRepository));

learnerTypeHierarchicalMenuItem.MenuItems.Add(new CreateMenuItem("Создать футболиста", "2", learnerRepository));

learnerTypeHierarchicalMenuItem.MenuItems.Add(new UpdateMenuItem("Обновить футболиста", "3", learnerRepository));

learnerTypeHierarchicalMenuItem.MenuItems.Add(new DeleteMenuItem("Удалить футболиста", "4", learnerRepository));

return learnerTypeHierarchicalMenuItem;

}

public static HierarchicalMenuItem GenerateUchetStatusMenu(KrugokContainer db, string position)

{

UchetRepository uchetRepository = new UchetRepository(db);

HierarchicalMenuItem uchetStatusHierarchicalMenuItem = new HierarchicalMenuItem("Учет", position);

uchetStatusHierarchicalMenuItem.MenuItems.Add(new ReadMenuItem("Список учета", "1", uchetRepository));

uchetStatusHierarchicalMenuItem.MenuItems.Add(new CreateMenuItem("Создать учет", "2", uchetRepository));

uchetStatusHierarchicalMenuItem.MenuItems.Add(new UpdateMenuItem("Обновить учет", "3", uchetRepository));

uchetStatusHierarchicalMenuItem.MenuItems.Add(new DeleteMenuItem("Удалить учет", "4", uchetRepository));

return uchetStatusHierarchicalMenuItem;

}

public static HierarchicalMenuItem GenerateReportMenu(KrugokContainer db, string position)

{

HierarchicalMenuItem reportHierarchicalMenuItem = new HierarchicalMenuItem("Отчеты", position);

reportHierarchicalMenuItem.MenuItems.Add(new ReportMenuItem("Список футболистов у тренера","1", new LearnerFOrTeacher(db)));

reportHierarchicalMenuItem.MenuItems.Add(new ReportMenuItem("Список занятий с полями", "2", new KursList(db)));

reportHierarchicalMenuItem.MenuItems.Add(new ReportMenuItem("Список футболистов на занятие", "3", new LearnerList(db)));

return reportHierarchicalMenuItem;

}

}

}

Выбрав в меню пункт «Тип футбольного поля», мы опустимся в этот раздел, в котором нам будет предложена одна из четырех команд.

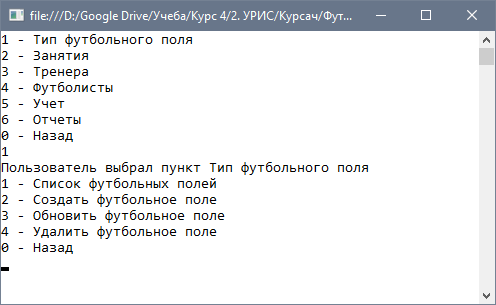


Рис. 6 Раздел с подразделами.

Исходный код данного класса, в котором имеются подразделы добавления, удаления, обновления и удаления представлен ниже.

using System;

using System.Linq;

namespace Krugok.Repositories

{

public class KabinetRepository : BaseRepository

{

public KabinetRepository(KrugokContainer db) : base(db)

{

}

public override void Create()

{

Kabinet kabinet = new Kabinet();

Console.WriteLine("Введите название футбольного поля");

kabinet.Name = Console.ReadLine();

db.KabinetSet.Add(kabinet);

db.SaveChanges();

}

public override void Read()

{

foreach (Kabinet kabinet in db.KabinetSet)

{

Console.WriteLine("{0} - {1}", kabinet.Id, kabinet.Name);

}

}

public override void Update()

{

Console.WriteLine("Введите ID футбольного поля");

int kabinetID = int.Parse(Console.ReadLine());

Kabinet kabinet = db.KabinetSet.SingleOrDefault(x => x.Id == kabinetID);

Console.WriteLine("Введите название футбольного поля");

kabinet.Name = Console.ReadLine();

db.SaveChanges();

}

public override void Delete()

{

Console.WriteLine("Введите ID футбольного поля");

int kabinetID = int.Parse(Console.ReadLine());

Kabinet kabinet = db.KabinetSet.SingleOrDefault(x => x.Id == kabinetID);

foreach (Kurs kurs in kabinet.Kurs)

{

db.UchetSet.RemoveRange(kurs.Uchet);

}

db.KursSet.RemoveRange(kabinet.Kurs);

db.KabinetSet.Remove(kabinet);

db.SaveChanges();

}

}

}

Всего имеется пять таких файлов: KabinetRepository, KursRepository, LearnerRepository, TeacherRepository, TeacherRepository, UchetRepository. Где аналогично прописаны те же функции. На Рис. 7. представлены все эти файлы в Обозреватели Решений.

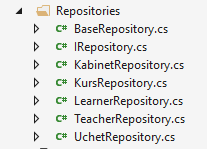


Рис. 7 Папка Repositories в обозревателе решений.

На Рис. 8 представлены папки с меню для каждого раздела, содержащие однотипные классы: создание, удаление, чтение и обновление.

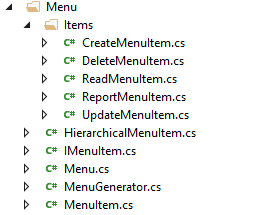


Рис. 8 Папка Menu в обозревателе решений.

## 6. Обзор решения

Ниже представлены экранные формы добавления записи, чтения всех записей, обновление записи и удаление ее, а так же все запросы:

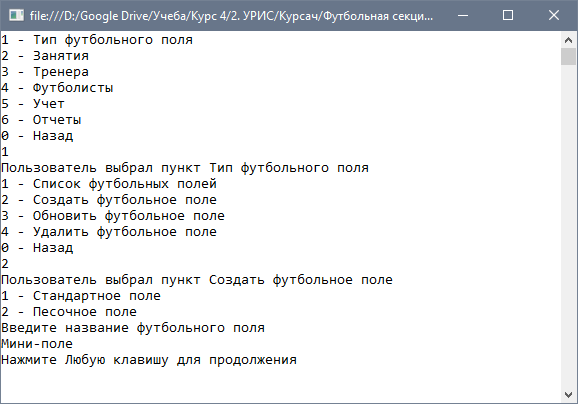


Рис. 9 Создание футбольного поля.

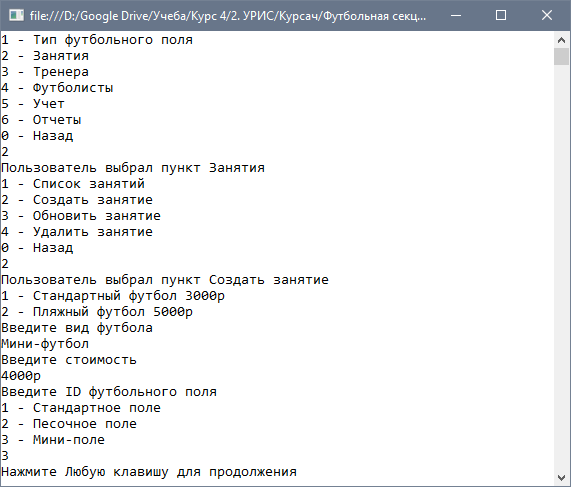


Рис. 10 Создание занятия.

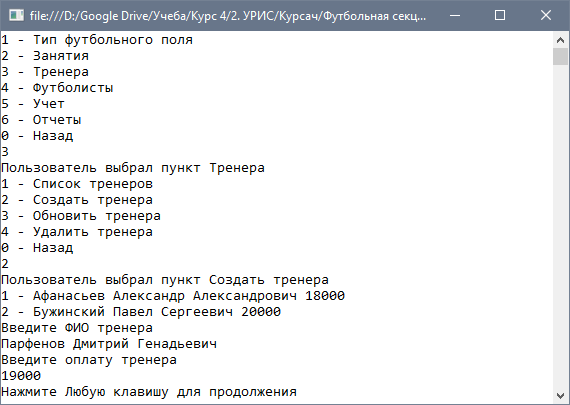


Рис. 11 Создание тренера.

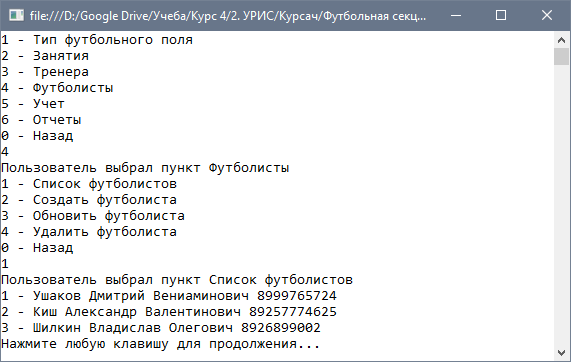


Рис. 12 Список футболистов.

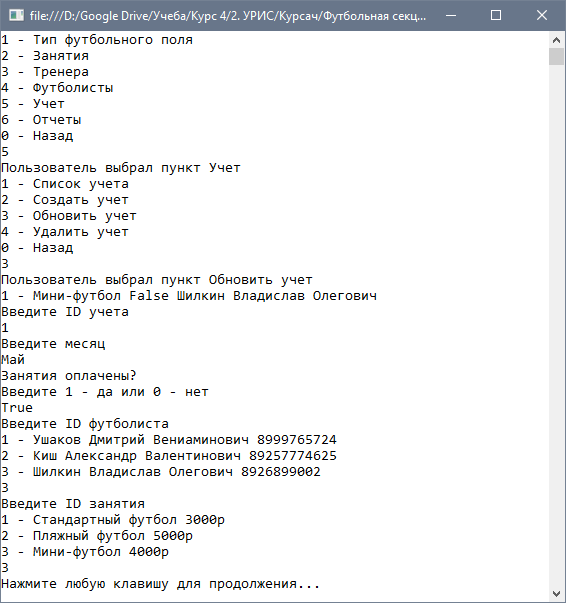


Рис. 13 Обновление учета.

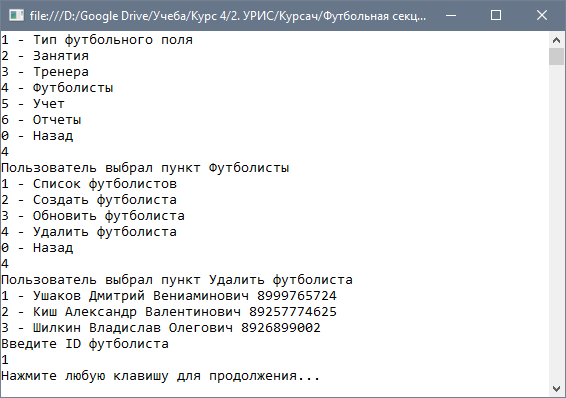


Рис. 14 Удаление футболиста.

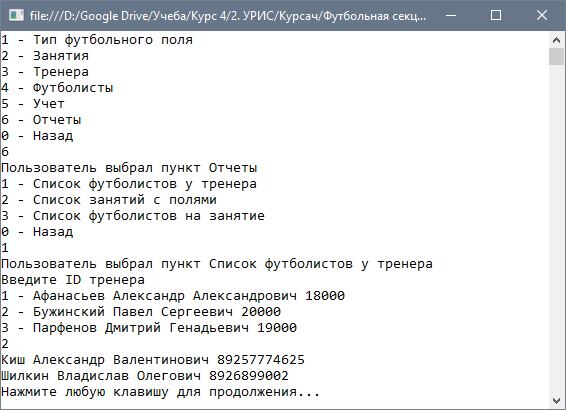


Рис. 15 Список футболистов у тренера.

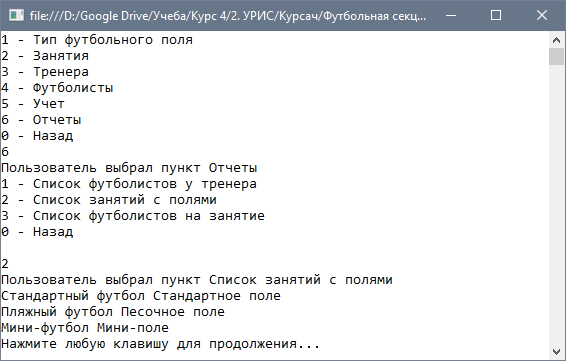


Рис. 16 Список занятий с полями.

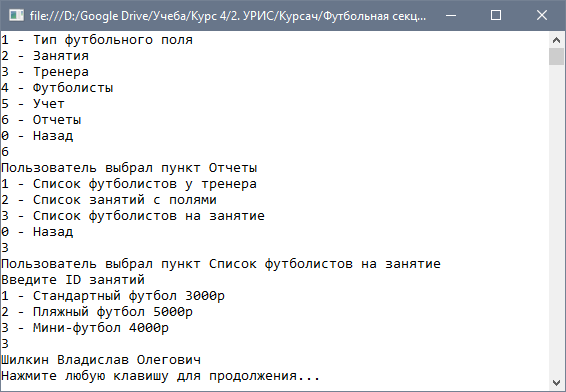


Рис. 17 Список футболистов на занятие.

## Заключение

В результате курсового проекта была спроектирована и реализована информационная система «Футбольная секция». Данная система удовлетворяет всем требованиям, предъявленным в задании, и реализует большинство необходимых сотрудникам секции функций. Приложение выполняет следующий перечень требований:

* представление данных в удобном для пользователя виде;
* добавление и удаление данных;
* фильтрация и сортировка данных.

В результате выполнения курсовой работы был сделан вывод, что сегодня внедрение информационных систем может способствовать:

* получению более рациональных вариантов решения управленческих задач за счет внедрения математических методов и интеллектуальных систем и т.д.
* освобождению работников от рутинной работы за счет ее автоматизации;
* обеспечению достоверности информации;
* замене бумажных носителей, данных на магнитные и оптические, что приводит к более рациональной организации переработки информации на компьютере и снижению объемов бумажных документов;
* уменьшению затрат на производство продуктов и услуг.

## Литература

1. [Фаулер М.](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D1%83%D0%BB%D0%B5%D1%80,_%D0%9C%D0%B0%D1%80%D1%82%D0%B8%D0%BD), Бек К., Брант Д., Робертс Д., Апдайк У. Рефакторинг: улучшение существующего кода = Refactoring: Improving the Design of Existing Code (2000). — Спб: Символ-Плюс, 2009. — 432 с. [ISBN 5-93286-045-6](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:%D0%98%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8_%D0%BA%D0%BD%D0%B8%D0%B3/5932860456).
2. Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя. 2-е изд.: Пер. с англ. Мухин Н. — М.: ДМК Пресс, 2006. — 496 с.: ил. ISBN 5-94074-334-X.
3. Стандарт ISO/IEC 2382:2015.
4. http://analyst.by/diagrams/logicheskaya-model-predmetnoy-oblasti
5. http://bibliofond.ru/download\_list.aspx?id=787590
6. http://obmendoc.ru/files/users/andrey/72/view/253102-253103
7. https://www.google.ru/url?sa=t&rct=j&q=&esrc=s&source=web&cd=4&ved=0ahUKEwjoiP-4lJbRAhVGWSwKHa1AApwQFggtMAM&url=http%3A%2F%2Fedulancer.ru%2Fupload.php%3Ffileid%3D6422&usg=AFQjCNH9N9V9Bitn3fch0UNovlXPBnhyww&cad=rjt
8. Microsoft Developer Network: https://msdn.microsoft.com/
9. Загрузка СУБД MS SQL Express: https://www.microsoft.com/en-us/cloud-platform/sql-server
10. LINQ. Карманный справочник: Пер. С англ. / Дж. Албахари, Б. Албахари. — СПб.: БХВ-Петербург, 2009. — 240 с.: ил. ISBN 978-5-9775-0317-4.