СОДЕРЖАНИЕ

ВВЕДЕНИЕ7

1 ОБЗОР ЛИТЕРАТУРЫ9

2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ17

2.1 Структура программной модели тотализатора игровых видов спорта с возможностью предсказания результата игры17

2.2 Обоснование выбора программных средств19

3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ22

3.1 Выбор алгоритмов предсказания и параметров команд 22

3.2 Классы блока нечетких предсказаний 23

3.3 Классы блока квалиметрического анализа 24

3.4 Классы блока анализа на основе нейронной сети 25

3.5 Основные классы блока предсказания 29

3.6 Классы блока работы с базами данных 30

3.7 Классы блока корректировки коэффициентов 36

3.8 Классы блока клиентского интерфейса 37

3.9 Классы блока управления 37

3.10 Структура данных 41

4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ45

4.1 Квалиметричесикй анализ 45

4.2 Нечеткие предсказания 46

4.3 Предсказание на основе нейронной сети РБФ 49

4.4 Вычисление итоговых вероятностей 52

5 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ54

5.1 Модульное тестирование54

5.2 Обработка исключительных ситуаций56

5.3 Регрессионное тестирование58

6 РУКОВОЛСТВО ПОЛЬЗОВАТЕЛЯ59

6.1 Требования к аппаратному и программному обеспечению59

6.2 Установка и настройка сервера приложений IIS59

6.3 Установка SQL Server 200861

6.4 Графический интерфейс пользователя61

7 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ПРОЕКТА65

7.1 Описание проекта 65

7.2 асчёт сметы затрат и цены ПО65

7.3 Расчёт экономического эффекта от применения программного средства у пользователя (заказчика)75

7.2 Выводы 80

ЗАКЛЮЧЕНИЕ81

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ82

ПРИЛОЖЕНИЕ А Листинг кода класса FuzzyPredicto83

ПРИЛОЖЕНИЕ Б Листинг кода класса GeneralPredictor 85

ПРИЛОЖЕНИЕ В Листинг кода класса QualimetricPredictor 86

ПРИЛОЖЕНИЕ Г Листинг кода класса TeamPowerCalculator 87

ПРИЛОЖЕНИЕ Д Листинг кода класса Serializer 89

ПРИЛОЖЕНИЕ Е Листинг кода класса NeuralNetworkPredictor 90

ПРИЛОЖЕНИЕ Ж Листинг кода класса

RadialBasisFunctionNetwork 92

ПРИЛОЖЕНИЕ И Листинг кода класса CoefficientsCalculator 99

ПРИЛОЖЕНИЕ К Листинг кода класса MatchInfo 100

ПРИЛОЖЕНИЕ Л Спецификация проекта 101

ПРИЛОЖЕНИЕ М Ведомость документов 102

ВВЕДЕНИЕ

Азартные игры – предмет интереса не только множества обычных людей, но многих ученых. Они являются неотъемлемой частью современного мира, а алгоритмы, используемые в азартных играх, представляют большой интерес для математики и теории вероятностей. В этой сфере вращаются огромные суммы денег и задействовано большое количество людей. Данная индустрия предлагает свои услуги всем слоям общества и позволяет каждому выбрать игру на свой вкус, в зависимости от предпочтений и возможностей. Одной из таких игр и является тотализатор.

Тотализатор позволяет сделать ставку практически на любую сферу деятельности человека, начиная от спорта и заканчивая полетами в космос. Но, все же, наибольшей популярностью в настоящее время пользуются ставки на спорт. С одной стороны, это способ болельщиков выразить свою поддержку команде или отдельному спортсмену. С другой же, тотализатор позволяет делающим ставки при правильном подходе зарабатывать до 15% от величины оборота денежных средств, вложенных в него. Прибыль владельца тотализатора без использования определенной стратегии не всегда стабильна. Если рассматривать достаточно длинный промежуток времени, то величина прибыли будет одинаковой, однако в данный конкретный момент времени убытки могут превышать прибыль. Но, варьируя коэффициенты определенным образом, прибыль можно получать всегда.

До повсеместного распространения компьютеров и интернета, тотализатор существовал только в реальном мире. Теперь же доля виртуального тотализатора едва ли не больше доли реального. С экономической точки зрения виртуальный тотализатор более выгоден для его владельца, так как не требует затрат на аренду помещения, зарплату сотрудников и прочее. Любой человек может ввести запрос в систему поиска и получить тысячи ссылок.

В любом тотализаторе предоставляются коэффициенты на тот или иной исход события. Поставивший на произошедшее событие, получает сумму, равную своей ставке, умноженной на коэффициент. Но сумма коэффициентов в процентах будет всегда больше 100. Прибылью владельца тотализатора и будет это превышение. Его задача, а в настоящее время задача программы, состоит в том, чтобы расставить коэффициенты максимально выгодно для себя, не сильно нарушая реальное их соотношение. Задача игрока – определить по выставленным коэффициентам реальные, основываясь на совем собственном опыте и интуиции.

Исходя из перечисленного выше, все современные тотализаторы нацелены на вычисление коэффициентов, выгодных их владельцам. Для этого необходимо рассчитать реальные коэффициенты, которые будут недоступны пользователям. Алгоритмы расчета обладают различной точностью и сложностью, что так же влияет на прибыль. Для работы этих алгоритмов необходима база данных с результатами прошлых событий подобного типа и данных об участниках события. Их анализ и выдает коэффициенты. Таким образом, задача данного проекта – создание онлайн-тотализатора с собственным алгоритмом вычисления коэффициентов, доступных некоторым пользователям.

После выполнения дипломного проекта в наличии будет по сути два отдельных продукта:

– программа, предсказывающая результаты матчей;

– интрернет-сайт тотализатора.

**1** ОБЗОР ЛИТЕРАТУРЫ

Тотализатор – это один из видов заключаемых пари [1]. Цели тотализатора и игры на тотализаторе различны: для его владельца это способ получить прибыль за предоставление услуг, а для игрока – возможность удовлетворить потребность в азартных играх не выходя из дома и заработать определенную сумму денег на победе любимой команды. В данный момент времени тотализаторы делятся на две большие категории: реальные и онлайн. Однако, онлайн тотализаторы набирают все большую популярность и в данном дипломном проекте в качестве аналогов рассматривались именно они. К преимуществам онлайн тотализатором можно отнести такие их особенности, как:

– доступность;

– распространенность;

– дешевизна;

– наглядность;

– прозрачность.

Существуют и определенные недостатки, основным из которых является меньшая надежность.

Само понятие тотализатора предполагает совершение ставок на результат определенных событий [2]. Для этого необходимо присвоить каждому из возможных результатов определенный коэффициент. Для владельца тотализатора точность определения реальных коэффициентов играет не столь значительную роль, так как его прибыль в любом случае будет заложена в откорректированные коэффициенты. К реальным коэффициентам добавляется по несколько процентов так, что бы их сумма была равна 115-120%. После этого коэффициенты могут корректироваться неограниченное количество раз с учетом текущих ставок игроков. Таким образом, при правильной установке коэффициентов, владелец тотализатора получит прибыль вне зависимости от результата игры. Однако, неправильно определяя вероятность того или иного результата и выставляя неправильные коэффициенты, владелец тотализатора потеряет аудиторию и перестанет получать прибыль. С этой точки зрения точность алгоритма предсказания результата события важна и для него.

Для человека, делающего ставки, как раз наоборот важна возможность узнать наиболее точное и нескорректированное в пользу кого-либо предсказание результата. От этого напрямую зависит, выиграет он или проиграет поставленную сумму. Опытные игроки умеют выделять из предоставляемых тотализатором коэффициентов приближенные к действительности величины, однако это является чисто субъективным мнением и зависит от опыта и умений человека. В связи с этим уже долгое время разрабатываются различные системы, позволяющие на основе определенных входных параметров предсказать результат события с различной степенью точности. Владельцы тотализаторов используют данные системы для вычисления первичных коэффициентов, которые потом корректируют в свою пользу. Игроки же могут получить на выходе реальные результаты, увеличивающие их шансы на выигрыш.

Наиболее популярными в Республике Беларусь на данный момент являются 3 онлайн тотализатора: Parimatch, Maxline, Maraphonbet. Так как информация об используемом алгоритме предсказания результатов и внутренних базах данных не выкладывается в публичный доступ, то с точки зрения пользователя зарубежные аналоги отличаются разве что масштабом и интерфейсом.

Сайт Parimatch [3] предоставляет большой выбор видов спорта и турниров. Для произведения ставки на данном ресурсе пользователю необходимо авторизироваться и выбрать интересующее событие. При желании пользователь может просмотреть историю команды, на которую ставит, и самостоятельно сделать вывод о результатах игры. База данных из данного сайта была частично использована в данном дипломном проекте. Владельцы данного сайта ввели ограничение на размер ставки в зависимости от размера коэффициента – чем он выше, тем максимальная ставка меньше. Интерфейс понятен пользователю, однако текстовое пояснение к элементам интерфейса во многих случаях отсутствует. Фрагмент таблицы доступных матчей с коэффициентами приведен на рисунке 1.1.

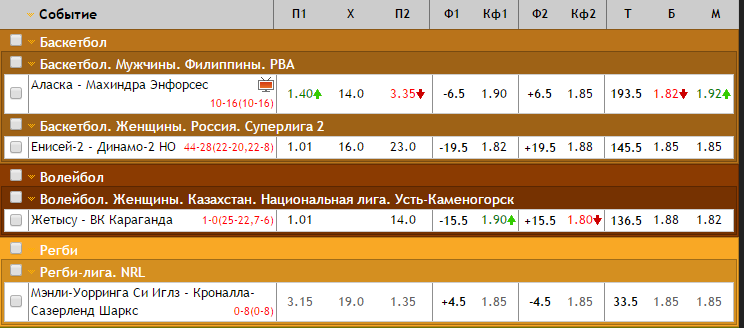


Рисунок 1.1 – Фрагмент таблицы доступных матчей Parimatch

Сайт Maxline [4] предоставляет такой же функционал, однако количество доступных видов спорта заметно меньше, чем на сайте Parimatch. Так же на этом сайте нет возможности просмотреть результаты предыдущих игр для конкретной команды, только результаты всех предыдущих событий. Таким образом, при желании самостоятельно оценить исход события по предыдущим играм, делающему ставку пользователю придется обращаться к сторонним источникам, что не особенно удобно. К преимуществам данного ресурса можно отнести хорошо разработанный интерфейс, практически для всех элементов присутствует пояснение. Он понятен даже новичку в сфере ставок. Фрагмент таблицы доступных матчей с коэффициентами приведен на рисунке 1.2.



Рисунок 1.2 – Фрагмент таблицы доступных матчей Maxline

Сайт Marathonbet [5] так же мало чем отличается от рассмотренных выше аналогов, однако предоставляет некоторые экзотические виды спорта, такие как собачьи бега. История предыдущих матчей на данном портале портале доступна и является наиблеее удобной из рассмотренных. Интерфест обладает теми же недостатками, что и интерфейс Parimatch, однако с визуальной точки зрения занимает лидирующее положение. Фрагмент таблицы доступных матчей с коэффициентами приведен на рисунке 1.3.

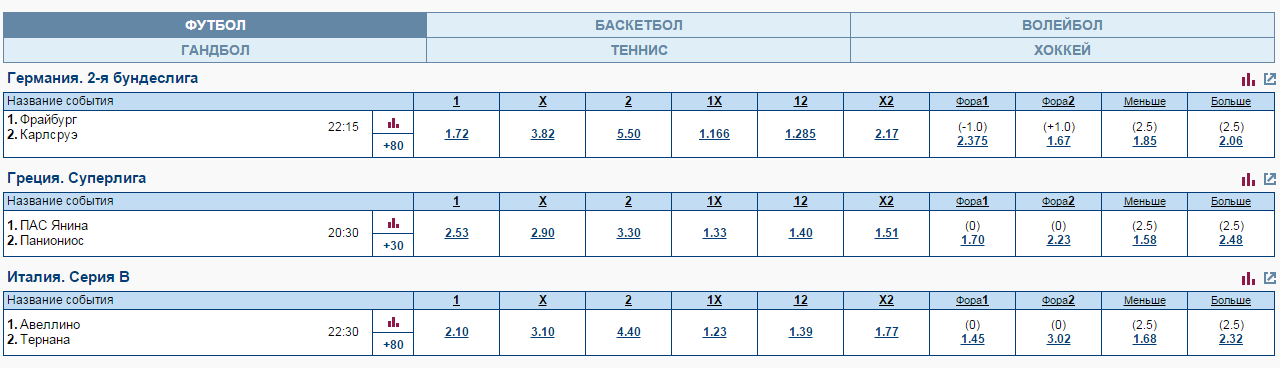


Рисунок 1.3 – Фрагмент таблицы доступных матчей Maxline

Однако, невозможно оценить сложность и точность алгоритма предсказания результатов событий рассмотренных тотализаторов, так как данная информация является конфиденциальной и не находится в открытом доступе. Для оценки результатов пользователями часто используются онлайн предсказатели, такие как Sgranks и предсказатель от Bi-Grouplabs. Данные предсказатели строятся на различных технологиях.

Продукт от Bi-Grouplabs [6] использует для анализа данных подход на основе технологий Data Mining:

– логистическая регрессия;

– деревья решений;

– карты Кохонена;

– нейронная сеть типа персептрон.

Данные технологии позволяют с достаточной точностью предсказать результат, однако являются сложными в реализации и сильно зависимы от базы данных для анализа. Также данный продукт предсказывает только два результата для футбола, тогда как на самом деле их три. Формат представления предсказания так же не очень удобен. Как итог, этот предсказатель показывает высокую точность, но не очень удобен для пользователя. Фрагмент таблицы предсказанных результатов приведен на рисунке 1.4.

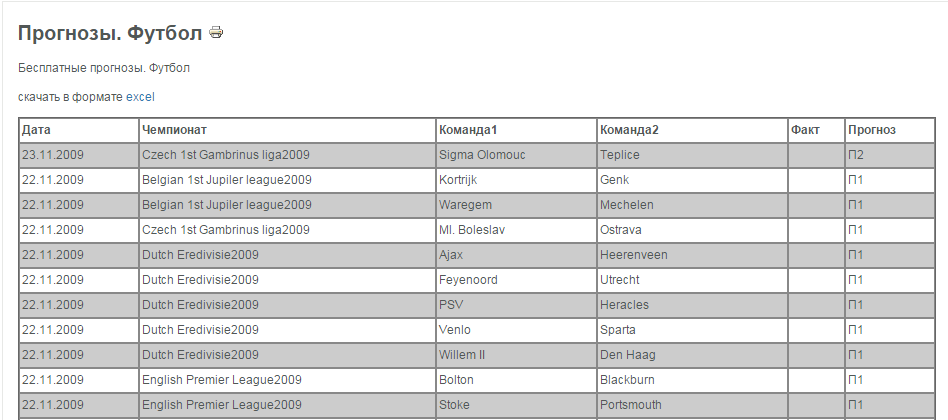


Рисунок 1.4 – Фрагмент таблицы предсказанных результатов от Bi-Grouplabs

Сайт Sgranks [7] предосатавляет схожий функционал, однако его алгоритм предсказания использует только нейронные сети. Данный продукт имеет большую базу данных, чем продукт от Bi-Grouplabs, что при использовании нейронных сетей играет важную роль. Интерфейс так же выгодно отличается от рассмотренного выше. Приводится анализ выданного результата с оценкой его достоверности. Фрагмент таблицы предсказанных результатов приведен на рисунке 1.5.

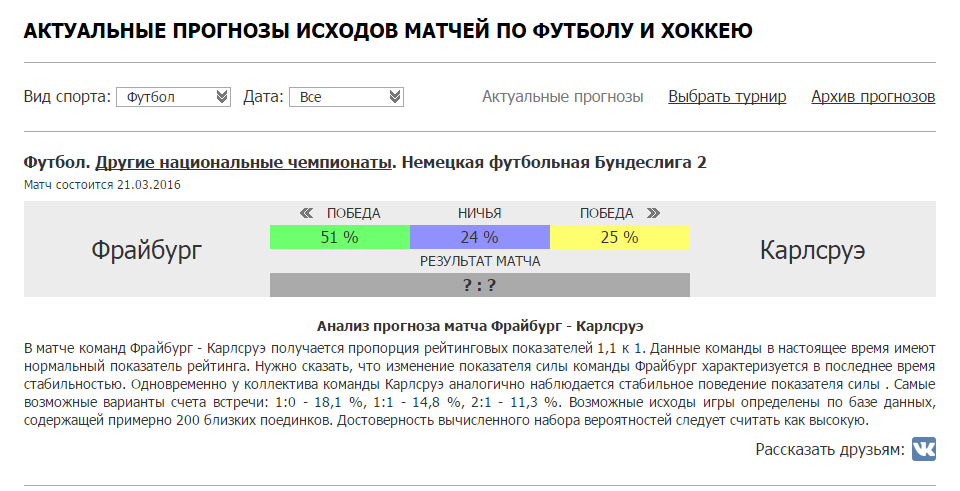


Рисунок 1.5 – Фрагмент таблицы предсказанных результатов Sgranks

Основываясь на рассмотренных аналогах и прочитанной литературе, в данном дипломном проекте было решено использовать сразу несколько технологий предсказания результата события, что позволит минимизировать недостатки каждого из них. А именно:

– квалиметрический анализ;

– нечеткие предсказания;

– нейронная сеть.

Каждый из этих методов имеет свои недостатки и достоинста и будет выдавать свой результат. Анализируя все три результата можно получить более достоверные коэффициенты, чем при использовании только одного метода.

Метод квалиметрического анализа [8] является наиболее простым из используемых. Для данного метода необходимо выбрать критерии оценки силы команд, задать их веса и привести к численному виду. Наиболее часто в качетсве критериев оценки выбираются следующие величины:

– место команды в турнирной таблице;

– количество травмированных игроков;

– сумма забитых мячей;

– сумма пропущенных мячей;

– количество травмированных игроков;

– результаты нескольких последних встреч сопоставляемых команд;

– количество дней отдыха до матча.

Далее на основе данных критериев вычисляется так называемая «сила» каждой команды и по отношению «сил» делается вывод об исходе матча. Недостатком данного метода является сильная зависимость от выбора весов критериев оценки.

Метод нечетких предсказаний [9] так же требует выбора критериев оценки команд, однако сам анализ строится на лексических конструкциях «если – то». Для большей точности данного метода будет предсказываться не три исхода (выигрышь – ничья – проигрышь), а пять (крупный выигрышь – выигрышь – ничья – проигрышь – крупный проигрышь). Недостатком данного метода является нестабильность результатов, то есть для одного турнира он может показывать высокую точность, а для другого ошибаться в 80% случаев.

Метод анализа на основе нейронных сетей [10] является самым мощным из рассматриваемых. Он позволяет проводить предсказание сколь угодно точно, однако для этого требуется расширять базу данных и величину самой нейронной сети в неограниченных размерах. Это и является главным недостатком данного алгоритма: сложность реализации и потребность в большой базе данных. В основе данного метода лежит нейронная сеть, веса нейронов которой устанавливаются во время обучения. В общем случае задача предсказания результата события с помощью нейронной сети сводится к задаче кластеризации. При этом каждый кластер должен содержать группы примеров с близкими по значению характеристиками объекта исследования, в нашем случае – статистическими показателями команд, участвующих в матче, прогноз на исход которого предстоит сделать. В самом простом случае необходимо 3 кластера (выигрышь – ничья – проигрышь), однако на практике их можно создать сколь угодно много. Все зависит от размера статистических данных. На рисунке 1.6 представлен пример такой нейронной сети.



Рисунок 1.6 – Схема нейронной сети

По результатам всех трех методов делается вывод об исходе события с той или иной долей достоверности. Данный результат доступен привелигерованным пользователям. В дальнейшем коэффициенты корректируются в пользу владельца тотализатора и становятся доступными всем пользователям.

Так как данный дипломный проект не только предсказывает результаты игр, но и реализует программную модель тотализатора, то необходимо использование клиент-серверных технологий. Всем необходимым условиям удовлетворяет ASP.NET MVC Framework [11]. Платформа ASP.NET MVC является легковесной платформой отображения с широкими возможностями тестирования и, подобно приложениям на основе веб-форм, интегрирована с существующими функциями ASP.NET, например с главными страницами и проверкой подлинности на основе членства. Пользовательский интерфейс будет изолирован от логики работы серверной части. На рисунке 1.7 приведена схема, иллюстрирующая данный шаблон.



Рисунок 1.7 – Шаблон проектирования MVC

Для реализации серверной части, в частности обработки ставок и рассчета коэффициентов на игры, был выбран язык программирования С#. Он позволяет как проводить вычисления, необходимые для предсказания, так и создавать структуру приложения, удобную для онлайн-тотализатора.

Клиенткая часть приложения реализована при помощи связки технологий Angularjs – HTML – CSS. Данные технологии позволяют постоить пользовательский интерфейс любой структуры и сложности, используя как уже существующие решения, так и собственные. Главным в данной связке является фреймворк Angularjs [12], который предназначен для расширения браузерных приложений на основе шаблона MVC, а также упрощения их разработки и тестирования. Взаимодействие серверной части на языке C# и клиентской части на Angularjs позволяет сделать работу приложения максимально быстрой, а интерфейс наиболее удобным для пользователя.

Веб-приложение написано как Single Page Application, то есть использует единственный HTML-документ как оболочку для всех веб-страниц и организует взаимодействие с пользователем через динамически подгружаемые HTML, CSS и JavaScript.

1. СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ
   1. Структура программной модели тотализатора игровых видов спорта с возможностью предсказания результата игры

Приложение построено с использованием клиент-серверной архитектуры и представляет собой одностраничное приложение (single-page application – SPA), реализующее паттерн MVC.

Структурная схема программной модели тотализатора игровых видов спорта с возможностью предсказания результата игры представлена на чертеже ГУИР.400201.075.С1 и состоит из следующих компонентов:

* блок управления;
* блок работы с базами данных;
* блок клиентского интерфейса;
* блок квалиметрического анализа;
* блок нечетких предсказаний;
* блок анализа на основе нейроной сети;
* блок предсказания;
* блок корректировки коэффициентов;
* база данных пользователей;
* база данных игр.

Далее каждый компонент рассматривается более подробно.

* + 1. Блок управления

Блок управления является одним из основных в приложении, так как все решения о действиях программы принимаются в нем. Из блока клиентского интерфейса в блок управления поступают данные о действиях пользователя, в соответствии с которыми либо отправляется запрос в блок работы с базами данных на корректировку или получение имеющихся данных, либо запускается процесс предсказания результата события в блоке предсказания. После получения предсказания блок управления передает данные блоку корректировки коэффициентов для вычисления видимых всем пользователям коэффициентов.

* + 1. Блок работы с базами данных

Блок работы с базами данных предназначен для управления всеми имеющимися в приложении базами данных, выборки и привединя к необходимому формату данных от них. Все процессы в нем начинаются по команде от блока управления, результаты этих процессов так же передаются в него.

* + 1. Блок клиентского интерфейса

Блок клиентского интерфейса представляет собой непосредственно сам интерфейс приложения. В программе он взаимодействует только с блоком управления, от которого получает все необходимые данные для отображения пользователю , и куда отправляет всю информацию о действиях пользователя. Интерфейс представляет собой одностраничное приложение, поэтому никаких задержек такой подход не создает.

* + 1. Блок квалиметрического анализа

Блок квалиметрического анализа взаимодействует только с блоком предсказания. От него он получает данные, необходимые для анализа, и ему же отдает результат своей работы. Предсказание результата происходит квалиметрическим методом, данные для которого поставляются из базы данных игр через несколько других блоков. Далее полученные данные проходят через процедуру оценки, результатом которой является так называемая «сила» обеих команд. Сравнивая эти «силы», блок возвращает блоку предсказания три коэффициенты на игру.

* + 1. Блок нечетких предсказаний

Блок нечетких предсказаний взаимодействует только с блоком предсказания. От него он получает данные, необходимые для анализа, и ему же отдает результат своей работы. На основе этих данных строится таблица нечетких аргументов, которая далее анализируется с помощью лексических выражений «если -- то». Результатом данного анализа являются выбор одного события. Для поддержания консистентности возвращаемых блоками анализа данных. Оставшиеся коэффициенты приравниваются к нулю.

* + 1. Блок анализа на основе нейронной сети

Блок анализа на основе нейронной сети взаимодействует только с блоком предсказания. От него он получает данные, необходимые для анализа, и ему же отдает результат своей работы. До выставления программы пользователю в данном блоке производится обучение нейронной сети на основе статистических данных, в процессе которого формируется структура сети и задаются веса составляющих ее нейронов. Задача данного блока обобщенно сводится к задаче кластеризации. Варьируя число кластеров можно улучшать результат предсказания. Собственно результатом работы блока будет являться отнесение результата события к некоторым из кластеров с определенной долей вероятности.

* + 1. Блок предсказания

Блок предсказания отвечает за обработку результатов от блоков квалиметрического анализа, нечетких предсказаний и анализа на основе нейронной сети. Данный блок получает данные от них в виде трех коэффициентов от каждого, и далее он производит их компоновку. На выходе получается итоговый результат предсказания, который отдается в блок управления для дальнейших действий с ним. Основой для предсказания являются данные от нейронной сети, достоверность которых в большинстве случаев выше достоверности результатов оставшихся блоков. Данные этих блоков используются для корректировки результата блока нейронной сети в ту или иную сторону. Каждый из трех упомянутых выше блоков получает данные для анализа из блока предсказания. Блок предсказания отправляет запрос на эти данные в блок управления, а после передает их блокам анализа.

* + 1. Блок корректировки коэффициентов

Блок корректировки коэффициентов получает от блока управления данные о предсказании результата события, вычисленные в блоке предсказания. Далее в эти коэффициенты закладывается прибыль владельца тотализатора в размере 15-20%. Так же данный блок предоставляет возможность корректировать коэффициенты по мере поступления ставок от пользователей. Чем больше человек поставило на такой исход события, тем меньше становится коэффициент для последующих игроков. Скорректированные коэффициенты возвращаются блоку управления.

* + 1. База данных пользователей

База данных пользователей хранит в себе всю информацию о зарегистрированных приложении пользователях и их правах. Для каждого пользователя сохраняется информаци о сделанных им ставках и полученных выплатах.

* + 1. База данных игр

База данных игр хранит в себе всю накопленную приложением статистическую информацию о прошедших играх, структурированную по видам спорта, а внутри них по турнирам. Здесь же хранится информация о предсказанных результатах событий с информацией о ее соответствии действительности.

* 1. Обоснование выбора программных средств

Клиент-серверные [9] приложения обычно принято называть трехуровневыми, так как они расположены на трех различных системах: клиентский компьютер, сервер ASP.Net и сервер базы данных или обычный сервер. Трехуровневые приложения, работающие данным способом, расширяют стандартную архитектуру клиент-сервер, добавляя многопоточный сервер приложений между клиентской частью и сервером базы данных. Эта модель решает, в основном, задачу развёртывания приложения. С использованием клиент-серверной модели созданы многие приложения для работы с базами данных, электронной почтой и для доступа к веб-ресурсам. Основные принципы данного архитектурного стиля:

* клиент инициирует один или несколько запросов, ожидает ответа на них, а затем обрабатывает ответы;
* в определённый момент времени клиент подключён к одному серверу для обработки запросов (реже – к небольшой группе серверов);
* клиент работает с пользователем напрямую, применяя графический интерфейс;
* сервер не инициирует запросов;
* обычно для выполнения запросов клиенты проходят аутентификацию на сервере.

Главными преимуществами клиент-серверной модели являются:

* все данные хранятся на сервере, обеспечивающем больший уровень безопасности, нежели отдельный клиент (высокая безопасность);
* так как данные хранятся только на сервере, ими легко управлять (централизованный доступ к данным);
* роль сервера могут выполнять несколько физических компьютеров, объединённых в сеть (устойчивость и лёгкость сопровождения).

Благодаря этому клиент не замечает сбоев или замены отдельного серверного компьютера. В системах клиент-очередь-клиент сервер исполняет роль очереди для данных клиентов. Пиринговые приложения– это вариация системы клиент-очередь-клиент, в которой любой клиент может играть роль сервера. Сервера приложений служат для размещения и выполнения программ, которыми управляет клиент.

Данный дипломный проект разрабатывался в рамках ASP.NET Framework. Это набор спецификаций и соответствующей документации для языка C# и платформы .Net, описывающей архитектуру серверной платформы для задач средних и крупных предприятий.

Спецификации детализированы настолько, чтобы обеспечить переносимость программ c одной реализации платформы на другую. Основная цель спецификаций — обеспечить масштабируемость приложений и целостность данных во время работы системы. .Net во многом ориентирована на использование её через веб как в интернете, так и в локальных сетях.

.Net является промышленной технологией и в основном используется в высокопроизводительных проектах, в которых необходима надежность, масштабируемость, гибкость.

ASP.Net — технология, позволяющая веб-разработчикам динамически генерировать HTML, XML и другие веб-страницы. Является составной частью единой технологии создания бизнес-приложений .Net. Технология позволяет внедрять .Net-код в статичное содержимое страницы.

Встроенная защита от различных видов нападений, предоставляет следующие возможности: SQL Injection, переполнение буфера, XSS, изменение скрытых полей и прочие. Технология ASP.NET повышает степень устойчивости к вредоносным действиям и различным видам хакерских атак сайтов, построенных на ней.

В качестве фреймворка для клиентской части был выбран AngularJS. AngularJS представляет собой opensource JavaScript-фреймворк, использующий шаблон MVC. Собственно, использование MVC является его одной из отличительных особенностей. Для описания интерфейса используется декларативное программирование, а бизнес-логика отделена от кода интерфейса, что позволяет улучшить тестируемость и расширяемость приложений. Другой отличительной чертой фреймворка является двустороннее связывание, позволяющее динамически изменять данные в одном месте интерфейса при изменении данных модели в другом. Таким образом, AngularJS синхронизирует модель и представление. Кроме того, AngularJS поддерживает такие функциональности, как Ajax, управление структорой DOM, анимация, шаблоны, маршрутизация и так далее. Мощь фреймворка, наличие богатого функционала во многом повлияла на то, что он находит свое применение во все большем количестве веб-приложений, являясь на данный момент наверное одним из самых популярных javascript-фреймворков.

В качестве базы данных была выбрана MySQL. Это одна из самых популярных СУБД в современных интернет-технологиях. К основным плюсам MySQL можно отнести высокую скорость работы, быстроту обработки данных и оптимальную надежность. Немаловажно и то, что данная СУБД распространяется бесплатно и представляет собой программное обеспечение с открытым кодом. За счет этого Вы можете вносить свои изменения и модифицировать код, что весьма полезно для веб-мастеров.

# ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

Рассмотрим подробно функционирование программы. Для этого проведем анализ всех классов, которые входят в состав кода программы, и рассмотрим назначение всех методов, свойств и переменных. Статические взаимоотношения классов представлены на диаграмме классов (см. чертеж ГУИР.400201.075 РР.1).

* 1. Выбор алгоритмов предсказания и параметров команд

**3.1.1** Выбор алгоритмов предсказания

На данный момент существует достаточно много алгоритмов предсказания результата события. Так как реализовать планировалось предсказание на основе нескольких из них, а не одного конкретного, то и при выборе учитывалась совместимость данных алгоритмов по входным данным.

Выбор делался из таких алгоритмов, как квалиметрический анализ, нечеткие предсказания, предсказания на основе нейронной сети, двумерное дискретное распределение.

Требовалось выбрать один основной алгоритм и несколько, в данном случае два, вспомогательных. К основному алгоритму предъявлялось требование точности, а также он должен был выдавать вероятности исхода всех трех результатов матча. Вспомогательные алгоритмы могли делать однозначный вывод об исходе матча, то есть присваивать ему вероятность 100%.

Квалиметрический анализ, также называемый методом взвешенной суммы показателей, является довольно простым и быстрым для расчета алгоритмом прогнозирования. Квалиметрия – научная дисциплина, в рамках которой изучаются методология и проблематика комплексной, количественной оценки качества объектов любой природы. Данный алгоритм не нуждается в наличии обучающей выборки, а принимает решение только на основании входных данных. Суть алгоритма состоит в вычислении «силы» каждой команды на основе ее параметров и их весов. Для приемлемой точности работы необходимо подобрать правильные веса параметров и границы «сил» команд для того или иного решения. Это делается вручную и требует затрат времени только на этапе разработки.

Метод нечетких предсказаний также демонстрирует хорошую скорость работы, однако имеет недостаточную точность для его использования в качестве основного. Данный алгоритм требует наличия массива прошедших матчей с их параметров, на основании которых принимается решение о принадлежности предсказываемого события к тому или иному кластеру. На выходе алгоритма мы получаем вероятности всех возможных исходов, что является его плюсом.

Предсказание на основе нейронной сети практически идеально подходит для использования в качестве основного алгоритма. Нейронная сеть может выдавать сколь угодно точный результат, зависящий только от величины обучающей выборки и времени, затрачиваемого на обучение сети. Так же плюсом данного алгоритма является возможность переобучить нейронную сеть уже после завершения разработки программного продукта. Из всех существующих нейронных сетей наиболее часто используются для предсказания результатов спортивных событий сети Кохонена и сети на основе РБФ. Обе эти сети демонстрируют приемлемую точность, однако сеть Кохонена является гораздо более сложной в реализации.

Метод на основе двумерного дискретного распределения не совпадает по входным данным со всеми рассмотренными ранее. Он требует двумерную величину, в то время как параметров у события гораздо больше. Приведение события из многомерного в двумерное потребует дополнительных затрат машинного времени.

Исходя из всего вышеперечисленного, в качестве основного алгоритма предсказания был выбран метод на основе нейронной сети РБФ, а в качестве вспомогательных метод квалиметрического анализа и метод нечетких предсказаний. Итоговые веса вероятностей, предсказанный данными методами, равняются 1, 0.3 и 0.33 соответственно.

**3.1.2** Выбор параметров команд

Так как данная программная модель должна иметь возможность предсказывать результаты матчей всех игровых видов спорта, то было принято решение выбрать такие параметры для команд, которые являются общими для всех игровых видов спорта. Также влияние данных параметров на исход матча должно быть существенном, то есть при сильном изменении любого из этих параметров с большой долей вероятности изменится и исход события.

Всем вышеперечисленным критериям отвечают следующие параметры:

– место команды в турнирной таблице;

– количество набранных очков за 5 последних игр;

– число забитых голов за 5 последних игр;

– число пропущенных голов за 5 последних игр;

– количество игроков, пропускающих матч;

– количество дней отдыха до матча;

– количество матчей подряд, в которых команда не пропускала голы;

– количество матчей подряд, в которых команда забивала голы;

– дома или на выезде играет команда.

* 1. Классы блока нечетких предсказаний

Классы данного блока осуществляют предсказание результата события по методу нечетких предсказаний на основании данных, приходящих от блока предсказания.

**3.2.1** Класс FuzzyPredictor

Класс FuzzyPredictor является центральным классом блока нечетких предсказаний. Осуществляет предсказание результата события по методу нечетких предсказаний. Класс FuzzyPredictor реализует методы, необходимые совершения такого предсказания. Основные методы приведены в таблице 3.1. Описание основных полей приведено в таблице 3.2

Таблица 3.1 – Методы класса FuzzyPredictor

|  |  |
| --- | --- |
| Метод | Описание |
| 1 | 2 |
| FuzzyPredictor  List<MatchInfo> \_matchesInfo) | Устанавливает начальную конфигурацию, а именно запоминает информацию о матчах, пришедшую в конструктор |
| predict  (MatchInfo matchInfo) | Осуществляет предсказание по методу нечетких предсказаний |
| analizeTeamInfo  (List<Parameter> teamParameters) | Анализирует информацию о команде и выдает результат в формате, необходимом для дальнейшей работы |

Таблица 3.2 – Описание полей класса FuzzyPredictor

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| 1 | 2 | 3 |
| matchesInfo | List<MatchInfo> | Информация о матчах, присутствующих в системе |

* 1. Классы блока квалиметрического анализа

Классы данного блока осуществляют предсказание результата события по методу квалиметрического анализа на основании данных, приходящих от блока предсказания.

**3.3.1** Класс QualimetricPredictor

Класс QualimetricPredictor является центральным классом блока квалиметрического анализа. Осуществляет предсказание результата события по методу квалиметрического анализа. Класс QualimetricPredictor реализует методы, необходимые для совершения такого предсказания. Основные методы приведены в таблице 3.3. Полей данный класс не содержит.

Таблица 3.3 – Основные методы интерфейса QualimetricPredictor

|  |  |
| --- | --- |
| Метод | Описание |
| 1 | 2 |
| predict  (MatchInfo matchInfo) | Осуществляет предсказание по методу квалиметрического анализа |

**3.3.2** Класс TeamPowerCalculator

Класс TeamPowerCalculator осуществляет расчет так называемой «силы» команды для последующего использования в классе QualimetricPredictor. Основные методы приведены в таблице 3.4. Полей данный класс не содержит.

Таблица 3.4 – Методы класса TeamPowerCalculator

|  |  |
| --- | --- |
| Метод | Описание |
| 1 | 2 |
| calculateTeamPower  (List<Parameter> firstTeamParameters, List<Parameter> secondTeamParameters) | Осуществляет расчет сил обеих команд, чьи параметры были переданы в качестве параметров |
| calculateRelativeParameters  (Parameter firstTeamParameter, Parameter secondTeamParameter) | На основе анализа однотипных параметров двух команд вычисляет относительные параметры данного типа для этих команд |

* 1. Классы блока анализа на основе нейронной сети

Классы данного блока осуществляют предсказание результата события при помощи нейронной сети, а именно радиально-базисной сети. Параметры обученной нейронной сети хранятся в базе данных и приходят в данный класс при инициализации. Если эти параметры не пришли, то происходит обучение нейронной сети.

**3.4.1** Класс NeuralNetworkPredictor

Класс NeuralNetworkPredictor является центральным классом блока анализа на основе нейронной сети. Осуществляет предсказание результата события при помощи нейронной сети. Класс NeuralNetworkPredictor реализует методы, необходимые для совершения такого предсказания. Основные методы приведены в таблице 3.5. Описание основных полей приведено в таблице 3.6

Таблица 3.5 – Описание основных методов класса NeuralNetworkPredictor

|  |  |
| --- | --- |
| Метод | Описание |
| 1 | 2 |
| NeuralNetworkPredictor  (List<MatchInfo> matchesInfo, List<OutputLayerWeights> outputLayerWeights, List<TeachedRBF> teachedRBFs, bool isTeachNeaded) | Инициализирует объект на основе данных, пришедших от блока предсказания, а именно информации о нейронной сети |
| initializeNeuralNetwork  (List<MatchInfo> matchesInfo, List<OutputLayerWeights> outputLayerWeights, List<TeachedRBF> teachedRBFs, bool isTeachNeaded) | В зависимости от пришедших данных либо инициализирует ими нейронную сеть, либо проводит ее обучение |
| prepareDataForSaving  (List<OutputLayerWeights> outputLayerWeights, List<TeachedRBF> teachedRBFs) | Вызывает метод нейронной сети, который подготавливает информацию о ней для сохранения в базу данных |
| predict  (MatchInfo matchInfo) | Выполняет предсказание результата события при помощи нейронной сети |
| isNetworkChanged() | Возвращает информацию о том, была ли нейронная сеть изменена |

Таблица 3.6 – Описание основных полей класса NeuralNetworkPredictor

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| 1 | 2 | 3 |
| rbfNetwork | RadialBasisFunctionNetwork | Нейронная сеть |
| \_isNetworkChanged | bool | Изменена ли сеть |

**3.4.2** Класс RadialBasisFunctionNetwork

Класс RadialBasisFunctionNetwork реализует нейронную сеть на основе радиально-базисных функций. Содержит в себе алгоритмы обучения нейронной сети и классификации объекта на основе обученной нейронной сети. Основные методы приведены в таблице 3.7. Описание основных полей приведено в таблице 3.8

Таблица 3.7 – Описание основных методов класса RadialBasisFunctionNetwor

|  |  |
| --- | --- |
| Метод | Описание |
| 1 | 2 |
| RadialBasisFunctionNetwork  (int \_inputCount, int \_outputCount, int \_rbfCount, List<TeachedRBF> \_teachedRBFs, List<OutputLayerWeights> \_outputLayerWeights, char[] \_classesNames) | Инициализирует нейронную сеть данными, которые раньше были получены при ее обучении |
| RadialBasisFunctionNetwork  (Dictionary<char, List<double[]>> classes, double alfa, double maxError) | Вызывает метод обучения нейронной сети и инициализирует ее полученными значениями |
| TeachNeuralNetwork  (Dictionary<char, List<double[]>> classes) | Обучает нейронную сеть на основе пришедшей обучающей выборки |
| TeachRBFNeurons(Dictionary<char, List<double[]>> classes) | Обучает нейроны нейронной сети |
| CalculateRBFNeuronsForTrainingSet  (Dictionary<char, List<double[]>> classes) | Вычисляет значения РБФ-нейронов для обучающей выборки |
| Randomize() | Случайным образом генерирует веса нейронов выходного слоя |
| ClassifyMatch  (double[] match) | Классифицирует матч на основе обученной нейронной сети |
| prepareDataForSaving  (List<OutputLayerWeights> outputLayerWeights, List<TeachedRBF> teachedRBFs) | Подготавливает данные о нейронах и весах связей для сохранения в базу данных |

Таблица 3.8 – Описание основных полей класса RadialBasisFunctionNetwork

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| 1 | 2 | 3 |
| rbf\_teached | RBF\_teachInfo[] | Обученные РБФ-ячейки |
| outputLayer\_weights | double[,] | Веса нейронов выходного слоя |
| inputCount | int | Число нейронов входного слоя |
| outputCount | int | Число нейронов выходного слоя |
| rbfCount | int | Число РБФ-ячеек |
| alfa | double | Скорость обучения |
| maxErrorTreshold | double | Максимальное значение ошибки классификации |
| iterationCount | int | Число итераций обучения |
| classesNames | char[] | Названия кластеров |

**3.4.3** Структура RBF\_teachInfo

Структура RBF\_teachInfo содержит свойств, описывающих обученную РБФ-ячейку Описание основных полей приведено в таблице 3.9.

Таблица 3.9 – Описание основных полей структуры RBF\_teachInfo

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| 1 | 2 | 3 |
| expectations | double[] | Ожидание |
| standardDeviation | double | Стандартное отклонение |

**3.4.4** Класс RandomProvider

Класс RandomProvider предназначен генерации случайного числа. Описание основных методов и полей класса RandomProvider приведены в таблицах 3.10 и 3.11 соответственно.

Таблица 3.10 – Описание основных методов класса RandomProvider

|  |  |
| --- | --- |
| Метод | Описание |
| 1 | 2 |
| Next() | Генерация случайного числа потокобезопасным методом |

Таблица 3.11 – Описание основных полей класса RandomProvider

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| 1 | 2 | 3 |
| Rnd | Random | Генератор случайных чисел |
| \_sync | object | Объект многопоточной синхронизации |

* 1. Основные классы блока предсказания

Данный блок является управляющим для всех блоков предсказания. Классы данного блока осуществляют вызов методов предсказания всех этих блоков и осуществляют их анализ и расчет итоговых вероятностей на их основе.

**3.5.1** Интерфейс IPredictor

Интерфейс IPredictor обеспечивает одинаковую сигнатуру для методов предсказания всех классов-предсказателей. Основные методы приведены в таблице 3.12

Таблица 3.12 – Описание основных методов интерфейса IPredictor

|  |  |
| --- | --- |
| Метод | Описание |
| 1 | 2 |
| Predict  (MatchInfo matchInfo) | Сигнатура метода предсказания результата |

**3.5.2** Класс Serializer

Класс Serializer производит сериализацию данных в строку и обратно для сохранения и чтения из базы данных. Основные методы описаны в таблице 3.13

Таблица 3.13 – Описание основных методов класса Serializer

|  |  |
| --- | --- |
| Метод | Описание |
| 1 | 2 |
| SerializeToString  (double[] array) | Сериализует массив в строку |
| DeserializeFromString  (string serializedArray) | Десериализует строку в массив |

**3.5.3** Класс GeneralPredictor

Класс GeneralPredictor является основным в блоке предсказания. Он вызывает все существующие в программе предсказатели и на их основе получает итоговый результат. Основные методы приведены в таблице 3.14. Описание основных полей приведены в таблице 3.15

Таблица 3.14 – Описание основных методов класса GeneralPredictor

|  |  |
| --- | --- |
| Метод | Описание |
| 1 | 2 |
| GeneralPredictor  (List<MatchInfo> matchesInfo, List<OutputLayerWeights> outputLayerWeights, List<TeachedRBF> teachedRBFs) | Инициализирует все предсказатели на основе полученных данных |
| Predict  (MatchInfo matchInfo) | Дает итоговое предсказание |

Таблица 3.15 – Описание основных полей класса GeneralPredictor

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| 1 | 2 | 3 |
| qualimetricPredictor | Qualimetric  Predictor | Квалиметрический предсказатель |
| fuzzyPredictor | FuzzyPredictor | Нечеткий предсказатель |
| neuralNetworkPredictor | NeuralNetwork  Predictor | Предсказатель на основе нейронной сети |

* 1. Классы блока работы с базами данных

Классы данного блока представляют собой модели сущностей, хранящихся в базах данный, позволяют считывать эти сущности и преобразовывать их к бизнес-моделям.

**3.6.1** Класс Rate

Класс Rate представляют собой модель сущности, хранящейся в базе ставок. Описание основных полей приведено в таблице 3.16

Таблица 3.16 – Описание основных полей класса Rate

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поле | Тип | | Описание | |
| 1 | 2 | | 3 | |
| Id | int | | Id ставки | |
| UserId | string | | Id пользователя, сделавшего ставку | |
| MatchId | int | | Id матча, на который сделана ставка | |
| Result | | int | | Результат, на который сделана ставка |
| Amount | | decimal | | Сумма ставки |
| Coefficient | | string | | Коэффициент на данный результат |

**3.6.2** Класс RatesContext

Класс RatesContext является входной точкой для работы с базой данных ставок. Описание основных полей приведено в таблице 3.17

Таблица 3.17 – Описание основных полей класса RatesContext

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| 1 | 2 | 3 |
| Rates | DbSet<Rate> | Все ставки, содержащиеся в базе данных ставок |

**3.6.3** Класс Team

Класс Team представляют собой модель сущности, хранящейся в таблице команд в базе игр. Описание основных полей приведено в таблице 3.18

Таблица 3.18 – Описание основных полей класса Team

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| 1 | 2 | 3 |
| Id | int | Id команды |
| Name | string | Название команды |
| Position | int | Место в турнирной таблице |
| Points | int | Набрано очков за последние 5 игр |
| Goals | int | Забито голов |
| MissingBalls | int | Пропущено голов |
| InjuredPlayers | int | Игроков пропускает матч |
| DaysOfRest | int | Дней отдыха |
| ZeroGames | int | Не пропускали голов подряд |

*Продолжение таблицы 3.18*

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| GoalGames | int | Забивали голы подряд |
| HomeMatch | int | Домашний или выездной матч |
| FuzzyInfo | string | Информация о команде в формате для нечеткого предсказания |

**3.6.4** Класс Match

Класс Match представляют собой модель сущности, хранящейся в таблице матчей в базе игр. Описание основных полей приведено в таблице 3.19

Таблица 3.19 – Описание основных полей класса Match

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| 1 | 2 | 3 |
| Id | int | Id матча |
| Result | string | Результат матча |
| FirstTeamId | int? | Id первой команды |
| SecondTeamId | int? | Id второй команды |
| FirstTeam | Team | Первая команда |
| SecondTeam | Team | Вторая команда |

**3.6.5** Класс GamesContext

Класс GamesContext является входной точкой для работы с базой данных игр. Описание основных полей приведено в таблице 3.20

Таблица 3.20 – Описание основных полей класса RatesContext

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| 1 | 2 | 3 |
| Matches | DbSet<Match> | Все матчи, содержащиеся в базе данных игр |
| Teams | DbSet<Team> | Все команды, содержащиеся в базе данных игр |

**3.6.6** Класс FuzzyLimit

Класс FuzzyLimit представляет собой информацию о верхнем и нижнем пределах параметров для нечеткого предсказания, которые задаются для каждого параметра команды отдельно. Основные методы приведены в таблице 3.21. Описание основных полей приведено в таблице 3.22

Таблица 3.21 – Описание основных методов класса FuzzyLimit

|  |  |
| --- | --- |
| Метод | Описание |
| 1 | 2 |
| FuzzyLimit  (int \_leftLimit,  int \_rightLimit) | Инициализирует объект входным данным |

Таблица 3.22 – Описание основных полей класса FuzzyLimit

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| 1 | 2 | 3 |
| leftLimit | int | Нижняя граница |
| rightLimit | int | Верхняя граница |

**3.6.7** Класс Parameter

Класс Parameter представляет собой расширенную информацию об определенном показателе команды. Основные методы приведены в таблице 3.23. Описание основных полей приведено в таблице 3.24

Таблица 3.23 – Описание основных методов класса Parameter

|  |  |
| --- | --- |
| Метод | Описание |
| 1 | 2 |
| Parameter  (bool \_isStimalutor, int \_weigth, double \_value) | Инициализирует объект без пределов параметра |
| Parameter  (bool \_isStimalutor, int \_weigth, double \_value, FuzzyLimit \_fuzzyLimit) | Инициализирует объект с пределами параметра |

Таблица 3.24 – Описание основных полей класса Parameter

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| 1 | 2 | 3 |
| isStimulator | bool | Является параметр полезным для команды |
| weight | int | Вес параметра |
| value | double | Значение параметра |
| fuzzyLimit | FuzzyLimit | Пределы параметра |

**3.6.8** Класс TeamInfo

Класс TeamInfo является бизнес-моделью класса Team. Основные методы приведены в таблице 3.25. Описание основных полей приведено в таблице 3.26

Таблица 3.25 – Описание основных методов класса TeamInfo

|  |  |
| --- | --- |
| Метод | Описание |
| 1 | 2 |
| TeamInfo  (Team teamInfo) | Инициализирует объект на основе модели Team |
| List<Parameter> getListOfParameters  () | Получает массив параметров команды |

Таблица 3.26 – Описание основных полей класса TeamInfo

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| 1 | 2 | 3 |
| name | string | Название команды |
| position | Parameter | Место в турнирной таблице в формате Parameter |
| points | Parameter | Набрано очков за последние 5 игр в формате Parameter |
| goals | Parameter | Забито голов в формате Parameter |
| missingBalls | Parameter | Пропущено голов в формате Parameter |
| injuredPlayers | Parameter | Игроков пропускает матч в формате Parameter |
| daysOfRest | Parameter | Дней отдыха в формате Parameter |
| zeroGames | Parameter | Не пропускали голов подряд в формате Parameter |
| goalGames | Parameter | Забивали голы подряд в формате Parameter |
| homeMatch | Parameter | Домашний или выездной матч в формате Parameter |
| fuzzyInfo | string | Информация о команде в формате для нечеткого предсказания |

**3.6.9** Класс MatchInfo

Класс MatchInfo является бизнес-моделью класса Match. Основные методы приведены в таблице 3.27. Описание основных полей приведено в таблице 3.28

Таблица 3.27 – Описание основных методов класса MatchInfo

|  |  |
| --- | --- |
| Метод | Описание |
| 1 | 2 |
| MatchInfo  (Match matchInfo, TeamInfo \_firstTeam, TeamInfo \_secondTeam) | Инициализирует объект на основе модели Match |
| getMatchParams  () | Получает массив параметров обеих команд матча |

Таблица 3.28 – Описание основных полей класса MatchInfo

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| 1 | 2 | 3 |
| id | int | Id матча |
| firstTeam | TeamInfo | Первая команда |
| secondTeam | TeamInfo | Вторая команда |
| realResult | int | Результат матча |

**3.6.10** Класс OutputLayerWeights

Класс OutputLayerWeights представляет собой модель сущности, хранящейся в таблице выходных нейронов в базе нейронной сети. Описание основных полей приведено в таблице 3.29

Таблица 3.29 – Описание основных полей класса OutputLayerWeights

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| 1 | 2 | 3 |
| Id | int | Id выходного нейрона |
| OutputLayerWeight | string | Веса выходного нейрона |

**3.6.11** Класс TeachedRBF

Класс TeachedRBF представляет собой модель сущности, хранящейся в таблице РБФ-ячеек в базе нейронной сети. Описание основных полей приведены в таблице 3.30

Таблица 3.30 – Описание основных полей класса TeachedRBF

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| 1 | 2 | 3 |
| Id | int | Id РБФ-ячейки |
| StandartDeviation | double | Стандартное отклонение |
| Expectation | string | Ожидание |

**3.6.12** Класс NeuralNetworkContext

Класс NeuralNetworkContext является входной точкой для работы с базой данных нейронной сети. Описание основных полей приведено в таблице 3.31

Таблица 3.31 – Описание основных полей класса NeuralNetworkContext

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| 1 | 2 | 3 |
| OutputLayer  Weights | DbSet<OutputLayer  Weights> | Все веса выходных нейронов, содержащиеся в базе нейронной сети |
| TeachedRBFs | DbSet<TeachedRBF> | Все РБФ-ячейки, содержащиеся в базе нейронной сети |

* 1. Классы блока корректировки коэффициентов

Данный блок вычисляет реальные коэффициенты на основе данных блока предсказания, а также производит корректировку реальных коэффициентов с учетом обязательной выгоды для владельца программы.

**3.7.1** Класс CoefficientsCalculator

Класс CoefficientsCalculator вычисляет реальные коэффициенты на основе данных блока предсказания, а также производит корректировку реальных коэффициентов с учетом обязательной выгоды для владельца программы. Основные методы приведены в таблице 3.32

Таблица 3.32 – Описание основных методов класса CoefficientsCalculator

|  |  |
| --- | --- |
| Метод | Описание |
| 1 | 2 |
| calculateRealCoefficients  (double[] prediction) | Вычисляет реальные коэффициенты |
| calculateAdjustedCoefficients  (double[] prediction) | Вычисляет откорректированные коэффициенты |

* 1. Классы блока клиентского интерфейса

Интерфейс пользователя представляет собой веб-приложение, спроектированное на основе многоуровневой архитектуре. *Многоуровневая архитектура* (multilayered architecture) сосредоточена на иерархическом распределении отдельных частей системы при помощи эффективного разделения отношений.

Уровень представления (presentation layer) ответственен за взаимодействие с пользователем, ввод и вывод информации.

Бизнес-уровень или уровень бизнес-логики (business logic layer) обрабатывает информацию, реализуя конкретные бизнес-правила.

Уровень доступа к данным (data access layer) обеспечивает загрузку и сохранение информации, используя источник данных (файл, база данных) или внешний сервис.

При этом уровень представления считается высшим, за ним идёт уровень бизнес-логики, а за уровнем бизнес-логики – уровень доступа к данным.

Классы данного блока предназначены для корректного представления данных пользователю, описанные выше функции реализуются в других блоках.

**3.8.1** Класс ResultsEncoder

Класс ResultsEncoder преобразует информацию о результате матча из внутреннего формата программы в формат, который понятен пользователю. Описание основных полей приведено в таблице 3.33

Таблица 3.33 – Описание основных методов класса ResultsEncoder

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| 1 | 2 | 3 |
| results | string[] | Массив строковых представлений результатов матчей, где индекс массива соответствует внутреннему представлению результата в программе |

* 1. Классы блока управления

Являясь центральным компонентом системы, отвечает за такие задачи, как запуск изменения баз данных, передачу данных на клиентский интерфейс,

запуск предсказания результата события.

**3.9.1** Класс GamesController

Класс GamesController является управляющим классом для работы с информацией о матчах и командах, участвующих в них. Позволяет добавлять и редактировать, а также выводить пользователю данную информацию. Основные методы приведены в таблице 3.34. Описание основных полей приведено в таблице 3.35

Таблица 3.34 – Описание основных методов класса GamesController

|  |  |
| --- | --- |
| Метод | Описание |
| 1 | 2 |
| Results  () | Подготавливает список всех матчей, результат которых уже известен, и выводит этот список пользователю |
| EditMatches  () | Выводит список всех доступных матчей с возможностью добавить новый или изменить уже существующий. Доступен только администратору. |
| CreateMatch  (bool? useTeamIds) | Возвращает страницу для создания нового матча. Если в параметрах пришло true, то в Id команд матча поместит \_firstTeamId и \_secondTeamId |
| CreateMatch  (Match match) | Создает новый матч и возвращает администратора на страницу редактирования матчей |
| EditMatch  (int id) | Возвращает страницу редактирования матча, id которого пришло в параметре |
| EditMatch  (Match match) | Сохраняет отредактированный матч в базе данных игр |
| CreateTeam  (bool isFirstTeam) | Доступен только при создании нового матча. Возвращает страницу для создания новой команды. Данная команда будет считаться первой, если в параметре пришло true |

*Продолжение таблицы 3.34*

|  |  |
| --- | --- |
| 1 | 2 |
| CreateTeam  (Team team) | Сохраняет созданную команду в базе данных игр и возвращает администратору к созданию матча |
| getMatchesInfo  () | Если к методу обратились первый раз, то подготавливает и запоминает массив всех матчей, а затем его возвращает, иначе просто возвращает ранее запомненный массив |
| updateTeamsTable  (List<Team> teamsToUpdate) | Сохраняет изменения в полученных командах в базе данных игр |

Таблица 3.35 – Описание основных полей класса GamesController

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| 1 | 2 | 3 |
| db | GamesContext | Подключение к базе данных игр |
| matchesInfo | List<MatchInfo> | Массив всех содержащихся в базе данных матчей |
| addFirstTeam | bool | Создается ли в данный момент первая команда для нового матча |
| \_firstTeamId | int? | Id первой созданной команды для нового матча |
| \_secondTeamId | int? | Id второй созданной команды для нового матча |

**3.9.2** Класс PredictionController

Класс PredictionController является управляющим классом для работы предсказаниями результатов матчей и непосредственно запускает переобучение нейронной сети. Основные методы приведены в таблице 3.36. Описание основных полей приведено в таблице 3.37

Таблица 3.36 – Описание основных методов класса PredictionController

|  |  |
| --- | --- |
| Метод | Описание |
| 1 | 2 |
| FutureEvents() | Подготавливает список всех матчей, результат которых еще не известен |

*Продолжение таблицы 3.36*

|  |  |
| --- | --- |
| 1 | 2 |
| ShowFutureEvents() | Выводит список пользователю. Также вычисляет и выводит коэффициенты на все возможные исходы данного матча |
| TeachNN() | Запускает процесс обучения нейронной сети и после его вызывает метод сохранения информации о нейронной сети |
| saveNeuralNetwokToDataBase  (List<OutputLayerWeights> outputLayerWeights, List<TeachedRBF> teachedRBFs) | Сохраняет информацию о нейронной сети |

Таблица 3.37 – Описание основных полей класса PredictionController

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| 1 | 2 | 3 |
| neuralNetworkDb | NeuralNetworkContext | Подключение к базе данных нейронной сети |

**3.9.3** Класс RatesController

Класс RatesController является управляющим классом для работы со ставками, позволяет совершить ставку на исход доступного матча, а также просмотреть информацию о всех своих ставках. Основные методы приведены в таблице 3.38. Описание основных полей приведено в таблице 3.39

Таблица 3.38 – Описание основных методов класса RatesController

|  |  |
| --- | --- |
| Метод | Описание |
| 1 | 2 |
| MakeRate(int matchId, int result, double coefficient) | Возвращает форму для совершения ставки на выбранный исход определенного матча |
| MakeRate(Rate rate) | Сохраняет совершенную ставку в базе данных ставок |

*Продолжение таблицы 3.38*

|  |  |
| --- | --- |
| 1 | 2 |
| RatesInfo() | Возвращает страницу с информацией о всех ставках данного пользователя. Перед этим проверяет, не произошли ли события, на которые сделана ставка. И если да, то изменяет состояние ставки в соответствии с результатом |

Таблица 3.39 – Описание основных полей класса RatesController

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| 1 | 2 | 3 |
| ratesDb | RatesContext | Подключение к базе данных ставок |

* 1. Структура данных

База данных программы хранит в себе информацию о зарегистрированных пользователях, их ставках, матчах, командах и информацию об обученной нейронной сети. Логически выделены блоки базы данных пользователей и базы данных игр, которые физически представлены четырьмя базами данных:

– база данных игр;

– база данных ставок;

– база данных пользователей;

– база данных нейронной сети.

Отношения между таблицами представлены в модели данных (см. чертеж ГУИР.400201.075 РР.2).

**3.10.1** База данных игр

База данных игр состоит из двух таблицы: Matches и Teams. Описание полей данных таблиц приведено в таблицах 3.40 и 3.41 соответственно.

Таблица 3.40 – Описание полей таблицы Matches

|  |  |
| --- | --- |
| Поле | Описание |
| 1 | 2 |
| Id | Хранит идентификатор данного матча, является уникальным первичным ключом таблицы |

*Продолжение таблицы 3.40*

|  |  |
| --- | --- |
| 1 | 2 |
| FirstTeamId | Хранит идентификатор первой команды, принимающей участие в матче. Является первым внешним ключом таблицы |
| SecondTeamId | Хранит идентификатор второй команды, принимающей участие в матче. Является вторым внешним ключом таблицы |
| Result | Хранит результат матча в числовом формате. Цифры ноль, один, два, три обозначают соответственно ничью, выигрыш, проигрыш первой команды или то, что матч еще не состоялся |

Таблица 3.41 – Описание полей таблицы Teams

|  |  |
| --- | --- |
| Поле | Описание |
| 1 | 2 |
| Id | Хранит идентификатор данной команды, является уникальным первичным ключом таблицы |
| Name | Название команды |
| Position | Место в турнирной таблице |
| Points | Набрано очков за последние 5 игр |
| Goals | Забито голов |
| MissingBalls | Пропущено голов |
| InjuredPlayers | Игроков пропускает матч |
| DaysOfRest | Дней отдыха |
| ZeroGames | Не пропускали голов подряд |
| GoalGames | Забивали голы подряд |
| HomeMatch | Домашний или выездной матч |
| FuzzyInfo | Информация о команде в формате для нечеткого предсказания |

**3.10.2** База данных ставок

База данных ставок состоит из одной таблицы Rates, в которой хранится информация о ставках пользователей. Описание полей данной таблицы приведено в таблице 3.42

Таблица 3.42 – Описание полей таблицы Rates

|  |  |
| --- | --- |
| Поле | Описание |
| 1 | 3 |
| Id | Хранит идентификатор данной ставки, является уникальным первичным ключом таблицы |
| UserId | Id пользователя, сделавшего ставку |
| MatchId | Id матча, на который сделана ставка |
| Result | Результат, на который сделана ставка |
| Amount | Сумма ставки |
| Coefficient | Коэффициент на данный результат |

**3.10.3** База данных пользователей

База данных пользователей содержит в себе пять таблиц с информацией о пользователях, их ролях и входах в систему, но интерес представляет только таблица Users. Описание полей данной таблицы приведено в таблице 3.43

Таблица 3.43 – Описание полей таблицы Users

|  |  |  |
| --- | --- | --- |
| Поле | Описание | |
| 1 | 3 | |
| Id | Хранит идентификатор данной пользователя в строковом формате, является уникальным первичным ключом таблицы | |
| Email | | Email пользователя, он же логин в системе |
| PasswordHash | | Хешированный пароль пользователя |

**3.10.4** База данных нейронной сети

База данных нейронной сети состоит из таблиц OutputLayerWeights и TeachedRBFs, содержащих информацию о весах выходного слоя нейронов и РБФ-ячейках сети соответственно. Описание полей данных таблиц приведено в таблицах 3.44 и 3.45 соответственно.

Таблица 3.44 – Описание полей таблицы OutputLayerWeights

|  |  |
| --- | --- |
| Поле | Описание |
| 1 | 3 |
| Id | Хранит идентификатор нейрона выходного слоя, первичный ключ |
| OutputLayerWeight | Веса выходного нейрона |

Таблица 3.45 – Описание полей таблицы TeachedRBFs

|  |  |
| --- | --- |
| Поле | Описание |
| 1 | 3 |
| Id | Хранит идентификатор РБФ-ячейки, является уникальным первичным ключом таблицы |
| StandartDeviation | Стандартное отклонение |
| Expectation | Ожидание |

1. РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

**4.1** Квалиметричесикй анализ

**4.1.1** Расчет силы команды

Сила команды является основным и единственным параметром, по которому квалиметрический анализатор принимает решение об исходе события. Поэтому расчет силы команды является основным методом данного анализатора.

У каждого параметра команды есть свой вес и определено, является ли он стимулятором, то есть увеличится ли сила команды при увеличении данного параметра. Однако просто сложить параметры, умноженные на веса, не является лучшим решением, так как результат не будет зависеть от взаимного отношения параметров, что неверно.

Для вычисления силы команды предназначен класс TeamPowerCalculator. Первым этапом квалиметрического анализа является вычисление относительного значения каждого из параметров, присутствующих у команды. Данные операции производятся в методе calculateRelativeParameters, который принимает в себя пару однотипных параметров обеих команд. Рассмотрим фрагмент его кода, приведенный ниже.

В данном коде вначале вычисляется сумма принятых параметров. Если данная сумма равна нулю, то дальнейшие действия бессмысленны и относительные параметры равны нулю. Иначе вычисляется значение каждого параметра относительно суммы и, если он не является стимулятором, то производится его инвертирование относительно единицы.

double sum = firstTeamParemeter.value + secondTeamParameter.value;

if (sum == 0)

{

relativeParameters[0] = 0;

relativeParameters[1] = 0;

}

else

{

double firstTeamRelativeParemeter = firstTeamParemeter.value / sum;

if (!firstTeamParemeter.isStimulator)

firstTeamRelativeParemeter = 1 - firstTeamRelativeParemeter;

double secondTeamRelativeParemeter = secondTeamParameter.value / sum;

if (!secondTeamParameter.isStimulator)

secondTeamRelativeParemeter = 1 - firstTeamRelativeParemeter;

relativeParameters[0] = firstTeamRelativeParemeter;

relativeParameters[1] = secondTeamRelativeParemeter;

}

Далее каждый из полученных относительных параметров добавляется к силе команды, которая изначально равна нулю. В конце расчета сила каждой команды нормируется относительно их суммы, что позволяет получить вероятность ее победы от нуля до единицы. Данное действие показано в коде ниже:

double teamPowersSum = teamPowers[0] + teamPowers[1];

teamPowers[0] /= teamPowersSum;

teamPowers[1] /= teamPowersSum;

**4.1.2** Вычисление вероятностей исходов матчей

Данный метод предсказания является вспомогательным, поэтому необходимость предсказания вероятностей каждого из исходов не является приоритетной для него.

Сила каждой команды после ее нормировки и будет являться вероятностью победы данной команды. Для поддержания консистентности и однотипности данных, возвращаемых всеми предсказателями, необходимо так же предусмотреть вероятность ничьей.

В данном случае вероятность ничьей принята равной отношению вероятности победы более слабой команды к вероятности победы более сильной команды. Такой выбор вероятностей исходов события позволяет получить не просто предсказание, а вероятности победы каждой команды относительно друг друга. Ниже приведен код предсказания:

double[] teamPowers = TeamPowerCalculator.calculateTeamPowers(matchInfo.firstTeam.getListOfParameters(), matchInfo.secondTeam.getListOfParameters());

double[] prediction = new double[] {

teamPowers.Min() / teamPowers.Max(),

teamPowers[0],

teamPowers[1]

};

return prediction;

* 1. Нечеткие предсказания

Следующим вспомогательным алгоритмом предсказания является алгоритм нечетких предсказаний, также известный как алгоритм на основе нечетких правил.

Суть данного алгоритма заключается в том, что определенный набор параметров команды сравнивается с наборами параметров всех других команд, имеющихся в системе, и при его совпадении выдается результат данного события. Однако, в связи с тем, что в программе используется довольно обширный набор параметров, для полного совпадения необходима огромная база данный. Учитывая это, алгоритм был немного изменен.

**4.2.1** Генерация параметра команды

Подготовительным этапом алгоритма нечетких предсказаний является генерация строкового параметра каждой команды, содержащего иформацию обо всех других параметрах команды. Рассмотрим код метода analizeTeamInfo, выполняющего данные действия:

string result = "";

for (int i = 0; i < teamParameters.Count; i++)

{

double value = teamParameters[i].value;

int leftLimit = teamParameters[i].fuzzyLimit.leftLimit;

int rigtLimit = teamParameters[i].fuzzyLimit.rightLimit;

if (value < leftLimit)

result += "1";

else if (value > rigtLimit)

result += "3";

else

result += "2";

}

return result;

В данном методе каждому параметру команды ставится в соответсвие цифра от одного до трех. Выбор цифры происходит в соответствии с границами, определенными для данного конкретного параметра. Все полученные цифры конкатенируются в одну строковую константу, которая и является параметром, анализируемым алгоритмом нечетких предсказаний.

**4.2.2** Вычисление вероятностей исходов матчей

Как было сказано выше, данный алгоритм был изменен в связи с недостаточным размером базы данных игр. Основные изменения заключаются в том, что поиск происходит не до полного совпадения строки-параметра, а проходит по всей базе данных, вычисляя схожесть строк-параметров для каждого из трех возможных исходов матча. Рассмотрим код метода predict, совершающего вычисление данных вероятностей:

string[] matchesFuzzyInfo = new string[matchesInfo.Count];

int i = 0;

string matchFuzzyInfo = "";

Для каждого матча вычисляется параметр для сравнения, получаемый путем склеивания строк-параметров участвующих в нем команд.

foreach (MatchInfo item in matchesInfo)

{

matchesFuzzyInfo[i++] = item.firstTeam.fuzzyInfo + item.secondTeam.fuzzyInfo;

if (item.id == matchInfo.id)

matchFuzzyInfo = matchesFuzzyInfo[i - 1];

}

int[] maxSimilarMatchesIds = new int[] { 0, 0, 0 };

int[] similarCounts = new int[] { 0, 0, 0 };

int currentSimilar;

int matchResultType;

Начинается цикл по всем матчам.

for (i = 0; i < matchesInfo.Count; i++)

{

if (matchesInfo[i].id == matchInfo.id || matchesInfo[i].realResult == 3)

continue;

currentSimilar = 0;

matchResultType = matchesInfo[i].realResult;

for (int j = 0; j < matchFuzzyInfo.Length; j++)

{

if (matchFuzzyInfo[j] == matchesFuzzyInfo[i][j])

currentSimilar++;

if (currentSimilar > similarCounts[matchResultType])

{

similarCounts[matchResultType] = currentSimilar;

maxSimilarMatchesIds[matchResultType] = i;

}

}

}

double[] prediction = new double[] {

similarCounts[0] / matchFuzzyInfo.Length,

similarCounts[1] / matchFuzzyInfo.Length,

similarCounts[2] / matchFuzzyInfo.Length,

};

return prediction;

В показанном выше цикле матчи классифицируются по их результату. Для каждого результата заведена переменная, показывающая степень подобия данного и предсказываемого матча. С помощью этих переменных в конце цикла получается характеристика подобия исхода анализируемого матча с каждым из возможных исходов матчей.

Для получения вероятности наступления каждого из исходов степень подобия делится на размер строки-параметра матча. Таким образом, изменения, внесенные в алгоритм, позволяют не только не зависеть от размера базы данных, но и улучшают выходные параметры алгоритма. Вместо одного конкретного результата без вероятности его наступления, алгоритм выдает вероятности всех возможных исходов.

**4.3** Предсказание на основе нейронной сети РБФ

**4.3.1** Обучение нейронной сети

Нейронная сеть состоит из нейронов трех слоев:

– входного;

– внутренного;

– выходного.

В сети РБФ нейроны внутреннего слоя представляют собой РБФ-ячейки, устанавливаемые в возбужденное состояние активационной функцией. Данная сеть и ее активационная функция приведены на рисунке 4.1.

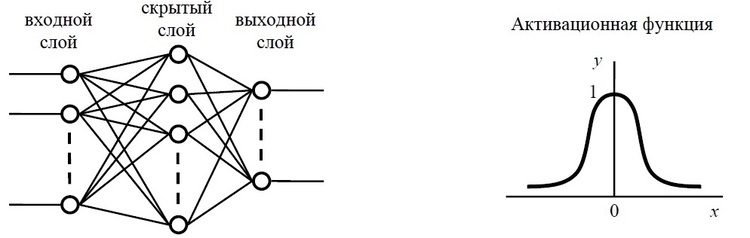


Рисунок 4.1 – Нейронная сеть РБФ и активационная функцияя нейронов внутреннего слоя

Основным этапом в реализации работы нейронной сети является ее обучение. Для этого используется обучающая выборка, от размеров и качества которой напрямую зависит последующая точность работы сети. Рассмотрим алгоритм обучения, представленный методом TeachNeuralNetwork класса RadialBasisFunctionNetwork.

Данный метод итерационно обучает нейронную сеть до тех пор, пока не будет достигнута максимальная величина ошибки, переданная в параметрах. Суть обучения состоит в следующем: обучение РБФ-ячеек происходит за одну итерацию, далее лишь подбираются подходящие веса нейронов выходного слоя. Рассмотрим фрагмент кода, отвечающий за обучения РБФ-ячейки:

List<RBF\_teachInfo> RBFs = new List<RBF\_teachInfo>();

Инициализация всех ячеек начальными значениями:

foreach (List<double[]> matchClass in classes.Values)

{

foreach (double[] match in matchClass)

{

RBF\_teachInfo rbf = new RBF\_teachInfo();

rbf.expectations = match;

RBFs.Add(rbf);

}

}

rbfCount = RBFs.Count;

rbf\_teached = RBFs.ToArray();

if (rbfCount == 1)

{

double sko = 0;

for (int k = 0; k < inputCount; k++)

sko += Math.Pow((rbf\_teached[0].expectations[k]),2);

rbf\_teached[0].standardDeviation = Math.Sqrt(sko);

return;

}

Если в сети более одной РБФ-ячейки, то для каждой из них вычисляем величину стандартного отклонения. Эта величина представляет собой расстояние между двумя соседними ячейками в многомерном пространстве:

List<double>[] deviations = new List<double>[rbfCount];

for (int i = 0; i < deviations.Length; i++)

deviations[i] = new List<double>();

for (int i = 0; i < rbfCount; i++)

{

for (int j = i + 1; j < rbfCount; j++)

{

double sko = 0;

for (int k = 0; k < inputCount; k++)

sko += Math.Pow((rbf\_teached[i].expectations[k] - rbf\_teached[j].expectations[k]), 2);

sko = Math.Sqrt(sko);

deviations[i].Add(sko);

deviations[j].Add(sko);

}

}

for (int i = 0; i < rbfCount; i++)

rbf\_teached[i].standardDeviation = deviations[i].Min();

После того, как все нейроны внутреннего слоя обучены, происходит вычисление весов нейронов выходного слоя. За это отвечает фрагмент кода, приведенный ниже:

for (int j = 0; j < rbfCount; j++)

{

for (int k = 0; k < outputCount; k++)

{

outputLayer\_weights[j, k] = mu \* outputLayer\_weights[j, k] + alfa \* outputErrors[k] \* rbfNeurons[classNumber][matchNumber][j];

}

}

Величина изменения веса выходного нейрона за одну итерацию зависит от величины ошибки в данной итерации и от заданной скорости обучения нейронной сети.

**4.3.2** Сохранение состояния обученной нейронной сети

Так как при каждом запуске программы заново обучать нейронную сеть глупо, то было принято решение хранить состояние обученной нейронной сети в базе данных. Для этого необходимо подготовить информацию о нейронах внутренного и выходного слоев к сохранению. Этим занимается метод prepareDataForSaving, код которого приведен ниже:

int i = 1;

foreach (var item in rbf\_teached)

{

TeachedRBF teachedRBF = new TeachedRBF();

teachedRBF.Id = i++;

teachedRBF.StandartDeviation = item.standardDeviation;

teachedRBF.Expectation = Serializer.SerializeToString(item.expectations);

teachedRBFs.Add(teachedRBF);

}

for (i = 0; i < rbfCount; i++)

{

OutputLayerWeights outputLayerWeightsItem = new OutputLayerWeights();

outputLayerWeightsItem.Id = i + 1;

outputLayerWeightsItem.OutputLayerWeight = Serializer.SerializeToString(

new double[] { outputLayer\_weights[i, 0],

outputLayer\_weights[i, 1],

outputLayer\_weights[i, 2] });

outputLayerWeights.Add(outputLayerWeightsItem);

}

**4.3.3** Классификация матча нейронной сетью

Классификация матча заключается в определении вероятностей каждого из его исходов нейронной сетью. Рассомтрим код метода ClassifyMatch, выполняющего дынные действия:

double[] rbfNeurons = new double[rbfCount];

double[] outputNeurons = new double[outputCount];

Вычисляем выходное значение каждой РБФ-ячейки:

for (int j = 0; j < rbfCount; j++)

{

double sum = 0;

for (int i = 0; i < inputCount; i++)

sum += (match[i] - rbf\_teached[j].expectations[i]) \* (match[i] - rbf\_teached[j].expectations[i]);

rbfNeurons[j] = Math.Exp(-sum / Math.Pow(rbf\_teached[j].standardDeviation, 2));

}

Вычисляем итоговое значение нейронов выходного слоя на основании значений нейронов внутреннего слоя и весов нейронов выходного слоя:

for (int k = 0; k < outputCount; k++)

{

double sum = 0;

for (int j = 0; j < rbfCount; j++)

sum += outputLayer\_weights[j, k] \* rbfNeurons[j];

if (sum < 0)

sum = 0;

outputNeurons[k] = sum;

}

Возвращаем результаты классификации, которые и являются искомыми вероятностями:

Dictionary<char, double> result = new Dictionary<char, double>();

int t = 0;

foreach (char className in classesNames)

{

result.Add(classesNames[t], outputNeurons[t]);

t++;

}

return result;

**4.4** Вычисление итоговых вероятностей

На основании трех описанных выше реализаций алгоритмов необходимо предоставить пользователю итоговые вероятности исходов матча.

Так как предсказание нейронной сети является наиболее точным, то его вес выбран максимальным из всех трех и равняется единице. Веса квалиметрического а нечеткого анализатора выбраны 0.3 и 0.33 соответственно. Также предусмотрена возможность того, что реализованный предсказатель допустит ошибку в вычислениях и предсказанная вероятность будет неоправданно малой. Ниже приведен код метода predict класса GeneralPredictor, реализующий вычисление итоговых вероятностей:

double[] qualimetricPrediction = qualimetricPredictor.predict(matchInfo);

double[] fuzzyPrediction = fuzzyPredictor.predict(matchInfo);

double[] neuralNetworkPrediction = neuralNetworkPredictor.predict(matchInfo);

double[] finalPrediction = new double[3];

for (int i = 0; i < 3; i++)

{

finalPrediction[i] = neuralNetworkPrediction[i];

finalPrediction[i] += fuzzyPrediction[i] / 3;

finalPrediction[i] += qualimetricPrediction[i] \* 0.3;

if (finalPrediction[i] < 0.15)

finalPrediction[i] = 0.15 + i / 100;

}

double sum = finalPrediction.Sum();

Нормализация вероятностей относительно единицы:

for (int i = 0; i < 3; i++)

finalPrediction[i] /= sum;

return finalPrediction;

**5** ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

**5.1** Модульное тестирование

Для постоянного контроля качества в процессе разработки системы широко используется модульное тестирование – процесс, позволяющий проверить модули исходного кода программы в процессе разработки. Смысл этого подхода заключается в том, чтобы писать тесты для каждого нетривиального метода исходного кода. Это позволяет достаточно легко проверить, не привело ли очередное изменение кода к регрессии, то есть к появлению ошибок в уже написанных и оттестированных местах программы, а также облегчает обнаружение и устранение таких ошибок.

Цель модульного тестирования — изолировать отдельные части программы и показать, что по отдельности эти части работоспособны. Модульное тестирование позволяет в дальнейшем проводить изменение уже существующего кода в целях его улучшения, будучи уверенным, что изменяемый модуль и модули, его использующие, по-прежнему работают корректно. Так же модульный тест можно рассматривать и в качестве документирования использования классов: описывая примеры использования класса и его методов, модульные тесты дают понять разработчикам, как правильно использовать класс.

В среде разработки Visual Studio 2013, в которой разрабатывался данный проект, встроены средства модульного тестирования. Для написания модульного теста для выбранного класса, необходимо создать класс, который будет инкапсулировать в себе выбранный класс и помеченный атрибутом [TestClass]. Тестовые методы в таком случае будут помечены с помощью атрибута [TestMethod]. Таким образом, в общем случае модульный тест имеет следующий вид:

[TestClass]

public class QualimetricPredictorTests

{

Team strongTeam;

Team weakTeam;

Match match;

TeamInfo strongTeamInfo;

TeamInfo weakTeamInfo;

QualimetricPredictor qualimetricPredictor;

public QualimetricPredictorTests()

{

strongTeam = new Team()

{

Id = 1, GoalGames = 5, Goals = 15,

DaysOfRest = 14, HomeMatch = 1,

InjuredPlayers = 0, MissingBalls = 0,

Points = 15, Position = 1, ZeroGames = 5

};

weakTeam = new Team()

{

Id = 2, GoalGames = 0, Goals = 0,

DaysOfRest = 1, HomeMatch = 0,

InjuredPlayers = 5, MissingBalls = 15,

Points = 0, Position = 15, ZeroGames = 0

};

match = new Match() { Id = 1, Result = "3" };

strongTeamInfo = new TeamInfo(strongTeam);

weakTeamInfo = new TeamInfo(weakTeam);

qualimetricPredictor = new QualimetricPredictor();

}

[TestMethod]

public void ShouldPredictVictoryIfFirstTeamIsStronger()

{

MatchInfo matchInfo = new MatchInfo(match, \_\_\_\_\_\_\_\_strongTeamInfo, weakTeamInfo);

double [] prediction = \_\_\_\_\_\_\_\_qualimetricPredictor.predict(matchInfo);

Assert.AreEqual(true, (prediction[1] > prediction[2]));

}

[TestMethod]

public void PredictorShouldPredictDrawIfTeamsAreEqual()

{

MatchInfo matchInfo = new MatchInfo(match, \_\_\_\_\_\_\_\_strongTeamInfo, strongTeamInfo);

double[] prediction = \_\_\_\_\_\_\_\_qualimetricPredictor.predict(matchInfo);

Assert.AreEqual(true, (prediction[1] == prediction[2]));

}

[TestMethod]

public void ShouldPredictDefeatIfSecondTeamIsStronger()

{

MatchInfo matchInfo = new MatchInfo(match, weakTeamInfo, \_\_\_\_\_\_\_\_strongTeamInfo);

double[] prediction = \_\_\_\_\_\_\_\_qualimetricPredictor.predict(matchInfo);

Assert.AreEqual(true, (prediction[1] < prediction[2]));

}

}

С помощью аннотации @Before помечается метод, который будет запущен перед каждым тестовым методом. В нем обычно производится инициализация всех нужных для тестирования данных. Аннотацией @After помечается метод, в котором производится освобождение ресурсов после того, как тест отработает. Результат работы тестов на примере тестирования класса QualimetricPredictor в среде разработки VisualStudio 2015 показан на рисунке 5.1.

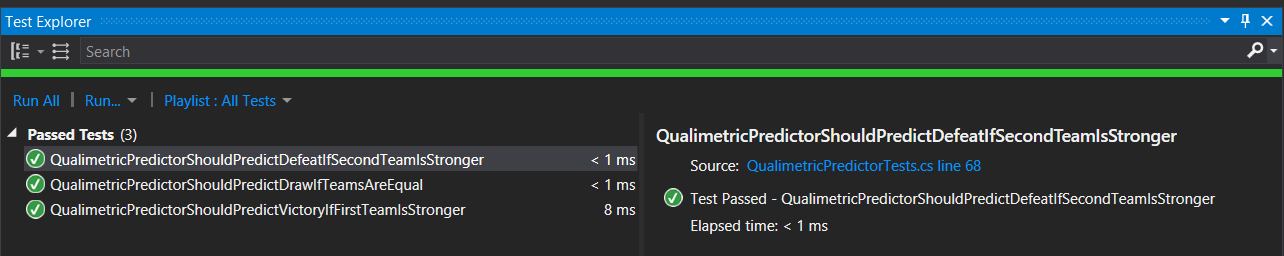


Рисунок 5.1 – Результат тестирования класса QualimetricPredictor

Все классы объектной модели приложения покрыты модульными тестами в среднем на 90%, что обеспечивает низкую вероятность регрессии при дальнейшей разработке и поддержке системы.

**5.2** Обработка исключительных ситуаций

Важной частью методики испытаний системы является корректная

обработка исключительных ситуаций. В случае возникновения исключения приложение должно оставаться в рабочем состоянии, а пользователь должен получить уведомление и иметь возможность продолжить свою работу с того места, где возникла исключительная ситуация.

Корректная обработка исключений в .Net веб-приложениях имеет особо важное значение, потому что в случае неправильно обработанного исключения пользователю предоставляется информация о стеке выполнения метода, в котором оно произошло, что является серьезным упущением в безопасности приложения.

Правильная обработка исключительных ситуаций включает в себя следующее:

* исключения должны перехватываться;
* информация об исключениях должна записываться в хранилище данных для возможности ее последующего анализа;
* пользователь не должен видеть полной информации о произошедшей ошибке: стека выполнения метода, деталей исключения и так далее.

Рассмотрим наличие правильной обработки исключений в случае, когда при работе приложения пропало соединение с базой данных, и GamesController не может далее нормально функционировать.

Ожидаемый результат работы программы в подобной исключительной ситуации представляет собой последовательность действий:

* вывести сообщение на графическом интерфейсе о случившейся исключительной ситуации;
* записать информацию об ошибке.

При попытке воспроизвести такое поведение системы, на экран вывелось сообщение о том, что при работе с базой данных произошла ошибка. Данное поведение можно увидеть на рисунке 5.2.

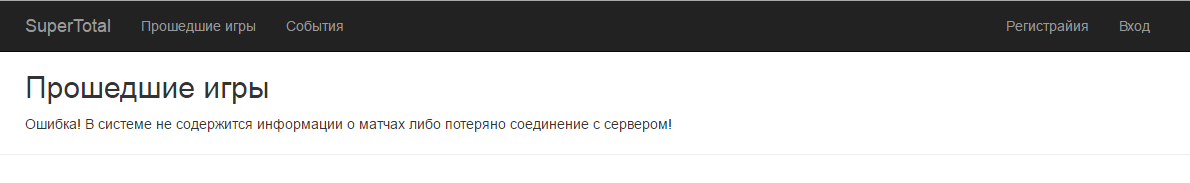


Рисунок 5.2 – Сообщение об ошибке при работе с базой данных

Также все поля для ввода данных в программу поддерживают проверку формата вводимых данных. На рсунке 5.3 жирной линией подчеркнуто поле ввода, которое специально предназначено для числовой информации и не поддерживает ввод строковых констант.

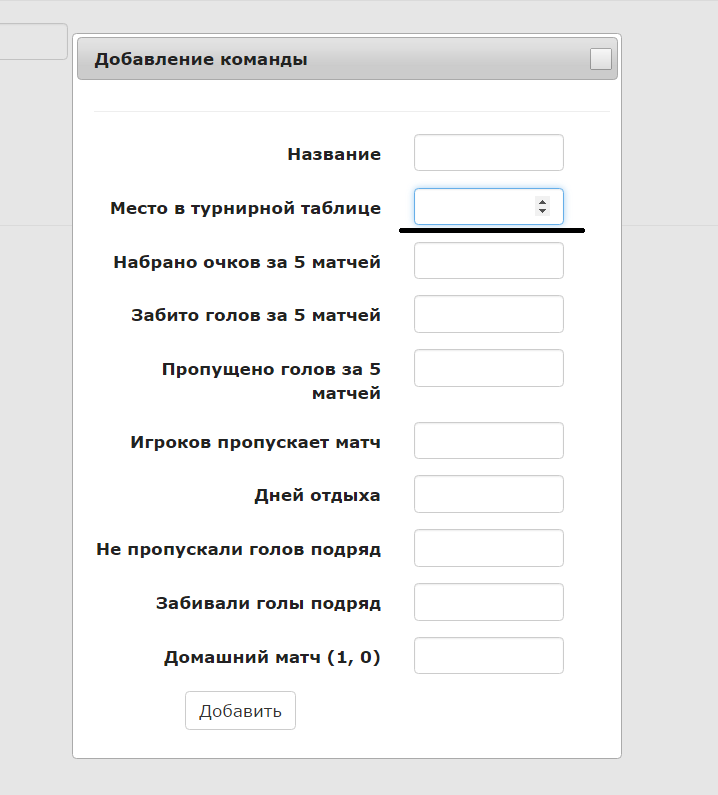


Рисунок 5.3 – Демонстрация валидации данных полем ввода

**5.3** Регрессионное тестирование

Регрессионное — проверка исправления вновь найденного дефекта, а такжепроверка, что исправленный ранее и верифицированный дефект не воспроизводится в системе снова. Сюда также можно включить проверку того, что не нарушилась работоспособность работающей ранее функциональности, если её код мог быть затронут при исправлении некоторых дефектов в другой функциональности.

Обычно используемые методы регрессионного тестирования включают повторные прогоны предыдущих тестов, а также проверки, не попали ли регрессионные ошибки в очередную версию в результате слияния кода. Поэтому считается хорошей практикой при исправлении ошибки создать тест на неё и регулярно прогонять его при последующих изменениях программы. Хотя регрессионное тестирование может быть выполнено и вручную, но чаще всего это делается с помощью специализированных программ, позволяющих выполнять все регрессионные тесты автоматически. В некоторых проектах даже используются инструменты для автоматического прогона регрессионных тестов через заданный интервал времени. Обычно это выполняется после каждой удачной компиляции (в небольших проектах) либо каждую ночь или каждую неделю.

Данный подход используется в связке с модульным-тестированием, поскольку регрессионное тестирование может быть использовано не только для проверки корректности программы, часто оно также используется для оценки качества полученного результата. В данном случае проект представляет собой набор структурируемых тестов.

**6** РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

**6.1** Требования к аппаратному и программному обеспечению

Минимальные системные требования для нормальной работы программы:

* процессор Intel Pentium 4 или старше;
* 2 гигобайта оперативного запоминающего устройства (ОЗУ).

Рекомендованными операционными системами являются Windows 7 или новее. На рабочей станции должна быть установлена .NET Framework версии 4.5 или выше соответствующей разрядности. Для развертывания ASP.NET приложения нужен сервер приложений. Рекомендуется использовать сервер приложений IIS версии, соответствующей версии используемой виртуальной машины CLI.

Для работы с приложением необходимо установить браузер. Для корректной работы приложения рекомендуется использовать следующие браузеры:

* Mozilla Firefox версии 16 или старше;
* Google Chrome версии 17 или старше;
* Internet Explorer версии 8 или старше.

**6.2** Установка и настройка сервера приложений IIS

По умолчанию IIS не устанавливается в выпусках Windows. Чтобы установить службу IIS, щелкните **Компоненты Windows** в дополнительных параметрах, которые расположены в «Программах» на панели управления.

Использование пользовательского интерфейса:

1. Нажмите кнопку Пуск и выберите Панель управления.
2. На панели управления выберите Программы, а затем Включение и отключение компонентов Windows.
3. В диалоговом окне «Компоненты Windows» нажмите Службы IIS, а затем кнопку ОК.

Компонент IIS включён как часть установки Windows (как для сервера, так и для рабочей станции) и требует активизации и конфигурирования непосредственно пользователем.

После выполнения данных инструкции на экране появится диалоговое окно с выбором компонентов, доступных для установки (см. рисунок 6.1).

Следующим шагом идет непосредственная конфигурация самого сервера приложений. Для этого требуется запустить Internet Information Servies (IIS) Manager. Для развёртывания веб-приложения нужно создать и сконфигурировать application pool и web-site. Даём Application Pool (пул приложения) имя TotalizatorPool.

Версию для .NET Framework следует выбрать 4 и выше. Модель управления контейнером следует выбрать Integrated mode (предполагает более тесное взаимодействие между IIS и ASP.NET, позволяет избежать дублирования функций и получить больший контроль над обработкой запроса).

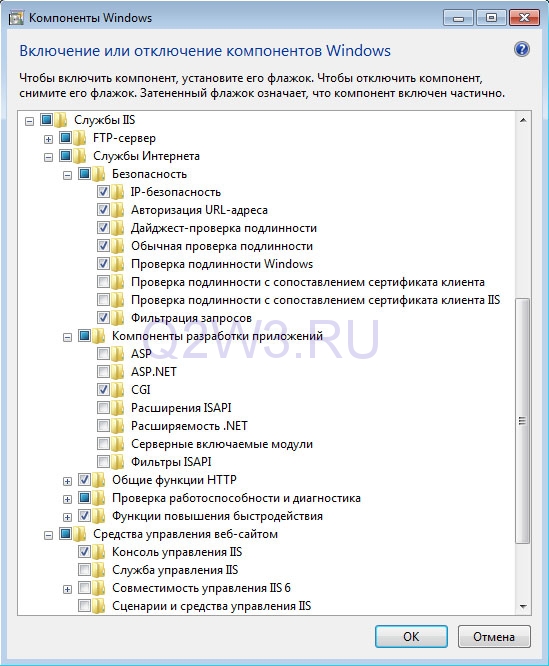


Рисунок 6.1 – Выбор нужных компонентов

Создавая веб-сайт следует выбрать уже созданный пул приложения, Имя хоста определяем в качестве totalizator.com. Пример конфигурации веб-приложения приведен на рисунке 6.2.

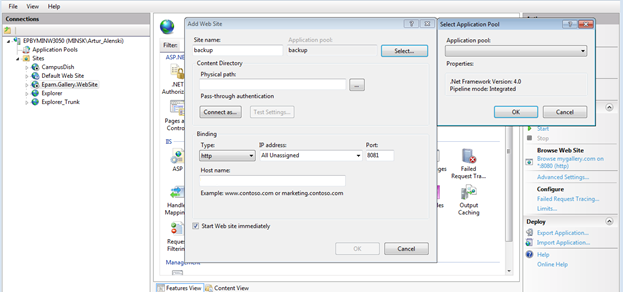


Рисунок 6.2 – Конфигурация веб-сайта

Стоит отметить, что значением по умолчанию для порта коннектора протокола HTTP является значение 8080. Этот порт используется многими приложениями. Чтобы избежать возможных конфликтов, рекомендуется изменить это значение. Так же рекомендуется создать учетную запись администратора сервера приложений.

После успешной установки и конфигурации сервера приложений нужно поместить архив, содержащий веб-приложение, в папку webapps сервера и перезапустить его.

**6.3** Установка SQL Server 2008

Использование пользовательского интерфейса:

* 1. Запустить программу-установщик с правами администратора на данном компьютере (см. рисунок 6.3).
  2. В разделе «Планирование» нажать пункт «Средство проверки конфигурации».
  3. Нажать на раздел «Установка» и затем пункт «Новая установка изолированного SQL Server или добавление компонентов …».
  4. Прочитать лицензию, установить галочку «Я принимаю условия…» и нажать кнопку «Далее».
  5. Нажать кнопку «Установить».

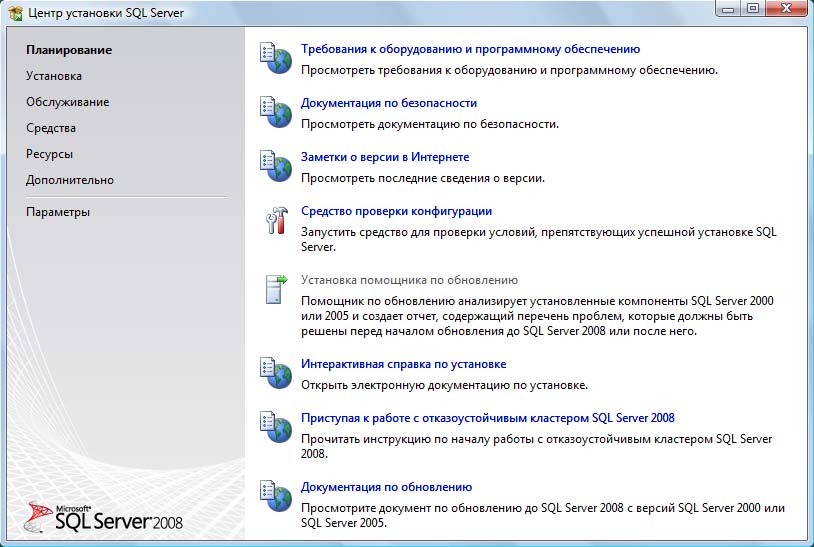


Рисунок 6.3 – Установка SQL Server

**6.4** Графический интерфейс пользователя

**6.4.2** Регистрация

В системе существует три типа пользователей: администратор, пользователь и привилегированный пользователь. Для администратора создана учетная запись с логином admin@gmail.com и паролем xX-123. При регистрации привилегированного пользователя необходимо в поле «Код подтверждения особых прав» ввести набор символов xX-123. На рисунке 6.4 представлена форма регистрации.

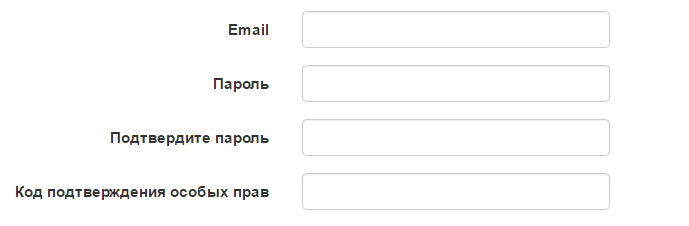


Рисунок 6.4 – Форма регистрации

**6.4.2** Главное меню

Главное меню представлено на рисунке 6.5. Данный вид меню будет доступен только администратору, пользователям будут видны только элементы «Прошедшие игры» и «События», а так же общие для всех элементы «SuperTotal» и «Корзина ставок».

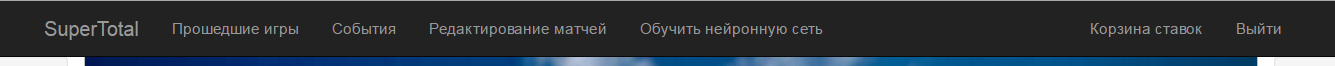


Рисунок 6.5 – Главное меню

**6.4.3** Редактирование матчей

Режим редактирования существующих и добавления новых матчей доступен только администратору по клику на элемент главного меню «Редактирование матчей». Результат матча вводится цифрами от нуля до трех, что соответствует ничьей, победе первой команды, победе второй команды и неопределенности результата соответственно.

У существующего матча возможно только изменить результат, для чего предназначено соответствующее диалоговое окно. При создании нового матча необходимо ввести его результат и создать обе участвующие в нем команды. На рисунке 6.6 представлена форма создания матча, на рисунке 6.7 – форма создания команды.

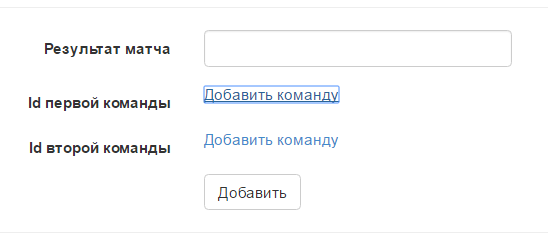


Рисунок 6.6 – Форма создания матча

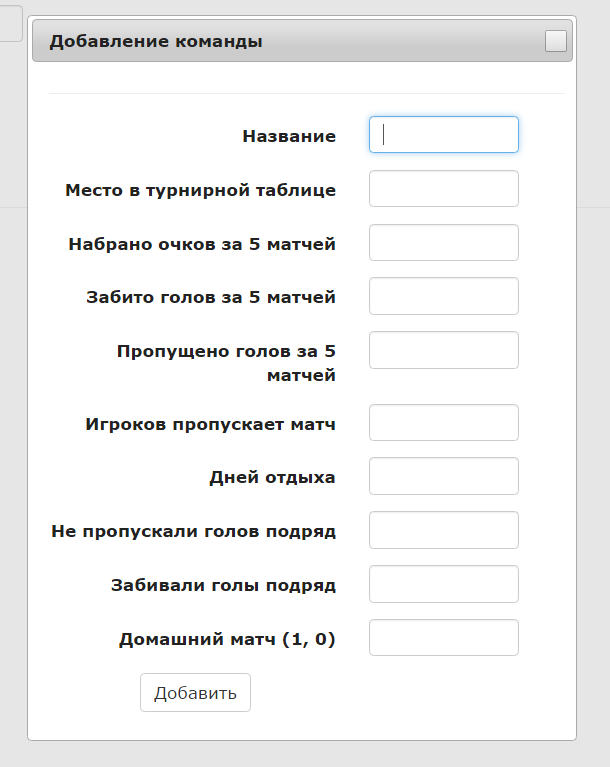


Рисунок 6.7 – Форма создания команды

**6.4.4** Ставки

Совершение ставок доступно всем пользователям по коэффициентам, откорректированным в сторону прибыли букмекера. Привилегированным пользователям также доступен просмотр реальных коэффициентов. Для совершения ставки необходимо нажать на соответствующий коэффициент и ввести в появившемся окне сумму ставки. На рисунке 6.8 представлен вид коэффициентов привилегированного пользователя, на рисунке 6.9 – окно совершения ставки.

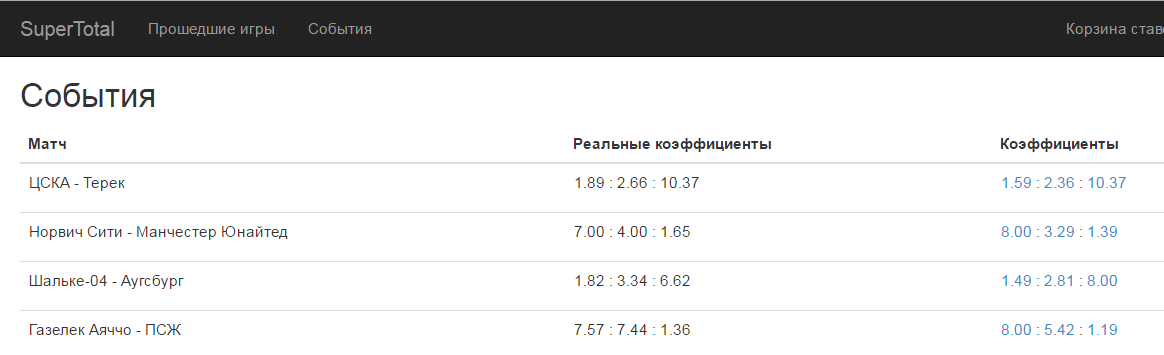


Рисунок 6.8 – Вид коэффициентов привилегированного пользователя

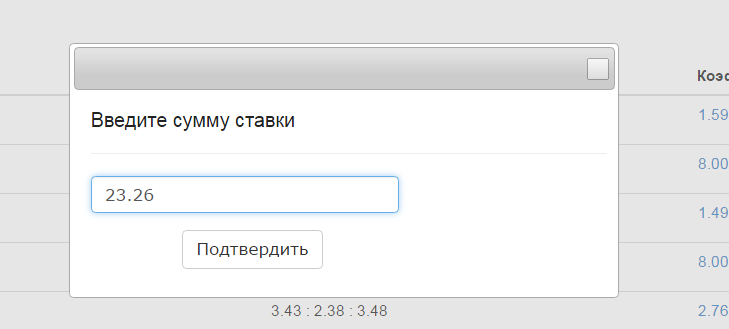


Рисунок 6.9 – Окно совершения ставки

Каждый пользователь также может просмотреть свою корзину ставок, в котороый отображается статус и результат всех его ставок.

1. ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ПРОЕКТА
   1. Описание проекта

Программная модель тотализатора игровых видов спорта выполняет несколько связанных задач. Основной из них является предсказание результата игры, вспомогательными – пользовательский интерфейс.  
 Цель проекта – создание программной модели тотализатора игровых видов спорта с возможностью: вычисления вероятностей исходов ставок, расчет реальных коэффициентов, расчет выгодных букмекеру коэффициентов, редактирования и добавления матчей в базу данных, совершения ставок, регистрацией и авторизацией.

Администратор данной системы, может самостоятельно и легко вносить коррективы в базу данных матчей.   
Предлагаемый электронный ресурс будет производить расчет коэффициентов на матчи в режиме онлайн и выдавать информацию в виде, понятном пользователю.

Сайт тотализатора предполагает удобное построение элементов управления, создающих интуитивно понятный пользовательский интерфейс. Математическая составляющая программы позволяет при определенном уровне опыта пользователя в данной сфере получать прибыль от ставок на спортивные матчи.

* 1. Расчёт сметы затрат и цены ПО

Разрабатываемый программный продукт относится к третьей категории сложности, поскольку не относится к ПО, планирующемуся к использованию под нагрузкой, либо в различных наукоёмких или требующих оптимизации системах. Программный продукт является ПО общего назначения и относится к категории новизны В (Кн = 0,7).

При расчете сметы затрат будут использоваться данные, приведенные в таблице 7.1. Они отражают текущую финансовую ситуацию.

Таблица 7.1 – Исходные данные для расчета

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование показателей | Буквенные обозначения | Единицы измерения | Количество |
| 1 | 2 | 3 | 4 |
| Группа сложности | - | - | 3 |
| Коэффициент новизны | Кн | - | 0,7 |
| Коэффициент, использования стандартных модулей | Кт | - | 0,8 |

*Продолжение таблицы 7.1*

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Коэффициент, использования стандартных модулей | Кт | - | 0,8 |
| Дополнительный коэффициент сложности | Ксд | - | 0,06 |
| Установленная плановая продолжительность разработки | Тр | лет |  |
| Количество дней в году | Дг | дней | 365 |
| Количество праздничных дней | Дп | дней | 6 |
| Количество выходных дней | Дв | дней | 104 |
| Количество дней отпуска | До | дней | 24 |
| Тарифная ставка 1-го разряда | Тм1 | руб. | 700 000 |
| Продолжительность рабочего дня | Тч | часов | 8 |
| Установленный фонд рабочего времени | Фрв | часов | 170 |
| Норматив дополнительной заработной платы | Нд | % | 20 |
| Ставка отчислений в ФСЗН | Нсз | % | 34 |
| Ставка отчислений на обязательное социальное страхование | Нсс | % | 0,6 |
| Норма расхода материалов от основной заработной платы | Нмз | % | 3 |
| Цена одного машинного часа | Цм | руб. | 4500 |
| Норматив расхода машинного времени | Нмв | ч. / 100 строк кода | 12 |
| Норматив расходов на командировки | Нрнк | % | 15 |
| Норматив прочих затрат | Нпз | % | 20 |
| Норматив накладных расходов | Нрн | % | 50 |
| Уровень рентабельности | Урп | % | 20 |
| Ставка налога на добавленную стоимость | Ндс | % | 20 |
| Норматив расходов на освоение | Но | % | 10 |
| Норматив расходов на сопровождение | Нс | % | 20 |
| Ставка налога на прибыль | Нп | % | 18 |

Отправной точкой для расчёта плановой сметы затрат на разработку ПО, требуется определить общий объем программного продукта (*V*о). В качестве единицы измерения примем количество строк исходного кода (Lines of Code, LOC). Прогнозируемый общий объём ПО определяется по каталогу функций. Каталог функций данного программного продукта представлен в таблице 7.2.

Таблица 7.2 – Каталог функций ПО

|  |  |  |
| --- | --- | --- |
| Код функции | Наименование (содержание) функции | Объем функций (LOC) |
| 101 | Организация ввода информации | 150 |
| 102 | Контроль, предварительная обработка информации | 450 |
| 109 | Организация ввода/вывода информации в интерактивном режиме | 320 |
| 207 | Генерация структуры базы данных | 4300 |
| 208 | Организация поиска и поиск в базе данных | 5480 |
| 506 | Обработка ошибочных и сбойных ситуаций | 410 |
| 507 | Обеспечение интерфейса между компонентами | 970 |
| 703 | Расчет показателей | 460 |
| 707 | Графический вывод результатов | 480 |
| – | Общий объём (*VО*) | 13020 |

На основе общего объёма и категории сложности ПО определяется нормативная трудоёмкость, которая, в данном случае, для *V*о = 13020 и третьей категории сложности, составит Тн = 278 человеко-дней.

Наличие интерактивного интерфейса позволяет применить к объёму ПО коэффициент Кс, который определяется по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (7.1) |

где К*i* – коэффициент, соответствующий степени повышения сложности ПО за счет конкретной характеристики;

*n* – количество, учитываемых характеристик.

|  |  |  |
| --- | --- | --- |
|  | . |  |

Исходя из нормативной трудоёмкости можно определить общую трудоёмкость То по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (7.2) |

где Кс – дополнительный коэффициент сложности;

Кт – коэффициент использования типовых программ и модулей;

Кн – коэффициент новизны.

Подставив значения в формулу (7.2), получим:

|  |  |  |
| --- | --- | --- |
|  | . |  |

Имея общую трудоёмкость, определяется численность исполнителей проекта, либо срок его разработки. Данный проект делался на заказ, при этом заранее было определено, что работа будет выполнена одним человеком, таким образом требуется определить второй параметр.

Для определения срока разработки проекта, необходимо рассчитать эффективный фонд времени одного работника (Фэф):

|  |  |  |
| --- | --- | --- |
|  |  | (7.3) |

где Дг – количество дней в году;

Дп – количество праздничных дней в году;

Дв ­– количество выходных дней в году;

До – количество дней отпуска.

Таким образом, по формуле (7.3) фонд эффективного времени составит:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Срок разработки проекта (Тр) определяется по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (7.4) |

где Чр – численность исполнителей проекта;

То – общая трудоемкость разработки проекта, человеко-дней;

Фэф – эффективный фонд времени работы одного работника.

Подставив значения в формулу (7.4), получим:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Основой для расчёта сметы затрат является основная заработная плата разработчиков проекта. В данном случае имеется один работник – инженер-программист II-й категории (тарифный разряд – 12, тарифный коэффициент – 2,84). Месячная тарифная ставка исполнителя (Тм) определяется по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (7.5) |

где Тм1 – месячная тарифная ставка первого разряда, руб.;

Тк – тарифный коэффициент.

Месячная тарифная ставка, определённая по формуле (7.5) составит:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Исходя из месячной тарифной ставки рассчитывается часовая тарифная ставка (Тч):

|  |  |  |
| --- | --- | --- |
|  |  | (7.6) |

где Фр – среднемесячная норма рабочего времени, ч.

При подстановке значений в формулу (7.6), получим:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Основная заработная плата исполнителей рассчитывается по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (7.7) |
|  |  |  |

где Тч*i* – часовая тарифная ставка *i*-го исполнителя, руб.;

Тч – количество часов работы в день, ч;

Фп – плановый фонд рабочего времени *i*-го исполнителя;

К – коэффициент премирования, принятый равным 1,4.

Учитывая число разработчиков *n* = 1, определим основную заработную плату по формуле (7.7):

|  |  |  |
| --- | --- | --- |
|  |  |  |

Дополнительная заработная плата (Зд*i*) включает в себя оплаты отпусков и другие выплаты, предусмотренные законодательством, и определяется по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (7.8) |
|  |  |  |

где Нд – норматив дополнительной заработной платы (15%).

Тогда получим:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Отчисления в фонды социальной защиты и социального страхования определяются по следующим формулам:

|  |  |  |
| --- | --- | --- |
|  |  | (7.9) |
|  |  |  |
|  |  | (7.10) |

где Нсз – норматив отчислений в фонд социальной защиты населения (34%);

Нсс – норматив отчислений в фонд социального страхования (0,6%).

По формулам (7.9) и (7.10) получим:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

Расходы по статье «Материалы» отражают расходы на бумагу, тонер и прочие вещи, необходимые для разработки ПО. Сумма затрат на расходные материалы определяется по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (7.11) |

где Нмз ­– норма расхода материалов от основной заработной платы (3%).

Подставив значения в формулу (7.11), получим:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Расходы по статье «Машинное время» включает оплату машинного времени, необходимого для разработки и отладки ПО и определяются по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (7.12) |

где Цм*i* – цена одного машино-часа, руб.;

*V*о*i* – общий объем ПО (строк исходного кода);

Нмв – норматив расхода машинного времени на отладку 100 строк исходного кода, машино-часов.

В современных условиях разработки используется понижающий коэффициент 0,4. Подставляя значения в формулу (7.12), с учётом понижающего коэффициента получим:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Расходы по статье «Научные командировки» определяются по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (7.13) |

где Нрнк – норматив расходов на командировки по организации (15%).

|  |  |  |
| --- | --- | --- |
|  |  |  |

Расходы по статье «Прочие затраты» включают затраты на приобретение и подготовку специальной научно-технической информации и специальной литературы. Определяются по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (7.14) |

где Нпз – норматив прочих затрат в целом по организации (20%).

|  |  |  |
| --- | --- | --- |
|  |  |  |

Затраты по статье «Накладные расходы» связаны с необходимостью содержания аппарата управления, вспомогательных хозяйств и опытных производств. Определяются по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (7.15) |

где Нрн – норматив накладных расходов в целом по организации.

|  |  |  |
| --- | --- | --- |
|  |  |  |

Общая сумма расходов по смете (Сп*i*) определяется как сумма выше рассчитанных показателей:

|  |  |  |
| --- | --- | --- |
|  |  | (7.16) |

Подставив рассчитанные ранее значения в формулу (7.16), получим:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Прогнозируемая прибыль от создаваемого ПО определяется как:

|  |  |  |
| --- | --- | --- |
|  |  | (7.17) |

где Сп*i* – себестоимость ПО, руб.;

Урп*i* – уровень рентабельности ПО (20%).

|  |  |  |
| --- | --- | --- |
|  |  |  |

На основе прогнозируемой прибыли определяется прогнозируемая цена ПО без налогов:

|  |  |  |
| --- | --- | --- |
|  |  | (7.18) |

Подставляя значения в формулу (7.18), получим:

|  |  |  |
| --- | --- | --- |
|  |  |  |

При расчёте отпускной цены дополнительно учитывается налог на добавочную стоимость (НДС):

|  |  |  |
| --- | --- | --- |
|  |  | (7.19) |

где Ндс – норматив налога на добавленную стоимость (20%).

По формуле (7.19) налог на добавочную стоимость равен:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Таким образом, с учётом НДС отпускная цена рассчитывается как:

|  |  |  |
| --- | --- | --- |
|  |  | (7.20) |

Подставив значения в формулу (7.20), получим:

|  |  |  |
| --- | --- | --- |
|  |  |  |

В дополнение к выше рассчитанным параметрам, определяются расходы на освоение (Ро*i*) и сопровождение (Рс*i*) ПО:

|  |  |  |
| --- | --- | --- |
|  |  | (7.21) |
|  |  |  |
|  |  | (7.22) |

где Но – норматив расходов на освоение ПО (10%);

Нс – норматив расходов на сопровождение ПО (20%).

Используя формулы (7.21) и (7.22), определим значения расходов:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

Все выше рассчитанные параметры, а также результирующие показатели сведены в таблицу 7.3.

Таблица 7.3 – Расчёт себестоимости и отпускной цены

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование статьи | Норматив | Формула расчёта | Значение, руб. |
| 1 | 2 | 3 | 4 |
| Основная ЗП | – |  | 30 255 021 |
| Дополнительная ЗП | Нд = 20% |  | 4 538 253 |

*Продолжение таблицы 7.3*

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Отчисления в фонд соцзащиты | Нсз = 34% |  | 11 829 713 |
| Отчисления в фонд соцстраха | Нсс = 0,6% |  | 208 760 |
| Машинное время | Нмв = 12 ч |  | 2 812 320 |
| Научные командировки | Ннк = 15% |  | 4 538 253 |
| Прочие затраты | Нпз = 20% |  | 6 051 004 |
| Накладные расходы | Нрн = 50% |  | 15 127 511 |
| Общая сумма по смете | – |  | 91 522 183 |
| Прогнозируемая прибыль | Урп = 20% |  | 15 253 697 |
| Прогнозируемая цена без налогов | – |  | 91 522 183 |
| НДС | Ндс = 20% |  | 18 304 437 |
| Отпускная цена | – |  | 109 826 620 |
| Освоение ПО | Но = 10% |  | 7 626 849 |
| Сопровождение ПО | Нс = 20% |  | 15 253 697 |

Учитывая налог на прибыль, можно рассчитать итоговую сумму, которая останется разработчику и будет является его экономическим эффектом:

|  |  |  |
| --- | --- | --- |
|  |  | (7.23) |

где ΔПч – чистая прибыль;

Ппс – прогнозируемая прибыль;

Нп – норматив налога на прибыль (18%).

Подставив значения в формулу (7.23), определим чистую прибыль:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Чистая прибыль от реализации ПО (ΔПч = 12 508 032 рублей) остается организации-разработчику и представляет собой экономический эффект от создания нового программного средства.

**7.3** Расчёт экономического эффекта от применения программного средства у пользователя (заказчика)

Для определения экономического эффекта от использования нового ПО у потребителя необходимо сравнить расходы по всем основным статьям сметы затрат на эксплуатацию нового ПО (расходы на заработную плату с начислениями, материалы, машинное время) с расходами по соответствующим статьям при использовании прежнего варианта ПО. При сравнении базового и нового вариантов ПО в качестве экономического эффекта будет выступать общая экономия всех видов ресурсов относительно базового варианта. При этом создание нового ПО окажется экономически целесообразным лишь в том случае, если все капитальные затраты окупятся за счет получаемой экономии в ближайшие 2–3 года.

Таблица 7.4 – Исходные данные для определения экономического эффекта

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование показателей | Обо­значе­ние | Единицы из­мерения | Значение показателя | |
| в базовом варианте | в новом варианте |
| 1 | 2 | 3 | 4 | 5 |
| Капитальные вложения, включая затраты поль­зователя на приобрете­ние ПO | Кпр | руб. | - | 109 826 620 |
| Затраты на освоение ПО | Кос | руб. | - | 7 626 849 |
| Затраты на сопровожде­ние ПО | Кс | руб. | - | 15 253 697 |
| Затраты на укомплекто­вание ВТ техническими средствами в связи с внедрением нового ПО | Ктс | руб. | - | 7 626 849 |
| Затраты на пополнение оборотных средств в связи с эксплуатацией нового ПО | Коб | руб. | - | 2 500 000 |

*Продолжение таблицы 7.4*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| Среднемесячная ЗП од­ного программиста | Зсм | руб. | 2 800 000 | 2 800 000 |
| Коэффициент начисле­ний на зарплату | Кнз |  | 1,5 | 1,5 |
| Среднемесячное коли­чество рабочих дней | Др | день | 21,5 | 21,5 |
| Количество типовых за­дач, решаемых за год | Зт1, Зт2 | задача | 1 300 | 1 300 |
| Объем выполняемых работ за год | А1, А2 | задача | 1 300 | 1 300 |
| Средняя трудоемкость работ | Тс1, Тс2 | чел.-час на задачу | 4 | 0,5 |
| Средний расход машин­ного времени | Мв1, Мв2 | маш.-час на задачу | 4 | 0,5 |
| Цена 1-го машино-часа работы ЭВМ | Цм | руб. | 4 500 | 4 500 |
| Количество часов ра­боты в день | Тч | ч | 8 | 8 |
| Ставка налога на при­быль | Нп | % | 18 | 18 |

Общие капитальные вложения заказчика (потребителя) рассчитываются по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (7.24) |

где Кпр – затраты пользователя на приобретение ПО по отпускной цене у разработчика с учетом стоимости услуг по эксплуатации, руб.;

Кос – затраты пользователя на освоение ПС, руб.;

Кс – затраты пользователя на оплату услуг по сопровождению ПО, руб.;

Ктс – затраты на доукомплектование ВТ техническими средствами в связи с внедрением нового ПО, руб.;

Коб – затраты на пополнение оборотных средств в связи с использованием нового ПО, руб.

|  |  |  |
| --- | --- | --- |
|  |  |  |

Экономия затрат на заработную плату (Сз) при использовании нового ПО в расчете на объем выполненных работ определяется по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (7.25) |
|  |  |  |

где Сзе – экономия затрат на заработную плату при решении задач c использованием нового ПО на 1 задачу, руб.;

А2 – объем выполненных работ с использованием нового ПО (задач).

Экономия затрат на заработную плату в расчете на 1 задачу рассчитывается по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (7.26) |

где Зсм – среднемесячная заработная плата одного программиста, руб.;

Тс1, Тс2 – трудоемкости работ в расчете на 1 задачу, человеко-часов;

Тч – количество часов работы в день, ч;

Др – среднемесячное количество рабочих дней.

|  |  |  |
| --- | --- | --- |
|  |  |  |

Тогда, подставляя значения в формулу (7.25), рассчитаем экономию затрат на заработную плату:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Экономия с учетом начисления на зарплату вычисляется по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (7.27) |

где Кнз – коэффициент начислений на зарплату (1,5).

|  |  |  |
| --- | --- | --- |
|  |  |  |

Экономия затрат на оплату машинного времени (См) в расчете на выполненный объем работ в результате применения нового ПО:

|  |  |  |
| --- | --- | --- |
|  |  | (7.28) |

где Сме – экономия затрат на оплату машинного времени в расчёте на 1 задачу с использованием нового ПО.

Экономия затрат на оплату машинного времени в расчете на 1 задачу определяется по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (7.29) |
|  |  |  |

где Цм – цена одного машино-часа работы ЭВМ;

Мв1, Мв2 – средний расход машинного времени при применении соответственно базового и нового ПО.

|  |  |  |
| --- | --- | --- |
|  |  |  |

Таким образом, по формуле (7.28) определим экономию затрат на оплату машинного времени в расчете на выполненный объем работ:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Общая годовая экономия текущих затрат, связанных с использованием нового ПО является важным фактором, влияющим в дальнейшем на расчеты, производимые в данном разделе и вычисляется по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (7.30) |

Подставляя ранее полученные значения в формулу (7.30), получим:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Внедрение нового ПО позволит заказчику сэкономить на текущих затратах 131 579 651 рубля, то есть практически получить на эту сумму дополнительную прибыль. Для заказчика в качестве экономического эффекта выступает лишь чистая прибыль – дополнительная прибыль, остающаяся в его распоряжении, которая определяется по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (7.31) |

где Нп – ставка налога на прибыль (18%).

|  |  |  |
| --- | --- | --- |
|  |  |  |

В процессе использования нового ПО чистая прибыль в конечном итоге возмещает капитальные затраты. Однако полученные при этом суммы результатов (прибыли) и затрат (капитальных вложений) по годам приводят к единому времени – расчетному году (за расчетный год принят 2016 год) путем умножения результатов и затрат за каждый год на коэффициент дисконтирования α*t*, который рассчитывается по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (7.32) |

где *E* – норматив приведения разновременных затрат и результатов (c учётом безрисковой ставки процента по валютным депозитам, уровня инфляции, роста спроса и стабильности дохода примем *E* = 24%);

*t* – номер года, результаты и затраты которого приводятся к расчётному (2016 год – 1, 2017 год – 2, 2018 год – 3, 2019 год – 4);

*tp* – номер расчётного года (2016).

Таким образом, получим следующие значения коэффициентов дисконтирования:

2016 год: 

2017 год: 

2018 год: 

2019 год: 

Сведем данные расчета экономического эффекта в таблицу 7.5.

Таблица 7.5 – Расчет экономического эффекта от использования нового ПС

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Показатели | Ед. изм. | Годы | | | | | |
| 2016 | | 2017 | 2018 | 2019 | |
| 1 | 2 | 3 | | 4 | 5 | 6 | |
| *Результаты:* | | | | | | | |
| Прирост прибыли за счет экономии затрат (Пч) | руб. |  | 107 895 314 | | 107 895 314 | | 107 895 314 |
| То же с учетом фактора времени | руб. |  | 87 012 350 | | 70 171 250 | | 56 589 718 |
| *Затраты:* | | | | | | | |
| Приобретение ПО (Кпр) | руб. | 109 826 620 |  | |  | |  |
| Освоение ПО (Кос) | руб. | 7 626 849 |  | |  | |  |
| Сопровождение (Кс) | руб. | 15 253 697 |  | |  | |  |
| Доукомплектование ВТ техническими средствами (Ктс) | руб. | 7 626 849 |  | |  | |  |

*Продолжение таблицы 7.5*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| Пополнение оборотных средств (Коб) | руб. | 2 500 000 |  |  |  |
| То же с учетом фактора времени | руб. | 142 834 014 |  |  |  |
| *Экономический эффект:* | | | | | |
| Превышение результатов над затратами | руб. | -142 834 014 | 87 012 350 | 70 171 250 | 56 589 718 |
| То же нарастающим итогом | руб. | -142 834 014 | -55 821 664 | 14 349 586 | 70 939 304 |
| Коэффициент приведения | ед. | 1 | 0,8065 | 0,6504 | 0,5245 |

* 1. Выводы

В данном разделе была рассчитана себестоимость проекта, на основании которой была сформирована рыночная цена. Был рассчитан экономический эффект для разработчика: чистая прибыль от реализации ПО составила 12 508 032 рублей.

Кроме этого, рассчитана экономическая эффективность для заказчика, выраженная в чистом дисконтированном доходе, составившем 70 939 304 рублей. Стоит отметить, что затраты заказчика окупятся менее, чем за два года. Данный эффект получается в результате снижения трудоемкости решения задач, снижения затрат машинного времени. Кроме этого, дополнительный положительный эффект может создаться за счёт пересмотра кадрового состава на основе анализа производимой конкретными сотрудниками работы, учитывая новые условия труда.

Таким образом, можно сделать вывод, что проект является полезным и экономически эффективным для заказчика.

## ЗАКЛЮЧЕНИЕ

В результате выполнения дипломного проекта были достигнуты все поставленные цели:

* подготовлен весь заявленный функционал;
* разработанный модуль экономически эффективен.

В результате было создано клиент серверное приложение, не имеющее полноценных аналогов. Отличительной особенностью разработанного программного модуля от потенциальных аналогов является применение трех алгоритмов предсказания результата события и возможность в любой момент времени перенастроить их без изменения кода программы. В результате разработки серверного приложения был разработано веб-приложение с удобным пользовательским интерфейсом (см. плакат ГУИР.400201.075 ПЛ.2).

Данное приложение позволяет просматривать информацию обо всех произошедших ранее и будущих матчах в отдельных вкладках, что является очень удобным для пользователя при принятии решения об исходе матча самостоятельно. Также пользователи, заплатившие при регистрации, могут просматривать не откорректированные коэффициенты на игру, которые являются более правдивыми.

Стоит отметить легкость и предсказуемость при работе с данным модулем. Это значит, что была выдержана общая концепция интерфейсов.

Реализованная архитектура приложения позволяет в качестве усовершенствования добавление новой логики. Также стоит отметить, что все модули данного приложения очень легко подвергаются модульному и регрессионному тестированию, что также благоприятно сказывается на дальнейшей поддержке и дальнейшему развитию и усовершенствованию данного приложения.