МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

|  |  |  |
| --- | --- | --- |
| Специальность | 1-40 05 01 | «Информационные системы и технологии» |
| Специализация | 1-40 05 01-03 | «Информационные системы и технологии (издательско-полиграфический комплекс)» |

**Пояснительная записка   
к дипломному проекту**

«Веб-приложение для ставок на футбольные события»

|  |  |  |
| --- | --- | --- |
| Дипломник |  | М.А. Гой |
|  |  |  |
| Руководитель работы |  | Комарова Е.И., ассистент |
|  |  |  |
| Заведующий кафедрой |  | Смелов В. В., к. т. н., доцент |
|  |  |  |
| Консультант |  | Семёнова Л.С., ассистент |
|  |  |  |
| Нормоконтроллер |  | Николайчук А.Н., ассистент |

|  |  |
| --- | --- |
| Дипломный проект защищен с оценкой |  |

|  |  |  |
| --- | --- | --- |
| Председатель ГЭК |  | В. К. Дюбков, к. т. н., доцент |

Минск 2023

Лист задания 1

Лист задания 2

Содержание

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*2*

*БГТУ 00.00.ПЗ*

Разраб*.*

*Иванова А.А.*

Провер*.*

*.*

*Блинова Е.А*

Н*.* контр*.*

*Николайчук А.Н.*

Утв*.*

*СмеловВ.В.*

Содержание

*Лит.*

*Листов*

*2*

11111111, 2023

[Содержание 5](#_Toc136390055)

[Введение 7](#_Toc136390056)

[1 Аналитический обзор аналогов и постановка задачи 8](#_Toc136390057)

[1.1 Обзор аналогичных решений 8](#_Toc136390058)

[1.2 Приложение «Betera» 8](#_Toc136390059)

[1.3 Приложение «Fonbet» 11](#_Toc136390060)

[1.4 Приложение «MaxLine» 14](#_Toc136390061)

[1.5 Постановка задачи 16](#_Toc136390062)

[1.6 Вывод по разделу 17](#_Toc136390063)

[2 Проектирование веб-приложения 18](#_Toc136390064)

[2.1 Проектирование структурной схемы веб-приложения 18](#_Toc136390065)

[2.2 Диаграмма вариантов использования 19](#_Toc136390066)

[2.3 Выбор средств реализации 19](#_Toc136390067)

[2.3.1 Основные языки программирования 19](#_Toc136390068)

[2.3.2 Фреймворки 20](#_Toc136390069)

[2.3.3 Система управления базами данных 20](#_Toc136390070)

[2.4 Логическая схема базы данных 20](#_Toc136390071)

[2.5 Вывод по разделу 25](#_Toc136390072)

[3 Реализация веб-приложения 26](#_Toc136390073)

[3.1 Разработка серверной части 26](#_Toc136390074)

[3.1.1 Разработка репозиториев 28](#_Toc136390075)

[3.1.2 Разработка сервисов 30](#_Toc136390076)

[3.1.3 Разработка контроллеров 31](#_Toc136390077)

[3.2 Разработка клиентской части 33](#_Toc136390078)

[3.3 Вывод по разделу 39](#_Toc136390079)

[4 Тестирование веб-приложения 40](#_Toc136390080)

[4.1 Ручное тестирование 40](#_Toc136390081)

[4.2 Тестирование валидации 42](#_Toc136390082)

[4.3 Модульное тестирование 45](#_Toc136390083)

[4.4 Вывод по разделу 47](#_Toc136390084)

[5 Руководство программиста 48](#_Toc136390085)

[6 Технико-экономическое обоснование проекта 49](#_Toc136390086)

[6.1 Общая характеристика разрабатываемого программного средства 49](#_Toc136390087)

[6.2 Исходные данные для проведения расчётов и маркетинговый анализ 49](#_Toc136390088)

[6.3 Обоснование цены программного средства 51](#_Toc136390089)

[6.3.1 Расчёт затрат рабочего времени на разработку программного средства 51](#_Toc136390090)

[6.3.2 Расчет основной заработной платы 52](#_Toc136390091)

[6.3.3 Расчет дополнительной заработной платы 52](#_Toc136390092)

[6.3.4 Расчет отчислений в Фонд социальной защиты населения и по обязательному страхованию 53](#_Toc136390093)

[6.3.5 Расчет суммы прочих прямых затрат 53](#_Toc136390094)

[6.3.6 Расчет суммы накладных расходов 54](#_Toc136390095)

[6.3.7 Сумма расходов на разработку программного средства 54](#_Toc136390096)

[6.3.8 Определение цены, оценка эффективности 54](#_Toc136390097)

[6.4 Вывод по разделу 55](#_Toc136390098)

[Заключение 56](#_Toc136390099)

[Приложение А 58](#_Toc136390100)

[Приложение Б 59](#_Toc136390101)

[Приложение В 60](#_Toc136390102)

Введение

Ставки на футбол – это один из самых популярных видов спортивных ставок в мире. Этот вид спорта обладает огромной популярностью, как среди профессионалов, так и среди любителей. Каждый год все больше людей начинают интересоваться футболом и делать ставки на результаты матчей. Однако, чтобы стать успешным игроком, нужно знать основные правила и стратегии, следить за новостями и анализировать статистику. Ставки на футбол могут принести не только увлекательный опыт, но и дополнительный доход. В этой области есть много возможностей для тех, кто хочет получить удовольствие от игры и испытать свою удачу. Перед матчем букмекеры оценивают шансы команд на их выигрыш и исходя из этого рассчитывают коэффициент. Далее игрок делают ставку по своему желанию. Если ставка срабатывает игрок выигрывает, если нет, то букмекер получает деньги поставил игрок. Для игрока предлагается широкий выбор ставок, на которой он сможет испытать свою удачу.

Задачей приложения создание такого продукта, в котором игроки смогут быстро и удобно пополнять свой личный счет, ставить на футбольное событие, а также отслеживать результат. Букмекерам также отведена особая роль, в которой он сможет эффективно оценить шансы команд на успех и рассчитать шанс любого события.

Целью дипломного проекта является создание веб-приложения, помочь игрокам получать максимальную прибыль и удовольствие от игры, предоставляя качественную информацию о командах, матчах, коэффициентах и стратегиях ставок.

Для достижения цели проекта сформулированы следующие задачи:

* определить обязательный функционал приложения;
* согласовать требования
* выбрать необходимые инструменты для разработки
* спроектировать приложение;
* разработать приложение, придерживаясь требований
* протестировать приложение;
* убрать найденные ошибки;
* составить руководство пользователя.

1 Аналитический обзор аналогов и постановка задачи

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*25*

*БГТУ 03.00.ПЗ*

Разраб*.*

*Гой М.А.*

Провер*.*

*.*

*Комарова Е.И.*

Н*.* контр*.*

*Николайчук А.Н.*

Утв*.*

*Смелов В.В.*

3 Разработка программного средства

*Лит.*

*Листов*

*7421877-C, 2023*

Перед тем как начать разрабатывать веб-приложение важно четко понимать, что именно должно получится в итоге. Поэтому для начала нужно выполнить несколько этапов: изучить, насколько актуальна данная задач, которую проект должен решать и проанализировать уже существующие аналоги, представляющие собой веб-приложение для букмекерских ставок на футбольные события.

1.1 Обзор аналогичных решений

На сегодняшний день доступно множество веб-приложений, которые принимают ставки на футбольные события. Рассмотрим некоторые из них и выявим сильные и слабые стороны.

1.2 Приложение «Betera»

Первым рассматриваемым проектом среди конкурентов является приложение «Betera» [1]. При открытии приложения нас встречает окно для входа. Интерфейс входа показан на рисунке 1.1.

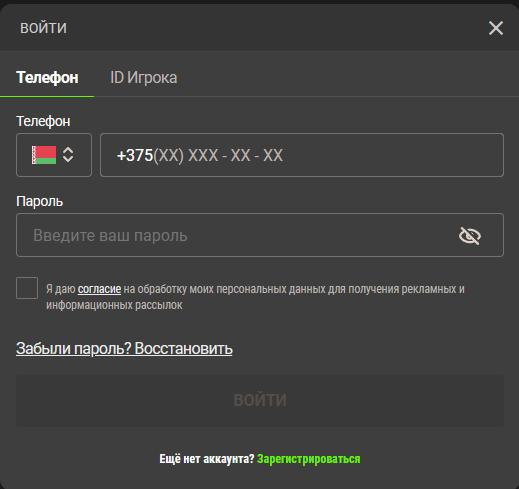


Рисунок 1.1 –Вход в приложение

Здесь видно, что можно выбрать 2 варианта входа, через номер телефона и через IDИгрока. Также можно нажать кнопку для регистрации и восстановить свой пароль. Кнопка «Войти» скрыта пока пользователь не введет свои данные для входа.

После входа мы попадаем на главную страницу. Пример главной страницы продемонстрировано на рисунке 1.2 ниже.

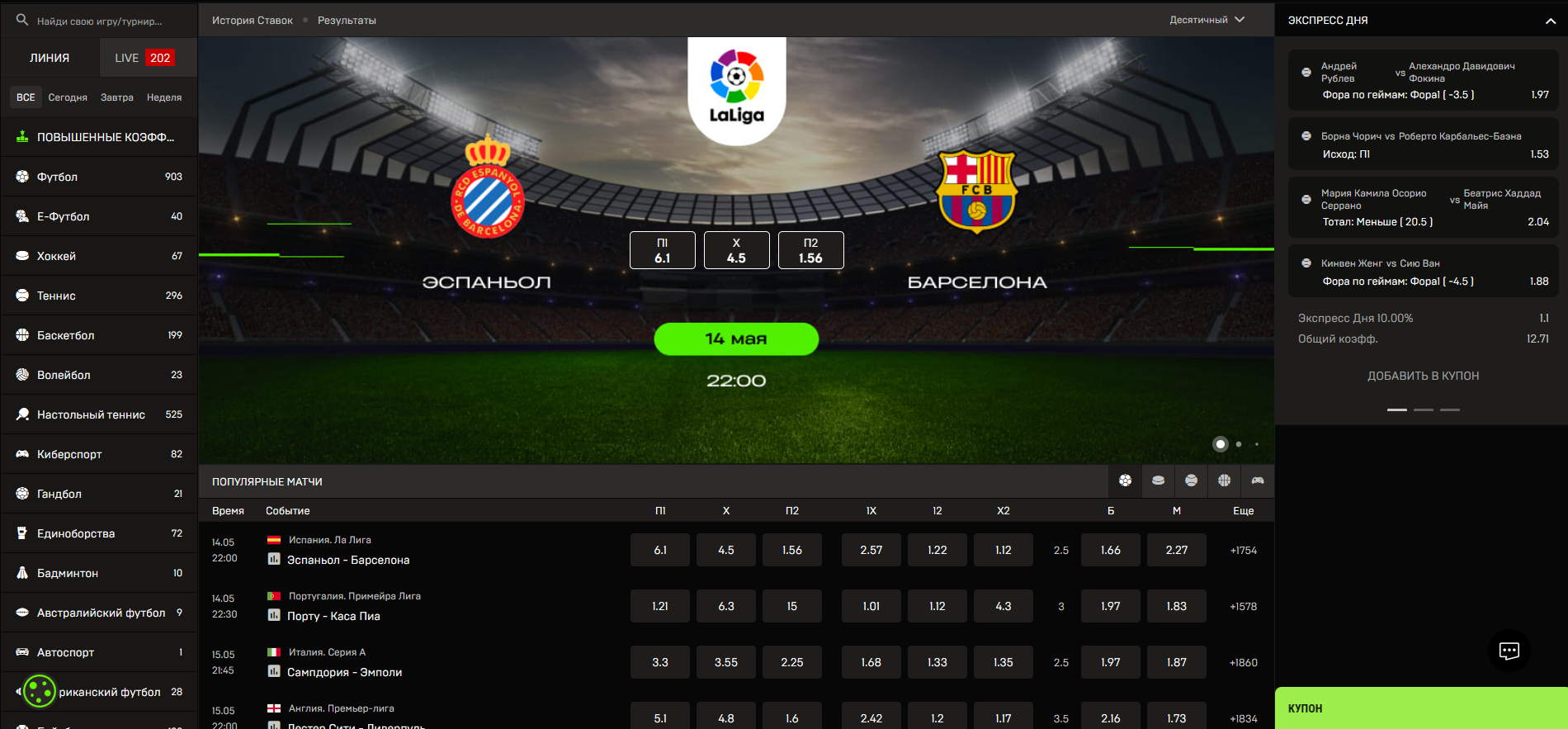


Рисунок 1.2 – Приложение «Betera»

«Betera» является одним из самых популярных приложений. В этом приложении большой выбор ставок для игроков, а также удобный интерфейс. На главной странице мы видим сразу же главные матчи дня. Сразу указываются основные исходы. С левой стороны находится навигация по странам и чемпионатам. Все матчи сегодняшнего дня разделены на популярные матчи и остальные. Также мы можем на главной странице ввести купон для получения бонусов в приложении. В правом верхнем углу мы можем пополнить наш счет, а также увидеть наш баланс. Как мы видим во всех матчах сразу доступна основная линия.

Для более детальной информации о матче мы можем открыть страницу матча. Страница матча показана на рисунке 1.3.

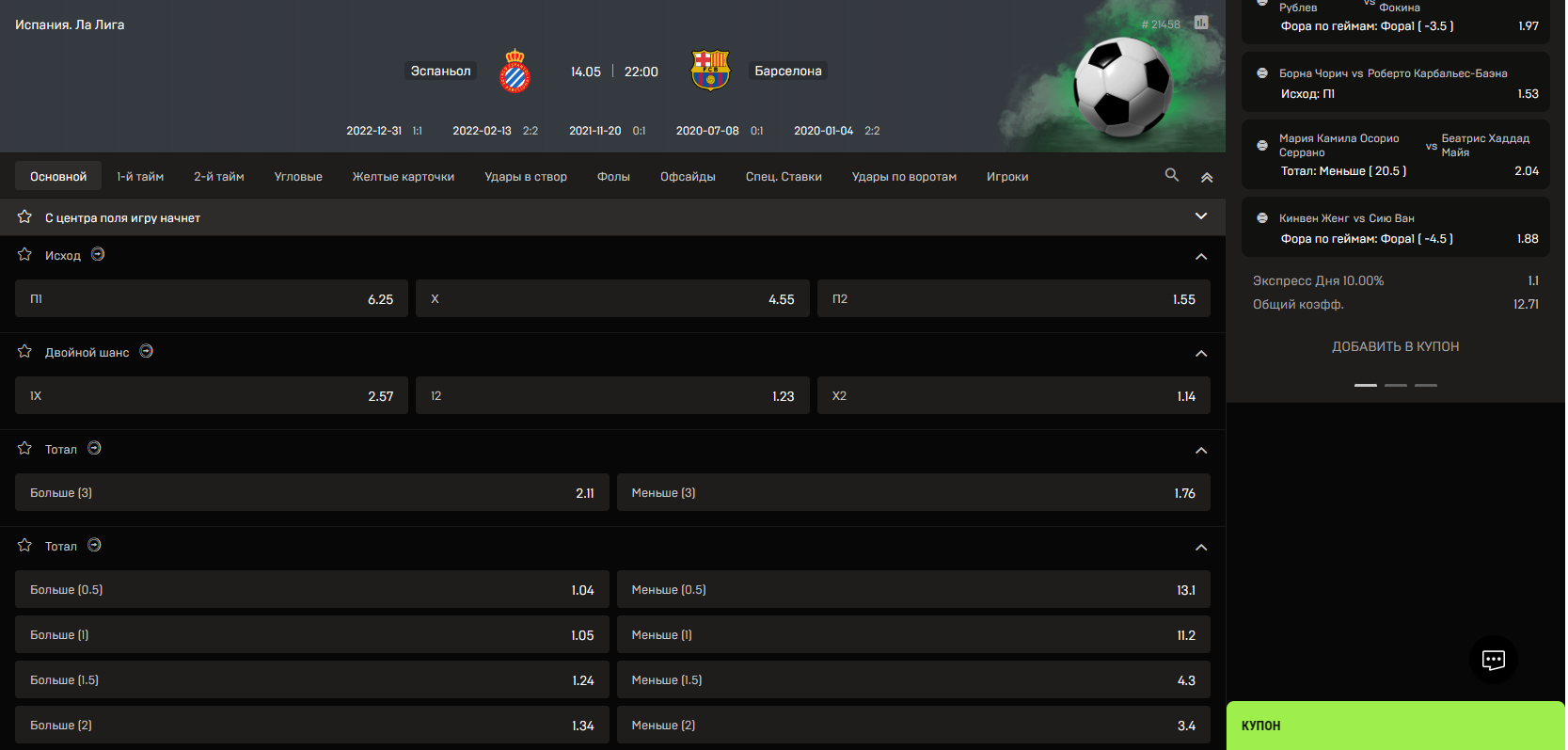


Рисунок 1.3 – Страница матча

На странице матча можно увидеть все исходы на главную игру. Мы также видим название двух команд, которые проводят встречу, время встречи, а также турнир в котором произойдет данная игра. Для более удобного поиска все исходы разделены на отдельные страница. Также при выборе исхода, можно сразу же на этой странице сделать ставку и начать игру. Еще можно найти линию через поиск что ускоряет время её нахождения. Для того чтобы сделать ставку необходимо нажать на исход и появится окно подтверждения ставки. Окно подтверждения ставки показано на рисунке 1.4.

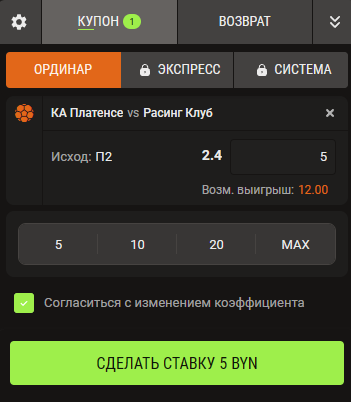


Рисунок 1.4 – Окно подтверждения ставки

Как видно здесь мы можем подтвердить нашу ставку, выбрать сумму денег, которую хотим поставить, а также увидеть выигрыш. Для быстрой ставки мы можем выбрать один из четырех перечисленных сумм и поставить на матч.

Для того чтобы посмотреть последние матчи команд необходимо нажать кнопку справа от команд. Последние матчи команд представлены на рисунке 1.5.



Рисунок 1.5 – Последние матчи команд

Здесь видим последние пять матчей хозяев, последние пять матчей гостевой команды и последние пять очных встреч.

Определим положительные и отрицательные стороны «Betera».

К положительным сторонам можно отнести:

* большой выбор исходов матча;
* вход в систему двумя способами;
* удобное подтверждение ставок.

К отрицательным сторонам можно отнести:

* неудобный поиск матчей;
* малое количество информации о командах.

1.3 Приложение «Fonbet»

Следующее приложение, которое мы рассмотрим является приложение «Fonbet» [2]. Окно входа в приложение показано на рисунке 1.6.

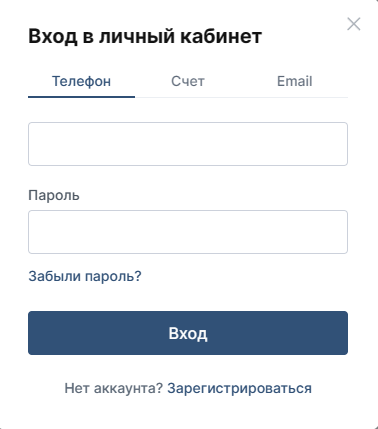


Рисунок 1.6 – Окно входа в приложение «Fonbet»

На рисунке видно, что есть 3 способа входа в приложение: с помощью телефона, с помощью номера счета и через Email. Как и в предыдущем приложении можно восстановить пароль, а также зарегистрироваться. После входа мы попадаем на главную страницу, которая показана на рисунке 1.7.

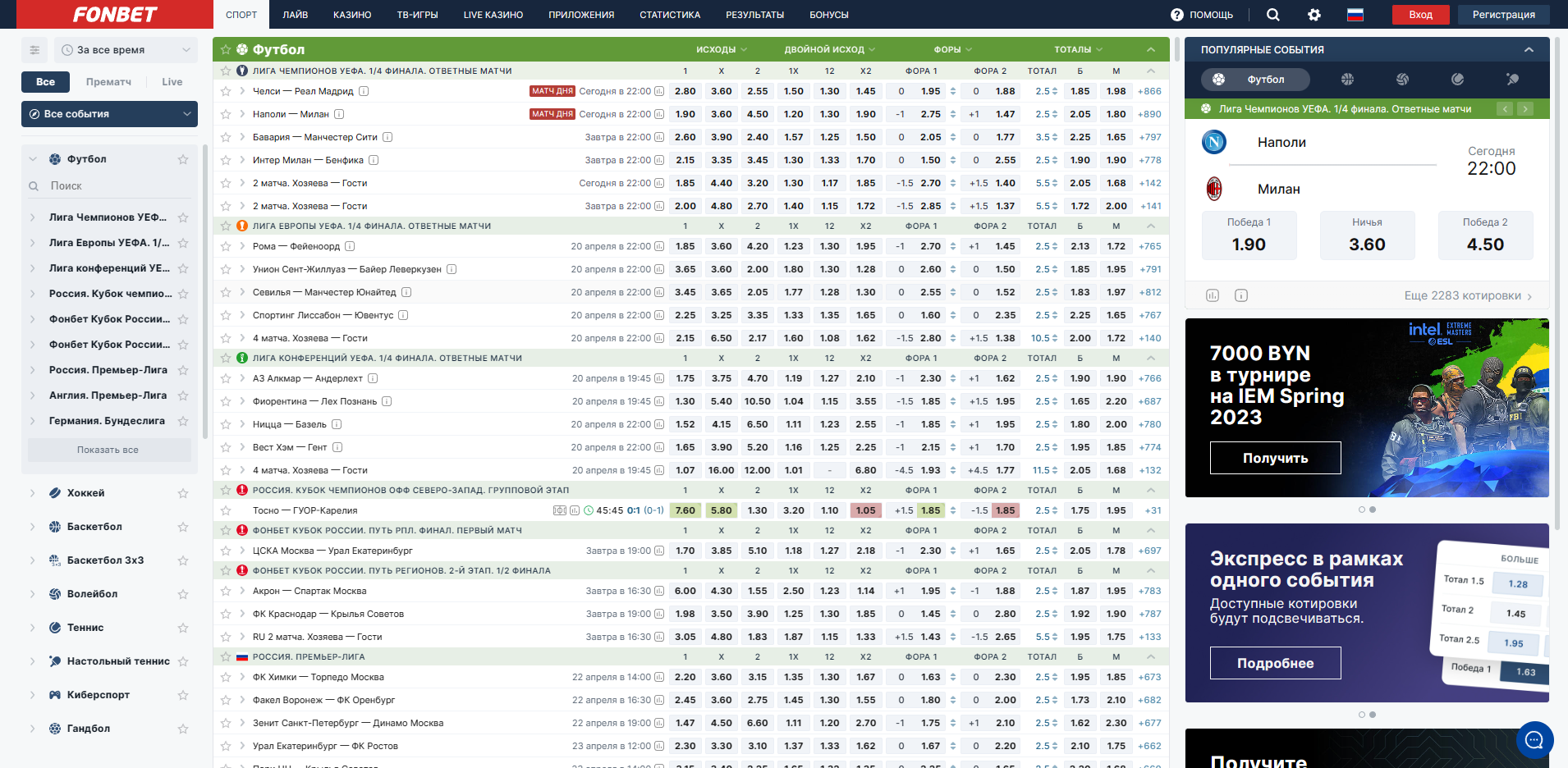


Рисунок 1.7 – Главная страница «Fonbet»

Также, как и в предыдущем приложении на главной странице мы видим все матчи. Первое отличие, которое мы видим это то что показаны не только матчи на сегодня, а также на ближайшую неделю. Возле них сразу представлены основные исходы и можно сделать ставку, не переходя на страницу матча. Справа мы видим также основные матчи сегодняшнего дня и основные коэффициенты. В отличие от первого приложения у нас матчи не сортированы как на популярные и не популярные, а идут все подряд. Как и в первом приложении мы можем выбрать турнир на панели слева.

Здесь также присутствует страница матча, она представлена на рисунке 1.8.

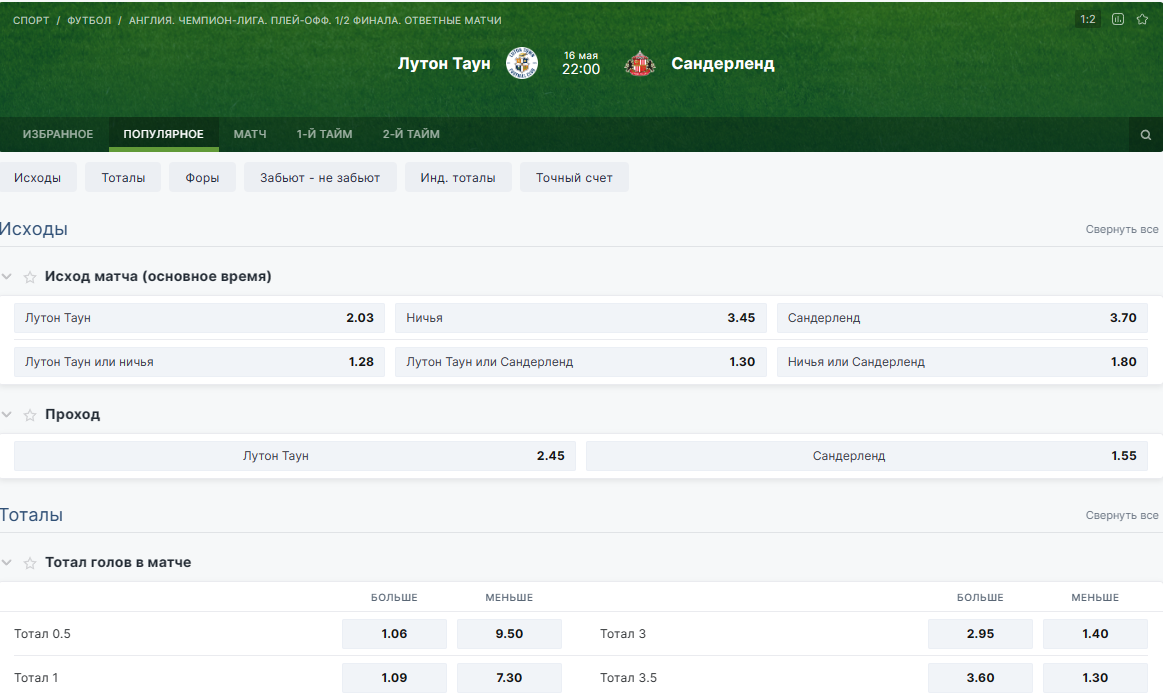


Рисунок 1.8 – Страница матча

На странице матча можно увидеть все исходы на главную игру. Мы также видим название двух команд, которые проводят встречу, время встречи, а также турнир в котором произойдет данная игра. Для более удобного поиска все исходы разделены на отдельные страница. Также при выборе исхода, можно сразу же на этой странице сделать ставку и начать игру. Еще можно найти линию через поиск что ускоряет время её нахождения. Для того чтобы сделать ставку необходимо нажать на исход, и справа на боковой панели можно подтвердить ставку. Окно подтверждения ставки показано на рисунке 1.9.

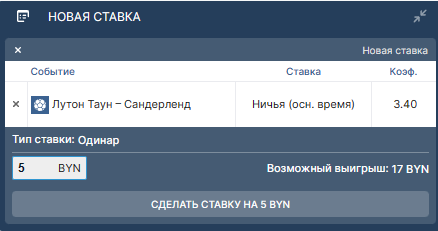


Рисунок 1.9 – Окно подтверждения ставки

В этом окне можно увидеть событие, на которое ставится, коэффициент, сумма ставки, а также возможный выигрыш.

Для того чтобы посмотреть аналитическую информацию о встрече необходимо нажать на кнопку в верхнем правом углу экрана. Информация о матче представлена на рисунке 1.10.

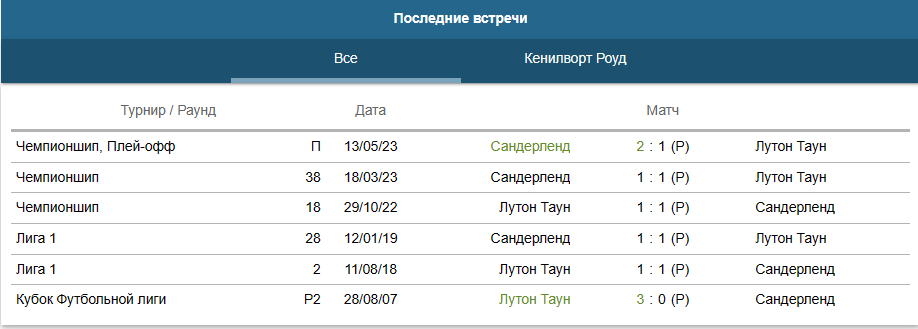


Рисунок 1.10 – Информация о матче

На этой странице в отличии от предыдущего приложения указано больше информации о двух командах. Помимо того, что мы видим предыдущие встречи обеих команд, а также их очные встречи, можно увидеть шансы на победу команд, эффективность забитых мячей, сколько было показано желтых и красных карточек командам, количество штрафных и угловых.

Определим положительные и отрицательные стороны «Fonbet».

К положительным сторонам можно отнести:

* огромный выбор исходов матча;
* вход в систему тремя способами;
* удобное подтверждение ставок.
* большое количество информации о командах.

К отрицательным сторонам можно отнести:

* медленный поиск матчей;
* слишком большое количество информации на главной странице.

1.4 Приложение «MaxLine»

Третьим и последним сравнение аналогом станет приложение «MaxLine» [3]. Вход в приложение показан на рисунке 1.11.



Рисунок 1.11 – Вход в приложение

В данном приложении вход сделан не через отдельное окно, а на верхней панели. Войти можно только через номер телефона. Также возможно восстановить пароль. Главная страница приложения показана на рисунке 1.12.

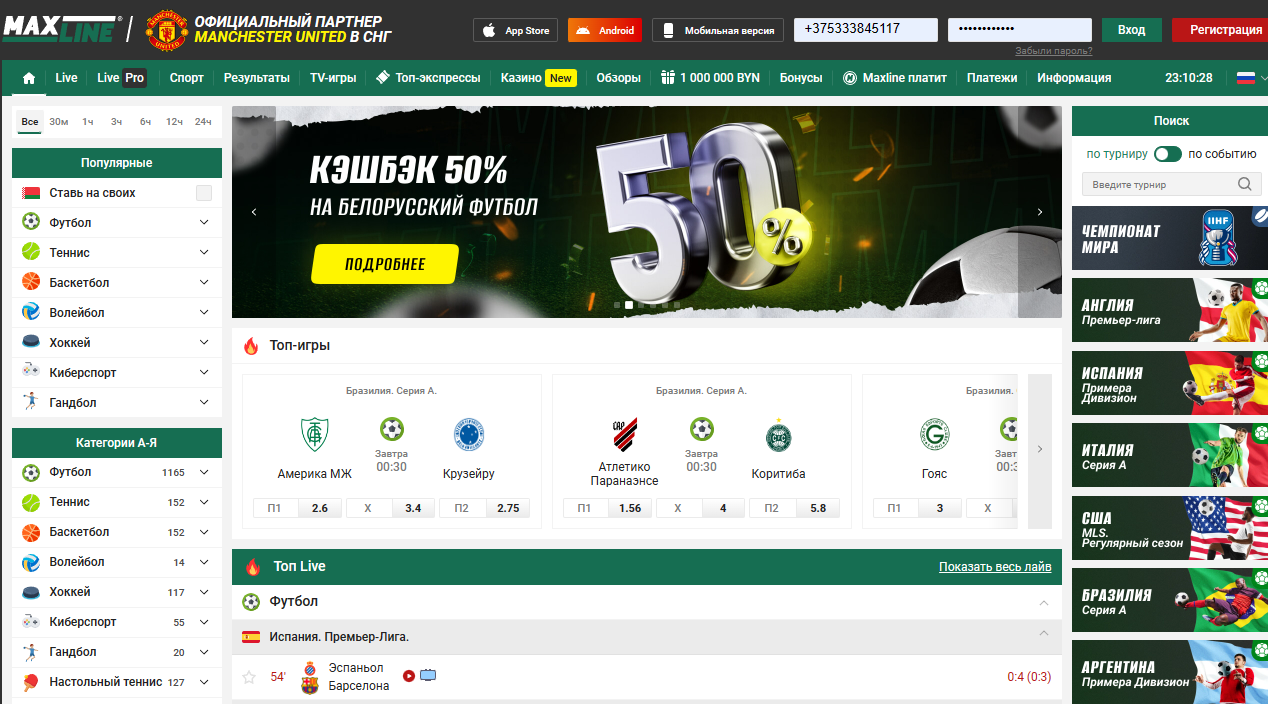


Рисунок 1.12 – Главная страница

Приложение «MaxLine» уже более 10 лет находится на рынке. На главной странице виден список матчей, который проходит на данный момент. В отличие от других приложений здесь не показаны основные исходы матча, а также не показаны матчи, которые будут сыграны в ближайшее время. Нажав на матч, переходим на её страницу. Страница матча показана на рисунке 1.12.

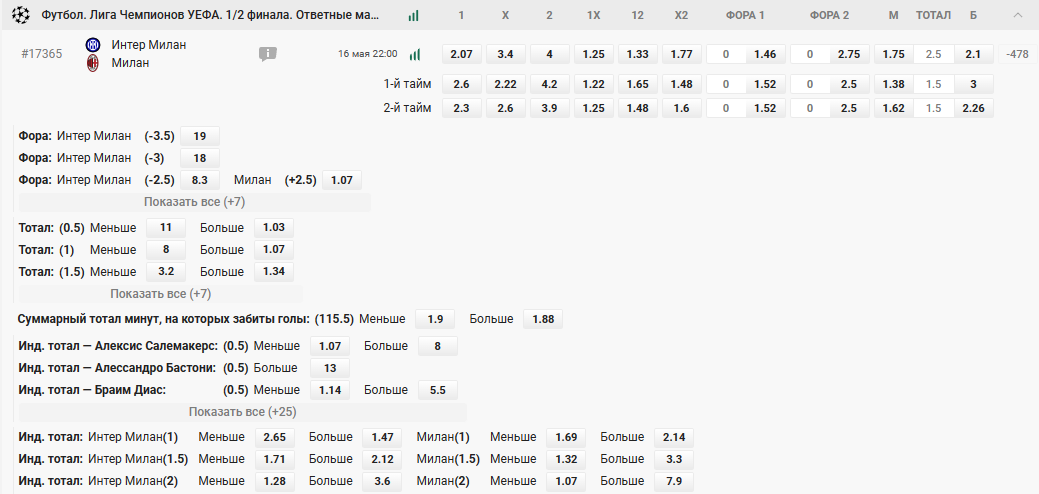


Рисунок 1.12 – Страница матча

На странице матче указаны команды, которые играют, время проведения игры, а также турнир, в котором сыграют. Исходы не сортированы по вкладкам, как было в предыдущих аналогах. Выбрав исход в правом окне появляется окно подтверждения ставки. Оно показано на рисунке 1.13.

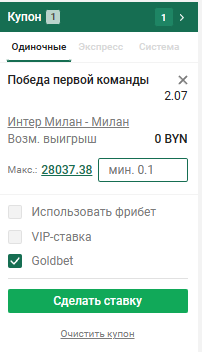


Рисунок 1.13 – Окно подтверждения ставки

В этом окне можно увидеть название ставки, коэффициент, сумма которую ставиться, возможный выигрыш, а также максимальную и минимальную сумму которую можно поставить.

Чтобы посмотреть информацию о матче, необходимо нажать на кнопку, которая находится справа от команд. Информация о матче можно увидеть на рисунке 1.14 ниже.

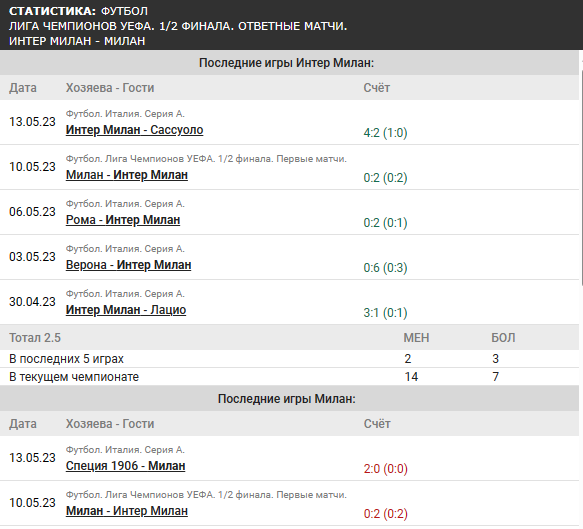


Рисунок 1.14 – Информация о матче

Здесь можно увидеть результаты последних пяти матчей команд, а также результаты последних пяти очных встреч.

Теперь можно определим положительные и отрицательные стороны букмекерской конторы «MaxLine».

К положительным сторонам можно отнести:

* большой выбор исходов матча;
* удобное подтверждение ставок.
* К отрицательным сторонам можно отнести:
* медленный поиск матчей;
* слишком маленькой количество информации на главной странице;
* малое количество информации о матче.

1.5 Постановка задачи

Аналогичные веб-приложения, рассмотренные выше имеют много общего с проектируемым веб-приложением, однако на данный момент существуют есть очевидные проблемы, связанные с качеством проектирования и реализации программного интерфейса.

В итоге были выделены следующие функциональные возможности веб-приложения для букмекерских ставок на футбольные события:

* регистрация и авторизация пользователя;
* поддержка ролей администратора, букмекера и пользователя;
* создание, удаление и изменения регионов, чемпионатов;
* создание, удаление и изменения команд и матчей;
* возможность для букмекера выставлять исходы и коэффициента для матча и чемпионата;
* возможность пользователям ставить на футбольные события;
* возможность для пользователя пополнять и возвращать деньги;
* возможность пользователя просматривать историю ставок;
* возможность пользователю просматривать историю счета.

1.6 Вывод по разделу

В данном разделе были описаны аналогичные веб-приложения разрабатываемого проекта. Из представленного краткого обзора «Betera», «Fonbet» и «MaxLine» имеют похожую функциональную возможность. Были определены положительные и отрицательные стороны каждого приложения.

На основании подобных приложений были сформулированы ключевые принципы разработки веб-приложения, упрощающие его использование и делающие его более интуитивно понятным для пользователей.

2 Проектирование веб-приложения

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*25*

*БГТУ 03.00.ПЗ*

Разраб*.*

*Гой М.А.*

Провер*.*

*.*

*Комарова Е.И.*

Н*.* контр*.*

*Николайчук А.Н.*

Утв*.*

*Смелов В.В.*

3 Разработка программного средства

*Лит.*

*Листов*

*10*

*7421877-C, 2023*

Проектирование программного средства – важная задача в процессе работы над приложением, потому что в зависимости от нее определяется уровень зависимости между компонентами приложения, и то, насколько легко его можно будет расширить и масштабировать. Хорошо продуманная архитектура веб-приложения может справляться с различными нагрузками и умело адаптироваться к изменяющимся бизнес-требованиям, обеспечивая быстрое взаимодействие с клиентами. Это, разумеется, повышает производительность приложения.

2.1 Диаграмма вариантов использования

Диаграмма вариантов использования нужна для того, чтобы представить проектируемую систему в виде множество клиентов с определенными ролями взаимодействующие с системой с помощью вариантов использования.

Диаграмма вариантов использования для администратора представлена в приложении А. Администратор отвечает за добавление регионов, чемпионатов, команд и матчей. Также он может их изменять или удалять. Всеми зарегистрированными пользователями тоже может управлять администратор, менять им роли, удалять или блокировать.

Диаграмма вариантов использования для букмекера представлена в приложении Б. Букмекеру доступна возможность добавления исходов, а также добавление коэффициентов на матч.

Диаграмма вариантов использования для Гостя и Пользователя представлена в приложении В. Для многих действий Пользователю необходимо авторизоваться в приложении. Для пополнения счета, просмотра личной страницы и делать ставки на футбольные события.

2.2 Выбор средств реализации

При разработке приложения необходимо подобрать языки программирования и фреймворки, для клиентской и серверной части, которую наилучшим образом соответствуют требуемому функционалу. При выборе правильного языка программирования, фреймворка и системы управления базами данных оказывает значительное влияние на проект, его сроки и сложность реализации.

2.3 Основные языки программирования

Перед началом разработки приложения необходимо выбрать языки программирования для написания клиентской и серверной части.

C# - современный объектно-ориентированный язык программирования [4]. На сегодняшний момент язык программирования C# один из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. В настоящий момент на нем пишутся самые различные приложения: от небольших десктопных программок до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей. Также огромным плюсом является то, что данный язык программирования получает обновления и поддерживается компанией Microsoft на протяжении долгих лет.

После выбора языка программирования для серверной части необходимо выбрать язык программирования для клиентской части.

TypeScript – язык программирования, представленный компанией Microsoft в 2012 году [5]. TypeScript - это расширенная версия JavaScript. То есть он содержит в себе все то же самое, что и JavaScript, но с некоторыми дополнениями. Главная причина использовать TypeSciprt - это возможность добавить статическую типизацию к JavaScript. Тип переменной со статической типизацией не может быть изменен после ее объявления. Это может предотвратить большое количество багов.

2.4 Выбор технологий и библиотек

Фреймворк – программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта. Как правило на основе фреймворков создаются приложения.

Для разработки серверной части веб-приложения был выбран кроссплатформенный фреймворк ASP.NET CORE [6]. ASP.NET Core является кроссплатформенной, высокопроизводительной средой с [открытым исходным кодом](https://github.com/dotnet/aspnetcore) для создания современных облачных приложений, подключенных к Интернету. Данный фреймворк может работать поверх кроссплатформенной среды .NET Core, которая может быть развернута на основных популярных операционных системах: Windows, Mac OS, Linux. И таким образом, с помощью ASP.NET Core мы можем создавать кроссплатформенные приложения. А для развертывания веб-приложения можно использовать традиционный IIS.

Для реализации клиентской части была выбрана библиотека React [7]. React представляет JavaScript-библиотеку для создания пользовательских интерфейсов, написанный на языке JavaScript и предназначенный для создания веб-приложений клиентского уровня.

2.5 Система управления базами данных

Для хранения данных я использовал Microsoft SQL Server [8]. Microsoft SQL Server – систему управления реляционными базами данных, разработанная корпорацией Microsoft. СУБД позволяет гибко управлять базами данных (БД). С ее помощью можно создавать, модифицировать или удалять записи, отправлять транзакцию — набор из нескольких последовательных запросов на особом языке запросов SQL. MSSQL базируется на языке SQL и поддерживает многочисленные возможности.

2.6 Проектирование структурной схемы веб-приложения

Структурная схема веб-приложения – это графическое представление архитектуры приложения, показывающее его основные компоненты и связи между ними. Она помогает разработчикам лучше понимать архитектуру приложения и оптимизировать его работу. Она также обеспечивает более эффективное управление проектом и позволяет более эффективно распределять задачи между разработчиками.

Для визуализации элементов и компонентов программы была разработана структурная схема приложения, представленная на рисунке 2.1.

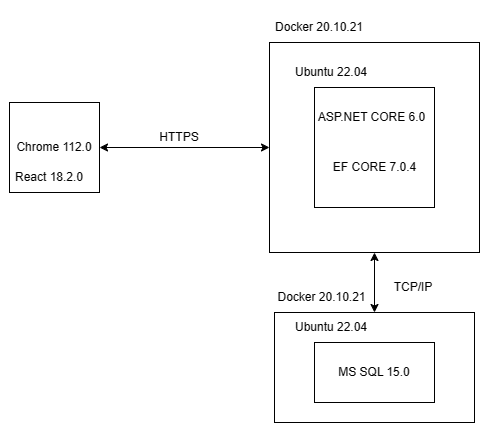


Рисунок 2.1 – Структурная схема веб-приложения

Данная схема показывает все компоненты веб-приложения и показывает взаимодействие между ними.

2.7 Логическая схема базы данных

Для разработки веб-приложения в рамках дипломного проекта понадобилась база данных с необходимой конфигурацией сущностей для хранения всей необходимой информации. Как было аргументировано ранее, в качестве системы управления базами данным выступает Microsoft SQL Server.

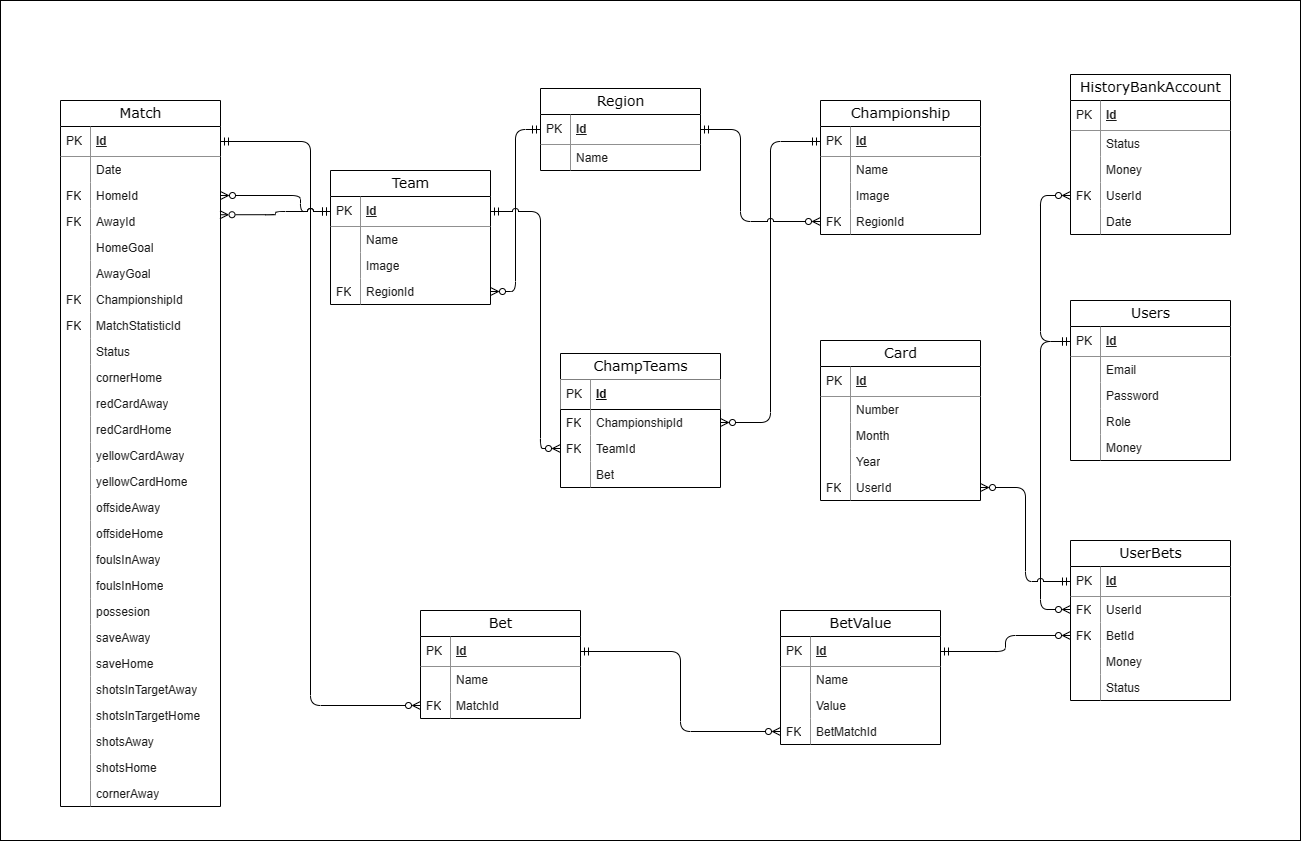


Рисунок 2.5 – Логическая схема базы данных

Далее приведена структура и описание всех таблиц, используемых в веб-приложении.

Таблица «Region» предназначена для хранения регионов. Полное описание данной таблицы представлена в таблице 2.1.

Таблица 2.1 – Структура таблицы «Region»

|  |  |  |
| --- | --- | --- |
| Написание | Тип | Описание |
| Id | Int | Идентификатор региона |
| Name | Nvarchar | Название региона |

Таблица «Team» предназначена для хранения команд. Полное описание данной таблицы представлена в таблице 2.2.

Таблица 2.1 – Структура таблицы «Team»

|  |  |  |
| --- | --- | --- |
| Написание | Тип | Описание |
| Id | Int | Идентификатор команды |
| Name | Nvarchar | Название команды |
| Image | Nvarchar | Ссылка на изображение |
| RegionId | Int | Идентификатор региона, внешний ключ |

Таблица «Championship» предназначена для хранения чемпионатов. Полное описание данной таблицы представлена в таблице 2.3.

Таблица 2.3 – Структура таблицы «Championship»

|  |  |  |
| --- | --- | --- |
| Написание | Тип | Описание |
| Id | Int | Идентификатор чемпионата |
| Name | Nvarchar | Название команды |
| Image | Nvarchar | Ссылка на изображение |
| RegionId | Int | Идентификатор региона, внешний ключ |

Таблица «Users» предназначена для хранения пользователей. Полное описание данной таблицы представлена в таблице 2.4.

Таблица 2.4 – Структура таблицы «Users»

|  |  |  |
| --- | --- | --- |
| Написание | Тип | Описание |
| Id | Int | Идентификатор пользователя |
| Email | Nvarchar | Электронная почта |
| Password | Nvarchar | Пароль пользователя |
| Role | Nvarchar | Роль пользователя |
| Money | Real | Личный счет пользователя |

Таблица «Match» предназначена для хранения матчей. Полное описание данной таблицы представлена в таблице 2.5.

Таблица 2.5 – Структура таблицы «Match»

|  |  |  |
| --- | --- | --- |
| Написание | Тип | Описание |
| Id | Int | Идентификатор матча |
| Date | DateTime2(7) | В |
| HomeGoal | Int | Голы домашней команды |
| AwayGoal | Int | Голы гостевой команды |
| HomeId | Int | Идентификатор домашней команды, внешний ключ |
| AwayId | Int | Идентификатор гостевой команды, внешний ключ |
| ChampionshipId | Int | Идентификатор чемпионата, внешний ключ |
| MatchStatisticId | Int | Идентификатор статистики, внешний ключ |
| Status | Nvarchar | Статус матча |

Таблица «ChampTeams» предназначена для хранения команд, которые участвуют в конкретном чемпионате. Полное описание данной таблицы представлена в таблице 2.6.

Таблица 2.6 – Структура таблицы «ChampTeams»

|  |  |  |
| --- | --- | --- |
| Написание | Тип | Описание |
| Id | Int | Идентификатор чемпионата |
| ChampionshipId | Int | Идентификатор чемпионата, внешний ключ |
| TeamId | Int | Идентификатор команды, внешний ключ |
| Bet | Real | Ставка на команду |

Таблица «Card» предназначена для карточек пользователей. Полное описание данной таблицы представлена в таблице 2.7.

Таблица 2.7 – Структура таблицы «Card»

|  |  |  |
| --- | --- | --- |
| Написание | Тип | Описание |
| Id | Int | Идентификатор записи |
| Number | Nvarchar | Номер карты |
| Month | Int | Месяц |
| Year | Int | Год |
| UserId | Int | Идентификатор пользователя, внешний ключ |

Таблица «MatchStatistic» предназначена для хранения статистики матча. Полное описание данной таблицы представлена в таблице 2.8.

Таблица 2.8 – Структура таблицы «MatchStatistic»

|  |  |  |
| --- | --- | --- |
| Написание | Тип | Описание |
| Id | Int | Идентификатор статистики |
| cornerHome | Int | Угловые домашней команды |
| cornerAway | Int | Угловые гостевой команды |
| shotsHome | Int | Удары домашней команды |
| shotsAway | Int | Удары гостевой команды |
| shotsInTargetHome | Int | Удары в створ домашней команды |
| shotsInTargetAway | Int | Удары в створ гостевой команды |
| saveHome | Int | Спасение домашней команды |
| saveAway | Int | Спасение гостевой команды |
| possession | Int | Владение мячом |
| foulsInHome | Int | Фолы домашней команды |
| foulsInAway | Int | Фолы гостевой команды |
| offsideHome | Int | Офсайды домашней команды |
| offsideAway | Int | Офсайды гостевой команды |
| yellowCardHome | Int | Желтые карточки домашней команды |
| yellowCardAway | Int | Желтые карточки гостевой команды |
| redCardHome | Int | Красные карточки домашней команды |
| redCardAway | Int | Красные карточки гостевой команды |

Таблица «BetMatch» предназначена для хранения всех ставок на матч. Полное описание данной таблицы представлена в таблице 2.9.

Таблица 2.9 – Структура таблицы «BetMatch»

|  |  |  |
| --- | --- | --- |
| Написание | Тип | Описание |
| Id | Int | Идентификатор чемпионата |
| Name | Nvarchar | Название ставки |
| MatchId | Int | Идентификатор матча, внешний ключ |

Таблица «Bet» предназначена для хранения коэффициентов на матч. Полное описание данной таблицы представлена в таблице 2.10.

Таблица 2.10 – Структура таблицы «Bet»

|  |  |  |
| --- | --- | --- |
| Написание | Тип | Описание |
| Id | Int | Идентификатор чемпионата |
| Name | Nvarchar | Название исхода |
| Value | Real | Коэффициент |
| BetMatchId | Int | Идентификатор матча, внешний ключ |

Таблица «UserBets» предназначена для хранения коэффициентов на матч. Полное описание данной таблицы представлена в таблице 2.11.

Таблица 2.11 – Структура таблицы «UserBets»

|  |  |  |
| --- | --- | --- |
| Написание | Тип | Описание |
| Id | Int | Идентификатор чемпионата |
| UserId | Int | Идентификатор пользователя, внешний ключ |
| BetId | Int | Идентификатор ставки, внешний ключ |
| Money | Real | Ставка |
| Status | Nvarchar | Статус ставки |

Таблица «HistoryBankAccount» предназначена для хранения истории счета пользователя. Полное описание данной таблицы представлена в таблице 2.12.

Таблица 2.12 – Структура таблицы «HistoryBankAccount»

|  |  |  |
| --- | --- | --- |
| Написание | Тип | Описание |
| Id | Int | Идентификатор чемпионата |
| Status | Nvarchar | Информация о счете |
| Money | Real | Величина перевода |
| Date | DateTime2(7) | Дата |
| UserId | Int | Идентификатор пользователя, внешний ключ |

2.5 Вывод по разделу

В данном разделе были обоснованы и выбраны основные языки программирования, фреймворки и СУБД для клиентской и серверной части веб-приложения. Были продемонстрированы диаграммы использования приложения, которые показывают возможные сценарии использования. Кроме того, была предоставлена логическая схема базы данных с описанием таблиц и всех полей.

3 Реализация веб-приложения

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*2*

*БГТУ 04.00.ПЗ*

Разраб*.*

*Гой М.А.*

Провер*.*

*.*

*Комарова Е.И.*

Н*.* контр*.*

*Николайчук А.Н.*

Утв*.*

*Смелов В.В.*

*Лит.*

*Листов*

*21*

11111111, 2023

*3 Реализация веб-приложения*

Целью дипломного проекта является разработка веб-приложения для букмекерских ставок на футбольные события. На диаграмме вариантов использования были определены функциональные возможности программного средства. Которые необходимо реализовать. Для достижения поставленной цели необходимо реализовать две части программного средства: серверную и клиентскую.

При реализации веб-приложения необходимо следовать правилам, которые были заложены при проектировании приложения, ведь следую этим правилам удастся выполнить все задачи дипломного проекта.

В этом разделе будет рассмотрен процесс разработки веб-приложения, в соответствии с пользовательскими потребностями и проектными требованиями, описанными в первом и во втором разделе данного дипломного проектирования.

3.1 Разработка серверной части

Разработка серверной части веб-приложения заключается в создании программного обеспечения, которое будет выполняться на сервере и предоставлять клиентской части приложения необходимые данные и функционал. Эта часть приложения отвечает за обработку запросов от клиентов, работу с базами данных, бизнес-логику и другие задачи, связанные с обработкой данных на стороне сервера. В соответствии с выбранной технологии будет использоваться фреймворк ASP.NET Core. Важным аспектом разработки серверной части является обеспечение безопасности и защиты от взлома, а также оптимизация производительности для обеспечения быстрого и стабильного функционирования приложения.

Для начала был создан проект ASP.NET Core Web API. Это платформа для создания веб-сервисов и API. Она позволяет разработчикам создавать высокопроизводительные, масштабируемые и безопасные веб-сервисы, которые могут быть использованы клиентскими приложениями на разных платформах, таких как веб-приложения, мобильные приложения и десктопные приложения.

Перед началом разработки серверной части была сформирована файловая структура проекта, которая будет соответствовать целям проекта, предоставляя весь доступный функционал. Структура серверной части приложения играет ключевую роль в обеспечении его эффективной работы и масштабируемости. Она помогает разработчикам создавать устойчивые, надежные и безопасные приложения, которые могут обрабатывать большие объемы данных и одновременных запросов.

На рисунке 3.1 представлена файловая структура серверной части веб-приложения, которая показывает организацию и расположение директорий внутри проекта. Такая структура помогает упрощает разработку, облегчает навигацию, а также разделяет компоненты внутри проекта.

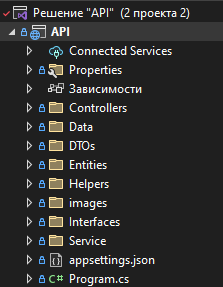


Рисунок 3.1 – Файловая структура веб-приложения

Папка Controllers содержит все контроллеры, которые обрабатывают запросы к приложению. При получении запроса система маршрутизации выбирает для обработки запроса нужный контроллер и передает ему данные запроса. Контроллер обрабатывает эти данные и посылает обратно результат обработки.

Папка Data хранит контекст класс ApplicationContext который отвечает за связь с базой данных. Кроме этого в папке хранятся классы репозитории, которые являются промежуточным звеном между классами и остальной программой.

Папка DTOs содержит классы, которые используются для обмена данными клиентская и серверная часть.

Папка Entities хранит те классы, которые представляют из себя объекты, которые будут хранится в базе данных.

Папка Helpers хранит классы, которые помогают настраивать наше приложение. Здесь находится классы AuthOptions и AutoMapperProfiles. Первый класс описывает ряд свойств, которые нужны для генерации токена. Второй класс позволяет проецировать одну модель данных на другую, что позволяет сократить объём кода в программе.

Папка images является статической папкой. В ней находятся все логотипы регионов, чемпионатов и команд. Хранение изображений на сервере, а не в базе данных снимает нагрузку с неё, что ускоряет работу базы данных.

Папка Interfaces хранит все интерфейсы, которые используются в приложении. Интерфейсы позволяют внутри программы абстрагироваться от реализации.

Папка Service содержит все сервисы, которые используются в приложении.

Таким образом, данная архитектура является наиболее приемлемой для данного веб-сервиса ввиду своей простоты и легкой возможности для дальнейшего расширения и внесения изменений в проект.

3.1.1 Разработка репозиториев

Один из начальных этапов нашего проекта - создание моделей и репозиториев для связи с базой данных. Для работы с базой данных на языке C# используется технология ORM "Entity Framework"[9]. Она позволяет получать данные из внешнего источника в форме объектов C#. Эта утилита начала свое развитие на платформе .Net Framework, но с появлением платформы Core, она получила значительные улучшения и изменилась. В особенности, была создана библиотека "EntityFramework.Core", которая обеспечивает более удобные асинхронные запросы к базе данных и удобную конфигурацию сущностей.

Для реализации цели дипломного проекта был выбран подход «Code First». Такой подход уже был успешно применен на платформе .Net Framework. Суть «Code First» заключается в создании всех необходимых объектов базы данных, таких как сущности, связи, ограничения целостности, индексы и представления, непосредственно в коде. После подготовки всей конфигурации выполняется специальная команда для создания миграций: «dotnet ef add-migration name». Такой подход может быть использован как для создания новой базы данных, так и для подключения к уже существующей.

Для начала необходимо дать определение модели и репозитория.

Модель в Entity Framework Core представляет собой набор классов, которые описывают структуру базы данных и связи между таблицами. Она используется для создания объектно-ориентированного представления базы данных, которое позволяет работать с данными в виде объектов, а не в виде SQL-запросов. Модель в Entity Framework Core может быть создана автоматически на основе существующей базы данных или ручным способом, при этом она может быть изменена в соответствии с требованиями приложения.

Было создано 11 моделей, представленных на рисунке 3.2.

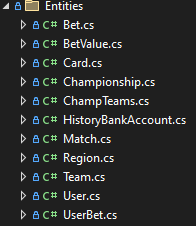


Рисунок 3.2 – Реализация моделей

Ниже приведен листинг 3.1 с кодом модели Team.

|  |
| --- |
| public class Team  {  [Key]  public int Id { get; set; }  public string Name { get; set; }  public string? Image { get; set; }  public int RegionId { get; set; }  public virtual Region Region { get; set; }  [JsonIgnore]  public virtual ICollection<ChampTeams> ChampTeams { get; set; }  [JsonIgnore]  public virtual ICollection<Match> HomeTeams { get; set; }  [JsonIgnore]  public virtual ICollection<Match> AwayTeams { get; set; }  public Team()  {  ChampTeams = new HashSet<ChampTeams>();  HomeTeams = new List<Match>();  AwayTeams = new List<Match>();  }  } |

Листинг 3.1 – Модель Team

Репозиторий в Entity Framework Core представляет собой класс, который предоставляет интерфейс для работы с данными в базе данных. Он инкапсулирует логику доступа к данным и скрывает детали реализации от остальной части приложения. Репозиторий позволяет выполнять операции CRUD (create, read, update, delete) с данными, а также выполнять более сложные запросы и операции. Он может быть реализован как обобщенный класс, который работает с любым типом сущностей в модели Entity Framework Core. Репозиторий также может использоваться для упрощения тестирования и повышения переносимости кода между различными базами данных.

Созданы девять репозиториев, представленных на рисунке 3.7.

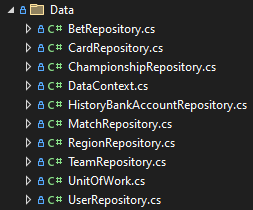


Рисунок 3.1 – Реализация репозиториев для базы данных

Листинг репозитория BetRepository представлен в приложении А. Он наследуется от интерфейса IBet, который содержит методы, который должен реализовать данный репозиторий. Далее все эти методы используют контроллеры, для обработки запросов.

3.1.2 Разработка сервисов

Основная задача сервисов, прежде всего, заниматься обработкой данных: вызывать инфраструктурный уровень для чтения и обработки данных во внешнем источнике, а также включать в себя логическую часть по преобразованию нужных данных и реализации соответствующих алгоритмов.

Всего было разработано три сервиса, которые могут вызываться любым контроллером. Сервисы продемонстрированы на рисунке 3.2.

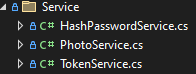


Рисунок 3.2 – Сервисы приложения

Пример программной реализации сервиса TokenService и его метод показан на листинге 3.2.

|  |
| --- |
| public class TokenService : ITokenService  {  public async Task<string> CreateToken(User user)  {  var claims = new List<Claim> {  new Claim("email", user.Email),  new Claim("role", user.Role),  new Claim("id", user.Id.ToString()),  new Claim("money", user.Money.ToString()),  };  var jwt = new JwtSecurityToken(  issuer: AuthOptions.ISSUER,  audience: AuthOptions.AUDIENCE,  claims: claims,  expires: DateTime.UtcNow.Add(TimeSpan.FromDays(30)),  signingCredentials: new SigningCredentials(AuthOptions.  GetSymmetricSecurityKey(), SecurityAlgorithms.HmacSha256));  return new JwtSecurityTokenHandler().WriteToken(jwt);}} |

Листинг 3.2 – Сервис TokenService

Все сервисы наследуются от интерфейса, который находятся в папке Interfaces. Это нужно для того, чтобы в уровне бизнес-логики не задумываться о реализации того или иного сервиса. Принцип работы каждого сервиса устроен следующим образом: данные он получает от уровня представления, или же контроллера. Затем сервис начинает обработку этих данных согласно алгоритмам. Здесь сервис имеет право вызывать инфраструктурный уровень для чтения и записи информации. Если же необходимо записать новую информацию или обновить существующие, сервис будет сопоставлять модели, которые ему предоставил контроллер к моделям базы данных. После того, как сервис закончит свою работу, он обязательно преобразует данные в тот формат, который будет компактным и удобным для чтения как клиентской части.

Каждый интерфейс наделен лишь одной задачей:

* HashPasswordService реализует интерфейс IHashPassword, который кэширует пароль пользователя при его регистрации.
* PhotoService реализует интерфейс IPhotoService, который сохраняет и удаляет изображения в приложении.
* TokenService, который реализует интерфейс ITokenService создает токен для всех пользователей, который используют для работы с серверной частью.

3.1.3 Разработка контроллеров

Контроллеры представляют собой классы, которые обрабатывают запросы от клиента, вызывают нужные методы модели и возвращают данные. Они являются посредниками между пользовательским интерфейсом и бизнес-логикой приложения. Контроллеры содержат методы действий, которые вызываются при получении запроса от клиента и обрабатывают его. Кроме того, контроллеры могут использоваться для управления состоянием приложения.

Чаще всего контроллер вызывает соответствующий уровень обработки данных, получает подходящего вида. Они инкапсулируют всю бизнес-логику приложения, так как именно благодаря контроллеру данные из модели будут обрабатываться и передаваться клиенту.

Всего было создано девять контроллеров, как показано на рисунке 3.3.

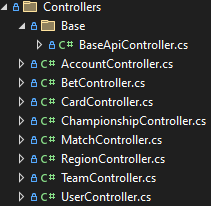


Рисунок 3.3 – Разработанные контроллеры

Каждый контроллер наследуется от класса BaseApiController, который находится в папке Base. Далее в списке приведены описания каждого контроллера:

* AccountController: данный контроллер отвечает за обработку запросов, связанных с авторизацией и регистрацией. Он содержит в себе два метода: авторизация и регистрация.
* BetController: этот контроллер отвечает за обработку запросов, которые связаны со ставками. Он содержит методы для создания ставок, принимает ставку, которую делает пользователь, а также сохраняет результат ставок.
* CardController: это контроллер, который занимается обработкой запросов связанных с картами клиентов. Он содержит методы для добавления карты, а также для вывода карт пользователя.
* ChampionshipController: это контроллер, который отвечает за обработку запросов связанных с чемпионатами. Он содержит методы для создания, изменения и удаления чемпионатов. Также он возвращает список команд, участвующих в чемпионате и может добавлять или удалять команду из чемпионата.
* MatchController: контроллер, который обрабатывает запросы связанные с матчами. Он может их добавлять и удалять, а также возвращать список матчей.
* RegionController: данный контроллер обрабатывает запросы связанных с регионами. Он может добавлять, изменять и удалять регионы.
* TeamController: этот контроллер отвечает за обработку запросов связанных к командами. Он может создавать, изменять и удалять команды.
* UserController: данный контроллер обрабатывает запросы связанные с пользователями. Он позволяет пользователям пополнять личный счет, выводить средства, получить историю личного счета.

В качестве примера в листинге 3.3 представлена часть кода AccountControllerдля авторизации пользователя.

|  |
| --- |
| [HttpPost("Login")]  public async Task<ActionResult<UserDto>> Login(LoginDto loginUser)  {  var user = await \_unitOfWork.User.  GetUserByEmail(loginUser.Email);  if (user is null)  return BadRequest("E-mail не найден");  var hash = \_hashPassword.CreateHash(loginUser.Password);  if (user.Password != \_hashPassword.CreateHash(loginUser.Password))  return BadRequest("Неверный пароль");  return new UserDto  {  Email = user.Email,  Token = await \_tokenService.CreateToken(user),  Role = user.Role,  Money = user.Money,  };  } |

Листинг 3.3 – Метод авторизации пользователя

Метод получает объект LoginDto с электронной почтой и паролем пользователя. Далее осуществляется поиск пользователя в базе данных по электронной почте. Если такой пользователь не находится, то возвращается объект BadRequest со статусным кодом 400, и с сообщение о том, что такой пользователь не найден.

Далее сравнивается пароль, который прислал пользователь, и пароль, который находится в базе данных, перед этим присланный пароль хэшируется. Если пароли не совпадают, то возвращается объект BadRequest с сообщение о том, что пароли не совпадают.

Если этапы проверки пользователя пройдены, то возвращается объект UserDto, который хранит в себе токен пользователя, для следующих взаимодействий с сервером.

3.2 Разработка клиентской части

Разработка клиентской части приложения является важным этапом процесса создания программного продукта. Клиентская часть отвечает за взаимодействие пользователя с приложением и обеспечивает удобный интерфейс для работы с функционалом.

Для эффективной разработки клиентской части приложения было принято решение использовать фреймворк React в сочетании с TypeScript, а в качестве редактора кода был выбран VS Code. Для создания динамических интерфейсов были предпочтены функциональные компоненты и хуки, которые обеспечивают работу со состоянием и другими функциональностями React без необходимости использования классов.

Для создания приложения на React с использованием TypeScript без дополнительных сложностей и конфигурации проекта нужно выполнить команду «npx create-react-app --template ». Это автоматически создаст проект с требуемой в дипломном проекте библиотекой и поддержкой TypeScript.

Для запуска проекта достаточно написать команду «npm start». Конфигурационный файл tsconfig.json хранит в себе всю информацию о проекте, а также скрипты для запуска. В нем уже находится скрипт с названием start, в котором прописана команда на запуск проекта. После этого можно приступить к самой реализации проекта, проверив, что он полностью в рабочем состоянии.

Как и в серверной части приложения, необходимо выбрать архитекуру клиентской части приложения. Архитектура в клиентской части приложения нужна для того, чтобы обеспечить структурированность и организованность кода, улучшить его читаемость и поддерживаемость, а также облегчить масштабирование и расширение приложения. Хорошая архитектура позволяет разделить логику приложения на отдельные компоненты, модули или слои, каждый из которых отвечает за свою задачу. Это делает код более гибким и позволяет изменять его без влияния на другие части приложения. Кроме того, хорошая архитектура позволяет улучшить производительность приложения и обеспечить его безопасность. В целом, архитектура в клиентской части приложения является необходимым элементом для создания качественного и эффективного приложения.

Архитектуру проекта можно увидеть на рисунке 3.4.

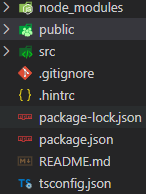


Рисунок 3.4 – Архитектура приложения

Каталог node\_modules содержит все внешние JavaScript-библиотеки, которые используются в приложении.

Директория public содержит базовые файлы HTML, JSON и изображений. Это корневые ресурсы проекта.

В директории src содержится код React-проекта. Она является основной директорией, где размещается весь код, относящийся к клиентской части приложения.

Файл .gitignore содержит список директорий и файлов, которые система контроля версий Git будет игнорировать, включая директорию node\_modules.

README.md - это файл разметки, который содержит полезную информацию о приложении Create React App, включая описание команд и ссылки на дополнительные настройки конфигурации.

Файл tsconfig.json используется для настройки компилятора TypeScript и определения параметров сборки проекта. Он содержит информацию о версии TypeScript, путях к файлам и директориям проекта, используемых библиотеках, настройках компиляции и других параметрах.

Более детально изучим содержимое каталога src, поскольку именно здесь располагается основной код клиентского приложения. Структура каталога src показана на рисунке 3.5.

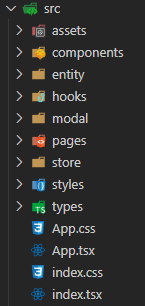


Рисунок 3.5 – Содержимое директории src

Рассмотрим подробнее папки, представленные на рисунке, и их назначение:

* assets - хранит в себе статические файлы такие как иконки и изображение;
* components - хранит в себе компоненты приложения;
* entity - содержит в себе файлы с определением типов данных;
* hooks - директория для хранения созданных хуков;
* modal – папка в которой хранится диалоговое окно;
* pages – хранит в себе все страницы приложения;
* store - содержит содержатся файлы и папки, связанные с управлением состоянием приложения;
* styles – хранит в себе все общие стили приложения;
* types - содержит файлы с определениями типов данных;
* app.tsx – этот файл содержит главный компонент приложения, который является точкой входа для компонентов и логики React-приложения, содержит маршрутизацию;
* index.tsx – этот файл является точкой входа React-приложения. В нем происходит инициализация и запуск приложения.

Рассмотрим реализацию компонентов. Компоненты являются основные основными блоками для создания пользовательских интерфейсов. Они могут быть созданы как классы или функции и могут содержать различные элементы, такие как текст, изображения, формы и другие компоненты. Компоненты могут быть многократно использованы и могут быть вложены друг в друга для создания более сложных интерфейсов. Также в React они позволяют использовать пропсы и состояние для динамического изменения содержимого и поведения компонента.

Все компоненты приложения показаны на рисунке 3.6.

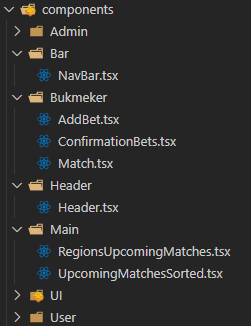


Рисунок 3.6 – Содержимое папки components

Как видно все компоненты рассортированы по страницам использования. Это нужно для более удобного поиска и использования. Пример кода компонента для создания команды представлен в приложении А.

Далее рассмотрим структуру директории pages. Она представлена на рисунке 3.7 ниже.

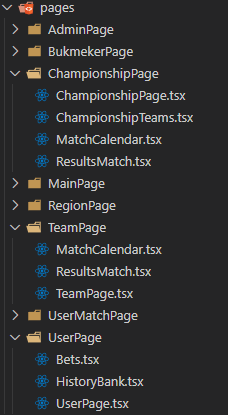


Рисунок 3.7 – Содержимое папки pages

Данная папка была разбита на следующие сущности:

* AdminPage – папка, которая содержит страницы администратора, к которым имеет доступ только администратор;
* BukmekerPage – папка, которая содержит страницы букмекера, к которым имеет доступ только букмекер;
* ChampionshipPage – папка, которая содержит страницы чемпионата, к которым имеет доступ только администратор;
* MainPage – папка, которая содержит главную страницу;
* RegionPage – папка, которая содержит страницы региона, к которым имеет доступ только администратор;
* TeamPage – папка, которая хранит страницы команды, к которым имеет доступ только администратор;
* UserMatchPage – папка которая содержит страницы матча;
* UserPage – папка, которая содержит страницы пользователя, к которым имеет доступ только пользователь.

Рассмотрим директории ChampionshipPage и UserPage. Ниже, на рисунке 3.8 представлена структура директории ChampionshipPage:

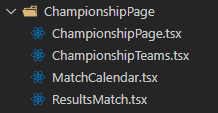


Рисунок 3.8 – Структура директории ChampionshipPage

Данная директория была разделена на следующие страницы:

* ChampionshipPage – это основная страница чемпионата, которая содержит подстраницы, что указаны ниже;
* ChampionshipTeams – страница, которая содержит команды, которые участвуют в чемпионате;
* MatchCalendar – страница, которая содержит в себе календарь матчей, которые будут сыграны в чемпионате;
* ResultsMatch – страница, которая результаты матчей, сыгранных в чемпионате.

Пример кода страницы календаря матчей указан в листинге 3.4.

|  |
| --- |
| import { useNavigate } from "react-router-dom";  import { useCalendarOfChampionshipMatches } from "../../hooks/match";  import { Match } from "../../components/Admin/Championship/Match";  import { useState } from "react";  import { Modal } from "../../modal/Modal";  import { CreateMatch } from "../../components/Admin/Championship/CreateMatch";  interface Props {  id: number;  }  export function MatchCalendar({ id }: Props) {  const [createMatch, setCreateMatch] = useState(false)  const { matches, error, loading } = useCalendarOfChampionshipMatches(id);  return (  <div>  <Modal active={createMatch} setActive={setCreateMatch}>  <CreateMatch id={id} />  </Modal>  <div><button style={{borderRadius:"3px"}} onClick={() => setCreateMatch(true)}>Создать матч</button></div>  <div>  {matches.map((m) => (  <Match m={m} key={m.id} />  ))}  </div>  </div>  );  } |

Листинг 3.4 – Пример кода календаря матчей

Теперь рассмотрим структуру директории UserPage, которая представлена на рисунке 3.9.



Рисунок 3.9 - Структура директории UserPage

Данная директория была разделена на следующие страницы:

* UserPage – это основная страница пользователя личного кабинета, которая содержит подстраницы, что указаны ниже;
* Bets – страница, которая содержит все ставки, которые делал пользователь;
* HistoryBank – страница, которая показывает пользователю история счета.
* Пример кода страницы всех ставок пользователя указан в листинге 3.5.

|  |
| --- |
| import { useUserBets } from "../../hooks/bet"  import { IUserBet } from "../../entity/UserBets";  import styled from "styled-components";  interface Props {  id: number  }  export function UserBets({ id }: Props) {  const { bets, loading, error } = useUserBets(id);  return <div>  <BetItem>  <div style={{ width: "50%" }}>Матч</div>  <div style={{ width: "15%" }}>Ставка</div>  <div style={{ width: "10%" }}>Сумма(BYN)</div>  <div style={{ width: "25%" }}>Статус</div>  </BetItem>  {loading && <>Загрузка</>}  {bets.map((b) => (  <Bet b={b} key={b.id} />  ))}  </div>  } |

Листинг 3.5 – всех ставок пользователя

Разработка большого количества страниц веб-приложения была осуществлена для обеспечения максимального комфорта и интуитивного взаимодействия пользователя с приложением. Разделение функциональности на отдельные страницы всегда является удобным решением для обеспечения удобства использования приложения.

3.3 Вывод по разделу

В разделе о реализации веб-приложения были описаны этапы разработки. Затронуты технологии и инструменты программирования, используемые в процессе разработки. Для соответствия поставленным задачам дипломного проекта были выбраны архитектуры проектов обеспечивают дальнейшую поддержку и расширение функционала. После этого были настроены и сконфигурированы обе части проекта таким образом, чтобы они взаимодействовали друг с другом. Кроме того, были реализованы основные функциональные возможности и применены ранее разработанные алгоритмы приложения.

4 Тестирование веб-приложения

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*25*

*БГТУ 03.00.ПЗ*

Разраб*.*

*Гой М.А.*

Провер*.*

*.*

*Комарова Е.И.*

Н*.* контр*.*

*Николайчук А.Н.*

Утв*.*

*Смелов В.В.*

3 Разработка программного средства

*Лит.*

*Листов*

*10*

*7421877-C, 2023*

Тестирование приложения - это процесс проверки функциональности, производительности, безопасности и качества приложения перед его выпуском на рынок. Тестирование проводится для обнаружения ошибок, уязвимостей и недостатков в приложении, а также для улучшения его работы и повышения удобства использования.

В процессе тестирования приложения используются различные методы и технологии, такие как ручное тестирование, автоматизированное тестирование, функциональное тестирование, тестирование производительности, тестирование безопасности и другие.

В процессе тестирования часто используются тест-кейсы. Простейший тест-кейс включает:

* краткое описание тестируемого объекта;
* последовательность действий для проверки его функциональности
* ожидаемый результат.

После выполнения тест-кейса тестировщик сравнивает фактический результат с ожидаемым. Если они не совпадают, то это указывает на наличие ошибки в программе.

В данном разделе описывается процесс тестирования функционала программного модуля, который был разработан в рамках данного дипломного проекта. Также проводится тестирование определенного функционала системы управления учебным процессом вуза, который необходим для доступа и корректной работы модуля успеваемости в этой системе.

4.1 Ручное тестирование

Ручное тестирование — это процесс поиска ошибок в программе без использования специального программного обеспечения, силами человека[]. Тестировщик имитирует реальные действия пользователя и старается охватить максимум функций продукта и найти ошибки Специалист по QA ищет недоработки в визуале, функционале, логике ПО, проверяет его надежность и удобство.

Для отображения тестируемых элементов были составлены таблицы тест-кейсов. В таблице 4.1 представлены тесты для роли Администратора.

Таблица 4.1 – Функциональные тест-кейсы

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Статус |
| Создание региона | Вывод окна для создания региона | Успешно |

Окончание таблицы 4.1

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Статус |
| Создание чемпионата | Вывод окна для создания чемпионата с выбором региона | Успешно |
| Создание команды | Вывод окна для создания команды с выбором региона | Успешно |
| Создание матча | Вывод окна для создания матча с выбором команд и назначением времени | Успешно |
| Удаление матча | Вывод возвращение на страницу чемпионата и обновление списка матчей | Успешно |
| Изменение роли пользователя | Вывод окна для выбора новой роли пользователя | Успешно |
| Изменение региона | Вывод окна для изменения названия региона и выбора изображения | Успешно |
| Создание региона без изображения | Создание нового региона без вывода изображения | Успешно |
| Добавление результата матча | Вывод счета результата матча | Успешно |
| Выход из учетной записи | Переход на главную страницу | Успешно |

Далее было проведено тестирование для Букмекера. Результаты тестирования представлены в таблице 4.2.

Таблица 4.2 Функциональные тест-кейсы интерфейса Букмекера

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Статус |
| Создание исхода матча | Вывод окна добавление исхода матча | успешно |
| Ввод уже существующего исхода матча | Вывод об ошибке, что такой исход уже Вуществует | успешно |
| Отправка невалидных данных при создания нового исхода | Вывод ошибки при создания исхода матча | успешно |
| Выбор нужного матча | Переход на страницу выбранного матча | успешно |
| Ввод невалидных данных при изменение исходов | Вывод ошибки при сохранении матча | успешно |

Также было проведено тестирование интерфейса для Пользователя, результаты тестирования представлены в таблице 4.3.

Таблица 4.3 Функциональные тест-кейсы интерфейса Букмекера

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Статус |
| Авторизация | Успешная авторизация. Переход на главную страницу | Успешно |
| Выход из аккаунта пользователя | Переход на главную страницу | Успешно |
| Авторизация без заполнения полей. | Приложение уведомляет о пустых полях | Успешно |
| Сделать ставку на сумму меньше двух рублей | Вывод об ошибке что минимальная сумма ставки равняется два рубля | Успешно |

Окончание таблицы 4.3

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Статус |
| Сделать ставку на тот же самый матч | Вывод об ошибке что на этот матч уже сделана ставка | Успешно |
| Пополнение счета без выбора карты | Вывод окна для добавления новой карты | Успешно |
| Пополение счета на сумму меньше двух рублей | Вывод ошибки о пополнения счета меньше двух рублей | Успешно |
| Выбор вкладки «История счета» | Переход на вкладку «История счета» | Успешно |

Описанные выше тест-кейсы помогают протестировать систему прямо из интерфейса, что значительно уменьшает время на написание тестов.

4.2 Тестирование валидации

Валидация - это процесс проверки соответствия продукта или системы требованиям и ожиданиям пользователей и заказчиков[]. Валидация помогает убедиться в том, что разработанный продукт или система выполняют свои функции и соответствуют заданным критериям качества.

Существует несколько видов валидации:

* мгновенная валидация - происходит одновременно с пользовательским вводом информации.
* валидация после потери фокуса - происходит после переключения на другое поле или после нажатия на другую область;
* валидация после отправки формы - начинается после нажатия кнопки отправки формы.

В разработанном веб-приложении в основном используется валидация после отправки формы.

Результат тестирования валидации пустых полей на форме регистрации показан на рисунке 4.1. Продемонстрированная валидация является после отправки формы.

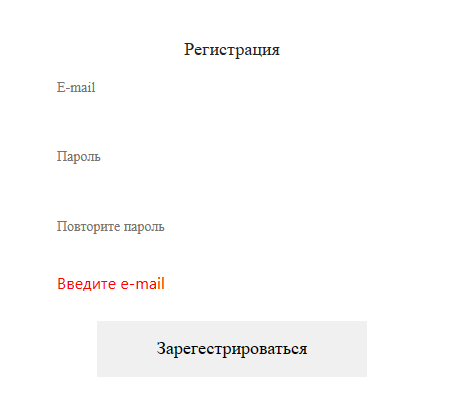


Рисунок 4.1 – Результат пустого поля электронной почты

Также валидация есть и для пароля, если пользователь вводить меньше восьми символов при регистрации. Результат показан на рисунке 4.2.

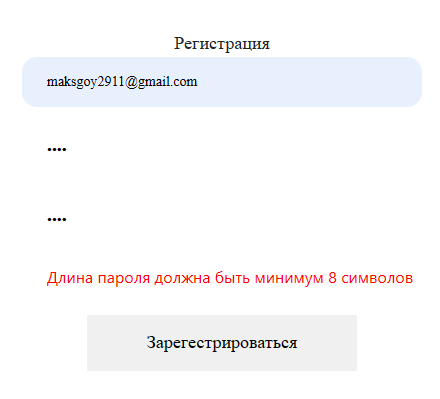


Рисунок 4.2 – Валидация пароля на форме регистрации

При ставке на футбольное событие пользователь должен ввести сумму ставки. При вводе суммы ставки меньше двух рублей пользователю показывается ошибка. Результат представлен на рисунке 4.3.

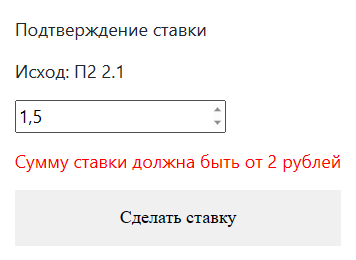


Рисунок 4.3 – Валидация на сумму ставки

При добавлении нового исхода на матча, Букмекер вводить название нового исхода. Если такой исход уже существует, выводится соответствующая ошибка. Результат показан на рисунке 4.4.

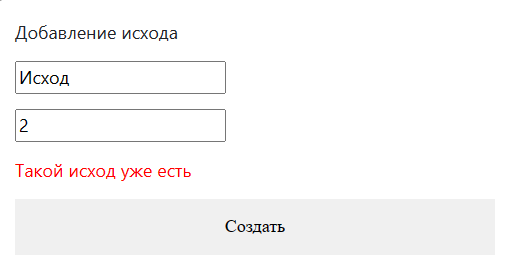


Рисунок 4.4 – Результат добавления существующего исхода

Также присутствует валидация на ввод количества исходов, который не должен превышать десяти. Результат ввода неверного количества исхода показан на рисунке 4.5.

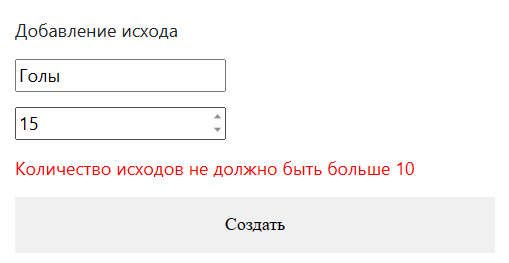


Рисунок 4.5 – Ввод неверного количества исходов

При создании нового региона Администратору неоходимо ввести название нового региона. Если регион с таким название уже существует ему будет показана ошибка. Результата выполнения показан на рисунке 4.6.

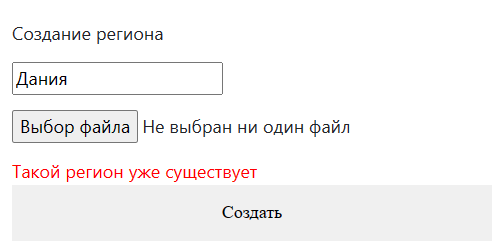


Рисунок 4.6 – Результат создания существующего региона

Также при создании нового региона необходимо чтобы название было минимум три символа. Если название будет меньше трёх символа, то Администратору покажется ошибка. Результат тестирования показан на рисунке 4.7.

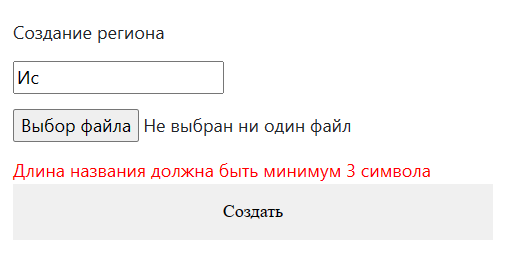


Рисунок 4.7 – Результат создания региона с двумя символами

4.3 Модульное тестирование

Модульное тестирование, иногда блочное тестирование или юнит-тестирование — процесс в [программировании](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), позволяющий проверить на корректность отдельные модули [исходного кода](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) программы, наборы из одного или более программных модулей вместе с соответствующими управляющими данными, процедурами использования и обработки[].

Идея состоит в том, чтобы писать тесты для каждой нетривиальной функции или метода. Это позволяет достаточно быстро проверить, не привело ли очередное изменение кода к [регрессии](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%81%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), то есть к появлению ошибок в уже оттестированных местах программы, а также облегчает обнаружение и устранение таких ошибок.

Для проведения Unit-тестирования в среде .Net доступно множество библиотек, которые можно использовать для проверки правильности работы кода. Я выбрал библиотеку XUnit, которая является бесплатной и поддерживается несколькими языками, использующими платформу .Net.

Для упрощения тестирования и абстрагирования его от основной логики приложения, была создана дополнительная DLL-библиотека, содержащая зависимости от двух предыдущих библиотек.

Одной из особенностей модульного тестирования является использование "фейковых" объектов, которые заменяют инфраструктурные элементы приложения, такие как базы данных или контекст контроллера. Для создания таких объектов я использовал фреймворк Moq, который позволяет создавать объекты классов, полностью соответствующие поведению инфраструктурных элементов.

Moq – это простой и легковесный изоляционный фреймворк, который построен на основе анонимных методов и деревьев выражений. Для создания моков он использует кодогенерацию, поэтому позволяет «мокать» интерфейсы, виртуальные методы (и даже защищенные методы) и не позволяет «мокать» невиртуальные и статические методы[].

Если описывать сами тесты, то на каждый метод была создана пара из «позитивного» и «негативного» теста, иногда было больше «позитивных», иногда «негативных» тестов для проверки работы модуля во всех ситуациях. Их отличие заключается в том, что в первом случае на вход тесту предоставляются некоторые данные и идет расчет, что эти данные будут корректно отработаны и получится желаемый результат.

Пример «позитивного» теста, в котором мы ожидаем получить токен пользователя, можно увидеть ниже на листинге 4.1.

|  |
| --- |
| [Fact]  public async void Login()  {  IUnitOfWork unitOfWork = new UnitOfWork();  ITokenService tokenService = new TokenService();  IHashPassword hashPassword = new HashPasswordService();  IMapper mapper;  var mapperConfig = new MapperConfiguration(mc =>  {  mc.AddProfile(new AutoMapperProfiles());  });  mapper = mapperConfig.CreateMapper();  var account = new AccountController(unitOfWork, tokenService, hashPassword, mapper);  LoginDto login = new()  {  Email = "444",  Password = "444",  };  var result = await account.Login(login);  var token = new JwtSecurityToken(jwtEncodedString: result.Value.Token);  string email = token.Claims.First(c => c.Type == "email").Value;  string role = token.Claims.First(c => c.Type == "role").Value;  Assert.Equal(email, login.Email);  } |

Листинг 4.1 -«Позитивный» Unit-тест

Во втором же случае, специально подаются некорректные данные, которые должны привести к ошибочному результату или вообще выбросить исключение. Делается это, прежде всего, для проверки корректной работы самого метода, затем обработчика ошибок, если он там присутствует, то убедиться, не вызовет ли это сбой в работе всего приложения, а также проверки отдельных условий в методе.

Пример созданного «негативного теста», в котором проверяется создание исключения в методе для хеширования пароля, представлен на листинге 4.2.

|  |
| --- |
| [Fact]  public void ShouldThrowInvalidPassword()  {  var password = string.Empty;  var hashPassword = new HashPasswordService();  Assert.Throws<Exception>(() => hashPassword.CreateHash(password));  } |

Листинг 4.3 - «Негативный» Unit-тест

Все тесты, которые были запланированы успешно прошли работу, что означает корректную работу модулей всего веб-сервиса. Все исключительные ситуации, которые могли привести к нестабильной работе приложения, были найдены и исправлены.

4.4 Вывод по разделу

В данном разделе был описан процесс тестирования разработанного веб-приложения для букмекерских ставок на футбольные события. Тестирование проводилось по предварительно написанным тест-кейсам. Также в процессе тестирования проводилась проверка валидации форм. Кроме ручного тестирования над задачами проводилось модульное тестирование. Все вышеперечисленные виды тестирования могут уверять в том, что модуль работает корректно и шанс появления непредвиденных ошибок у пользователей минимизирован.

Полученные результаты при ручном и модульном функциональном тестировании показали, что разработанное программное средство работает правильно и корректно.

5 Руководство программиста

В данном разделе дипломного проектирования будут рассмотрены основные аспекты работы с приложением. Чтобы облегчить изучение, раздел разбит на подразделы в соответствии с ролями пользователей.

5.1 Роль «Администратор»

5.2 Роль «Пользователь»

5.3 Роль «Букмекер»

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*25*

*БГТУ 03.00.ПЗ*

Разраб*.*

*Гой М.А.*

Провер*.*

*.*

*Комарова Е.И..*

Н*.* контр*.*

*Николайчук А.Н.*

Утв*.*

*Смелов В.В.*

3 Разработка программного средства

*Лит.*

*Листов*

*10*

*7421877-C, 2023*

6 Технико-экономическое обоснование проекта

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *БГТУ 06.00 ПЗ* | | | | | |
|  |  |  |  |  |
|  |  | Ф.И.О | Подпись | Дата |
| Разраб. | | Гой М.А.. |  |  | 6 Технико-экономическое обоснование проекта | Лит. | | | Лист | Листов |
| Пров. | | Комарова Е.И. |  |  |  |  |  | 1 |  |
| Консульт. | | Семёнова Л.С. |  |  | \_\_\_\_\_\_\_, 2023 | | | | |
| Н. контр. | | Николайчук А.Н. |  |  |
| Утв. | | Смелов В.В. |  |  |

Разработка программного обеспечения требует вложения различных ресурсов, включая трудовые, материальные и финансовые. Поэтому каждый проект должен быть обоснован как с технической, так и экономической точек зрения, чтобы определить его целесообразность и эффективность.

6.1 Общая характеристика разрабатываемого программного средства

Целью дипломного проекта является разработка веб-приложения для букмекерских ставок на футбольные события, которое позволяет клиентам пополнять личный счет и делать ставки на футбольные события.

На рынке приложений имеется некоторое количество аналогов представленному в дипломном проекте программному средству, предназначенных для ставок на спорт. Для использования программного средства достаточно иметь браузер. Программный продукт относится к программным средствам общего назначения.

Пользователю разработанного веб-приложения будут доступны такие функции как:

* авторизация и регистрация;
* просмотр списка матчей;
* сортировка матчей по чемпионатам
* пополнение личного счета и вывод средств;
* просмотр истории счета;
* просмотр истории ставок;
* ставить на футбольные события.

Во время разработки дипломного проекта были использованы технологии ASP.NET Core 6.0, Entity Framework Core 7.0.4, MS SQL 15.0, React.JS 18.2.0.

В данном разделе требуется определить все расходы, которые были сделаны на каждой из стадий в процессе разработки программного продукта. Также нужно провести расчет экономии основных ресурсов, которая может быть достигнута при использовании данного программного продукта.

6.2 Исходные данные для проведения расчётов и маркетинговый анализ

Источниками исходных данных для данных расчетов выступают действующие нормативные правовые акты. Исходные данные для расчета приведены в таблице 6.1.

Таблица 6.1 – Исходные данные для расчета (приводится то, что используется в расчетах)

|  |  |  |
| --- | --- | --- |
| Наименование показателя | Условные обозначения | Норматив |
| Численность разработчиков, чел | Чр | 1 |
| Норматив дополнительной заработной платы, % | Ндз | 15 |
| Ставка отчислений в Фонд социальной защиты населения, % | Нфсзн | 34 |
| Ставка отчислений по обязательному страхованию в БРУСП «Белгосстрах», % | Нбгс | 0,6 |
| Норматив прочих прямых затрат, % | Нпз | 20 |
| Норматив накладных расходов, % | Нобп, обх | 50 |
| Ставка НДС, % | ННДС | 0 |
| Налог на прибыль, % | Нп | 0 |

Ставка НДС и налог на прибыль равняются 0, так как практика проходила в компании-резиденте ПВТ.

Источниками исходных данных для расчёта средней цены разработки программного средства данного дипломного проекта, выступает таблица 6.2 с ценами аналогичных программных продуктов.

Таблица 6.2 – Стоимость разработки аналогичных программных продуктов

|  |  |  |  |
| --- | --- | --- | --- |
| Продукт-аналог | Источник | Стоимость | Примечание |
| Betera | https://pm.by/ru | 10700 | Букмекерская контора для ставок на спорт. Имеет функционал, которого не будет в нашем приложении, например, карту движения поезда. В нашем приложении этого не будет, что позволит сократить цену разработки. |
| Фонбет | https://www.fonbet.by/ | 15500 | Букмекерская компания, которая имеет огромный выбор исходов матчей, позволяет быстро поплнять счет и выводить средства на карту. |
| MaxLine | https://maxline.by/ | 9800 | Букмекерская компания, которая позволяет отслеживать коэффициента в онлайн режиме |

Исходя из данных таблицы 6.2, стоимость разработки аналогичных программных продуктов составляет 9800 – 15500 рублей. Возьмём среднее арифметическое полученных стоимостей.

Таким образом, общая стоимость разработки данного программного средства, выбранного в качестве базы сравнения, составит 12000 рублей.

6.3 Обоснование цены программного средства

В условиях современной рыночной экономики программное обеспечение, как правило, разрабатывается и выпускается организациями в форме готовых функциональных продуктов, которые продается потребителям по рыночной цене. Разработки программного обеспечения тесно связаны с научно-технической продукцией.

Для разработчика программного обеспечения экономический эффект проявляется через создание чистой прибыли от продаж программного обеспечения, которая остается в распоряжении организации. При оценке стоимости программного обеспечения у компаний-разработчиков учитываются различные расходы, включающие в себя следующие статьи:

* оплата труда исполнителей, включая основную и дополнительную заработную плату;
* взносы в фонд социальной защиты населения;
* платежи по обязательному страхованию от несчастных случаев на производстве и профессиональных заболеваний;
* накладные расходы;
* другие прямые расходы.

На основании затрат рассчитывается себестоимость и отпускная цена конечного программного средства.

6.3.1 Расчёт затрат рабочего времени на разработку программного средства

В таблице 6.3 в укрупнённом виде указаны все работы, реально выполненные для создания, указанного в дипломной работе программного средства и количество рабочих часов, реально потраченных для выполнения этих работ.

Таблица 6.3 – Затраты рабочего времени на разработку ПС

|  |  |
| --- | --- |
| Содержание работ | Затраты рабочего времени, часов |
| Разработка концепции приложения | 43 |
| Организация ввода информации | 25 |
| Контроль, предварительная обработка | 27 |
| Управление вводом/выводом | 18 |
| Взаимодействие между компонентами системы | 50 |
| Взаимодействие с базой данных | 40 |
| Вспомогательные методы | 35 |

Продолжение таблицы 6.3

|  |  |
| --- | --- |
| Обработка ошибочных и сбойных ситуаций | 40 |
| Графический вывод результатов | 40 |
| Тестирование программных модулей | 25 |
| Доработка программного средства | 40 |
| Всего | 383 |

Общее время разработки занимает 383 часов.

6.3.2 Расчет основной заработной платы

Было проведено исследование заработных плат специалистов в области веб-программирования, работающих с платформой ASP.NET и React.js. Для этого использовались открытые веб-порталы, форумы, официальная отчетность и средний уровень заработка в сфере информационных технологий. Изучив и проанализировав полученные данные, было установлено, что средняя месячная заработная плата на позиции junior составляет 1600 рублей, часовая ставка составляет 9,52 руб/час.

Согласно таблице 6.2, проект разрабатывался одним специалистом на протяжении 383 часов. Таким образом, основная заработная плата будет рассчитываться по формуле 6.3

|  |  |  |
| --- | --- | --- |
|  | , | (6.3) |

где Соз– основная заработная плата, руб.;

Траз – время раработки (часов);

Сзп – средняя часовая ставка руб./час;

Краз – количество разработчиков, человек.

|  |  |  |
| --- | --- | --- |
|  | руб*.* |  |

В дальнейшем для других расчётов используется основная заработная плата, рассчитанная по указанной выше методике.

6.3.3 Расчет дополнительной заработной платы

Дополнительная заработная плата представляет собой выплаты, предусмотренные законодательством о труде, включает компенсирующие выплаты (например, доплаты за работу в сверхурочное время, в государственные праздники, праздничные и выходные дни) и определяется по нормативу в процентах к основной заработной плате по формуле (6.4).

|  |  |  |  |
| --- | --- | --- | --- |
|  | , | (6.4) | (6.4) |

где Соз– основная заработная плата, руб.;

Ндз – норматив дополнительной заработной платы, %.

руб*.*

6.3.4 Расчет отчислений в Фонд социальной защиты населения и по обязательному страхованию

Отчисления в Фонд социальной защиты населения (ФСЗН) и по обязательному страхованию от несчастных случаев на производстве и профессиональных заболеваний в БРУСП «Белгосстрах» определяются в соответствии с действующими законодательными актами по нормативу в процентном отношении к фонду основной и дополнительной зарплаты исполнителей.

Отчисления в Фонд социальной защиты населения вычисляются по формуле 6.5

|  |  |  |
| --- | --- | --- |
|  |  | (6.5) |

где – основная заработная плата, руб.;

– дополнительная заработная плата на конкретное ПС, руб.;

– норматив отчислений в Фонд социальной защиты населения, %.

Отчисления в БРУСП «Белгосстрах» вычисляются по формуле 6.6

|  |  |  |
| --- | --- | --- |
|  |  | (6.6) |

Таким образом, общие отчисления в БРУСП «Белгосстрах» составили 25,16 руб., а в фонд социальной защиты населения – 1425,65 руб.

6.3.5 Расчет суммы прочих прямых затрат

Сумма прочих затрат Спз определяется как произведение основной заработной платы исполнителей на конкретное программное средство Соз на норматив прочих затрат в целом по организации Нпз, и находится по формуле 6.9.

|  |  |  |
| --- | --- | --- |
|  |  | (6.7) |

Сумма прочих затрат составит:

Спз = = 546,92 руб.

6.3.6 Расчет суммы накладных расходов

Сумма накладных расходов Собп,обх – произведение основной заработной платы исполнителей на конкретное программное средство Соз на норматив накладных расходов в целом по организации Нобп,обх, по формуле 6.8.

|  |  |  |
| --- | --- | --- |
|  |  | (6.8) |

Сумма накладных расходов составит:

= 3646,16 ⋅ 50 / 100 = 1823,08 руб.

6.3.7 Сумма расходов на разработку программного средства

Сумма расходов на разработку программного средства Ср определяется как сумма основной и дополнительной заработных плат исполнителей на конкретное программное средство, отчислений на социальные нужды, суммы прочих затрат и суммы накладных расходов, по формуле 6.9. (если что-то еще использовали в расчетах, то тоже указываем тут).

|  |  |  |
| --- | --- | --- |
|  | Ср = Соз + Сдз + Сфсзн + Сбгс + Спз + Собп,обх. | (6.9) |

Все данные необходимые для вычисления есть, поэтому можно определить сумму расходов на разработку программного средства.

Ср = 3646,16 + 364,62 + + 25,16 + 546,92 + 1823,08 = 7831,59 руб.

Сумма расходов на разработку программного средства была вычислена на основе данных, рассчитанных ранее в данном разделе, и составила 7831,59 рублей.

6.3.8 Определение цены, оценка эффективности

Исходя из цен аналогов, определим цену реализации ПС равную 12000 рублей. Прибыль, рассчитывается по формуле 6.10.

|  |  |  |
| --- | --- | --- |
|  |  | (6.10) |

руб.

Поскольку компании на базе которой проводится расчет является резидентом ПВТ, налог на прибыль составляет 0, прибыль равняется чистой прибылью

Рентабельность, определяется как процентное отношение прибыли к себестоимости и рассчитывается по формуле 6.11.

|  |  |  |
| --- | --- | --- |
|  |  | (6.11) |

%.

Таким образом рентабельность составит 53,098 процента.

6.4 Вывод по разделу

В таблице 6.4 представлены результаты расчётов для основных показателей данной главы в краткой форме.

Таблица 6.4 – Результаты расчетов

|  |  |
| --- | --- |
| Наименование показателя | Значение |
| Время разработки, ч. | 383 |
| Количество разработчиков, чел. | 1 |
| Основная заработная плата, руб. | 3646,16 |
| Дополнительная заработная плата, руб. | 364,62 |
| Отчисления в Фонд социальной защиты населения и БРУСП «Белгосстрах», %, руб. | 1425,65 |
| Отчисления в БРУСП «Белгосстрах», руб. | 25,16 |
| Прочие прямые затраты, руб | 546,92 |
| Полная себестоимость, руб. | 7851,39 |
| Чистая прибыль, руб | 4158,41 |
| Рентабельность, % | 53,098 |

Разработка программного средства, осуществляемая одним программистом в течении 383 часов, при заданных условиях обойдется в 7851,39 руб.

При разработке данного программного продукта были учтены преимущества технологии ASP.Net Core, которая обеспечивает легкую расширяемость функционала и независимость компонентов. Также использование бесплатной технологии позволило снизить полную себестоимость проекта. Однако, следует учитывать, что проект был реализован только одним программистом, поэтому для его дальнейшего развития и поддержания необходимы качественная документация, понятное объяснение кода и правильная архитектура проекта.

Приложение может распространятся бесплатно, но при условии его продажи заказчику можно получить экономический эффект, в виде чистой прибыли в 4158,41 руб., исходя из цен аналогов и рентабельности в 53,098 %.

Заключение

Список используемых источников

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *БГТУ 06.00 ПЗ* | | | | | |
|  |  |  |  |  |
|  |  | Ф.И.О | Подпись | Дата |
| Разраб. | | Гой М.А.. |  |  | 6 Технико-экономическое обоснование проекта | Лит. | | | Лист | Листов |
| Пров. | | Комарова Е.И. |  |  |  |  |  | 1 |  |
|  | |  |  |  | \_\_\_\_\_\_\_, 2023 | | | | |
| Н. контр. | | Николайчук А.Н. |  |  |
| Утв. | | Смелов В.В. |  |  |

1. «Betera» [Электронный ресурс]. – Режим доступа: <https://pm.by/>. – Дата доступа: 18.04.2023
2. «Fonbet» [Электронный ресурс]. – Режим доступа: https://www.fonbet.by/. – Дата доступа: 18.04.2023
3. «MaxLine» [Электронный ресурс]. – Режим доступа: https://maxline.by. – Дата доступа: 18.04.2023
4. Документация по C# [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/csharp/ – Дата доступа: 20.04.2023
5. TypeScript [Электронный ресурс]. – Режим доступа: https://www.typescriptlang.org – Дата доступа: 20.04.2023
6. Документация по ASP.NET [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/aspnet/core/?view=aspnetcore-6.0 – Дата доступа: 20.04.2023
7. React JS [Электронный ресурс]. – Режим доступа: <https://ru.reactjs.org> – Дата доступа: 20.04.2023
8. Microsoft SQL Server [Электронный ресурс]. – Режим доступа: <https://www.microsoft.com/en-us/sql-server> – Дата доступа: 20.04.2023
9. Entity Framework [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/ru-ru/ef/> - Дата доступа: 20.05.2023
10. Ручное тестирование [Электронный ресурс]. – Режим доступа: <https://blog.skillfactory.ru/chto-takoe-ruchnoe-testirovanie> - Дата доступа: 23.05.2023
11. Валидация [Электронный ресурс]. – Режим доступа: https://secrets.tinkoff.ru/glossarij/validaciya/ - Дата доступа: 23.05.2023
12. Модульное тестирование [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Модульное_тестирование> - Дата доступа: 23.05.2023
13. Примеры использования Moq [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/150859/ - Дата доступа: 23.05.2023

Приложение А

Приложение Б

Приложение В