

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения 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 букмекерских ставок на футбольные события

Дипломник _____ Гой М.А.

Руководитель проекта _____ Комарова Е.И., преп.-стажер

Заведующий кафедрой _____ Смелов В. В., к.т.н., доцент

Консультант _____ Семенова Л. С., ст. преп.-стажер

Нормоконтролер _____ Николайчук А. Н., преп.-стажер

Дипломный проект защищен с оценкой _____

Председатель ГЭК _____ Дюбков В. К., к.т.н., доцент

Лист задания 1

Лист задания 2

Реферат

Пояснительная записка содержит 71 страницу, 56 рисунков, 18 таблиц, 15 источников литературы, 8 приложений.

ВЕБ-ПРИЛОЖЕНИЕ, C#, ASP.NET CORE, ENTITY FRAMEWORK CORE, MSSQL SERVER, TYPESCRIPT, STYLED COMPONENTS, REACT, REDUX, РЕДАКТОР КОДА VISUAL STUDIO CODE

Целью данного дипломного проекта является разработка веб-приложения для букмекерских ставок на футбольные события.

Пояснительная записка состоит из введения, шести разделов и заключения.

В первом разделе дипломного проекта приведен аналитический обзор предметной области и постановка задачи по теме дипломного проекта.

Во втором разделе представлены основные подходы к построению программной части дипломного проекта и описан выбор технологий для разработки веб-приложения.

В третьем разделе рассмотрены этапы разработки и программной реализации дипломного проекта.

В четвертом разделе описано тестирование программного средства.

В пятом разделе находится руководство пользователя по эксплуатации разработанной программы.

Шестой раздел содержит экономические расчеты.

В заключении приведены результаты проделанной работы, приводятся соображения насчет использования данного программного средства.

				БГТУ 00.00.ПЗ		
				Abstract		
ФИО	Подпись	Дата		Лит.	Лист	Листов
Разраб. Гой М.А.						
Провер. Комарова Е.И.					1	1
Н. контр. Николайчук А.Н.				74218010, 2023		
Утв. Смелов В.В.						

Abstract

The explanatory note of the diploma project contains 71 pages, 56 figures, 18 tables, 15 literature sources, 8 appendices.

WEB APPLICATION, C#, ASP.NET CORE, ENTITY FRAMEWORK CORE, MS SQL SERVER, TYPESCRIPT STYLE COMPONENTS, REACT, REDUX, VISUAL STUDIO CODE EDITOR

The goal is to develop a web application for betting on football events.

The explanatory note consists of an introduction, six sections and a conclusion.

The first section of the diploma project provides an analytical overview of the subject area and a statement of the problem on the topic of the diploma project.

The second section presents the main approaches to building the software part of the diploma project and describes the choice of technologies for developing a web application.

In the third section, the stages of development and program implementation of the diploma project are considered.

The fourth section presents the conducted testing of the functional part of the application.

The fifth section contains the user's manual for the operation of the developed program.

The sixth section contains economic calculations.

In conclusion, the results of the work done are presented, considerations are given about the use of this software tool.

				БГТУ 00.00.ПЗ		
	ФИО	Подпись	Дата			
Разраб.	Гой М.А.			<i>Abstract</i>	Лит.	Лист
Провер.	Комарова Е.И.				У	1
Н. контр.	Николайчук А.Н.					1
Утв.	Смелов В.В.				74218010, 2023	

Содержание

Введение	7
1 Постановка задачи и анализ аналогичных решений	8
1.1 Обзор аналогичных решений	8
1.2 Приложение «Betera»	8
1.3 Приложение «Fonbet».....	11
1.4 Приложение «MaxLine»	14
1.5 Постановка задачи	16
1.6 Вывод по разделу	17
2 Проектирование веб-приложения	18
2.1 Диаграмма вариантов использования.....	18
2.2 Выбор средств реализации.....	18
2.3 Основные языки программирования	18
2.4 Выбор технологий и библиотек	19
2.4.1 Технология ASP.NET Core	19
2.4.2 Технология EntityFramework	20
2.4.3 Библиотека React	20
2.4.4 Библиотека Styled Components	21
2.4.5 Библиотека Redux	22
2.5 Система управления базами данных.....	22
2.6 Проектирование структурной схемы веб-приложения.....	23
2.7 Логическая схема базы данных	23
2.8 Вывод по разделу	28
3 Реализация веб-приложения	29
3.1 Разработка серверной части.....	29
3.1.1 Разработка репозиториев	31
3.1.2 Разработка сервисов	33
3.1.3 Разработка контроллеров	34
3.2 Разработка клиентской части	36
3.3 Вывод по разделу	41
4 Тестирование веб-приложения	42
4.1 Ручное тестирование	42
4.2 Тестирование валидации.....	44
4.3 Модульное тестирование	47
4.4 Вывод по разделу	49
5 Руководство пользователя	50
5.1 Роль «Пользователь»	50
5.2 Роль «Администратор»	53

				БГТУ 00.00.ПЗ		
	ФИО	Подпись	Дата			
Разраб.	Гой М.А.			Lит.	Лист	Листов
Провер.	Комарова Е.И.				1	2
Н. контр.	Николайчук А.Н.			Содержание		
Утв.	Смелов В.В.			74218010, 2023		

5.3 Роль «Букмекер»	59
5.4 Вывод по разделу	62
6 Технико-экономическое обоснование проекта.....	63
6.1 Общая характеристика разрабатываемого программного средства.....	63
6.2 Исходные данные для проведения расчётов и маркетинговый анализ.....	63
6.3 Обоснование цены программного средства.....	65
6.3.1 Расчёт затрат рабочего времени на разработку программного средства.....	65
6.3.2 Расчет основной заработной платы	66
6.3.3 Расчет дополнительной заработной платы	66
6.3.4 Расчет отчислений в Фонд социальной защиты населения и по обязательному страхованию	67
6.3.5 Расчет суммы прочих прямых затрат	67
6.3.6 Расчет суммы накладных расходов	68
6.4 Вывод по разделу	69
Заключение	70
Список используемых источников.....	71
ПРИЛОЖЕНИЕ А	72
ПРИЛОЖЕНИЕ Б	73
ПРИЛОЖЕНИЕ В	74
ПРИЛОЖЕНИЕ Г	75
ПРИЛОЖЕНИЕ Д	76
ПРИЛОЖЕНИЕ Е	77
ПРИЛОЖЕНИЕ Ж	80
ПРИЛОЖЕНИЕ И	81
ПРИЛОЖЕНИЕ К	86
ПРИЛОЖЕНИЕ Л	88
ПРИЛОЖЕНИЕ М	89
ПРИЛОЖЕНИЕ Н	90
ПРИЛОЖЕНИЕ П	91

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 делает ставку по своему желанию. Если ставка срабатывает, игрок выигрывает, если нет, то букмекер получает деньги, которые поставил игрок. Для игрока предлагается широкий выбор ставок, на которых он сможет испытать свою удачу.

Целью дипломного проекта является разработка веб-приложения для букмекерских ставок на футбольные события, который должен предоставлять интерфейс пользователю для ставок на футбольные события, а также предоставлять интерфейс администратору и модератору для управления матчами и исходами.

Для достижения цели проекта сформулированы следующие задачи:

- выбрать необходимые инструменты для разработки;
- разработать клиентскую часть;
- разработать серверную часть;
- реализовать возможность регистрации и авторизации пользователя;
- реализовать возможность делать ставку на футбольное событие;
- разработать руководство пользователя;
- рассчитать экономические показатели.

Практическая значимость: разработка веб-приложения которое позволит пользователям пополнять игровой счет, выводить средства, делать ставку на футбольное событие и отслеживать результаты ставок.

				БГТУ 00.00.ПЗ		
	ФИО	Подпись	Дата			
Разраб.	Гой М.А.			Lит.	Лист	Листов
Провер.	Комарова Е.И.			У	1	1
Н. контр.	Николайчук А.Н.					
Утв.	Смелов В.В.			74218010, 2023		
Введение						

1 Постановка задачи и анализ аналогичных решений

Перед тем как начать разрабатывать веб-приложение важно четко понимать, что именно должно получится в итоге. Поэтому для начала нужно проанализировать уже существующие аналоги, представляющие собой веб-приложение для букмекерских ставок на футбольные события и определить функционал будущего веб-приложения, который необходимо реализовать.

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 модального окна для входа в приложение показан на рисунке 1.1.

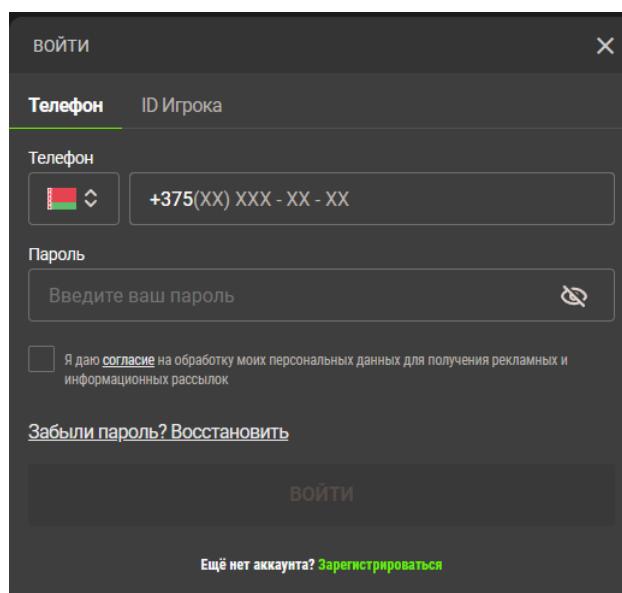


Рисунок 1.1 – Скриншот модального окна входа в приложение

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

				БГТУ 01.00.П3		
	ФИО	Подпись	Дата			
Разраб.	Гой М.А.				Lит.	Лист
Провер.	Комарова Е.И.				У	10
Н. контр.	Николайчук А.Н.			1 Постановка задачи и анализ аналогичных решений		
Утв.	Смелов В.В.					
				74218010, 2023		

После входа пользователь попадает на главную страницу. Скриншот главной страницы продемонстрировано на рисунке 1.2 ниже.

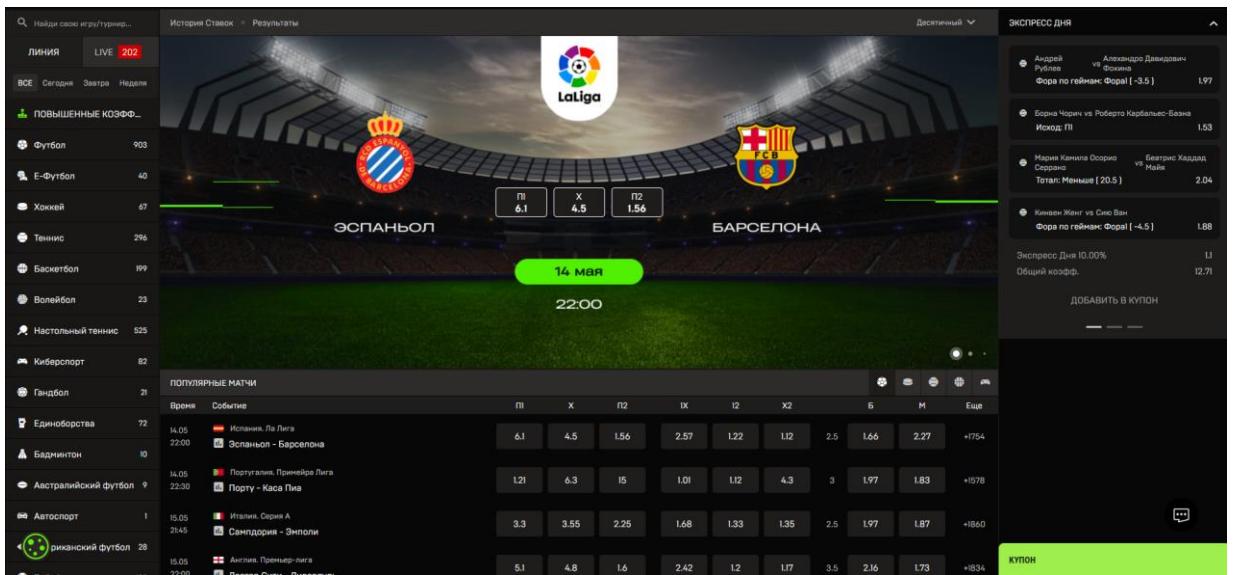


Рисунок 1.2 – Скриншот главной страницы приложения «Betera»

«Betera» является одним из самых популярных приложений. В этом приложении большой выбор ставок для игроков, а также удобный интерфейс. На странице 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 можно на этой странице ввести купон для получения бонусов в приложении. В правом верхнем углу можно пополнить счет, а также увидеть баланс. Возле всех матчей сразу доступна основная линия.

Для более детальной информации о матче можно открыть страницу матча. Скриншот страницы матча показан на рисунке 1.3.

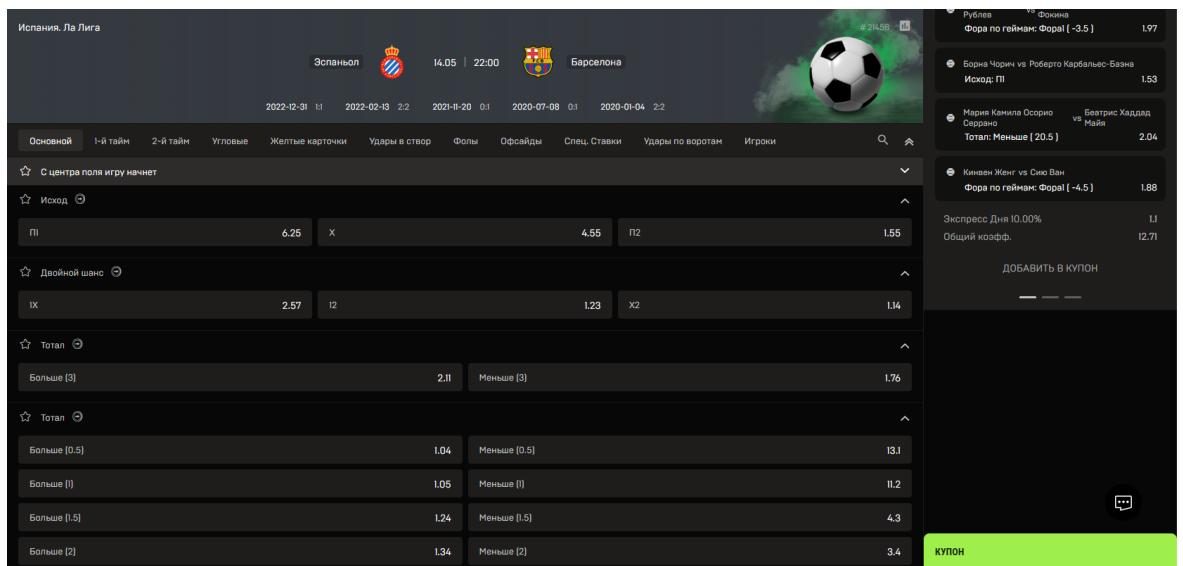


Рисунок 1.3 – Скриншот страницы матча

На странице матча можно увидеть все исходы на главную игру. Также видно название двух команд, которые проводят встречу, время и турнир. Для более удобного поиска все исходы разделены на отдельные страницы. 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. Скриншот модального окна подтверждения ставки показан на рисунке 1.4.

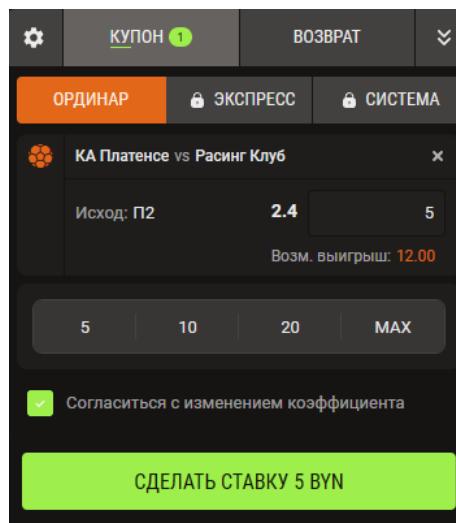


Рисунок 1.4 – Скриншот модального окна подтверждения ставки

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

Для того чтобы посмотреть последние матчи команд требуется нажать кнопку, находящуюся справа от команд. Скриншот страницы последних матчей команд представлены на рисунке 1.5.

ПОСЛЕДНИЕ ИГРЫ: ЭСПАНЬОЛ			
04.05.2023	Севилья	Эспаньол	3 : 2
30.04.2023	Эспаньол	Хетафе	1 : 0
27.04.2023	Вильярреал	Эспаньол	4 : 2
21.04.2023	Эспаньол	Кадис	0 : 0
15.04.2023	Бетис	Эспаньол	3 : 1
08.04.2023	Эспаньол	Атлетик	1 : 2
ПОСЛЕДНИЕ ИГРЫ: БАРСЕЛОНА			
02.05.2023	Барселона	Осасуна	1 : 0
29.04.2023	Барселона	Бетис	4 : 0
26.04.2023	Райо Вальекано	Барселона	2 : 1
23.04.2023	Барселона	Атлетико	1 : 0
16.04.2023	Хетафе	Барселона	0 : 0
10.04.2023	Барселона	Жирона	0 : 0

Рисунок 1.5 – Скриншот страницы последних матчей команд

Здесь видно 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»

Следующим аналогом, на который следует обратить внимание, является приложение «Fonbet» [2]. Скриншот модального окна для входа в приложение показан на рисунке 1.6.

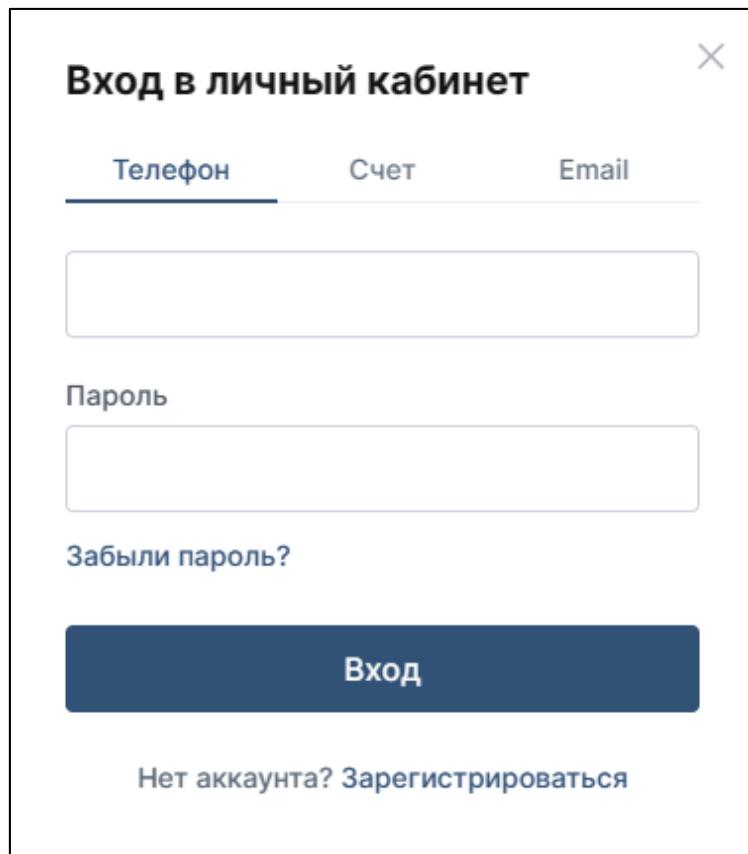


Рисунок 1.6 – Скриншот модального окна для входа в приложение «Fonbet»

На рисунке видно, что есть три 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 приложение перенаправляет на главную страницу, скриншот которой показан на рисунке 1.7.

The screenshot shows the Fonbet website's main page. At the top, there's a navigation bar with links like SPORT, ЛАЙВ, КАЗИНО, ТВ-ИГРЫ, LIVE КАЗИНО, ПРИЛОЖЕНИЯ, СТАТИСТИКА, РЕЗУЛЬТАТЫ, БОНУСЫ, ПОМОЩЬ, and Вход/Регистрация. Below the navigation, there's a search bar and a dropdown menu for 'All events'. The main content area is divided into sections for different sports: Football, Basketball, Basketball 3x3, Volleyball, Tennis, Table Tennis, Cyber Sport, and Handball. The Football section is expanded, showing various football tournaments and their fixtures. For example, the 'Лига Чемпионов УЕФА. 1/4 финала. Ответные матчи' section lists matches between Real Madrid vs Napoli, Manchester City vs Bayern Munich, and others. Each match entry includes team names, kick-off times, and detailed odds tables for outcomes like '1', 'X', '2', '1X', '12', 'X2', 'Фора 1', 'Фора 2', 'Total', and 'Баланс'.

Рисунок 1.7 – Скриншот главной страницы «Fonbet»

Также, как и в предыдущем приложении на главной странице видны все матчи, которые будут в ближайшее время. Первое отличие, которое видно 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 можно разглядеть еще основные матчи сегодняшнего дня и основные коэффициенты. В отличие от первого приложения матчи не сортированы как на популярные и не популярные, а идут все подряд. Как и в первом приложении можно выбрать турнир на панели слева. Здесь также присутствует страница матча, скриншот представлен на рисунке 1.8.

The screenshot shows the match page for Luton Town vs Sanderson. At the top, it displays the teams, date (16 мая 22:00), and stadium (Сандерленд). Below this, there are tabs for ИЗБРАННОЕ, ПОПУЛЯРНОЕ, МАТЧ, 1-Й ТАЙМ, and 2-Й ТАЙМ. The main content is organized into sections: Исходы (Outcomes), Проход (Run), and Тоталы (Totals). The 'Исходы' section shows odds for 'Лутон Таун' (2.03), 'Ничья' (3.45), 'Сандерленд' (3.70), and 'Лутон Таун или ничья' (1.28) with odds 1.30 for 'Лутон Таун или Сандерленд'. The 'Проход' section shows odds for 'Лутон Таун' (2.45) and 'Сандерленд' (1.55). The 'Тоталы' section shows odds for 'БОЛЬШЕ' (2.95) and 'МЕНЬШЕ' (1.40) for 'Тотал 0.5', and 'БОЛЬШЕ' (3.60) and 'МЕНЬШЕ' (1.30) for 'Тотал 1'.

Рисунок 1.8 – Скриншот страницы матча

На странице матча можно увидеть все исходы на главную игру. Также на странице видно название двух команд, которые проводят встречу, время встречи, и турнир. Для более удобного поиска все исходы разделены на отдельные страницы. 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. Скриншот модального окна подтверждения ставки показано на рисунке 1.9.

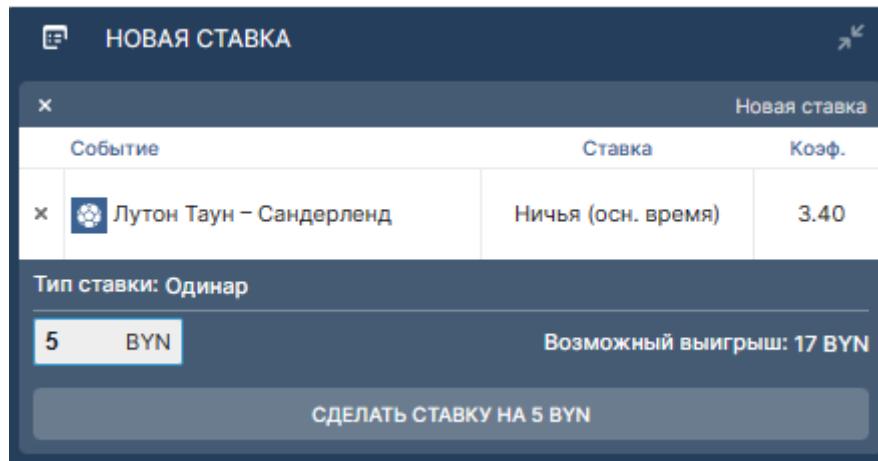


Рисунок 1.9 – Скриншот модального окна подтверждения ставки

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.

Последние встречи				
	Все	Кенилворт Роуд		
Турнир / Раунд	Дата	Матч		
Чемпионшип, Плей-офф	П 13/05/23	Сандерленд	2 : 1 (P)	Лутон Таун
Чемпионшип	38 18/03/23	Сандерленд	1 : 1 (P)	Лутон Таун
Чемпионшип	18 29/10/22	Лутон Таун	1 : 1 (P)	Сандерленд
Лига 1	28 12/01/19	Сандерленд	1 : 1 (P)	Лутон Таун
Лига 1	2 11/08/18	Лутон Таун	1 : 1 (P)	Сандерленд
Кубок Футбольной лиги	P2 28/08/07	Лутон Таун	3 : 0 (P)	Сандерленд

Рисунок 1.10 – Скриншот страницы информации о матче

На этой странице в отличие от предыдущего приложения указано больше информации о двух командах. Помимо того, можно ознакомиться с предыдущими встречами 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 приложения «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:

- иногда дублируются ставки;
- слишком большое количество информации на главной странице.

1.4 Приложение «MaxLine»

Последним аналогом для сравнения станет приложение «MaxLine» [3]. Скриншот модального окна для входа в приложение показан на рисунке 1.11.



Рисунок 1.11 – Скриншот модального окна для входа в приложение

В данном приложении вход сделан не через отдельное окно, а на верхней панели. Войти можно только через номер телефона. Также есть возможность восстановить пароль. Скриншот главной страницы приложения показан на рисунке 1.12.

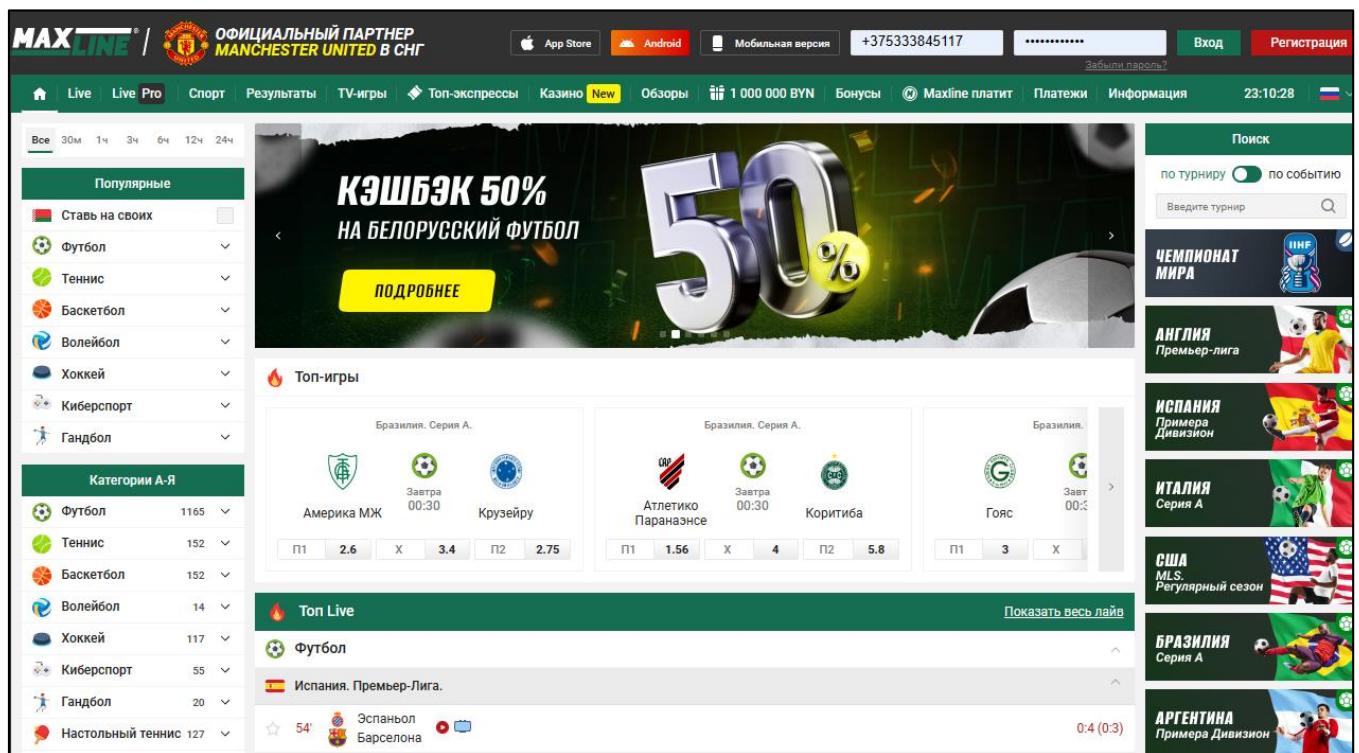


Рисунок 1.12 – Скриншот главной страницы

Приложение «MaxLine» уже более десяти 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 страницу матча. Скриншот страницы матча показана на рисунке 1.13.

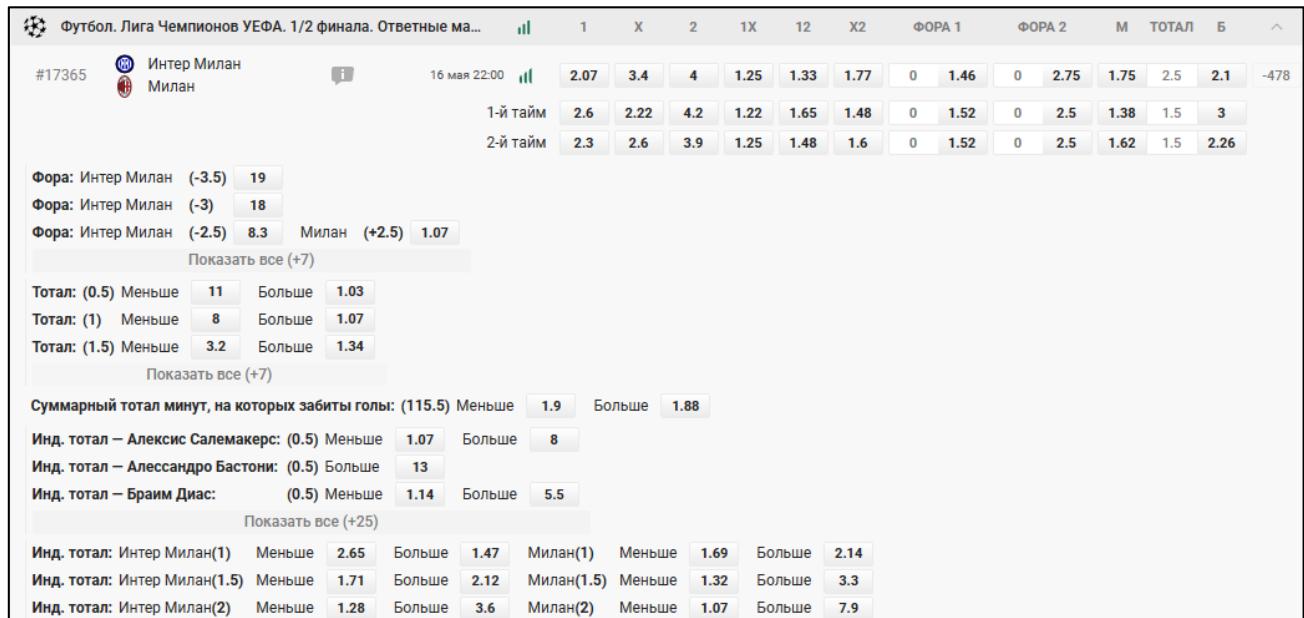


Рисунок 1.13 – Скриншот страницы матча

На странице матча указаны команды, которые играют, время проведения игры и турнир. Исходы не сортированы по вкладкам, как было в предыдущих аналогах. Выбрав исход в правом окне появляется модальное окно подтверждения ставки. Скриншот модального окна показан на рисунке 1.14.

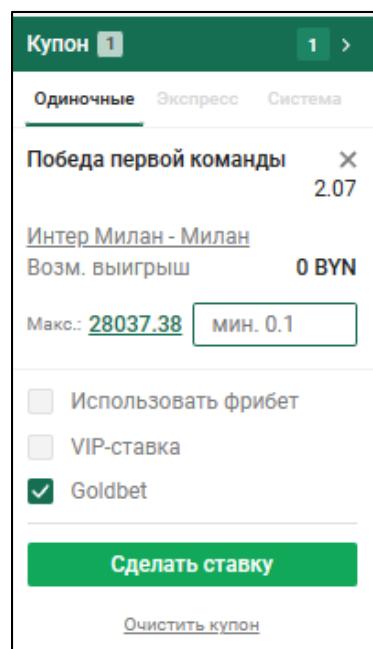


Рисунок 1.14 – Скриншот модального окна подтверждения ставки

В этом окне можно увидеть название ставки, коэффициент, сумму которую ставит игрок, 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. Скриншот страницы информации о матче можно увидеть на рисунке 1.15.

СТАТИСТИКА: ФУТБОЛ ЛИГА ЧЕМПИОНОВ УЕФА. 1/2 ФИНАЛА. ОТВЕТНЫЕ МАТЧИ. ИНТЕР МИЛАН - МИЛАН		
Последние игры Интер Милан:		
Дата	Хозяева - Гости	Счёт
13.05.23	Футбол. Италия. Серия А. Интер Милан - Сассуоло	4:2 (1:0)
10.05.23	Футбол. Лига Чемпионов УЕФА. 1/2 финала. Первые матчи. Милан - Интер Милан	0:2 (0:2)
06.05.23	Футбол. Италия. Серия А. Рома - Интер Милан	0:2 (0:1)
03.05.23	Футбол. Италия. Серия А. Верона - Интер Милан	0:6 (0:3)
30.04.23	Футбол. Италия. Серия А. Интер Милан - Лацио	3:1 (0:1)
Тотал 2.5		МЕН БОЛ
В последних 5 играх		2 3
В текущем чемпионате		14 7
Последние игры Милан:		
Дата	Хозяева - Гости	Счёт
13.05.23	Футбол. Италия. Серия А. Специя 1906 - Милан	2:0 (0:0)
10.05.23	Футбол. Лига Чемпионов УЕФА. 1/2 финала. Первые матчи. Милан - Интер Милан	0:2 (0:2)

Рисунок 1.15 – Скриншот страницы информации о матче

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

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

К положительным сторонам можно отнести:

- большой выбор исходов матча;
- удобное подтверждение ставок.

К отрицательным сторонам можно отнести:

- неудобный поиск матчей;
- небольшое количество информации на главной странице.

1.5 Постановка задачи

При обзоре аналогов были выявлены и рассмотрены их главные достоинства и недостатки. В итоге были выделены следующие функциональные возможности

веб-приложения для букмекерских ставок на футбольные события, которые необходимо реализовать:

- регистрация и авторизация пользователя;
- поддержка ролей администратора, букмекера и пользователя;
- создание, удаление и изменения регионов, чемпионатов и команд;
- создание и удаление матчей;
- изменение роли пользователя;
- возможность для букмекера добавлять исходы на матч
- возможность для букмекера изменять коэффициенты на матч;
- возможность для букмекера изменять название исходов
- возможность для пользователя ставить на футбольные события;
- возможность для пользователя пополнять игровой;
- возможность для пользователя выводить средства;
- возможность для пользователя просматривать историю ставок;
- возможность для пользователя просматривать историю счета.

Для реализации данного функционала были выделены следующие задачи:

- спроектировать веб-приложение;
- выбрать необходимые инструменты для разработки;
- спроектировать базу данных;
- разработать клиентскую часть;
- разработать серверную часть;
- реализовать возможность регистрации и авторизации пользователя;
- реализовать возможность сделать ставку на футбольное событие.

1.6 Вывод по разделу

В данном разделе были описаны аналогичные веб-приложения на рынке. В ходе обзора аналогов был приведен краткий обзор к каждому из них, рассмотрены их возможности и недостатки, выделены минусы и плюсы, которые можно было бы позаимствовать при разработке собственного продукта. На основании данных приложений были сформулирован список функционала, который необходимо реализовать в веб-приложении.

На основании функционала, который должен быть реализован в дипломном проекте были выделены задачи, которые помогут создать веб-приложение для букмекерских ставок на футбольные события.

2 Проектирование веб-приложения

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 множества 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. Всем зарегистрированным пользователям Администратор может менять роль

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, чтобы иметь возможность пополнить счет, просмотреть личную страницу и сделать ставку на футбольное событие.

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.

				БГТУ 02.00.ПЗ		
	ФИО	Подпись	Дата			
Разраб.	Гой М.А.			2 Проектирование веб-приложения	Лит.	Лист
Провер.	Комарова Е.И.				У	1
Н. контр.	Николайчук А.Н.				Листов	
Утв.	Смелов В.В.				11	
				74218010, 2023		

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 программ 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. Главная причина использовать TypeScript – 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

2.4.1 Технология ASP.NET Core

ASP.NET Core является кроссплатформенной, высокопроизводительной средой с открытым исходным кодом для создания современных облачных приложений, подключенных к Интернету [6]. Он является эволюцией ASP.NET и предоставляет множество новых возможностей, таких как кроссплатформенность, высокая производительность, поддержка облачных платформ и контейнеризации.

Одной из главных особенностей ASP.NET Core является его модульная архитектура. Он состоит из набора модулей, которые можно подключать и отключать по мере необходимости. Это позволяет разработчикам создавать более гибкие и масштабируемые приложения.

ASP.NET Core также поддерживает множество протоколов и форматов данных, таких как HTTP, WebSocket, JSON и XML. Это обеспечивает универсальность фреймворка и позволяет разработчикам создавать приложения, которые могут работать с различными клиентами и серверами.

Еще одной важной особенностью ASP.NET Core является его высокая производительность. Он оптимизирован для работы в условиях высокой нагрузки и может обрабатывать тысячи запросов в секунду. Это делает его идеальным выбором для создания масштабируемых веб-приложений.

ASP.NET Core также предоставляет множество инструментов для упрощения разработки приложений. Он включает в себя интегрированную среду разработки (IDE) Visual Studio, который обеспечивает автоматическую генерацию кода и поддержку отладки. Кроме того, он предоставляет множество библиотек и пакетов NuGet, которые упрощают работу с базами данных, аутентификацией и авторизацией, кэшированием и другими задачами.

В целом, ASP.NET Core является мощным и гибким фреймворком для создания веб-приложений и микросервисов. Он обеспечивает высокую

производительность, масштабируемость и удобство разработки, что делает его идеальным выбором для создания современных веб-приложений.

2.4.2 Технология EntityFramework

Entity Framework – это технология от Microsoft, которая позволяет разработчикам работать с базами данных в объектно-ориентированном стиле [7]. EF является частью .NET Framework и позволяет создавать приложения, которые могут работать с различными базами данных, такими как SQL Server, Oracle, MySQL, PostgreSQL и другими.

Одной из главных особенностей Entity Framework является его способность автоматически генерировать код для работы с базой данных. Это позволяет разработчикам сосредоточиться на бизнес-логике приложения, а не на деталях работы с базой данных. EF также предоставляет множество инструментов для упрощения работы с данными, таких как LINQ to Entities, который позволяет разработчикам писать запросы к базе данных на языке C#.

Еще одной важной особенностью Entity Framework является его способность работать с различными типами связей между таблицами в базе данных. EF поддерживает связи один-ко-многим, многие-ко-многим и один-к-одному. Это делает его идеальным выбором для создания сложных приложений, которые работают с большим количеством данных.

Entity Framework также поддерживает механизм миграций базы данных, который позволяет разработчикам изменять структуру базы данных без необходимости пересоздания ее с нуля. Это упрощает процесс обновления приложения и позволяет сохранять данные при обновлении структуры базы данных.

Кроме того, Entity Framework предоставляет множество инструментов для управления транзакциями, кэшированием и безопасностью данных. EF также поддерживает асинхронную работу с базой данных, что позволяет создавать более быстрые и отзывчивые приложения.

В целом, Entity Framework является мощной технологией для работы с базами данных в .NET Framework. Он обеспечивает высокую производительность, удобство использования и множество инструментов для упрощения работы с данными. Это делает его идеальным выбором для создания сложных приложений, которые работают с большим количеством данных.

2.4.3 Библиотека React

React – это библиотека JavaScript, разработанная компанией Facebook для создания пользовательских интерфейсов. Она позволяет создавать масштабируемые и быстрые веб-приложения с использованием компонентной архитектуры [8].

Одной из главных особенностей React является его способность работать с виртуальным DOM. Виртуальный DOM – это копия реального DOM, которая хранится в памяти. React использует виртуальный DOM для минимизации количества обновлений реального DOM, что позволяет улучшить производительность приложения.

Еще одной важной особенностью React является создавать и использовать компонентную архитектуру. Компоненты – это независимые блоки кода, которые могут быть переиспользованы в различных частях приложения. Компоненты также могут быть вложены друг в друга, что позволяет создавать сложные пользовательские интерфейсы.

React также предоставляет множество инструментов для управления состоянием приложения. Redux – это один из наиболее популярных инструментов для управления состоянием в React. Он позволяет хранить состояние приложения в единственном месте и обновлять его при необходимости.

Кроме того, React предоставляет множество инструментов для упрощения работы с данными и API. Axios – это один из наиболее популярных инструментов для работы с API в React. Он позволяет отправлять запросы на сервер и получать ответы в формате JSON.

React также поддерживает множество инструментов для тестирования приложений. Jest – это один из наиболее популярных инструментов для тестирования React-приложений. Он позволяет создавать простые тесты для компонентов, состояния и функций.

В целом, React является мощной библиотекой для создания пользовательских интерфейсов. Он обеспечивает высокую производительность, удобство использования и множество инструментов для упрощения работы с данными. Это делает его идеальным выбором для создания масштабируемых и быстрых веб-приложений.

2.4.4 Библиотека Styled Components

Styled-components – это библиотека для React, которая позволяет создавать стилизованные компоненты с помощью CSS в JavaScript [9]. Она позволяет управлять стилями компонентов напрямую в коде, что делает их более гибкими и использовать их в любой части приложения.

Одной из главных особенностей styled-components является ее способность использовать функциональные преимущества JavaScript, такие, как шаблонные строки и переменные. Это позволяет создавать динамические стили, которые могут изменяться в зависимости от состояния компонента.

Styled-components также предоставляет множество инструментов для управления стилями компонентов. Она позволяет создавать глобальные стили, определять темы и использовать медиа-запросы для адаптивной верстки.

Кроме того, styled-components обеспечивает высокую производительность приложения, так как она использует оптимизированные CSS-правила и генерирует уникальные классы для каждого компонента.

Одной из главных преимуществ в использовании библиотеки styled-components является ее возможность создания переиспользуемых компонентов. Это позволяет создавать более эффективный и гибкий код, который может быть использован в различных частях приложения.

Styled-components также интегрируется с другими инструментами React, такими как Redux и React Router. Это позволяет создавать более сложные приложения с помощью единообразного подхода к стилизации компонентов.

В целом, styled-components является мощной библиотекой для создания стилизованных компонентов в React. Она обеспечивает высокую производительность, гибкость и переиспользуемость кода, что делает ее идеальным выбором для создания эффективных и красивых пользовательских интерфейсов.

2.4.5 Библиотека Redux

Redux – это библиотека для управления состоянием приложения в React [10]. Она позволяет создавать предсказуемые и легко тестируемые приложения, которые могут быть масштабированы на любом уровне.

Одной из главных особенностей Redux является ее способность хранить все состояние приложения в одном объекте – так называемом хранилище (store). Это позволяет управлять состоянием приложения централизованно и делает его более прозрачным и управляемым.

Redux также предоставляет множество инструментов для управления состоянием приложения. Она позволяет создавать действия (actions) – объекты, которые описывают, что произошло в приложении, и редукторы (reducers) – функции, которые обрабатывают эти действия и изменяют состояние хранилища.

Кроме того, Redux обеспечивает высокую производительность приложения, так как она использует неизменяемые объекты и оптимизированные алгоритмы обновления состояния.

Одной из главных преимуществ Redux является ее возможность создания переиспользуемых компонентов и легкость тестирования. Это позволяет создавать более эффективный и гибкий код, который может быть использован в различных частях веб-приложения.

Redux также интегрируется с другими инструментами React, такими как React Router и Redux Saga. Это позволяет создавать более сложные приложения с помощью единообразного подхода к управлению состоянием приложения.

В целом, Redux является мощной библиотекой для управления состоянием приложения в React. Она обеспечивает высокую производительность, гибкость и переиспользуемость кода, что делает ее идеальным выбором для создания эффективных и масштабируемых приложений.

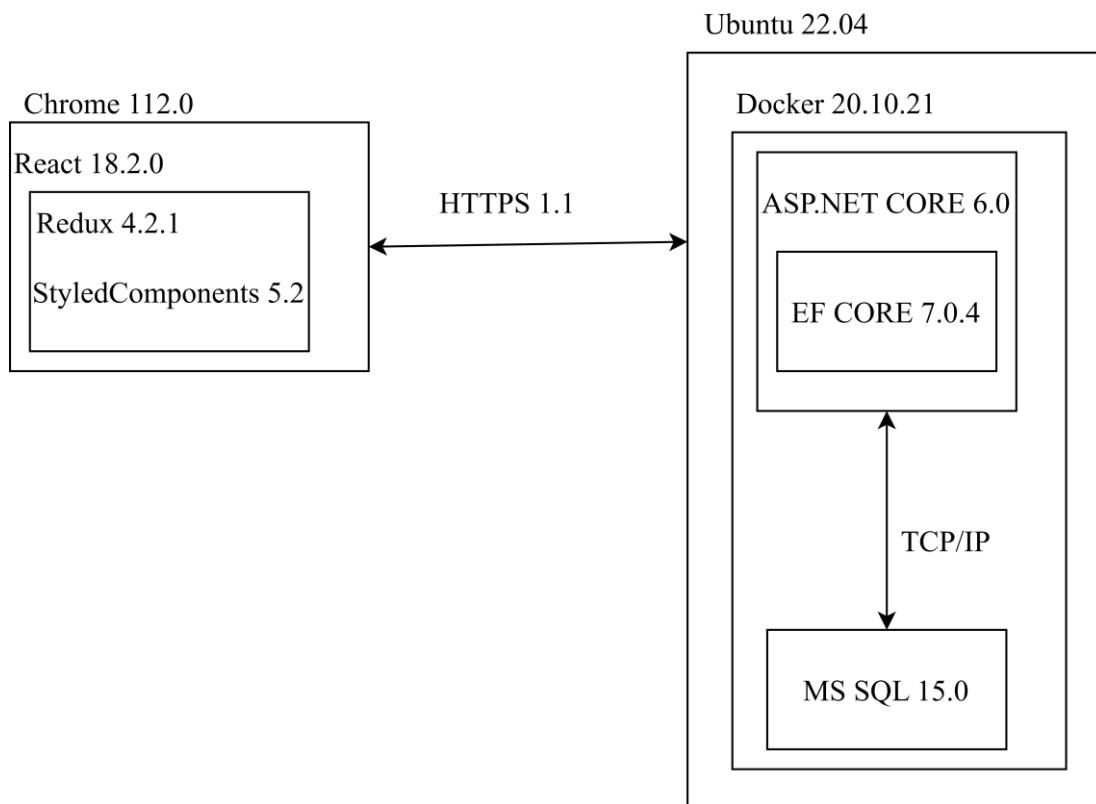
2.5 Система управления базами данных

Для хранения данных будет использована Microsoft SQL Server [11]. Microsoft SQL Server – система управления реляционными базами данных, разработанная корпорацией Microsoft. СУБД позволяет гибко управлять базами данных. 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. Выбранная база данных предлагает множество функций и возможностей, таких как поддержка транзакций, масштабируемость, репликация данных, индексирование и многое другое. MS SQL также поддерживает различные языки программирования, такие как C#, Java, Python и другие.

2.6 Проектирование структурной схемы веб-приложения

Структурная схема веб-приложения является неотъемлемой частью процесса разработки веб-приложений. Она представляет собой диаграмму, которая отображает все компоненты приложения и их взаимосвязи. Структурная схема помогает разработчикам лучше понять архитектуру приложения, что позволяет им оптимизировать его работу и сделать его более эффективным.

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



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

2.7 Логическая схема базы данных

Логическая схема базы данных является важным инструментом для проектирования и создания базы данных. Она определяет структуру и связи между таблицами, атрибутами и ограничениями, которые используются для хранения и управления данными в базе данных. Как было аргументировано ранее, в качестве системы управления базами данных выступает Microsoft SQL Server. Логическая схема базы данных позволяет разработчикам и администраторам баз данных лучше понимать структуру данных и их отношения. Она также помогает определить правила целостности данных, которые гарантируют, что данные в базе остаются точными и надежными. Логическая схема базы данных представлена в приложении Д на рисунке 2.2.

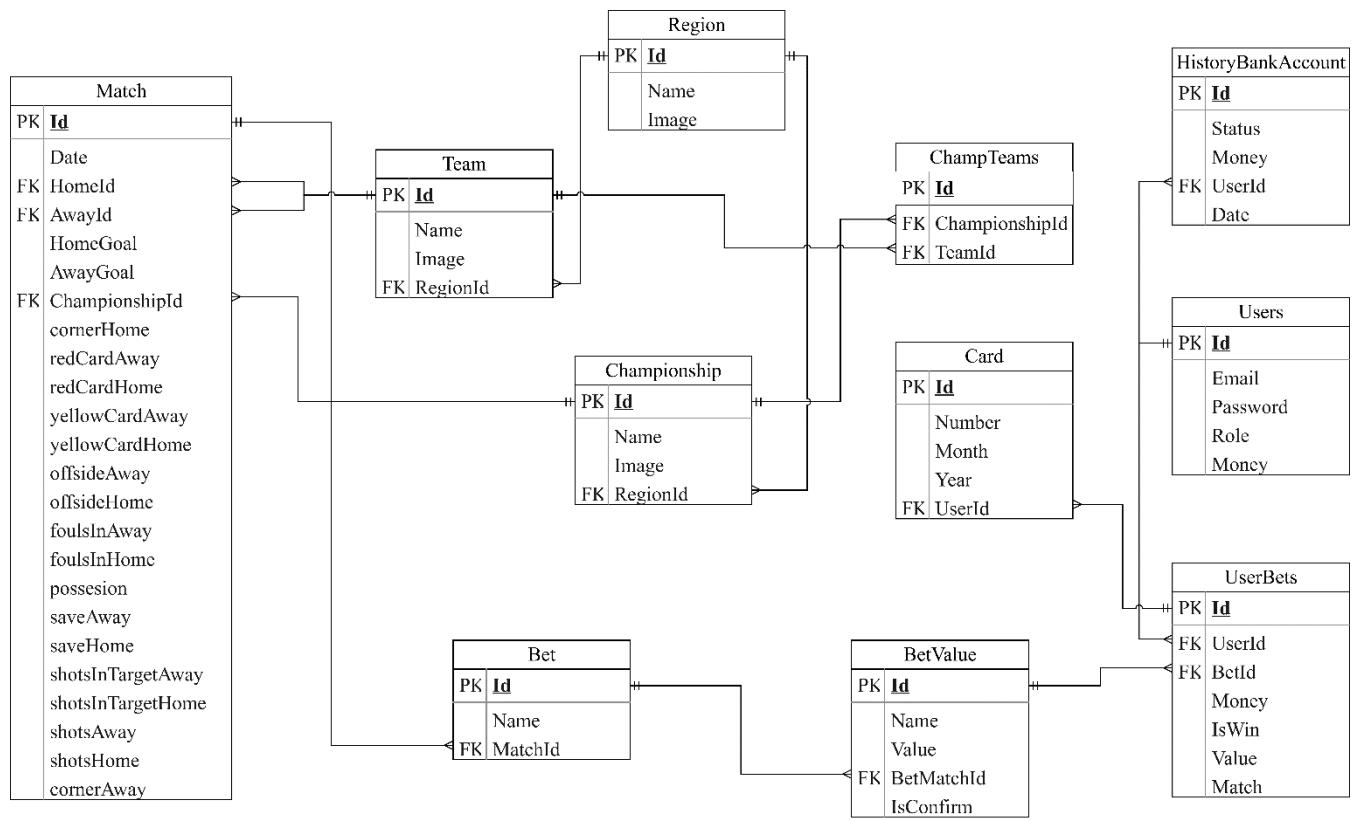


Рисунок 2.2 – Логическая схема базы данных

Далее для полноты картины 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
Image	Nvarchar	Ссылка на изображение

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
1	2	3
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
cornerHome	Int	Угловые домашней команды
cornerAway	Int	Угловые гостевой команды
shotsHome	Int	Удары домашней команды

Окончание таблицы 2.5

1	2	3
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 «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

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 «BetMatch» предназначена для хранения всех ставок на матч. Полное описание данной таблицы представлена в таблице 2.8.

Таблица 2.8 – Структура таблицы «BetValue»

Написание	Тип	Описание
Id	Int	Идентификатор чемпионата
Name	Nvarchar	Название ставки
BetMatchId	Int	Идентификатор матча, внешний ключ
Value	Real	Сумма ставки
IsConfirm	Bit	Выиграна ли ставка

Таблица «Bet» предназначена для хранения коэффициентов на матч. Полное описание данной таблицы представлена в таблице 2.9.

Таблица 2.9 – 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 2.10 – Структура таблицы «UserBets»

Написание	Тип	Описание
1	2	3
Id	Int	Идентификатор чемпионата
UserId	Int	Идентификатор пользователя, внешний ключ

Окончание таблицы 2.10

1	2	3
BetId	Int	Идентификатор ставки, внешний ключ
Money	Real	Ставка
Match	Nvarchar	Матч
IsWin	Bit	Выиграна ли ставка
Value	Real	Коэффициент ставки

Таблица «HistoryBankAccount» предназначена для хранения истории счета пользователя. Полное описание данной таблицы представлена в таблице 2.11.

Таблица 2.11 – 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

Описанные выше таблицы в полной мере удовлетворяют всем функциональным требованиям разрабатываемого программного модуля.

2.8 Вывод по разделу

В данном разделе были обоснованы и выбраны основные языки программирования, технологии, библиотеки 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 каждой таблицы и описанием всех полей.

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 для каждой роли. 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 следуя 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.

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.

				БГТУ 03.00.ПЗ		
	ФИО	Подпись	Дата			
Разраб.	Гой М.А.			Lит.	Лист	Листов
Провер.	Комарова Е.И.			У	1	13
Н. контр.	Николайчук А.Н.			3 Реализация веб-приложения		
Утв.	Смелов В.В.			74218010, 2023		

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

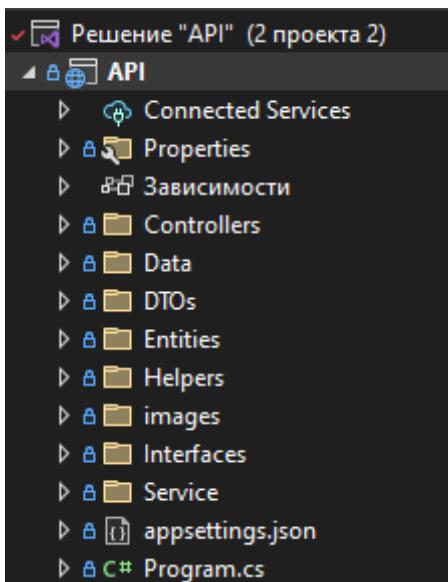


Рисунок 3.1 – Файловая структура веб-приложения

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 посыпает 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 между клиентом и сервером

Папка **Entities** хранит те классы, которые описывают логику, взаимодействие и характеристики используемых данных.

Папка **Helpers** хранит классы, которые помогают настраивать приложение. Здесь находятся 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 при разработке приложения.

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 нужны в программировании для определения набора методов и свойств, которые должны быть реализованы классом, который имплементирует данный интерфейс

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 разрабатываемого проекта – создание моделей и репозиториев для связи с базой данных. Для работы с базой данных на языке C# используется технология ORM Entity Framework. 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 одиннадцать моделей, скриншот списка представлен на рисунке 3.2.

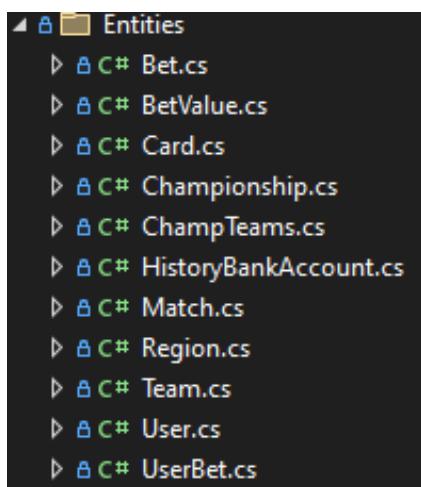


Рисунок 3.2 – Скриншот списка моделей

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, скриншот списка представлен на рисунке 3.3.

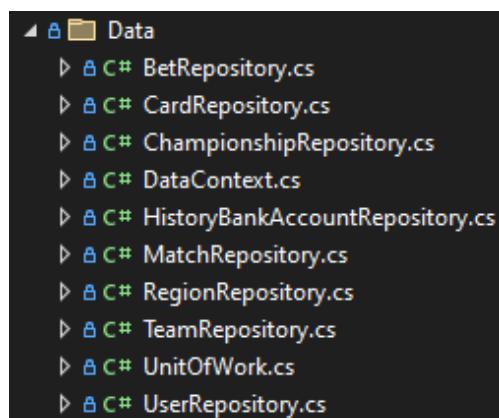


Рисунок 3.3 – Скриншот списка репозиториев

Листинг репозитория BetRepository представлен в приложении Е. Он наследуется от интерфейса IBet, который содержит методы, реализуемые данным репозиторием. Далее 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. Скриншот списка сервисов продемонстрирован на рисунке 3.4.

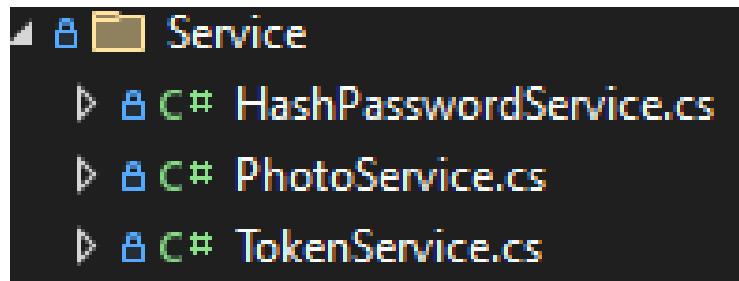


Рисунок 3.4 – Скриншот списка сервисов приложения

Программная реализация сервиса TokenService и его метод CreateToken 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 интерфейсов, которые 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. 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.

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 и 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, чтобы получить данные в подходящем виде для дальнейшей работы с ними.

Всего было создано девять контроллеров. Скриншот списка разработанных контроллеров показано на рисунке 3.5.

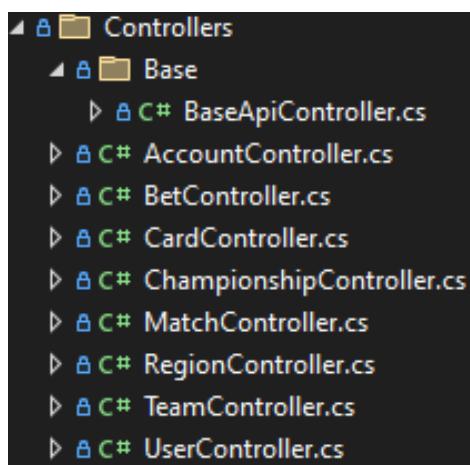


Рисунок 3.5 – Скриншот списка разработанных контроллеров

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. Блок-схема алгоритма подтверждения ставки представлена в приложении Ж.

– 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 с 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 AccountController для 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, который содержит сообщение о том, что пользователь с такой электронной почтой не найден.

Далее сравнивается пароль, который прислал пользователь, и пароль находящийся в базе данных, перед этим присланный пароль хэшируется. Если пароли не совпадают, то возвращается объект BadRequest со статусным кодом 400, который содержит сообщение о том, что пароли не совпадают.

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

Пример кода ChampionshipController, который обрабатывает запросы, связанные с чемпионатами, представлен в приложении И.

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 Visual Studio 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 архитектуру 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.

Скриншот архитектуры приложения можно увидеть на рисунке 3.6.

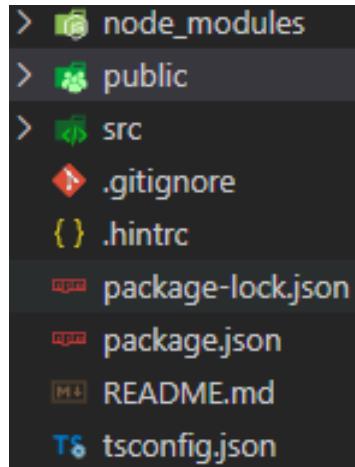


Рисунок 3.6 – Скриншот архитектуры приложения

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.

Директория `src` содержит исходный код приложения, включая компоненты, страницы, стили, изображения и другие ресурсы. Эта папка является основной точкой входа для React приложения.

Файл `.gitignore` содержит список директорий и файлов, которые система контроля версий Git будет игнорировать, включая директорию `node_modules`.

Файл `.hintrc` содержит конфигурации, используемые популярным инструментом компоновки ESLint для включения правил, плагинов и других опций.

`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. Использование папки `src` позволяет разработчикам легко организовывать свой код и управлять зависимостями между компонентами. Кроме того, это позволяет быстро находить нужные файлы и изменять код без необходимости пересобирать всё приложение.

Также, использование папки `src` является хорошей практикой для разработки масштабируемых и поддерживаемых приложений, так как разделение кода на компоненты и модули упрощает его понимание и обслуживание. Скриншот директории `src` показан на рисунке 3.7.

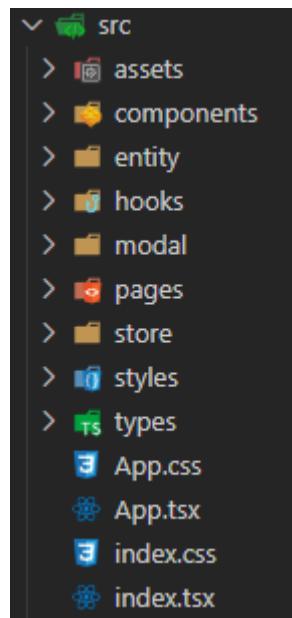


Рисунок 3.7 – Скриншот директории src

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 – хранит хранилище, которое содержит 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 более подробно. Компоненты являются основными блоками для создания пользовательских интерфейсов в React. Они могут быть созданы как классы или функции и могут содержать различные элементы, такие как текст, изображения, формы и другие компоненты.

Компоненты в React могут быть многократно использованы и могут быть вложены друг в друга для создания более сложных интерфейсов. Это позволяет создавать гибкие и масштабируемые приложения.

Одной из основных возможностей компонентов в React является использование пропсов и состояния для динамического изменения содержимого и поведения компонента. Пропсы позволяют передавать данные в компонент, а состояние позволяет хранить и изменять данные внутри компонента.

Скриншот всех компонентов приложения показан на рисунке 3.8.

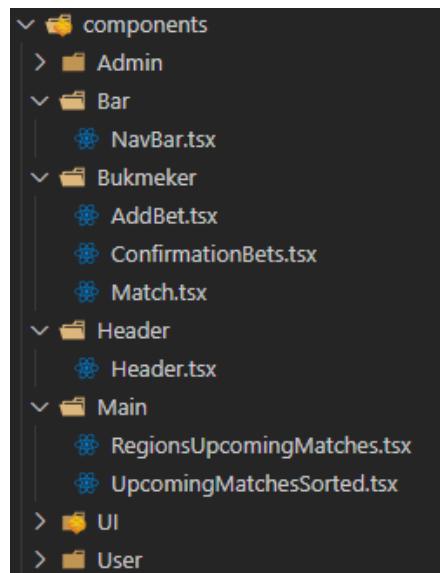


Рисунок 3.8 – Скриншот всех компонентов приложения

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 К.

Далее будет рассмотрено содержимое директории pages. Скриншот директории представлен на рисунке 3.9 ниже.

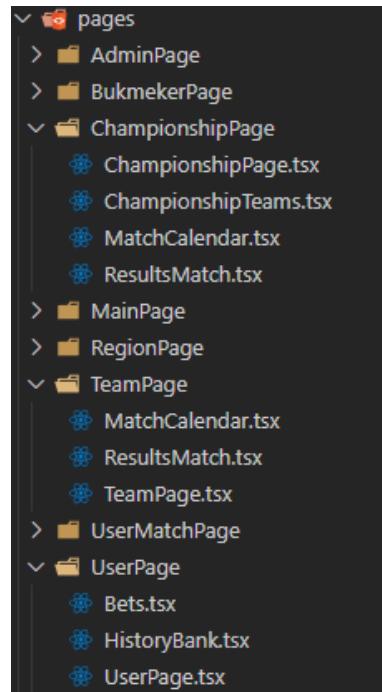


Рисунок 3.9 – Скриншот директории pages

Данная папка была разбита на следующие сущности:

- AdminPage – папка 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 3.10 представлен скриншот директории ChampionshipPage.

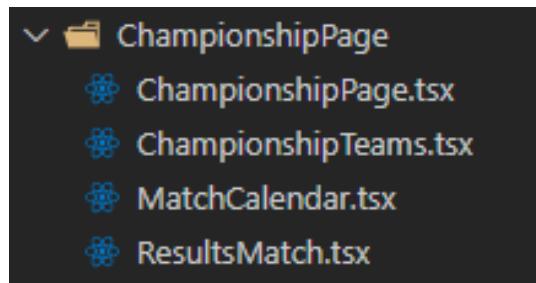


Рисунок 3.10 – Скриншот директории ChampionshipPage

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 показывает результаты матчей.

Пример кода страницы календаря матчей указан в приложении Л.

Теперь рассмотрим структуру директории UserPage. Скриншот директории представлен на рисунке 3.11.

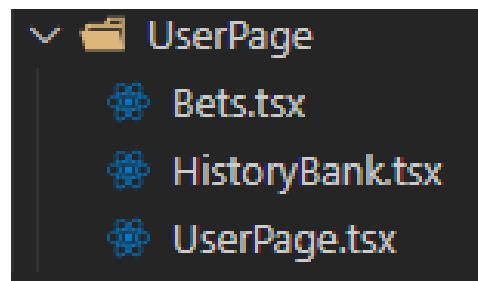


Рисунок 3.11 – Скриншот директории UserPage

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 3.4.

```
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 3.4 – Пример кода 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. Перед разработкой проектов были выбраны архитектуры обеспечивающие дальнейшую поддержку и расширение функционала. После этого были настроены и сконфигурированы обе части проекта таким образом, чтобы они взаимодействовали друг с другом. Были разработаны все функциональные компоненты приложения. Итогом выполнения реализации программного средства стало веб-приложение для букмекерских ставок на футбольные события отвечающее всем требованиям, предъявленных ему в начале разработки.

4 Тестирование веб-приложения

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 разработанные 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, которые необходимо исправить.

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 [12]. Тестировщик имитирует реальные действия пользователя и старается охватить максимум функций продукта и найти ошибки. Специалист по QA ищет недоработки в визуальной части, 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
1	2	3	
Создание региона	Вывод окна для создания региона		Успешно

				БГТУ 04.00.ПЗ			
	ФИО	Подпись	Дата				
Разраб.	Гой М.А.			Lит.	Лист	Листов	
Провер.	Комарова Е.И.			У	1	8	
Н. контр.	Николайчук А.Н.			4 Тестирование веб-приложения			
Утв.	Смелов В.В.			74218010, 2023			

Окончание таблицы 4.1

Описание теста	Ожидаемый результат	Статус
1	2	3
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 существует	Успешно
Отправка невалидных данных при создании нового исхода	Вывод ошибки при создании исхода матча	Успешно
Выбор нужного матча	Переход на страницу выбранного матча	Успешно
Ввод невалидных данных при изменении 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 Пользователя

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
Сделать ставку на тот же самый матч	Вывод об ошибке что на этот матч уже сделана ставка	Успешно
Сделать ставку на сумму превышающую игровой счет	Вывод об ошибке что не хватает денег на игровом счету	Успешно
Пополнение счета без выбора карты	Вывод окна для добавления новой карты	Успешно
Пополнение 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 [13]. 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 на сервер.

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

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

The image shows a registration form titled 'Регистрация'. It has four input fields: 'E-mail', 'Пароль', 'Повторите пароль', and 'Ведите e-mail'. Below the last field is a large red error message 'Ведите e-mail'. At the bottom is a grey button labeled 'Зарегистрироваться'.

Рисунок 4.1 – Результат пустого поля электронной почты

При создании пароля необходимо вести пароль два раза, для того чтобы пользователь не ошибся. Также валидация присутствует и для пароля, если вводится меньше восьми символов при регистрации. Если Пользователь допустит ошибку, ему выведется сообщение об ошибке. Результат показан на рисунке 4.2.

The image shows a registration form titled 'Регистрация'. It has three input fields: 'maks@gmail.com', '...', and '...'. Below the second field is a red error message 'Длина пароля должна быть минимум 8 символов'. At the bottom is a grey button labeled 'Зарегистрироваться'.

Рисунок 4.2 – Результат валидации пароля на форме регистрации

При ставке на футбольное событие пользователь должен ввести сумму ставки. Перед этим выводится модальное окно с подтверждение ставки, где видно название исхода, и коэффициент ставки. При вводе суммы ставки меньше двух рублей Пользователю показывается ошибка. Результат представлен на рисунке 4.3.

Подтверждение ставки
Исход: П2 2.1
1,5
Сумму ставки должна быть от 2 рублей
Сделать ставку

Рисунок 4.3 – Валидация на сумму ставки

При добавлении нового исхода на матч, Букмекер вводить название нового исхода. Если такой исход уже существует, сервер сообщает об ошибке, после чего выводится соответствующее сообщение. Результат показан на рисунке 4.4.

Добавление исхода
Исход
2
Такой исход уже есть
Создать

Рисунок 4.4 – Результат добавления существующего исхода

Также присутствует валидация на ввод количества исходов, который не должен превышать десяти. Если превысить это значение, выведется ошибка. Результат ввода неверного количества исходов показан на рисунке 4.5.

Добавление исхода
Голы
15
Количество исходов не должно быть больше 10
Создать

Рисунок 4.5 – Ввод неверного количества исходов

При создании нового региона Администратору необходимо ввести название нового региона. Если регион с таким названием уже существует, сервер сообщит об ошибке и выведет сообщение об этом. Результат выполнения создания региона показан на рисунке 4.6.

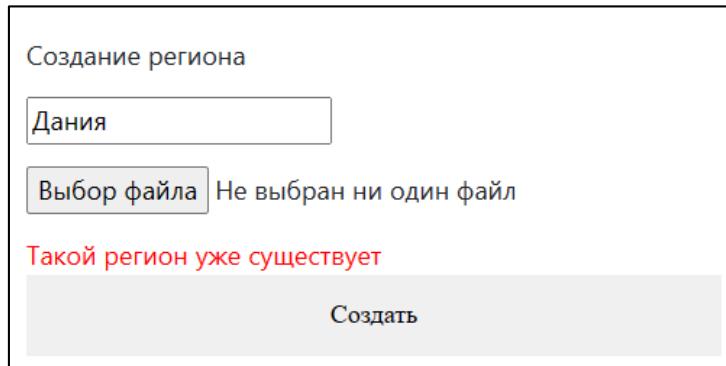


Рисунок 4.6 – Результат создания существующего региона

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

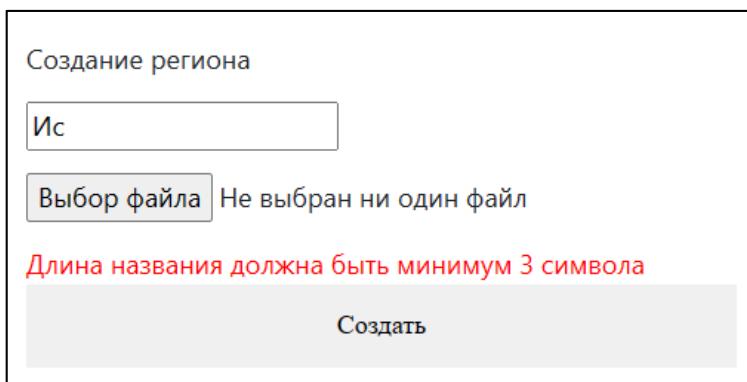


Рисунок 4.7 – Результат создания региона с двумя символами

Далее, кроме ручных тестов и тестов валидации, необходимо провести модульное тестирование. Для полной уверенности в правильной работе программного средства, так как модульные тесты могут протестировать не только пользовательский интерфейс, но и логику, обращаясь напрямую к контроллерам.

4.3 Модульное тестирование

Модульное тестирование, иногда блочное тестирование или юнит-тестирование – процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы, наборы из одного или более программных модулей вместе с соответствующими управляющими данными, процедурами использования и обработки [14].

Тестирование является важной частью разработки программного обеспечения, и идея заключается в том, чтобы написать тесты для каждой

нетривиальной функции или метода. Это позволяет быстро проверить, не привело ли изменение кода 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. Одной из наиболее популярных и удобных библиотек является XUnit. Она бесплатна и поддерживается несколькими языками, использующими платформу .Net. Благодаря своей простоте и гибкости, XUnit позволяет быстро и эффективно создавать тесты для любого типа приложений. С ее помощью можно проверять как отдельные методы, так и целые классы, а также проводить интеграционное тестирование. В целом, использование XUnit значительно упрощает процесс тестирования и повышает качество программного кода.

Для упрощения тестирования и абстрагирования его от основной логики приложения, была создана дополнительная DLL-библиотека. В этой библиотеки для каждого контроллера и сервиса был создан класс, в котором создавались все тесты.

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 использовался 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 [15].

Всего в проекте было создано 94 теста, которые проверяли все функциональные части, которые участвовали в обработке данных: мапперы, контроллеры, сервисы, репозитории.

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. Во втором случает расчет был на то, что код сработает, но вернет ошибку вместо нужного результата.

Пример «позитивного» теста, в котором ожидается получить токен пользователя и посмотреть основную информацию о нем можно в приложении M.

Во втором случае на вход тесту предоставляются данные, которые не соответствуют требованиям или ограничениям модуля, и проверяется, как система обработает такие данные. Таким образом, тесты позволяют проверить работу модуля в различных условиях и выявить возможные ошибки или недочеты. Кроме того, тестирование помогает убедиться в корректности работы модуля и повысить уверенность в его качестве перед внедрением в продуктивную среду.

Пример созданного «негативного теста», в котором проверяется создание существующего пользователя, представлен на листинге 4.1.

```
[Fact]
    public async void FakeRegister()
    {
        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);
        RegisterDto register = new()
        {
            Email = "maksgoy2911@gmail.com",
            Password = "12345678",
        };
        var result = account.Register(register).Result;
        Assert.Null(result);
    }
}
```

Листинг 4.1 – «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 пользователя

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 «Пользователь»

В рамках разработанного веб-приложения Пользователь может выполнять следующие задачи:

- работать с личным счетом;
- просматривать личную страницу;
- делать ставку на футбольное событие;
- просматривать главную страницу;
- просматривать информацию о матче.

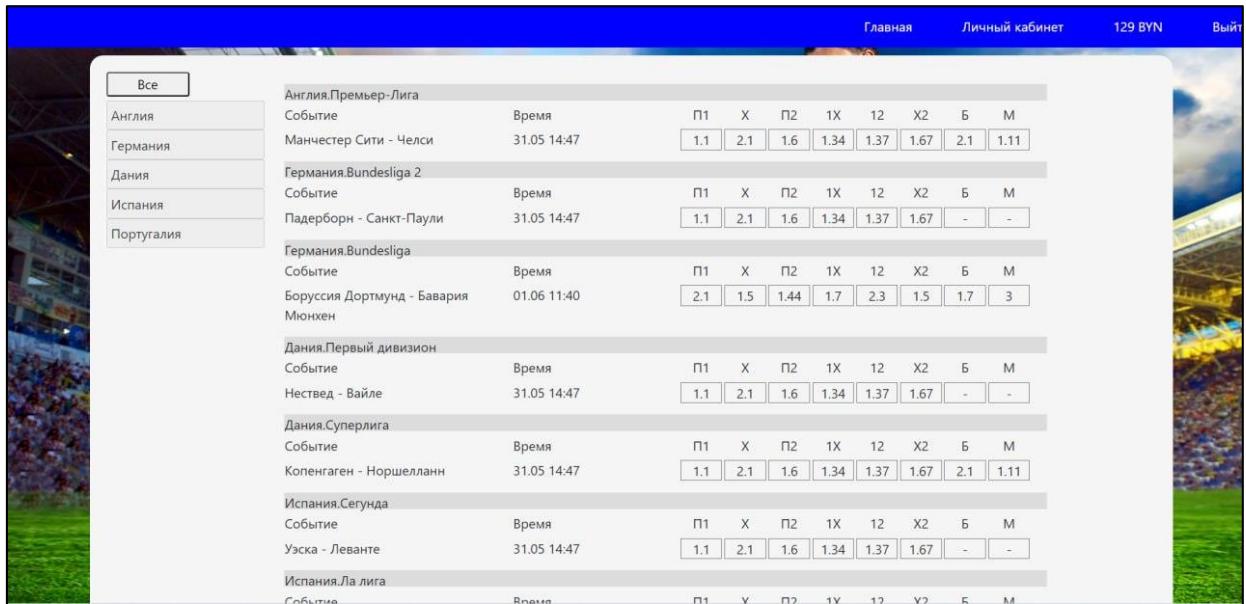
Для того чтобы выполнять все вышеперечисленные задачи, пользователю необходимо авторизоваться. Для этого необходимо на главной странице нажать кнопку «Войти». Далее появится модальное окно входа, с помощью которого можно пройти авторизацию. Скриншот модального окна для входа в приложение показан на рисунке 5.1.



Рисунок 5.1 – Скриншот модального окна для входа в приложение

Для авторизации необходимо ввести электронную почту и пароль учетной записи. Если все данные указаны верно, Пользователь попадает на главную страницу, где может посмотреть расписание ближайших матчей и т.д. Скриншот главной страницы продемонстрирован в приложении Н на рисунке 5.2.

				БГТУ 05.00.ПЗ		
	ФИО	Подпись	Дата			
Разраб.	Гой М.А.			5 Руководство пользователя	Лит.	Лист
Провер.	Комарова Е.И.				У	1
Н. контр.	Николайчук А.Н.					13
Утв.	Смелов В.В.				74218010, 2023	



		Главная	Личный кабинет	129 BYN	Выход						
<input checked="" type="checkbox"/>	Все										
<input type="checkbox"/>	Англия										
<input type="checkbox"/>	Германия										
<input type="checkbox"/>	Дания										
<input type="checkbox"/>	Испания										
<input type="checkbox"/>	Португалия										
Англия.Премьер-Лига		Событие	Время	П1	X	П2	1X	12	X2	Б	М
Манчестер Сити - Челси		31.05 14:47		1.1	2.1	1.6	1.34	1.37	1.67	2.1	1.11
Германия.Bundesliga 2		Событие	Время	П1	X	П2	1X	12	X2	Б	М
Падерборн - Санкт-Паули		31.05 14:47		1.1	2.1	1.6	1.34	1.37	1.67	-	-
Германия.Bundesliga		Событие	Время	П1	X	П2	1X	12	X2	Б	М
Боруссия Дортмунд - Бавария Мюнхен		01.06 11:40		2.1	1.5	1.44	1.7	2.3	1.5	1.7	3
Дания.Первый дивизион		Событие	Время	П1	X	П2	1X	12	X2	Б	М
Нествед - Вайле		31.05 14:47		1.1	2.1	1.6	1.34	1.37	1.67	-	-
Дания.Суперлига		Событие	Время	П1	X	П2	1X	12	X2	Б	М
Копенгаген - Норшелланн		31.05 14:47		1.1	2.1	1.6	1.34	1.37	1.67	2.1	1.11
Испания.Сегунда		Событие	Время	П1	X	П2	1X	12	X2	Б	М
Уэска - Леванте		31.05 14:47		1.1	2.1	1.6	1.34	1.37	1.67	-	-
Испания.Ла лига		Событие	Время	П1	X	П2	1X	12	X2	Б	М

Рисунок 5.2 – Скришонт главной страницы

На главной странице вверху можно заметить, что Пользователь может перейти в личную страницу, посмотреть баланс на личном счету, а также выйти из учетной записи. Также на странице присутствует расписание ближайших матчей, сортированных по чемпионату, а слева на боковой панели для быстрого доступа можно выбрать нужный чемпионат. Для того чтобы сделать ставку на футбольное событие, необходимо выбрать матч и перейти на страницу матча. Скриншот страницы матча показан на рисунке 5.2.

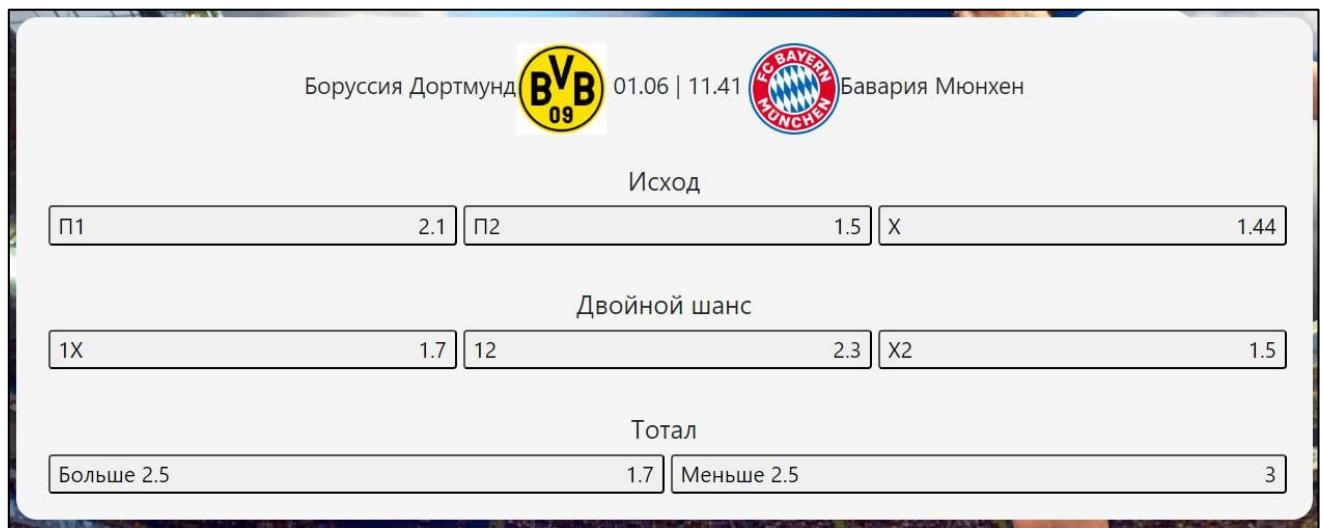


Рисунок 5.3 – Скришот страницы матча

На странице матча Пользователь может просмотреть информацию о матче, увидеть список исходов на матч и их коэффициенты. Для того чтобы сделать ставку, Пользователю необходимо выбрать исход, после чего появится окно подтверждения ставки. Стоит отметить что Пользователь может сделать только одну ставку на матч. Скриншот модального окна подтверждения ставки показано на рисунке 5.4 ниже.

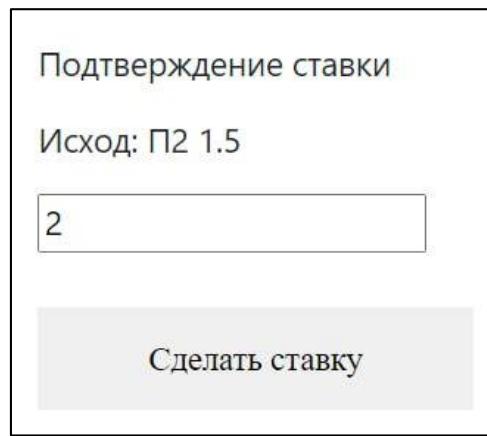


Рисунок 5.4 – Скриншот модального окна подтверждения ставки

В данном окне виден исход на который делается ставка, коэффициент исхода. Также Пользователю необходимо ввести сумму ставки и нажать кнопку «Сделать ставку». Если сумма введена верно, и на этот матч ставки у данного Пользователя нет, ставка принимается, после чего окно закроется.

Для того чтобы посмотреть все ставки, которые делал пользователь ему необходимо перейти в личный кабинет. Скриншот страницы личного кабинета показан на рисунке 5.5.

Матч	Ставка	Сумма(ВН)	Статус
Уэска - Леванте	П2 2.1	2	Играется
Бенфика - Порту	Больше 2.5 2.1	25	Играется
Суонси - Норвич	2 2.5	20	Выиграна
Манчестер Сити - Челси	П1 1.1	20	Играется
Бавария Мюнхен - Боруссия Дортмунд	П1 1.2	20	Проиграна
Боруссия Дортмунд - Бавария Мюнхен	П2 1.5	3	Играется

Рисунок 5.5 – Скриншот страницы личного кабинета

При входе в личный кабинет Пользователь видит свою учетную запись, ставки, которые делал пользователь, а также же возможно пополнить счет и вывести средства. Также он может увидеть матч, на который онставил, коэффициент исхода, сумму которую он поставил, а также статус ставки.

При пополнении счета появляется модальное окно, в котором выбирается карта, с которой произойдет списание средств, и сумма пополнения.

Вся работа со счетом Пользователя фиксируется в приложении. В любой момент можно увидеть какая сумма была поставлена на исход, все пополнения игрового счета. При выводе средств так же запись фиксируется в базе данных.

Для того чтобы посмотреть историю счета, необходимо нажать на кнопку «Историю счета», после чего Пользователя переадресует на страницу истории счета. Скриншот страницы истории счета показана на рисунке 5.6.

Время	Операция	Сумма(BYN)
26.05.2023 14:49	Ставка Уэска - Леванте Исход: П2 2,1	2 BYN
26.05.2023 14:49	Ставка Бенфики - Порту Тотал: Больше 2.5 2,1	25 BYN
26.05.2023 15:44	Ставка Bayern Munich - Borussia Dortmund Тотал: Больше 2.5 2,1	20 BYN
27.05.2023 11:58	Выигрыш	42 BYN
28.05.2023 21:06	Пополнение средств	20 BYN
28.05.2023 21:06	Вывод средств	15 BYN
28.05.2023 21:49	Ставка Суонси - Норвич Угловые: 2 2,5	20 BYN
28.05.2023 21:50	Выигрыш	50 BYN
28.05.2023 21:52	Ставка Манчестер Сити - Челси Исход: П1 1,1	20 BYN
29.05.2023 11:15	Ставка Реал Мадрид - Леванте Исход: П1 1,9	10 BYN
29.05.2023 11:17	Выигрыш	19 BYN
29.05.2023 11:22	Удаление матча	10 BYN

Рисунок 5.6 – Скриншот страницы истории счет

На странице истории счет, показываются все транзакции со счетом. Можно увидеть дату и время транзакции, наименование операции и сумма, которая была задействована.

Для того чтобы сделать ставку новому Пользователю, сначала необходимо пополнить счет. Для этого на странице личного кабинета нужно нажать кнопку «Пополнить счет», после чего появится окна пополнения счета. Скриншот модального окна для пополнения счета показано на рисунке 5.7.

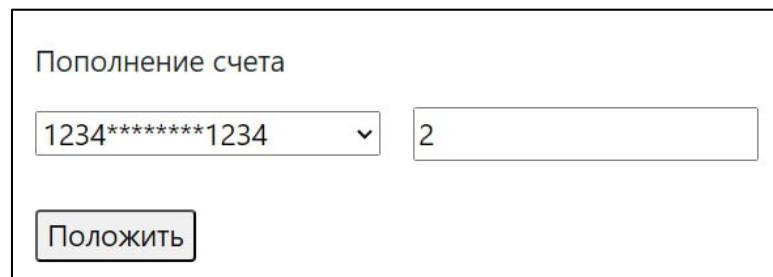


Рисунок 5.7 – Скриншот модального окна для пополнения счета

Для того чтобы пополнить счет, необходимо выбрать карту, с которой Пользователь хочет пополнить счет, далее выбрать сумму. Затем следует нажать кнопку «Положить», после чего игровой счет будет пополнен и окно пополнения счета пропадет.

5.2 Роль «Администратор»

В рамках разработанного веб-приложения Администратор может выполнять следующие задачи:

- сортировка регионов;
- поиск региона;
- создание регион, чемпионата и команды;

- изменение региона, чемпионата и команды;
 - создание матча;
 - удаление матча;
 - добавление результатов матча;
 - изменение роли пользователя.
- После авторизации Администратор попадает на страницу администрирования. Страница Администратора показана на рисунке 5.8.



Рисунок 5.8 – Страница администратора

На данной странице Администратор может перейти в любую выбранную вкладку, искать регион по названию, а также создать регион.

Для того чтобы создать регион, Администратору необходимо войти в учетную запись, после чего его переадресует на страницу регионов. Далее следует нажать соответствующую кнопку «Создать регион». Скриншот модального окна для создания региона показано на рисунке 5.9.

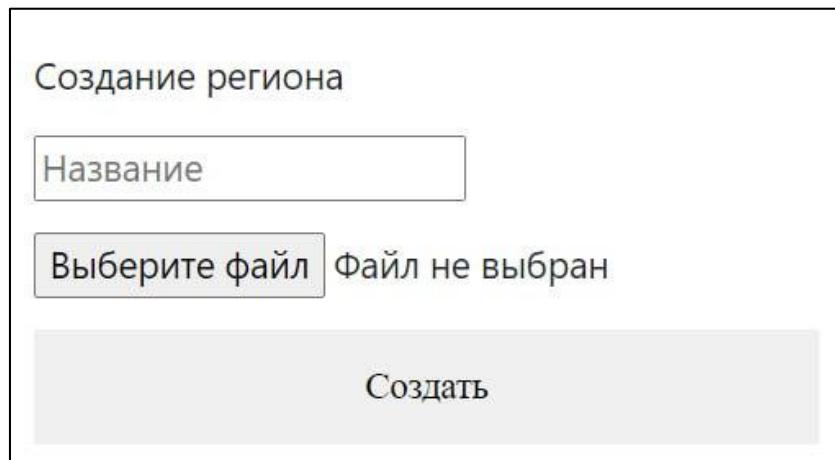


Рисунок 5.9 – Скриншот модального окна для создания региона

В окне создания региона Администратору необходимо ввести название нового региона, а также по желанию выбрать изображения региона.

Для создания чемпионата Администратору следует перейти на вкладку страницу чемпионатов, показанную на рисунке 5.10.

Регионы	Чемпионаты	Команды	Пользователи
<input type="text" value="Поиск чемпионата"/>	<input type="button" value="назначению"/>	<input type="button" value="Создать чемпионат"/>	
	Ла лига	Испания	
	Сегунда	Испания	
	Бундеслига	Германия	
	Бундеслига 2	Германия	
	Премьер-Лига	Англия	
	Чемпионшип	Англия	
	Чемпионат Португалии	Португалия	

Рисунок 5.10 – Страница чемпионатов

На странице чемпионатов можно искать чемпионат по названию, сортировать чемпионаты по региону и создать чемпионат, нажав на кнопку «Создать чемпионат». Окно создания чемпионат показано на рисунке 5.11.

Создание чемпионата

Название

Выберите файл

Файл не выбран

Англия
▼

Рисунок 5.11 – Создание чемпионата

Для создания чемпионата необходимо ввести название чемпионат, по желанию выбрать логотип, а также выбрать регион, в котором находится создаваемый чемпионат.

Для изменения чемпионата необходимо перейти на страницу чемпионата. Страница чемпионата показана на рисунке 5.12.

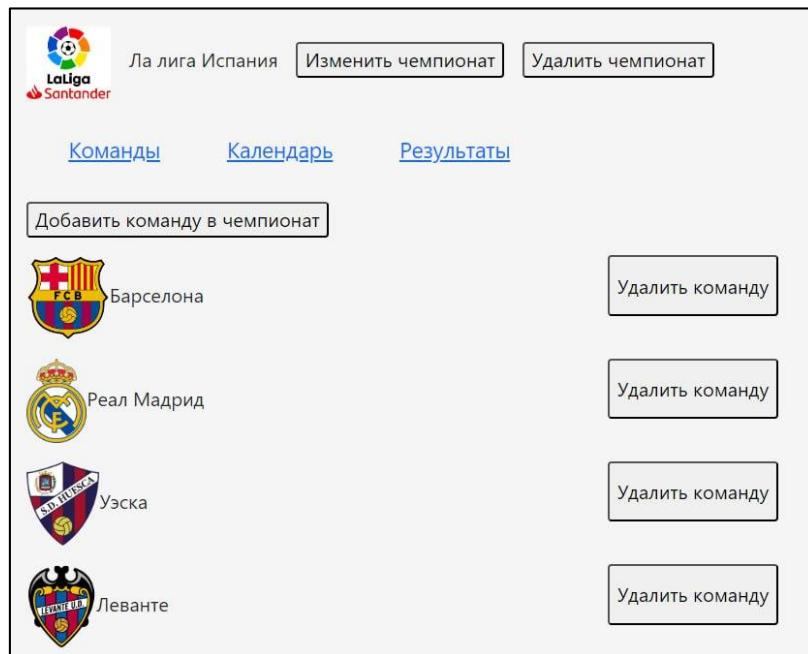


Рисунок 5.12 – Страница чемпионата

На странице чемпионата Администратор может просмотреть какие команды участвуют в чемпионате, расписание матчей чемпионата, результаты матчей, изменить и удалить чемпионат.

Нажав на кнопку «Изменить чемпионат» появляется окно для изменения чемпионата. Окно изменения показано на рисунке 5.13.

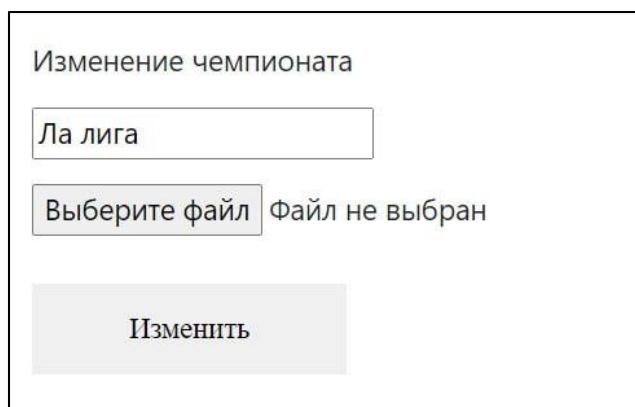


Рисунок 5.13 – Окно изменения чемпионата

При изменении чемпионата можно изменить название, а также выбрать новый логотип, затем нажать кнопку «Изменить». Если нет необходимости менять логотип, следует не выбирать изображение.

Для создания матча, сначала нужно перейти на вкладку «Календарь», затем нажать кнопку «Создать матч». Скриншот модального окна для создания матча показан на рисунке 5.14.

Создание матча

Дата:

Время:

Домашняя команда:

Гостевая команда:

Создать

Рисунок 5.14 – Скриншот модального окна для создания матча

При создании матча Администратор должен указать дату и время проведения матча, выбрать домашнюю команду и гостевую команду. При выборе одинаковых команд выведется ошибка о том, что выбранные команды одинаковые. После создания, матч будет добавлен в расписание матчей.

Для того чтобы удалить чемпионат Администратор должен на странице чемпионата нажать кнопку «Удалить чемпионат». После этого появится окно подтверждения удаления чемпионата. Окно подтверждения удаления чемпионата показано на рисунке 5.15.



Рисунок 5.15 – Окно подтверждения удаления чемпионата

После нажатия кнопку «Подтвердить» удалится чемпионат, а Администратора переадресует на главную страницу.

Для того чтобы внести результаты матча, необходимо на странице чемпионата, во вкладке «Календарь» нажать на нужный матч, после чего

Администратора переадресует на страницу матча. Страница матча показана на рисунке 5.16.

The screenshot shows a match statistics page with two team logos at the top: FC Barcelona (red and yellow) and Real Madrid (blue and white). The match details are as follows:

Статистика	Барселона	Реал Мадрид
Голы	2	0
Угловые	3	2
Удары	14	15
Удары в створ	7	6
Сэйвы	5	6
Владение	50	50
Фолы	5	7
Оффсайды	3	3

At the top right are two buttons: 'Сохранить' (Save) and 'Удалить матч' (Delete Match).

Рисунок 5.16 – Страница матча

Для того чтобы внести результаты матча, следует указать счет матча, заполнить статистику матча, далее нажать кнопку «Сохранить». При сохранении матча будет перенесен из вкладки с расписанием матча, во вкладку с результатами матча. Для того чтобы удалить матча, необходимо на этой же странице нажать соответствующую кнопку «Удалить матч». После этого появится окно с подтверждением удаления матча. Окно показано на рисунке 5.17.

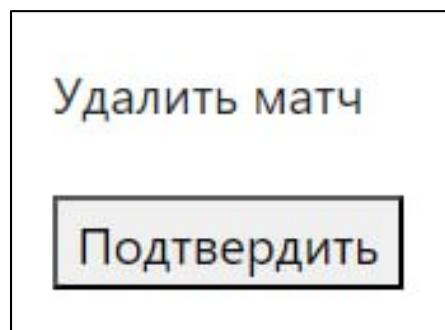


Рисунок 5.17 – Окно удаления матча

Также Администратор может менять роли существующих пользователей. Для этого необходимо перейти на страницу «Пользователи». Стоит отметить что в приложении существует Администратор, которому будет невозможно изменить роль. Страница показана на рисунке 5.18.

Регионы	Чемпионаты	Команды	Пользователи
admin@gmail.com		Администратор	Изменить роль
user@gmail.com		Пользователь	Изменить роль
bukmeker@gmail.com		Букмекер	Изменить роль
maksgoy2911@gmail.com		Администратор	Изменить роль
maks@gmail.com		Пользователь	Изменить роль
bukmeker1@gmail.com		Букмекер	Изменить роль
bukmeker2@gmail.com		Букмекер	Изменить роль
user1@gmail.com		Пользователь	Изменить роль
user2@gmail.com		Пользователь	Изменить роль

Рисунок 5.18 – Страница «Пользователи»

Сначала нужно нажать на кнопку «Изменить роль» у выбранного пользователя, после чего в появившемся окне выбрать новую роль. Окно выбора роли показано на рисунке 5.19.

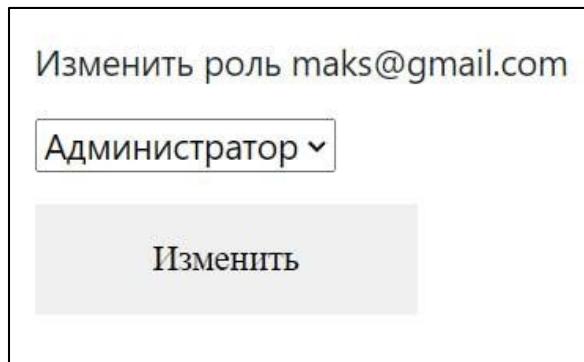


Рисунок 5.19 – Окно выбора роли пользователя

После того, как Администратор выбрал роль пользователю необходима нажать на кнопку «Изменить», после чего у пользователя измениться роль.

5.3 Роль «Букмекер»

В рамках разработанного веб-приложения Букмекер может выполнять следующие задачи:

- добавление исходов матча;
- ввод названия исхода и количества исходов;
- изменение названия исхода;
- изменение коэффициентов;
- просмотр статистики матча;
- выбор выигрышных исходов.

– После авторизации Букмекер попадает на страницу с чемпионатами. Страница Букмекера показана на рисунке 5.20.

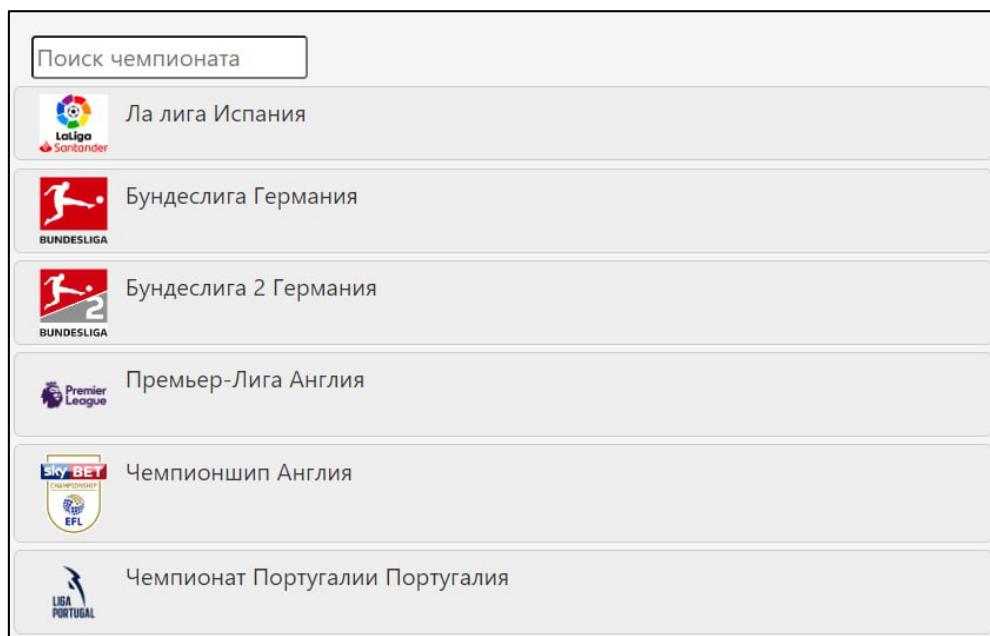


Рисунок 5.20 – Страница Букмекера

На странице для более быстрого поиска чемпионата можно воспользоваться поиском по названию, после чего перейти на страницу нужного чемпионата.

На странице матча уже будут определены первые три исхода, в которых Букмекеру необходимо поменять только коэффициенты.

Для того чтобы добавить новый исход, необходимо сначала перейти на страницу матча. Скриншот страницы матча показана на рисунке 5.21.

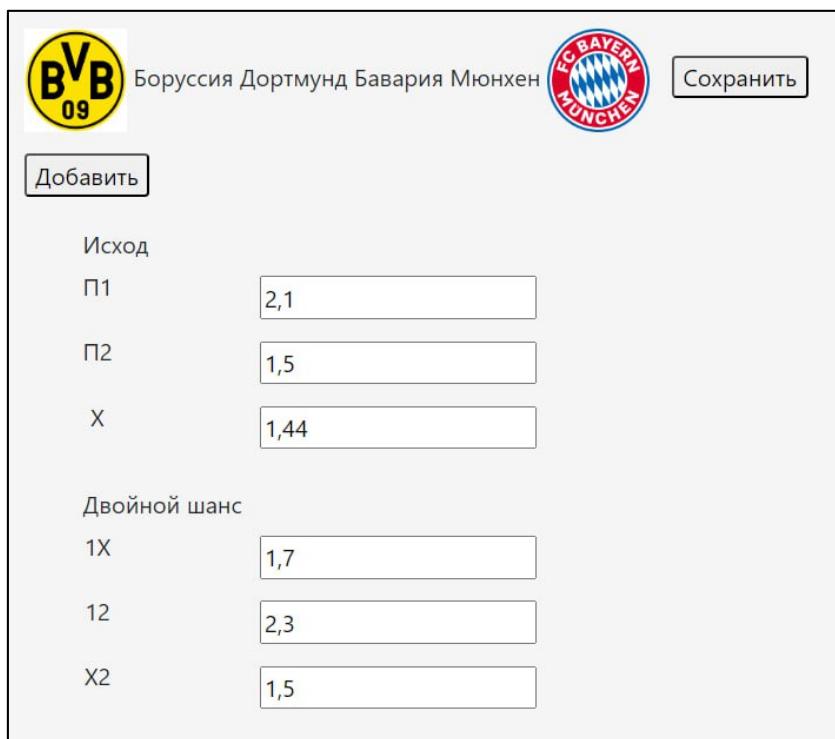


Рисунок 5.21 – Скриншот страницы матча

На данной страницу Букмекер должен нажать кнопку «Добавить», после чего появится окно для создания исхода. Окно создания исхода показано на рисунке 5.22.

Добавление исхода

Голы

2

Добавить

Рисунок 5.22 – Добавление нового исхода

При добавлении нового исход следуют указать название и количества исходов. Количество новых исходов должно быть не больше десяти и не меньше двух. После нажатия кнопки «Добавить» на странице матча появится добавленный исход, где Букмекер может поменять его название и коэффициенты. Новый исход на странице матча показан на рисунке 5.23.

Голы

Исход 1

1

Исход 2

1

Рисунок 5.23 – Созданный исход

Здесь букмекер может поменять название нового исхода, поменять название исходов, а также поменять коэффициенты на матч.

После того как матч будет сыгран, Букмекеру необходимо указать выигрышные исходы. Для этого следует перейти на страницу результатов матчей. Скриншот страницы результатов матчей показана на рисунке 5.24.

BUNDESLIGA

		Календарь		Результаты		
31.04 13:20		Бавария Мюнхен	1	2	Боруссия Дортмунд	
02.05 19:47		Штутгарт	1	0	Фрайбург	
02.05 19:53		Вольфсбург	3	4	Айнтрахт	

Рисунок 5.24 – Скриншот страницы результатов матчей

Далее нужно выбрать нужный матч и нажать на него. Скриншот страницы результата матча представлен на рисунке 5.25.



Рисунок 5.25 – Страница результата матча

На странице результата матча Букмекер может посмотреть всю информацию о матче, статистику, а также выбрать те исходы, которые являются выигрышными. Для этого Букмекер из выпадающего списка выбирает нужный исход. Для сохранения результатов необходимо нажать кнопку «Сохранить», после чего результаты исходов будут сохранены.

5.4 Вывод по разделу

Руководство пользователя было составлено в соответствии с функционалом разработанного веб-приложения для букмекерских ставок на футбольные события. В данном разделе представлены ключевые аспекты использования приложения для всех ролей, доступных в системе. Благодаря простому и интуитивно понятному интерфейсу, пользователи не столкнутся с трудностями при использовании приложения.

Разработанное руководство должно поспособствовать упрощенному и быстрому пониманию логики работы веб-приложения и благоприятно повлиять на начало работы пользователей как с имеющимся опытом работы с подобными сервисами, так и пользователей без богатого опыта.

6 Технико-экономическое обоснование проекта

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 будет доступен следующий функционал, который включает в себя:

- авторизацию и регистрацию;
- просмотр списка матчей;
- сортировка матчей по чемпионатам
- пополнение личного счета и вывод средств;
- просмотр истории счета;
- просмотр истории ставок;
- делать ставки 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.

Стратегия монетизации предполагает продажу приложения заказчику.

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.

				БГТУ 06.00.ПЗ		
	ФИО	Подпись	Дата			
Разраб.	Гой М.А.			6 Технико-экономическое обоснование проекта	Лит.	Лист
Провер.	Комарова Е.И.				у	1
Консульт.	Семенова Л.С.					7
Н. контр.	Николайчук А.Н.				74218010, 2023	
Утв.	Смелов В.В.					

Таблица 6.1 – Исходные данные для расчета

Наименование показателя	Условные обозначения	Норматив
Численность разработчиков, чел	$Ч_p$	1
Норматив дополнительной заработной платы, %	$H_{дз}$	15
Ставка отчислений в Фонд социальной защиты населения, %	$H_{фсзн}$	34
Ставка отчислений по обязательному страхованию в БРУСП «Белгосстрах», %	$H_{бгс}$	0,6
Норматив прочих прямых затрат, %	$H_{пз}$	20
Норматив накладных расходов, %	$H_{обп, обх}$	50
Ставка НДС, %	$H_{ндс}$	0
Налог на прибыль, %	$H_{п}$	0

Ставка НДС и налог на прибыль равняются 0, так как расчеты производились на базе компании-резидента 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	25306,15 руб.	Букмекерская контора для ставок на спорт. Имеет функционал, которого не будет в разработанном приложении, например ставки на другие виды спорта, что позволит сократить цену разработки.
Фонбет	https://www.fonbet.by/	22658,73 руб.	Букмекерская компания, которая имеет огромный выбор исходов матчей, позволяет быстро пополнять счет и выводить средства на карту.
MaxLine	https://maxline.by/	18474,80 руб.	Букмекерская компания, которая позволяет отслеживать коэффициента в онлайн режиме

Исходя из данных сайтов-калькуляторов, стоимость разработки аналогичных программных продуктов составляет 19000 – 26000 рублей.

Таким образом, общая стоимость разработки данного программного средства, выбранного в качестве базы сравнения, составит 22146,56 рублей.

6.3 Обоснование цены программного средства

В условиях современной рыночной экономики программное обеспечение, как правило, разрабатывается, тестируется 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
1	2
Разработка концепции приложения	43
Организация ввода информации	27
Контроль, предварительная обработка	27
Управление вводом/выводом	21
Взаимодействие между компонентами системы	50
Взаимодействие с базой данных	40
Вспомогательные методы	35
Обработка ошибочных и сбойных ситуаций	40
Графический вывод результатов	40

Окончание таблицы 6.3

1	2
Обработка ошибочных и сбойных ситуаций	40
Графический вывод результатов	40
Тестирование программных модулей	30
Доработка программного средства	40
Всего	393

Общее время разработки занимает 393 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 1667 рублей, часовая ставка составляет 9,92 руб/час.

Согласно таблице 6.2, проект разрабатывался одним специалистом на протяжении 393 часов. Таким образом, основная заработная плата будет рассчитываться по формуле 6.3

$$C_{оз} = T_{раз} \cdot C_{зп} \cdot K_{раз}, \quad (6.3)$$

где $C_{оз}$ – основная заработная плата, руб.;

$T_{раз}$ – время разработки (часов);

$C_{зп}$ – средняя часовая ставка руб./час;

$K_{раз}$ – количество разработчиков, человек.

$$C_{оз} = 393 \cdot 9,92 = 3898,56 \text{ руб.}$$

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).

$$C_{дз} = \frac{C_{оз} \cdot H_{дз}}{100}, \quad (6.4)$$

где $C_{оз}$ – основная заработная плата, руб.;

$H_{дз}$ – норматив дополнительной заработной платы, %.

$$C_{дз} = 3898,56 \cdot 15 / 100 = 584,78 \text{ руб.}$$

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

$$C_{\text{фсзн}} = \frac{(C_{\text{оз}} + C_{\text{дз}}) \cdot H_{\text{фсзн}}}{100}, \quad (6.5)$$

где $C_{\text{оз}}$ – основная заработная плата, руб.;

$C_{\text{дз}}$ – дополнительная заработка плата на конкретное ПС, руб.;

$H_{\text{фсзн}}$ – норматив отчислений в Фонд социальной защиты населения, %.

Отчисления в БРУСП «Белгосстрах» вычисляются по формуле 6.6

$$C_{\text{бгс}} = \frac{(C_{\text{оз}} + C_{\text{дз}}) \cdot H_{\text{бгс}}}{100}, \quad (6.6)$$

$$C_{\text{фсзн}} = \frac{(3898,56 + 584,78) \cdot 34}{100} = 1524,34 \text{ руб.}$$

$$C_{\text{бгс}} = \frac{(3898,56 + 584,78) \cdot 0,6}{100} = 26,90 \text{ руб.}$$

Таким образом, общие отчисления в БРУСП «Белгосстрах» составили 26,90 руб., а в фонд социальной защиты населения – 1524,34 руб.

6.3.5 Расчет суммы прочих прямых затрат

В экономическом разделе пояснительной записки подробно рассматриваются прочие прямые затраты, которые включают в себя несколько ключевых компонентов. Подробный анализ и учет каждого из этих компонентов позволит определить точную сумму прочих прямых затрат.

Таблица 6.4 – Прочие прямые затраты на разработку ПС

Статья затрат	Сумма
1	2
Регистрация доменного имени сроком на год	33,00
Оплата хостинга сроком на один год	378,00

Окончание таблицы 6.3

1	2
Среда разработки Visual Studio Professional сроком на три месяца	394,20
Итого	805,20

Далее найдём сумму прочих затрат по формуле:

$$C_{pz} = 33,00 + 378,88 + 394,20 = 805,20 \text{ руб.}$$

6.3.6 Расчет суммы накладных расходов

Сумма накладных расходов $C_{обп,обх}$ – произведение основной заработной платы исполнителей на конкретное программное средство C_{o3} на норматив накладных расходов в целом по организации $H_{обп,обх}$, по формуле 6.8.

$$C_{обп,обх} = \frac{C_{o3} \cdot H_{обп,обх}}{100}. \quad (6.8)$$

Сумма накладных расходов составит:

$$C_{обп,обх} = 3898,56 \cdot 50 / 100 = 1949,28 \text{ руб.}$$

6.3.7 Сумма расходов на разработку программного средства

Сумма расходов на разработку программного средства C_p 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.

$$C_p = C_{o3} + C_{дз} + C_{фсн} + C_{бгс} + C_{пз} + C_{обп,обх}. \quad (6.9)$$

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

$$C_p = 3898,56 + 584,78 + 1524,34 + 26,90 + 1949,28 + 805,20 = 8789,06 \text{ руб.}$$

Сумма расходов на разработку программного средства была вычислена на основе данных, рассчитанных ранее в данном разделе, и составила 8789,06 рублей.

6.3.8 Определение цены, оценка эффективности

Прибыль от реализации программного средства вычисляется по формуле 6.12.

$$\Pi_{nc} = \frac{C_n \cdot Y_{рент}}{100} \quad (6.12)$$

где $Y_{рент}$ – уровень рентабельности, %;

$C_{\text{п}}$ – полная себестоимость программного средства, руб.

Цена разработки программного средства без налогов находится по следующей формуле:

$$\Pi_{\text{р}} = C_{\text{п}} + \Pi_{\text{nc}} \quad (6.13)$$

Сумма налога на добавленную стоимость не рассчитывается, так как компания является резидентом ПВТ.

Исходя из вышеописанных данных рассчитаем прибыль от реализации программного средства, а также планируемую отпускную цену.

$$\Pi_{\text{nc}} = 8789,06 \cdot 30 / 100 = 2636,72 \text{ руб.}$$

$$\Pi_{\text{р}} = 8789,06 + 2636,72 = 11425,78 \text{ руб.}$$

Чистая прибыль равняется прибыли от реализации, так как компания является резидентом ПВТ и не облагается налогом.

6.4 Вывод по разделу

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

Таблица 6.4 – Результаты расчетов

Наименование показателя	Значение
Время разработки, ч.	393
Количество разработчиков, чел.	1
Основная заработка плата, руб.	3898,56
Дополнительная заработка плата, руб.	584,78
Отчисления в Фонд социальной защиты населения и БРУСП «Белгосстрах», %, руб.	1524,34
Отчисления в БРУСП «Белгосстрах», руб.	26,90
Прочие прямые затраты, руб	805,20
Накладные расходы, руб.	1949,28
Полная себестоимость, руб.	8789,06
Чистая прибыль, руб.	2636,72
Уровень рентабельности, %	30
Цена реализации, руб.	11425,78

Разработка программного средства одним работником за 393 часов, при заданных условиях, обойдется в 8789,06 рублей. Цена разработанного приложения составила 11425,78 рублей, что в два раза ниже чем у аналогичных решений на рынке. Реализация веб-приложения приложения по цене в 11425,78 рублей принесет прибыль организации в сумме 2636,72 рублей, при этом уровень рентабельности составит 30%.

Заключение

Целью дипломного проекта была разработка веб-приложения для букмекерских ставок на футбольные события.

Были решены следующие поставленные задачи:

– обзор аналогов: было рассмотрено три приложения и были выявлены их положительные и отрицательные стороны. Данная информация была использована для улучшения приложения и пользовательского интерфейса;

– проектирования веб-приложения: были проанализированы пользовательские потребности, построена диаграмма вариантов использования, архитектура веб-приложения и разработана структура базы данных. Были описаны и выбраны технологии и библиотеки для создания веб-приложения;

– реализация веб-приложения: было разработано веб-приложение, которое позволяет создавать, изменять и удалять регионы, чемпионаты и команды, создавать и удалять матчи, вносить результаты матчей, а также сделать ставку на футбольное событие. Серверная часть программного обеспечения была реализована с использованием технологии ASP.NET Core. Доступ к базе данных осуществлялся с помощью Entity Framework Core. Клиентская часть приложения была разработана с помощью библиотеки React. Для хранения состояния приложения использовалась библиотека Redux. Для стилизации пользовательского интерфейса использовалась библиотека Styled Components;

– тестирование веб-приложения: были протестированы, а также была проведена проверка на возможные ошибки. При тестировании использовались ручной и модульный методы тестирования. Анализ результатов показал, что система функционирует правильно и соответствует установленным требованиям;

– руководство пользователя: разработано руководство по использованию, для всех ролей были описаны основные возможности в системе. Это позволит легко ориентироваться в веб-приложении и быстро выполнять необходимые задачи;

– технико-экономическое обоснование проекта: было выявлено, что разработка программного средства одним работником за 393 часов, при заданных условиях, обойдется в 8789,06 рублей. Цена разработанного приложения составила 11425,78 рублей, что в два раза ниже чем у аналогичных решений. Реализация веб-приложения приложения по цене в 11425,78 рублей принесет прибыль организации в сумме 2636,72 рублей, при этом уровень рентабельности составит 30%.

Таким образом, была полностью достигнута поставленная цель по разработке программного средства и создан проект для ставок на футбольные события. Были учтены все требования, а задачи дипломного проекта выполнены в полном объеме.

				БГТУ 00.00.ПЗ		
		ФИО	Подпись	Дата	Лист.	
Разраб.	Гой М.А.				у	1
Провер.	Комарова Е.И.					1
Н. контр.	Николайчук А.Н.				74218010, 2023	
Утв.	Смелов В.В.					

Заключение

Список используемых источников

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> – Дата доступа: 20.04.2023
7. Entity Framework [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/ru-ru/ef/> – Дата доступа: 20.05.2023
8. React JS [Электронный ресурс]. – Режим доступа: <https://ru.reactjs.org> – Дата доступа: 20.04.2023
9. Styled Components – идеальная стилизация React-приложений [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/articles/591381/> – Дата доступа: 20.04.2023
10. Redux [Электронный ресурс]. – Режим доступа: <https://blog.skillfactory.ru/glossary/redux/> – Дата доступа: 25.04.2023
11. Microsoft SQL Server [Электронный ресурс]. – Режим доступа: <https://www.microsoft.com/en-us/sql-server> – Дата доступа: 20.04.2023
12. Ручное тестирование [Электронный ресурс]. – Режим доступа: <https://blog.skillfactory.ru/chto-takoe-ruchnoe-testirovanie> – Дата доступа: 23.05.2023
13. Валидация [Электронный ресурс]. – Режим доступа: <https://secrets.tinkoff.ru/glossarij/validaciya/> – Дата доступа: 23.05.2023
14. Модульное тестирование [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Модульное_тестирование – Дата доступа: 23.05.2023
15. Примеры использования Моq [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/articles/150859/> – Дата доступа: 23.05.2023

				БГТУ 00.00.ПЗ		
	ФИО	Подпись	Дата	Список используемых источников		
Разраб.	Гой М.А.			Lит.	Лист	Листов
Провер.	Комарова Е.И.			У	1	1
Н. контр.	Николайчук А.Н.			74218010, 2023		
Утв.	Смелов В.В.					

ПРИЛОЖЕНИЕ А

Диаграмма вариантов использования для Администратора

ПРИЛОЖЕНИЕ Б

Диаграмма вариантов использования для Букмекера

ПРИЛОЖЕНИЕ В

Диаграмма вариантов использования для Гостя и Пользователя

ПРИЛОЖЕНИЕ Г
Схема архитектуры веб-приложения

ПРИЛОЖЕНИЕ Д
Логическая схема базы данных

ПРИЛОЖЕНИЕ Е

Листинг класса BetRepository

```

using API.Entities;
using API.Interfaces;
using Microsoft.EntityFrameworkCore;
namespace API.Data
{
    public class BetRepository : IBet
    {
        private DataContext _context;
        public BetRepository(DataContext context)
        {
            _context = context;
        }
        public void AddBet(Match match, string name, int count)
        {
            var bet = new Bet
            {
                Name = name,
                MatchId = match.Id,
            };
            _context.Bet.Add(bet);
            _context.SaveChanges();
            var betValues = new List<BetValue>();
            for (int i = 0; i < count; i++)
            {
                _context.BetValue.Add(new BetValue
                {
                    Name = "Исход " + (i + 1),
                    Value = 1f,
                    BetId = bet.Id
                });
            }
        }
        public void Create(Bet item)
        {
            _context.Bet.Add(item);
        }
        public void DefineUserBets(int matchId)
        {
            var match = _context.Match.Find(matchId);
            if (match == null) return;
            foreach (var bet in match.Bets)
            {
                foreach (var betValue in bet.Values)
                {
                    var userBets = _context.UserBets.
                        Where(b => b.BetValueId == betValue.Id);
                    foreach (var userBet in userBets)
                    {
                        var user = _context.User.
                            FirstOrDefault(u => u.Id == userBet.UserId);
                        if (user != null)
                    }
                }
            }
        }
    }
}

```

```

{
if (betValue.IsConfirm == true && userBet.IsWin == null)
{
var history = new HistoryBankAccount
{
Status = "Выигрыш",
Date = DateTime.Now,
UserId = user.Id};
float money = (float)Math.Round(userBet.Money * userBet.Value, 2);
history.Money = money;
user.Money += money;
userBet.IsWin = true;
_context.HistoryBankAccounts.Add(history);
}
else if (betValue.IsConfirm == false && userBet.IsWin == null)
{
userBet.IsWin = false;
_context.Entry(userBet).State = EntityState.Modified;
_context.Entry(user).State = EntityState.Modified; } } } }
public void Delete(Bet bet)
{
context.Bet.Remove(bet); }
public void DoBet(int betId, int userId, float amount)
{
var betValue = _context.BetValue.Find(betId);
if (betValue == null) { return; }
var user = _context.User.Find(userId);
if (user == null) { return; }
var status = betValue.Bet.Match.Home.Name + " - " +
betValue.Bet.Match.Away.Name;
var userBet = new UserBet
{
UserId = userId,
BetValueId = betId,
Money = amount,
Value = betValue.Value,
Match = status };
_context.UserBets.Add(userBet);
}
public async Task<Bet> Get(int id)
{
return await _context.Bet.FindAsync(id);
}
public async Task<IEnumerable<Bet>> GetAll()
{
return await _context.Bet.ToListAsync();
}
public async Task<BetValue> GetBetValue(int id)
{
return await _context.BetValue.FindAsync(id);
}
public async Task<UserBet?> GetUserBet(int userId, int betValueId)
{
var userBets = await _context.
UserBets.
Where(ub => ub.UserId == userId).
ToListAsync();
var betValue = _context.BetValue.Find(betValueId);
if (betValue == null)
}

```

```
return null;
if (userBets.Count == 0)
return null;
foreach (var userBet in userBets)
{
var id = userBet.BetValue.Bet.Match.Id;
var id1 = betValue.Bet.Match.Id;
if (userBet.BetValue.Bet.Match.Id == betValue.Bet.Match.Id)
return userBet;}
public async Task<IEnumerable<UserBet>> GetUserBets(int userId)
{
var userBets = await _context.UserBets.
Where(i=>i.UserId==userId).
ToListAsync();
return userBets;}
public async Task<bool> IsOutcomeInMatch(Match match, string name)
{
var bet = match.Bets.FirstOrDefault(b => b.Name == name);
if (bet == null)
{
return false;}
return true;}
public void SaveBetsResultMatch(IEnumerable<Bet> bets)
{
foreach(var bet in bets)
{
foreach (var betValue in bet.Values)
{
_context.Entry(betValue).State = EntityState.Modified;
}}}
public void Update(Bet item)
{_context.Entry(item).State = EntityState.Modified;
}}}
```

ПРИЛОЖЕНИЕ Ж

Блок-схема алгоритма подтверждения ставки

ПРИЛОЖЕНИЕ И

Листинг кода контроллера ChampionshipController

```

using API.Controllers.Base;
using API.DTOs;
using API.Entities;
using API.Interfaces;
using API.Service;
using AutoMapper;
using Microsoft.AspNetCore.Mvc;

namespace API.Controllers
{
    public class ChampionshipController : BaseApiController
    {
        private readonly IUnitOfWork _unitOfWork;
        private readonly IMapper _mapper;
        private readonly IPhotoService _photoService;

        public ChampionshipController(IUnitOfWork unitOfWork, IMapper mapper, IPhotoService photoService)
        {
            _unitOfWork = unitOfWork;
            _mapper = mapper;
            _photoService = photoService;
        }

        [HttpGet("GetChampionships")]
        public async Task<IEnumerable<Championship>> GetAllChampionships()
        {
            return await _unitOfWork.Championship.GetAll();
        }

        [HttpPost("AddTeam/{championshipId:int}-{teamId:int}")]
        public async Task<ActionResult> AddTeamInChampionship(int championshipId, int teamId)
        {
            Team team = await _unitOfWork.Team.Get(teamId);

            Championship championship = await
            _unitOfWork.Championship.Get(championshipId);

            if (team == null) return NotFound("Команда не найдена");

            if(championship == null) return NotFound("Чемпионат не
найден");

            var IsTeamInChampionship = await
            _unitOfWork.Championship.TeamExistsInChampionship(championship.Id,
team.Id);
        }
    }
}

```

```

        if (IsTeamInChampionship) return BadRequest("Команда уже
в чемпионате");

_unitOfWork.Championship.AddTeamInChampionship(championship.Id,
team.Id);

if(await _unitOfWork.Complete())
    return Ok("Команда добавлена в чемпионат");

return BadRequest("Failed to add team in championship");
}

[HttpDelete("DeleteTeam/{championshipId:int}-{teamId:int}")]
public async Task<ActionResult>
DeleteTeamFromChampionship(int championshipId, int teamId)
{
    var championship = await
_unitOfWork.Championship.Get(championshipId);

    if (championship == null)
        return NotFound("Чемпионат не найден");

    var team = await _unitOfWork.Team.Get(teamId);

    if (team == null) return NotFound("Команда не найдена");

    var teamExists = await _unitOfWork.Championship.
        TeamExistsInChampionship(championshipId, teamId);

    if(!teamExists) return BadRequest("Такой команды нет в
чемпионате");

_unitOfWork.Championship.DeleteTeamFromChampionship(championshipId,
teamId);

if (await _unitOfWork.Complete())
    return Ok("Команда удалена из чемпионата");

return BadRequest("Ошибка при удалении команды из
чемпионата");
}

[HttpDelete("DeleteChampionship/{championshipId:int}")]
public async Task<ActionResult> deleteChampionship(int
championshipId)
{
    Championship championship = await
_unitOfWork.Championship.Get(championshipId);

    if (championship == null) return NotFound("Чемпионат не
найден");
}

```

```

        _unitOfWork.Championship.Delete(championship);

        if (await _unitOfWork.Complete())
            return Ok("Чемпионат удален");

        return BadRequest("Не удалось удалить чемпионат");
    }

[HttpPost("CreateChampionship")]
public async Task<ActionResult> CreateChampionship()
{
    var req = Request.Form;

    string? name = req["name"];
    string? regionName = req["region"];
    var image = req.Files["image"];

    if (name == null || name.Length < 3)
        return BadRequest("Длина названия должна быть минимум
3 символа");

    var region = await
_unitOfWork.Region.GetRegionByName(regionName);

    if (region == null) return NotFound("Регион не найден");

    var championshipExists = await _unitOfWork.
Championship.
GetChampionshipInRegionByName(name, region);

    if (championshipExists != null)
        return BadRequest("Такой чемпионат уже существует в
регионе");

    var championship = new Championship
    {
        Name = name,
        Region = region,
        Image = null
    };

    _unitOfWork.Championship.Create(championship);

    if (!await _unitOfWork.Complete())
        return BadRequest("Не удалось сохранить чемпионат");

    var pathImage = image == null ? "" :
        _photoService.AddPhoto(Request,
"images/championships/" + championship.Id + ".png", image);

    championship.Image = pathImage;

    _unitOfWork.Championship.Update(championship);
}

```

```

        if (await _unitOfWork.Complete())
            return Ok("Чемпионат создан");

        return BadRequest("Не удалось создать");
    }
    [HttpGet("GetRegionalChampionships")]
    public async Task<IEnumerable<Championship>>
GetRegionalChampionships(int regionId)
{
    return await
    _unitOfWork.Championship.GetRegionalChampionships(regionId);
}

[HttpGet("GetChampionship")]
public async Task<Championship> GetChampionship(int id) {
    return await _unitOfWork.Championship.Get(id);
}

[HttpGet("GetChampionshipTeams")]
public async Task<IEnumerable<Team>> GetChampionshipTeams(int
id)
{
    return await _unitOfWork.Team.GetChampionshipTeams(id);
}

[HttpDelete("DeleteChampionship")]
public async Task<ActionResult> DeleteChampionship(int id)
{
    var championship = await
    _unitOfWork.Championship.Get(id);

    if (championship == null)
        return BadRequest("Чемпионат не найден");

    _unitOfWork.Championship.Delete(championship);

    if (await _unitOfWork.Complete())
        return Ok("Чемпионат удален");

    return BadRequest("Не удалось удалить чемпионат");
}

[HttpPost("EditChampionship")]
public async Task<ActionResult> EditChampionship(int id)
{
    var championship = await
    _unitOfWork.Championship.Get(id);

    if (championship == null)
        return BadRequest("Чемпионат не найден");

    var req = Request.Form;
}

```

```

        string? name = req["name"];
        var image = req.Files["image"];

        if (name == null & image == null)
            return BadRequest();

        if (name.Length < 3)
            return BadRequest("Длина названия должна быть минимум
3 символа");

        if (championship.Name != name)
        {
            var region = await
_unitOfWork.Region.Get(championship.RegionId);

            var chExists = await _unitOfWork.Championship.
GetChampionshipInRegionByName(name, region);

            if (chExists != null)
                return BadRequest("Такой чемпионат в регионе уже
существует");
        }

        if (name != null)
            championship.Name = name;

        if (image != null)
        {
            if (championship.Image != null)

(photoService.RemovePhoto(championship.Image.Replace(Request.Scheme +
"://" + Request.Host.ToUriComponent() + "/", ""));

            championship.Image = _photoService.AddPhoto(Request,
"images/championships/" + image.FileName, image);
        }

        _unitOfWork.Championship.Update(championship);

        if (await _unitOfWork.Complete())
            return Ok("Чемпионат изменен");

        return BadRequest("Не удалось изменить чемпионат");
    }
}
}

```

ПРИЛОЖЕНИЕ К

Листинг кода компонента создания команды

```

import axios, { AxiosError } from "axios";
import { useEffect, useState } from "react";
import { useAllRegions } from "../../hooks/region";
import styled from "styled-components";

export function CreateTeam() {
  const [name, setName] = useState("");
  const [image, setImage] = useState(null);
  const [buttonState, setButton] = useState("Создать");
  const [errorCreate, setErrorCreate] = useState("");
  const [region, setRegion] = useState("Испания");
  const { regions, error, loading } = useAllRegions();
  useEffect(() => {
    setRegion(regions == null || regions.length < 1 ? "Мир" :
regions[0].name);
  }, [regions]);
  const FetchCreateTeam = async (event: any) => {
    event.stopPropagation();
    setButton("Создание");
    const formData = new FormData();
    formData.append("name", name);
    formData.append("region", region);
    if (image != null) formData.append("image", image);
    try {
      const response = await axios.post(
        "https://localhost:7167/api/Team/CreateTeam",
        formData);
      window.location.assign("/admin/teams")
    } catch (e: unknown) {
      const error = e as AxiosError;
      const message = error.response?.data as String;
      setErrorCreate(message.toString());
      setButton("Создать");
    }
    const AddImage = (e: any) => {
      setImage(e.target.files[0]);
    }
    return (
      <FormContainer>
        <Div>Создание команды</Div>
        <Div>
          <input value={name} placeholder="Название" onChange={(e) =>
            setName(e.target.value)} />
        </Div>
        <Div>
          <input
            type="file"
            accept="image/*,.png,.jpg,.gif,.web"
            onChange={AddImage}/>
        </Div>
      </FormContainer>
    );
  };
}

```

```
</Div>
<Div>
<select onChange={(e) => setRegion(e.target.value)}>
{regions.map((region) => (
<option>{region.name}</option>
)) }
</select>
</Div>
<Div>{errorCreate}</Div>
<StyledButton onClick={FetchCreateTeam}>{buttonState}</StyledButton>
</FormContainer>
);
}
```

```
const FormContainer = styled.div`  
font-size: 18px;  
display: flex;  
justify-content: center;  
flex-direction: column;  
width: 25vw;  
`;
```

```
const StyledButton = styled.button`  
justify-content: center;  
align-items: center;  
display: flex;  
height: 56px;  
padding-left: 60px;  
padding-right: 60px;  
font-size: 18px;  
font-family: "Montserrat-Bold";  
text-align: center;  
border: none;  
:hover {  
cursor: pointer;  
}  
transition-duration: 0.4s;  
`;
```

```
const Div = styled.div`  
margin-bottom: 15px;`;
```

ПРИЛОЖЕНИЕ Л

Листинг кода страницы календаря матчей

```

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>
  );
}

```

ПРИЛОЖЕНИЕ М

Листинг кода «позитивного» теста

```
[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);
    }
```

ПРИЛОЖЕНИЕ Н
Скриншот главной страницы веб-приложения

ПРИЛОЖЕНИЕ П

Таблица результатов расчета экономических показателей