МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка веб-приложения по прогнозу итогов спортивных встреч

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.04.2024.308-333.ВКР

Научный руководитель,
профессор кафедры СП, д.г.н.,
к.ф.-м.н.
_____ С.М. Абдуллаев

Автор работы,
студент группы КЭ-403
_____ С.Н. Берестюк

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студенту группы КЭ-403
Берестюку Сергею Николаевичу,
обучающемуся по направлению
09.03.04 «Программная инженерия»

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12)

Разработка веб-приложения по прогнозу итогов спортивных встреч.

2. **Срок сдачи студентом законченной работы:** 03.06.2024 г.

3. **Исходные данные к работе**

3.1. Форсье Д. Django. Разработка веб-приложений на Python / Д. Форсье, П. Биссекс, У. Чан; [пер. с англ. А. Киселева]. – Санкт-Петербург: Символ-Плюс, 2010. – 456 с.

3.2. The Python Standard Library – Python 3.11. [Электронный ресурс] URL: https://docs.python.org/3.11/library (дата обращения: 10.02.2024 г.).

3.3. Django Documentation. [Электронный ресурс] URL: https://docs.djangoproject.com (дата обращения: 10.02.2024 г.).

3.4. MySQL Documentation. [Электронный ресурс] URL: https://dev.mysql.com/doc/ (дата обращения: 10.02.2024 г.).

4. **Перечень подлежащих разработке вопросов**

4.1. Анализ предметной области.

4.2. Обзор существующих решений.

4.3. Проектирование веб-приложения.

4.4. Реализация и тестирование веб-приложения.

5. Дата выдачи задания: 29.01.2024 г.

Научный руководитель,
профессор кафедры СП, д.г.н., к.ф.-м.н. С.М. Абдуллаев

Задание принял к исполнению С.Н. Берестюк

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

Спорт играет огромную роль в современном обществе, и его популярность продолжает расти. События, связанные с спортом, привлекают огромное количество зрителей и участников, что создает повышенный спрос на ставки. Развитие веб-технологий открывает новые возможности для создания инновационных продуктов в сфере спортивных ставок. В современном мире широко распространены букмекерские конторы, предоставляющие возможность делать ставки на спортивные события как в реальном времени, так и заранее.

В России букмекерские конторы достаточно популярны, они рекламируются на федеральных телеканалах. Три главных футбольных лиги страны и национальный Кубок забрендированы ведущими беттинговыми компаниями, большинство клубов РПЛ имеют спонсорские контракты с букмекерскими компаниями. Некоторые команды в рамках сотрудничества с букмекерами даже идут на смену названия.

С развитием интернета и мобильных технологий ставки можно делать прямо из дома с мобильного устройства. Это значительно упрощает доступ к букмекерским услугам, делая их более удобными и доступными для широкой аудитории. За 2023 год объем принятых российскими легальными букмекерами ставок составил 1,224 триллиона рублей, из которых 1,158 триллиона рублей пришлось на онлайн-сегмент, а остальные 65,6 миллиарда рублей – на ставки наличными в пунктах приема. А общее количество игроков на ставках превысило 15 миллионов человек, среди которых более 40% являются активными пользователями.

Однако важно помнить, что азартные игры, включая ставки в букмекерских конторах, могут вызывать проблемы с игорной зависимостью и иметь негативное влияние на финансовое положение человека. Поэтому создание веб-приложения по прогнозу итогов спортивных встреч, которое не

предусматривает каких-либо финансовых вложений, может иметь актуальность, поскольку оно может быть полезным инструментом для развития аналитических навыков и способствовать формированию здорового отношения к азартным играм.

Постановка задачи

Целью выпускной квалификационной работы является разработка веб-приложения по прогнозу итогов спортивных встреч. Для достижения поставленной цели необходимо решить следующие задачи:

1) проанализировать предметную область;
2) произвести обзор существующих решений;
3) спроектировать веб-приложение;
4) реализовать и протестировать веб-приложение.

Структура и содержание работы

Работа состоит из введения, трех глав, заключения и списка литературы. Объем работы составляет 36 страниц, объем списка литературы – 17 источников.

В первой главе описывается анализ предметной области, включающий обзор аналогичных решений и выбор средств разработки.

Вторая глава посвящена проектированию веб-приложения.

В третьей главе происходит реализация и тестирование веб-приложения.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Целью данной работы является реализация веб-приложения по прогнозу итогов спортивных встреч. Веб-приложение представляет собой платформу, на которой участники могут делать прогнозы на исходы спортивных событий и коммуницировать друг с другом. Сайт должен предоставлять информацию о предстоящих спортивных мероприятиях, о результатах прогнозов пользователей и успешности прогнозов пользователя в общем зачете.

1.1. Обзор существующих решений

Прежде чем приступить к разработке веб-приложения, необходимо проанализировать существующие сайты схожей тематики. Это поможет определиться с необходимой структурой и содержанием, а также выявить положительные стороны реализации, которые можно включить в работу.

Fon.bet [1]

На рисунке 1 изображена главная страница сайта. На ней располагается логотип, меню для навигации по сайту, карусель с рекламными баннерами. Также в шапке сайта расположены кнопки для авторизации и регистрации пользователей. На сайте доступен поиск по ключевому слову.

Основное пространство сайта занимает информация о предстоящих событиях. Есть возможность фильтрации событий по виду спорта и страны в которой оно проходит. Также можно отображать события, которые проходят в данный момент.

Из плюсов можно отметить, что есть возможность настроить сайт под себя, можно изменить вид навигации по событиям, вид росписи событий, тему сайта и тип коэффициентов.

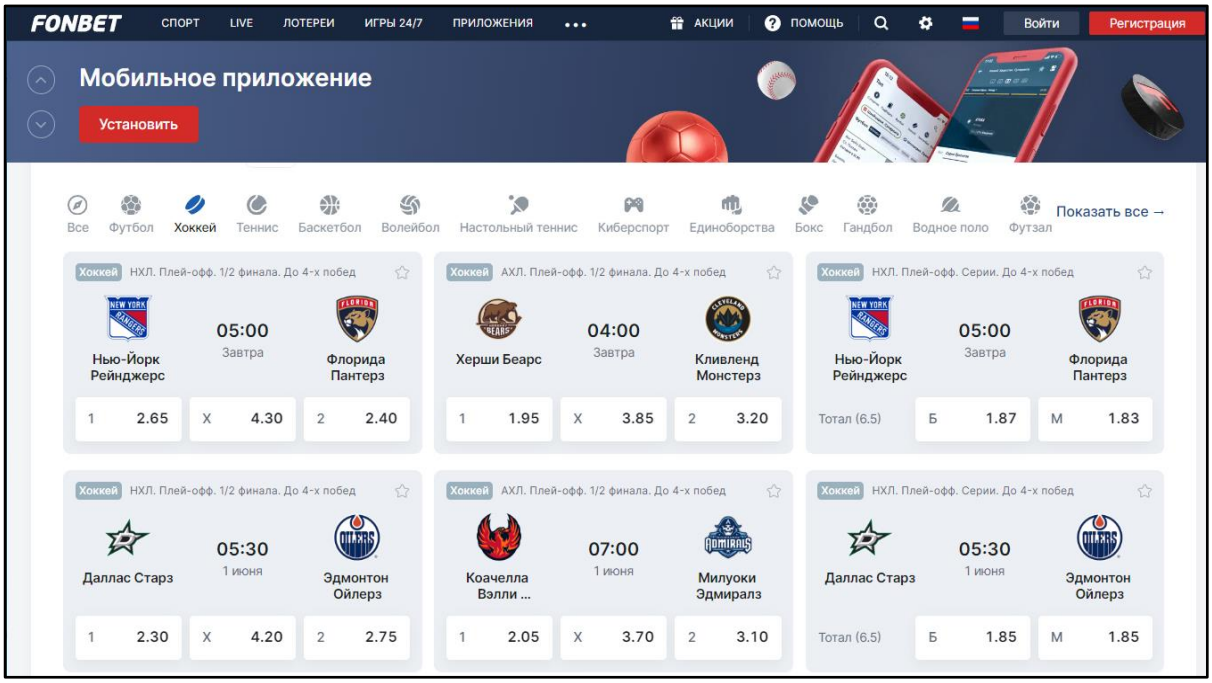

Рисунок 1 – Скриншот главной страницы сайта «Fon.bet»

Winline.ru [2]

На рисунке 2 изображена главная страница сайта. Шапка сайта схожа с предыдущем проектом и также содержит логотип, меню для навигации и кнопки для авторизации.

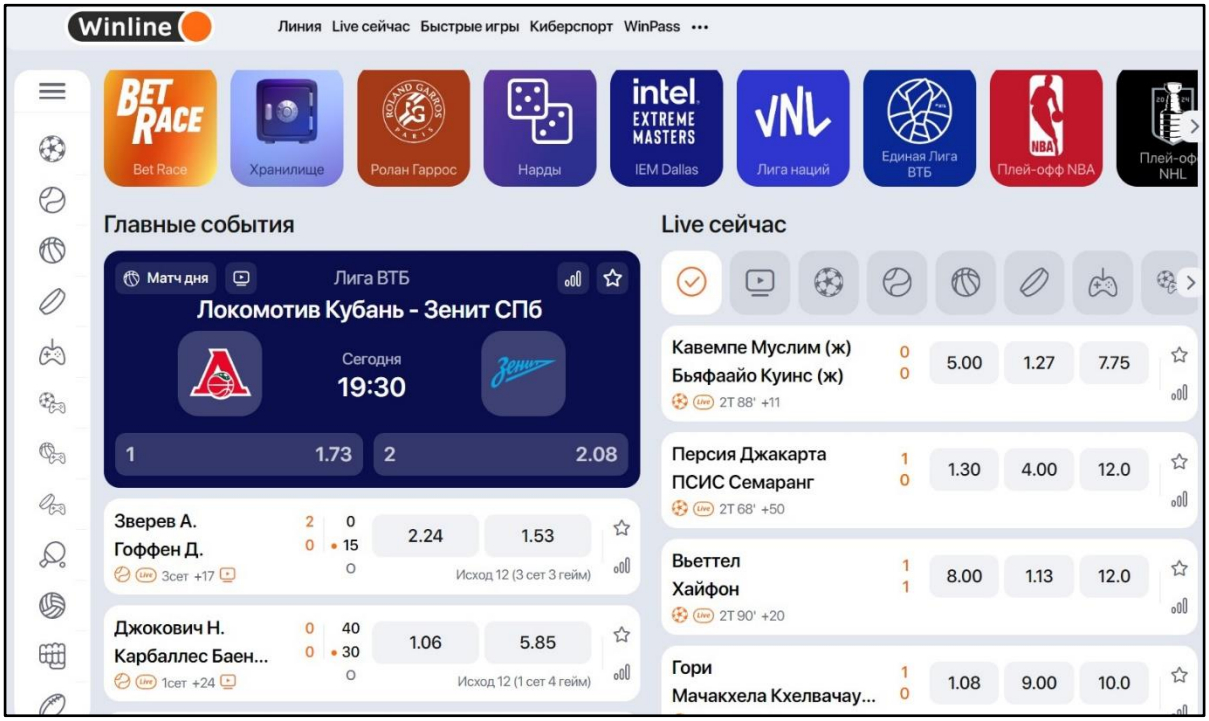

Рисунок 2 – Скриншот главной страницы сайта «Winline.ru»

Основное пространство сайта также занимает информация о предстоящих событиях, в отличии от предыдущего сайта информация здесь отображается более компактно и ее больше, из-за этого может возникать ощущение информационного перегруза, информация может сложнее восприниматься. Также есть возможность фильтрации событий по виду спорта.

Настройки отображения информации менее обширны, есть возможность только поменять тему на светлую/темную и увеличить шрифты.

Pari.ru [3]

На рисунке 3 изображена главная страница сайта Pari.ru. Как и на предыдущих аналогах в шапке странице располагается логотип компании, навигационное меню и кнопки для авторизации.

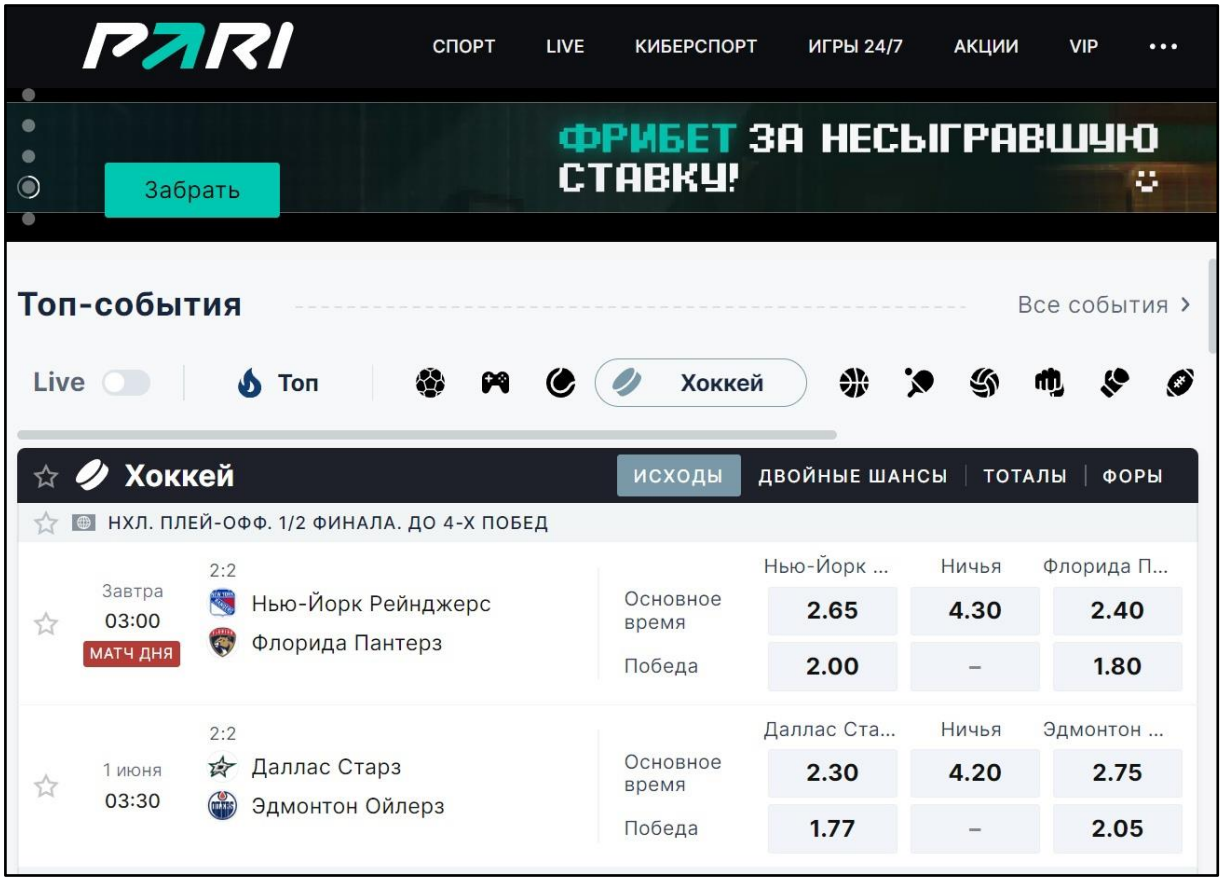

Рисунок 3 – Скриншот главной страницы сайта «Pari.ru»

Основное пространство страницы занимает информацию о предстоящих спортивных встречах. Информация хорошо структурирована и удобна для восприятия. Функциональность сайта позволяет фильтровать события

по виду спорта, а также по типу ставки, которую вы хотите сделать. Также стоит отметить обширные параметры настройки отображения сайта под себя.

1.2. Обзор средств разработки

Бэкенд представляет собой важнейший компонент проекта, который не виден конечному пользователю. В отличие от бэкенда, фронтенд отвечает за представление пользовательского интерфейса программного обеспечения или веб-сайта. Фронтенд называют презентационным слоем, а бэкенд – слоем доступа и обработки данных.

Бэкенд включает в себя несколько процессов, некоторые из которых выполняют:

1) хранение данных в базе данных;
2) доступ к данным из базы данных с помощью SQL запросов;
3) обработка входящих запросов веб-страниц;
4) применение JavaScript для обработки пользовательского ввода.

Python

Это универсальный язык программирования высокого уровня, известный своей интерпретируемостью и ориентацией на удобство чтения и понимания. Обладая обширной библиотекой стандартных функций, Python поддерживает множество парадигм программирования, включая объектно-ориентированное, структурированное и функциональное программирование [4]. Его главными преимуществами являются простой синтаксис и кроссплатформенная совместимость, позволяющая без проблем работать на широком спектре операционных систем. Python отлично справляется с упрощением сложных задач с помощью всего нескольких строк кода и обеспечивает простую интеграцию с такими языками, как Java, C и C++.

Несмотря на многочисленные достоинства, Python имеет и некоторые недостатки. Например, его использование может привести к увеличению использования памяти и может быть невыгодным, особенно когда речь идет

об оптимизации памяти. Кроме того, построчное выполнение кода Python может привести к снижению скорости его обработки, из-за этого клиентские или мобильные приложения могут не воспринимать его. Поскольку это язык с динамической типизацией, тип данных может неожиданно измениться во время выполнения, что может привести к ошибкам во время выполнения.

Основанный на языке Python фреймворк Django обладает надежной и адаптируемой архитектурой, что делает его одним из лучших кандидатов для создания обширных и сложных веб-приложений, таких как платформы электронной коммерции. Известный своим обширным набором функций, Django предлагает разработчикам множество готовых решений, включающих аутентификацию пользователей, обработку форм, управление базами данных и многое другое. Такой богатый набор встроенных функций значительно ускоряет жизненный цикл разработки и одновременно снижает вероятность ошибок. Более того, следование принципу «Не повторяй себя» (DRY) в Django способствует упорядоченной практике разработки, улучшая сопровождаемость и масштабируемость кода.

Кроме того, Python и Django имеют обширные сообщества разработчиков, которые создают и поддерживают множество библиотек, пакетов и плагинов, которые могут быть использованы для создания надежного, функционального и эффективного веб-приложения. Такое обилие ресурсов значительно упрощает задачи разработчиков, повышая продуктивность проектов за счет предоставления легкодоступных решений для различных задач разработки.

JavaScript

JavaScript – это универсальный язык программирования, используемый в основном для создания динамического и интерактивного контента на веб-сайтах [5]. Его универсальность заключается в возможности работать как на стороне клиента, так и на стороне сервера, что делает его важной частью веб-разработки.

Одно из ключевых преимуществ JavaScript – его повсеместность. Практически каждый веб-браузер поддерживает JavaScript, что делает его доступным инструментом для разработчиков, позволяющим улучшить пользовательский опыт. Его синтаксис относительно прост для понимания, что делает его доступным для новичков и в то же время предоставляет мощные возможности для более опытных разработчиков. С помощью JavaScript разработчики могут манипулировать HTML и CSS, создавать анимацию, обрабатывать пользовательский ввод и асинхронно взаимодействовать с серверами. Благодаря таким технологиям, как AJAX (Asynchronous JavaScript and XML) [6], разработчики могут получать данные с серверов без необходимости перезагрузки всей веб-страницы, что приводит к более плавному взаимодействию с приложением.

Кроме того, JavaScript имеет большое количество библиотек и фреймворков, таких как React, Angular и Vue.js, которые упрощают и ускоряют процесс разработки, предоставляя уже готовые компоненты и оптимизированные рабочие процессы.

MySQL

Django совместим с различными системами управления базами данных (СУБД), включая MySQL. MySQL – это мощная реляционная база данных, которая широко используется в веб-разработке [7]. Рассмотрим преимущества и недостатки использования Django с MySQL.

Преимущества использования MySQL с Django.

1. Надежность и производительность: MySQL известна своей высокой производительностью и надежностью, что делает ее отличным выбором для крупных и масштабируемых веб-приложений.

2. Богатый функционал: MySQL предлагает обширный набор функций и возможностей, включая транзакции, хранимые процедуры, триггеры, полнотекстовый поиск и многое другое, что может быть полезно при проектировании сложных приложений.

3. Широкая поддержка и сообщество: MySQL имеет обширное сообщество разработчиков и пользователей, что обеспечивает доступ к большому количеству документации и учебных материалов.

4. Масштабируемость: MySQL позволяет масштабировать базу данных вертикально и горизонтально, что обеспечивает адаптацию к изменяющимся потребностям проекта.

Недостатки использования MySQL с Django.

1. Требуется установка и настройка: для работы с MySQL необходимо установить и настроить отдельный сервер баз данных, что требует дополнительных шагов по сравнению с SQLite.

2. Сложность настройки: первоначальная настройка MySQL может потребовать много времени и усилий, особенно для новых пользователей Django, по сравнению с более простыми СУБД, такими как SQLite.

3. Управление соединениями: в случае большого количества одновременных соединений или неоптимизированных запросов может возникнуть необходимость более тщательного управления соединениями с базой данных для обеспечения оптимальной производительности.

Вывод по первой главе

По результатам обзора аналогичных проектов и средств разработки для реализации функциональности сайта было решено использовать язык программирования Python и фреймворк Django. В качестве базы данных будет использована MySQL.

2. ПРОЕКТИРОВАНИЕ

2.1. Требования к веб-приложению

После рассмотрения предметной области и обзора доступных средств разработки были выявлены функциональные и нефункциональные требования, которые должны быть учтены при разработке веб-приложения.

Функциональные требования

Были выявлены следующие функциональные требования.

1. Система должна предоставлять возможность регистрации и входа для пользователей.
2. Система должна предоставлять возможность делать прогнозы на спортивные события.
3. Система должна рассчитывать результат прогноза пользователя.
4. Система должна предоставлять информацию об общих успехах пользователей.
5. Пользователь должен иметь возможность просматривать историю своих прогнозов.
6. Администратор должен иметь возможность добавления и редактирования спортивных событий.

Нефункциональные требования

Были выявлены следующие нефункциональные требования.

1. Система должна быть реализовано на языке Pyhton в фреймворке Django.
2. Данные системы должны храниться в базе данных MySQL.

2.2. Диаграмма вариантов использования

Диаграмма вариантов использования была составлена с использованием унифицированного языка моделирования UML [8]. Были выделены следующие актеры, взаимодействующие с системой: администратор, авторизованный пользователь и неавторизованный пользователь.

Администратор – это лицо, имеющее права на создание и редактирование спортивных событий.

Авторизованный пользователь – это человек зарегистрированный и прошедший авторизацию на сайте, ему доступен основной функционал веб-приложения.

Неавторизованный пользователь – это любой незарегистрированный на сайте человек, он не имеет доступа к личному кабинету.

Диаграмма вариантов использований представлена на рисунке 4, она визуально отображает взаимодействие актеров с прецедентами в системе, создавая наглядное представление о функциональности и возможностях системы.

Рисунок 4 – Диаграмма вариантов использования

Краткое описание вариантов использования для актера «Администратор» представлено ниже.

1. Добавить событие. «Администратор» может добавить новое спортивное событие.

2. Изменить данные события. «Администратор» может редактировать событие: изменять статус мероприятия, записывать итоговый счет.

Краткое описание вариантов использования для актера «Авторизованный пользователь» представлено ниже.

1. Просмотреть события. Пользователь может ознакомиться с предстоящими матчами.
2. Сделать прогноз. Возможность сделать прогноз на результат выбранного матча.
3. Просмотреть историю прогнозов. Пользователь может посмотреть историю своих прогнозов и их результаты.
4. Просмотреть список лидеров. Пользователь может посмотреть на каком месте он находится в таблице общего зачета.
5. Открыть чат. Пользователь может открыть общий чат, расположенный на главной странице сайта.
6. Отправить сообщение. Пользователь может отправить сообщение в общий чат.

Краткое описание вариантов использования для актера «Неавторизованный пользователь» представлено ниже.

1. Регистрация. Возможность зарегистрироваться в системе.
2. Авторизация. Возможность авторизоваться в системе под своими учетными данными.

2.3. Диаграмма компонентов системы

В статье представлен краткий обзор традиционной трехслойной архитектуры, присущей веб-приложениям [9]. В ней выделены основные компоненты, составляющие любое веб-приложение: презентационный слой, отвечающий за визуализацию пользовательского интерфейса; слой бизнес-логики, управляющий функциональностью и обработкой данных приложения; и слой хранения данных, обеспечивающий управление и сохранение данных.

В качестве графического представления структуры разрабатываемого веб-приложения была спроектирована диаграмма компонентов, представленная на рисунке 5.

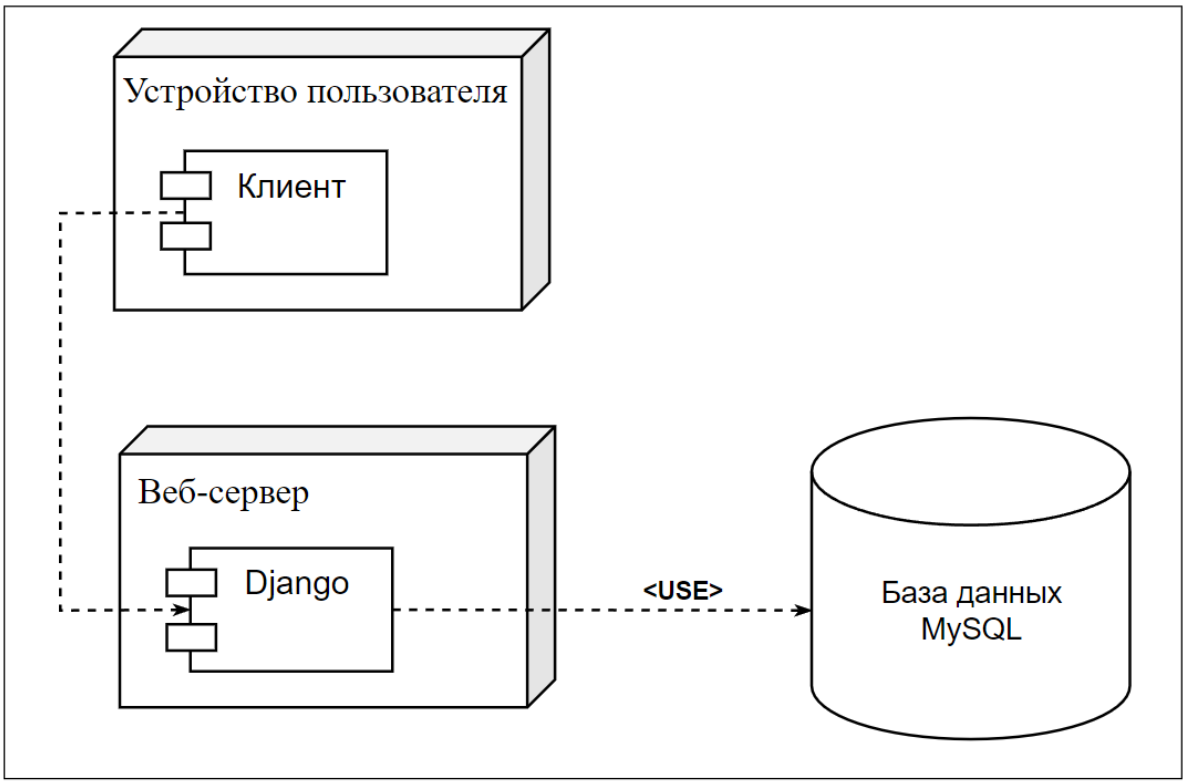

Рисунок 5 – Диаграмма компонентов

Ниже представлено описание каждого из компонентов.

1. Слой представления (клиентская часть). Этот слой отвечает за представление пользовательского интерфейса и обрабатывает пользовательское взаимодействие с системой. Этот уровень базируется на визуальном представлении веб-приложения, обычно при разработке пользовательского интерфейса (UI) основным выбором индустрии является HTML в сочетании с JavaScript или такими фреймворками, как React, Angular, Ember.

2. Слой бизнес-логики (веб-сервер). Этот слой отвечает за обработку и управление основной функциональности веб-приложения. Он получает запросы от слоя представления, выполняет необходимую обработку, взаимодействует с уровнем хранения данных и возвращает выходные данные на слой представления в пользовательский интерфейс, а также в базу данных.

Он реализуется с помощью серверных языков программирования, в нашем случае применяется Python совместно с фреймворком Django.

3. Слой хранения данных (база данных). Данный слой отвечает за хранение данных, которые используются при работе веб-приложения. Он может включать в себя базы данных, файловые системы или прочие механизмы хранения информации.

2.4. Диаграмма деятельности

Была разработана диаграмма деятельности, которая описывает процесс создания прогноза на сайте, являющийся одним из вариантов использования описанным в пункте 2.2. Диаграмма представлена на рисунке 6.

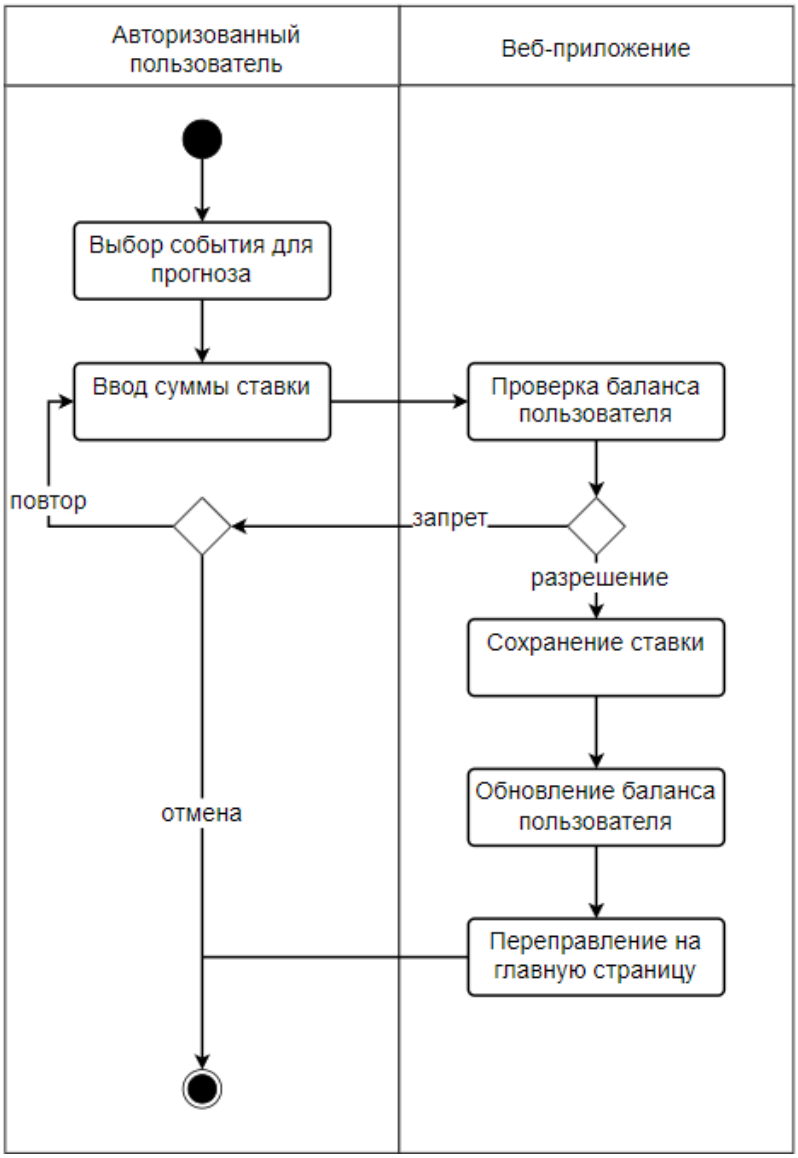

Рисунок 6 – Диаграмма деятельности для прецедента «сделать прогноз»

Авторизованный пользователь выбирает интересующее его событие и делает прогноз на его результат. Далее пользователь указывает сумму ставки на выбранный исход события. Система проверяет имеет ли пользователь указанную сумму на своем балансе. Если баланса недостаточно, то происходит переход обратно к вводу суммы ставки. Если пользователь отменяет создание прогноза, то процесс завершается.

Если пользователь имеет достаточный баланс, то ставка сохраняется, и система обновляет баланс пользователя, вычитая из него сумму ставки. Далее происходит перенаправление на главную страницу.

Вывод по второй главе

В этой главе были рассмотрены различные аспекты проектирования веб-сайта, включая определение функциональных и нефункциональных требований. Была разработана диаграмма вариантов использования, которая позволяет определить и визуализировать основные сценарии взаимодействия пользователя с системой, диаграмма компонентов системы, которая помогают выделить основные модули и компоненты системы, их взаимосвязи и зависимости, а также диаграмма деятельности. Разработка диаграмм происходила с использованием языка моделирования UML, что позволило создать четкое и наглядное представление о стуктуре и функциональности веб-приложения.

3. РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ

3.1. Создание Django проекта

Создание нового проекта Django включает в себя несколько этапов, которые представлены ниже.

1. Создание виртуального окружения.
2. Установка фреймворка Django в виртуальное окружение Python.
3. Создание нового проекта с помощью команды «`django-admin startproject название_проекта`».

После создания проекта будут автоматически сгенерированы следующие файлы [10]:

– `manage.py`: это утилита командной строки для различных задач управления Django;

– `settings.py`: файл, содержащий в себе настройки конфигурации проекта;

– `urls.py`: это файл, содержащий в себе конфигурации URL для проекта;

– `asgi.py` и `wsgi.py`: конфигурационные файлы для запуска проекта в качестве ASGI или WSGI приложения;

– `__init__.py`: пустой файл, который помечает каталог в качестве Python пакета.

ASGI и WSGI являются протоколами, которые позволяют Django взаимодействовать с веб-сервером и обрабатывать входящие запросы.

После создания проекта внутри него нужно создать приложение с помощью команды «`python manage.py startapp название`». Создадим приложение «betsite». Каждое приложение представляет определенную функциональность или модуль проекта. Внутри приложения Django автоматически генерирует следующие файлы:

– `models.py`: определяет модели базы данных для приложения;

– `views.py`: содержит в себе представления (функции или классы) для обработки HTTP-запросов;

– `urls.py`: конфигурация URL, специфичная для приложения;

– `admin.py`: конфигурационный файл для панели администратора Django;

– `apps.py`: используется для управления и конфигурирования приложений в проекте, позволяет настроить различные аспекты приложений, такие как их имена, поведение и дополнительные настройки;

– `migrations/`: каталог для миграции базы данных.

В итоге была получена следующая структура проекта, представленная на рисунке 7.

```
echoserver/
      manage.py
      echoserver/
            __init__.py
            asgi.py
            settings.py
            urls.py
            wsgi.py
      betsite/
            migrations/
            __init__.py
            admin.py
            apps.py
            models.py
            views.py
```

Рисунок 7 – Начальная структура проекта

3.2. Создание базы данных сайта

В Django архитектура базы данных управляется и организуется с помощью компонентов, известных как модели [11]. Модель представляет собой класс Python, который наследуется от класса Django.db.models.Model. Это наследование наделяет модель рядом функциональных возможностей, определяемых объектно-реляционным отображением (ORM) [12]. ORM упрощает трансляцию между объектами Python и записями базы данных, позволяя разработчикам манипулировать данными с помощью кода Python, не прибегая к прямым SQL-запросам. Модели служат для разграничения структуры таблиц базы данных, определяя их соответствующие поля и атрибуты, включая типы данных, ограничения и связи с другими моделями.

Система моделей Django предлагает широкую поддержку для определения сложных отношений между различными сущностями, таких как отношения один-к-одному, один-ко-многим и многие-ко-многим. Эта возможность позволяет разработчикам создавать сложные структуры данных и обеспечивать ссылочную целостность в базе данных.

Кроме того, Django обеспечивает встроенную поддержку миграции баз данных, позволяя легко управлять изменениями схемы базы данных с течением времени. Это обеспечивает согласованность и целостность структуры базы данных на разных этапах жизненного цикла приложения, от разработки до развертывания.

Для работы веб-приложения был создан ряд основных моделей.

1. `Role`: модель, описывающая роли пользователей.
2. `User`: модель представляет сущность пользователя системы и используется для хранения информации о нем и связанных прогнозов.
3. `Event`: представляет сущность события, на которое пользователь может делать ставки. Содержит информацию об участниках, коэффициентах и других аспектах события.
4. `Bet`: модель представляет информацию о ставке пользователя на определенное событие, связана с пользователем и событием через внешние ключи.
5. `ChatMessage`: модель представляет информацию о сообщениях в чате между пользователями. Каждое сообщение содержит текст сообщения, дату и время отправки, а также связь с пользователем, который отправил сообщение, через внешний ключ.

Была получена следующая структура базы данных, представленная на рисунке 8.

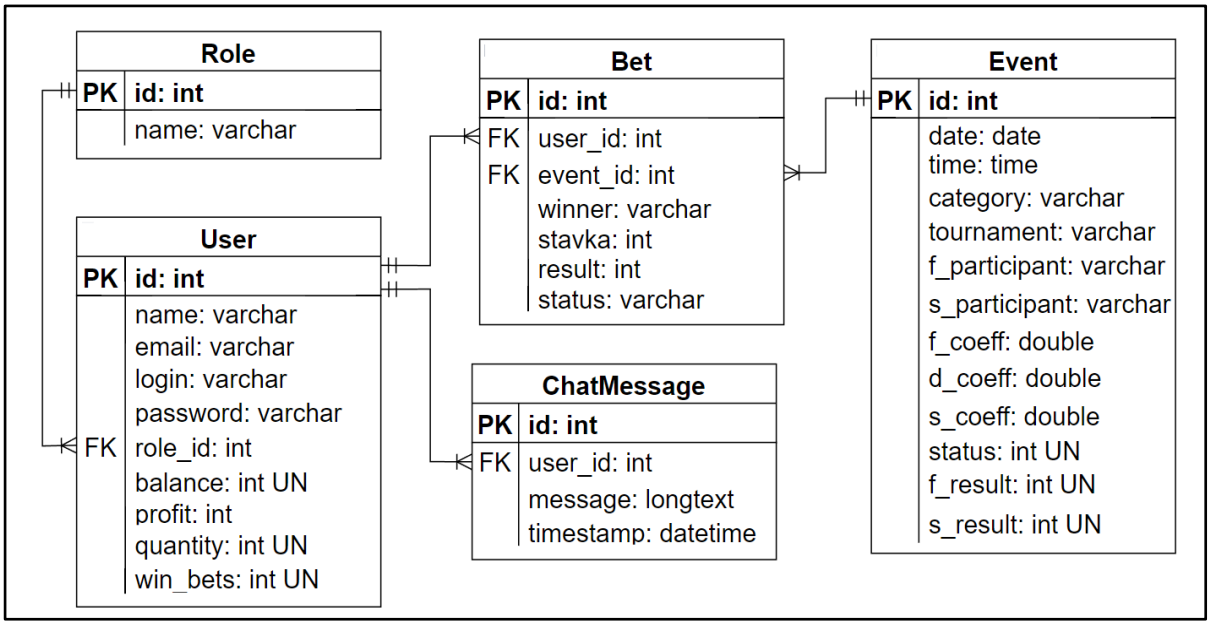

Рисунок 8 – Структура базы данных

Разберем структуру модели на примере модели Bet. Исходный код модели представлен в листинге 1.

Листинг 1 – Модель Bet

```
class Bet(models.Model):
    id = models.AutoField(primary_key=True)
    user = models.ForeignKey('user', on_delete=models.PROTECT)
    event = models.ForeignKey('event', on_delete=models.PROTECT)
    winner = models.CharField(max_length=30)
    stavka = models.IntegerField()
    result = models.IntegerField(blank=True, null=True)
    status = models.CharField(max_length=30, default='Не рассчитана')

    def __str__(self):
        return str(self.id)
```

Описание полей модели:

– `id: AutoField` первыичный ключ модели;

– `user: ForeignKey` поле, связанное с моделью User, которая описана выше;

– `event: ForeignKey` поле, связанное с моделью Event, которая описана выше;

– `winner: CharField` поле для хранения исхода выбранного события;

– `stavka: IntegerField` поле для хранения суммы ставки;
– `result: IntegerField` поле для хранения результата прогноза;
– `status: Charfield` поле для хранения статуса прогноза.

3.3. Разработка модуля авторизации и регистрации

Основная задача данного модуля – предоставление возможности пользователям регистрироваться и входить в систему. В файле forms.py были созданы формы `UserSignInForm` и `UserSignUpForm`, которые наследуются от встроенного в Django класса `ModelForm`.

Класс `UserSignInForm` определяет поля формы для ввода логина и пароля, представлен в листинге 2.

Листинг 2 – Класс `UserSignInForm`

```
class UserSignInForm(forms.ModelForm):
    login = forms.CharField(max_length=100)
    password = forms.PasswordInput()

    class Meta:
        model = User
        fields = ['login', 'password']
```

Класс `UserSignUpForm` определяет поля для регистрации нового пользователя. Реализована валидация данных с помошью JavaScript. Система проверяет наличие введенного имени, проверяет корректность указанного почтового ящика, проверяет уникальность логина и количество символов в пароле.

Этот модуль использует модель пользователя, представленную классом User, для управления хранением пользовательских данных. Когда пользователи заполняют форму регистрации, данные сохраняются в модели User. Впоследствии Django автономно генерирует новую запись в таблице пользователей, содержащую предоставленную информацию и сохраняет ее в базу данных.

3.4. Настройка маршрутизации и представления страниц

Маршрутизация URL в Django осуществляется с помощью модуля URLconf (конфигурация URL) [13], который определяется в каждом приложении Django. Этот модуль содержит набор шаблонов URL, сопоставленных с соответствующими функциями или классами представлений. Когда к приложению Django поступает запрос, диспетчер URL проверяет запрашиваемый URL на соответствие заданным шаблонам, чтобы определить, какое представление должно обрабатывать запрос. Шаблоны URL могут включать регулярные выражения, строковые шаблоны или именованные группы, что позволяет осуществлять гибкую и динамическую маршрутизацию на основе различных критериев.

Представления страниц (Views) – это функции или классы веб-приложения, которые определяют логику обработки запросов и получения выходных данных, представляющих собой содержимое веб-страницы. Представления получают данные от моделей (если это необходимо), обрабатывают их и возвращают шаблоны или другие формы ответов (например, JSON) для отображения пользователю [14].

Рассмотрим параметры маршрутизации основного файла проекта, его содержание представлено в листинге 3.

Листинг 3 – Содержание основного файла маршрутизации

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('betsite.urls')),
]
```

Маршруты проекта хранятся в списке `urlpatterns`. Каждый маршрут задается с помощью функции `path()`. В первом параметре указывается шаблон пути, а во втором может быть указан список маршрутов для конкретного приложения.

В приведенном листинге 3 определяются два маршрута.

1. Функция `path('admin/', admin.site.urls)` обрабатывает запросы, которые начинаются с admin/.

2. Функция `path('', include("betsite.urls"))` обрабатывает все остальные запросы, которые не начинаются с `admin/`. В нем используется функция `include()`, для включения маршрутов из приложения «betsite», о создании которого рассказывается в пункте 3.1. Таким образом, все запросы, направленные на данные маршрут, будут обрабатываться маршрутами, определенными в файле urls.py приложения «betsite», которые представлены в листинге 4.

Листинг 4 – Файл маршрутизации приложения «betsite»

```
urlpatterns = [
    path('', start),
    # Главная
    path('betting/', betting, name='betting'),
    # Создание события
    path('create/', create, name='create'),
    # Редактирование события
    path('edit/<int:id>/', edit, name='edit'),
    # Список лидеров
    path('leaders/', leaders, name='leaders'),
    # Регистрация
    path('signup/', signUp, name='signup'),
    # Авторизация
    path('signin/', signIn, name='signin'),
    path('logout/', logout, name='logout'),
    # Профиль пользователя
    path('profile/', profile, name='profile'),
    # Создание прогноза
    path('makedep/<int:id_event>/', makedep, name='makedep'),
    # История прогнозов
    path('history/', history, name='history'),
    # Проверка уникальности логина
    path('check/', checkLogin, name='check'),
]
```

В файле маршрутизации приложения «betsite» определены пути для основных URL-адресов веб-сайта и соответствующие представления, обрабатывающие запросы по этим путям. В нашем приложении в качестве представлений используются функции, которые принимают на входе запрос – объект класса HttpRequest.

В качестве примера обработчика представления страниц рассмотрим функцию `makedep()`. Код функции `makedep()` представлен в листинге 5.

Листинг 5 – Функция `makedep()`

```
def makedep(request, id_event):
    # Проверяем авторизован ли пользователь
    role = request.session.get('role', 'none')
    if role != 'user':
        return redirect('/betting')

    user_id = request.session.get('user_id')  # Получаем ID пользователя из
сессии
    if Event.objects.filter(id=id_event, status=0).exists():
        event = Event.objects.get(id=id_event)
        user = User.objects.get(id=user_id)

        if request.method == 'POST':
            form = MakeBetForm(request.POST)
            if form.is_valid():
                stavka = form.cleaned_data['stavka']
                winner = request.GET.get('winner', '')
               # Проверяем, что ставка пользователя не превышает его баланс
                if stavka <= user.balance:
                    user.balance -= stavka
                    user.quantity +=1
                    user.save()
                    bet = form.save(commit=False)
                    bet.user = user
                    bet.event =event
                    bet.winner = winner
                    bet.save()
                    return redirect("/")
                else:
                    form.add_error('stavka', f'У вас недостаточно средств.
Ваш баланс: {user.balance}')
        else:
            form = MakeBetForm()

    return render(request, "bet.html", {'form': form})
```

Функция `makedep()` представляет собой представление веб-страницы для создания прогноза в системе. С помощью метода `get` получаем данные о пользователя из сессии и проверяем авторизованный пользователь ли он. Если данные не проходят проверку, то перенаправляем на главную страницу.

Далее обрабатывается POST-запрос [15], который отправляется после заполнения и отправки формы. Создается экземпляр формы `MakeBetForm`, используя данные из POST-запроса. Если данные формы проходят проверку, то прогноз успешно создается и сохраняется в базе данных.

Когда спортивное событие завершилось, администратор вносит информацию о результате встречи и присваивает ей статус «Завершено». После чего система автоматически вызывает функцию `calculation()`. Эта функция рассчитывает результаты прогнозов для каждого пользователя, кто сделал ставку на данное событие и вносит изменения в базу данных. Код функции представлен в листинге 6.

Листинг 6 – Функция `calculation()`

```
def calculation():
    if Event.objects.filter(id=id_event).exists():
        event = Event.objects.get(id=id_event)
        if event.f_result > event.s_result:
            event.d_coeff = 0
            event.s_coeff = 0
        elif event.f_result == event.s_result:
            event.f_coeff = 0
            event.s_coeff = 0
        else:
            event.f_coeff = 0
            event.d_coeff = 0
        event.save()
        bets = Bet.objects.filter(event=event)
        for bet in bets:
            user = bet.user
            if bet.winner == 'f':
                bet.result = round(bet.stavka * event.f_coeff)
            elif bet.winner == 'd':
                bet.result = round(bet.stavka * event.d_coeff)
            else:
                bet.result = round(bet.stavka * event.s_coeff)
            if bet.result > 0:
                user.balance += bet.result
                user.profit += (bet.result-bet.stavka)
                user.win_bets +=1
                bet.status = 'Выигрыш'
            else:
                bet.status = 'Проигрыш'
                user.profit -= bet.stavka
            user.save()
            bet.save()
```

Рассмотрим также функцию `signin()`. Она обрабатывает входящие запросы, содержащие данные, введенные пользователем в форму, представленной классом UserSignInForm, код которого приведен в пункте 3.3. Функция `signin()`, отвечающая за процесс авторизации пользователей в системе, приведена в листинге 7.

Листинг 7 – Функция `signin()`

```
def signIn(request):
    role = request.session.get('role', 'none')
    if role != 'none':
        return redirect('/betting')

    if request.method == 'POST':
        form = UserSignInForm(request.POST)
        if form.is_valid():
            if    User.objects.filter(login=form.cleaned_data['login']).ex-
ists():
                user = User.objects.get(login=form.cleaned_data['login'])
                if (user.password == form.cleaned_data['password']):
                    request.session['name'] = user.name
                    request.session['role'] = user.role.name
                    request.session['user_id'] = user.id
                    return redirect('/')
                else:
                    result = 'Неправильный пароль'
            else:
                result = 'Несуществующий логин'
            form = UserSignInForm()

            return render(request, 'sign_in.html',
                          {'form': form, 'result': result})
```

Функция обрабатывает POST-запрос, в котором содержится логин и пароль пользователя. Система проверяет существует ли пользователь с введенным логином в системе, если пользователь существует, то проверяется правильность пароля. Если данные были введены корректно, то пользователя переадресует на главную страницу, в противном случае ему будет указано, какую ошибку он совершил.

3.5. Реализация интерфейса приложения

HTML язык разметки служит основой веб-страниц, определяя их структуру и содержание [16]. Он состоит из ряда элементов или тегов, которые содержат различные типы контента. Блочная верстка в HTML позволяет организовать содержимое веб-страницы в виде блоков, каждый из которых может содержать текст, изображения или другие элементы. Такой подход создания страниц с иерархическим размещением элементов способствует удобству чтения и восприятия информации.

Использование адаптивного дизайна на веб-сайтах необходимо для обеспечения оптимального пользовательского опыта на различных устройствах. В современном мире доступ к интернету осуществляется через различные устройства: от настольных компьютеров и ноутбуков до планшетов и смартфонов. Адаптивный дизайн позволяет сайту автоматически адаптироваться к различным размерам экранов, обеспечивая удобство использования для пользователей независимо от того, на чем они просматривают сайт. Одним из способов создания адаптивного дизайна сайта может служить использование фреймворка Bootstrap [17]. Bootstrap – это открытый и бесплатный HTML, CSS и JS фреймворк, который используется веб-разработчиками для быстрого создания адаптивных дизайнов сайтов. Он предоставляет готовые компоненты и стили, которые можно легко интегрировать в проект, что упрощает и ускоряет процесс разработки.

В шаблонах Django доступны операторы для работы с данными и выполнения условных операций. Операторы используются внутри конструкции {% if %} для определения условий и принятия решений о выполнении определенных блоков кода. Они позволяют создавать более гибкие и динамические шаблоны, которые могут адаптироваться к различным условиям и значениям данных.

В листинге 8 представлен шаблон навигационного меню, который написан с использованием фреймворка Bootstrap.

Листинг 8 – Код шаблона навигационного меню

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand" href="#">Прогнозирование спортивных встреч</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false"
aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav">
            {% if role == 'none' %}
            <li class="nav-item">
                <a class="nav-link" href="{% url 'signin' %}">Авторизация</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="{% url 'signup' %}">Регистрация</a>
            </li>
```

```
            {% else %}
            <li class="nav-item">
                <a class="nav-link" href="{% url 'profile' %}">Профиль</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="{% url 'history' %}">Мои ставки</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="{% url 'leaders' %}">Список лидеров</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="{% url 'logout' %}">Выйти</a>
            </li>
            {% endif %}
            {% if role == 'admin' %}
            <li class="nav-item">
                <a class="nav-link" href="{% url 'create' %}">Добавить событие</a>
            </li>
            {% endif %}
        </ul>
    </div>
</nav>
```

На рисунке 9 показан вид главной страницы для авторизованного пользователя.

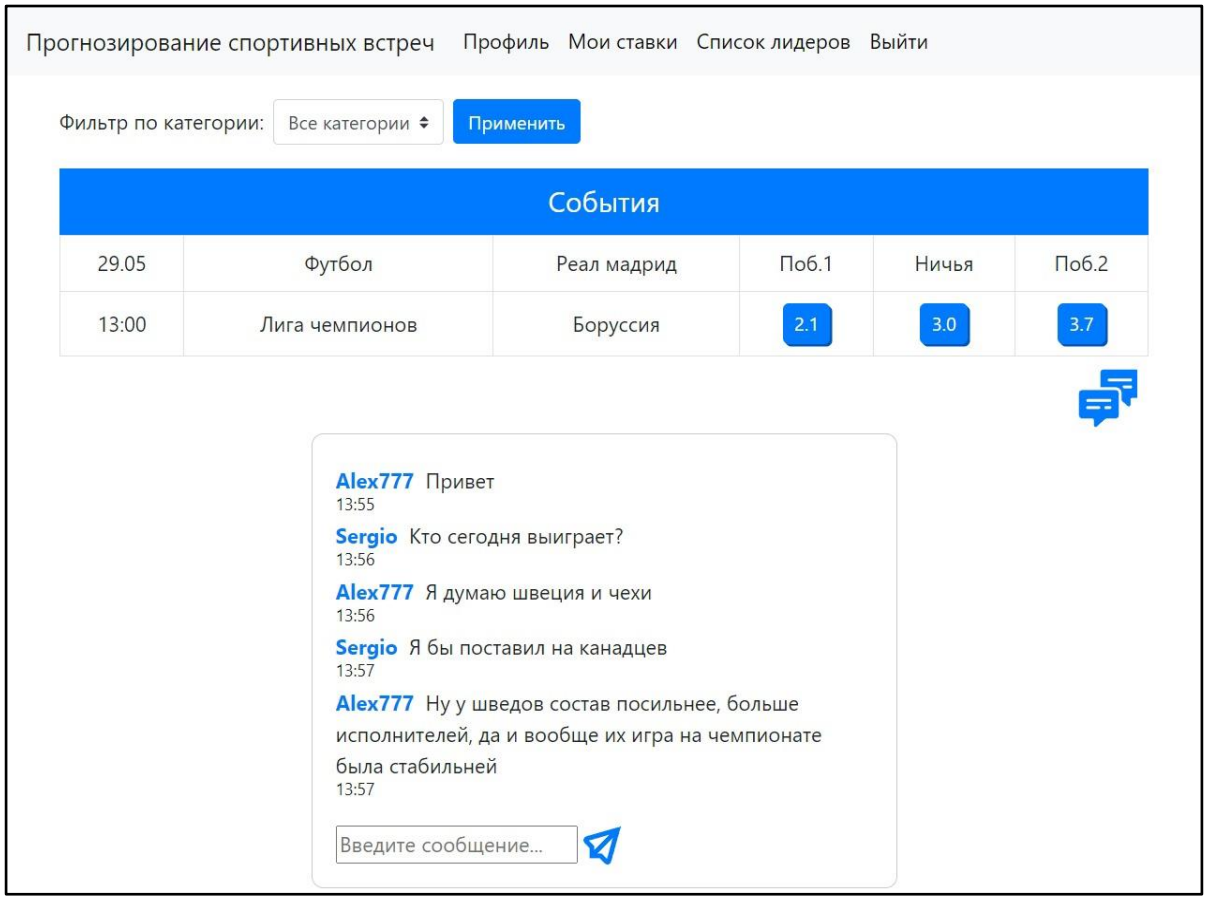

Рисунок 9 – Вид главной страницы

Веб-интерфейс главной страницы сайта имеет следующие элементы: навигационное меню, таблицу событий, кнопка открытия чата, окно чата (если он был открыт).

Для удобства использования чата были разработаны скрипты на языке JavaScript. Пример скрипта, который обрабатывает отправку сообщения, приведен в листинге 9.

Листинг 9 – Код скрипта обработки отправки сообщения

```
$('#chat-form').on('submit', function(event) {
    event.preventDefault();
    var form = $(this);
    $.ajax({
        type: form.attr('method'),
        url: form.attr('action'),
        data: form.serialize(),
        dataType: 'json',
        success: function(data) {
            // Обновление сообщений чата
            $('.chat-messages').html(data.chat_messages_html);
            // Очистка поля ввода сообщения
            $('#message-input').val('');
            scrollChatToBottom();
        }
    });
});
```

Технология AJAX позволяет отправлять и получать данные асинхронно, что позволяет динамически обновлять содержимое веб-страницы без перезагрузки всей страницы. В данном случае, AJAX используется для обновления сообщений чата без перезагрузки страницы.

3.6. Функциональное тестирование

Функциональное тестирование выполняется с целью проверки соответствия разработанного программного обеспечения изначально заявленной функциональности системы и ее первоначальным функциональным требованиям. Функциональное тестирование также помогает выявить любые потенциальные дефекты или несоответствия в работе программного продукта. Результаты функционального тестирования системы представлены в таблице 1.

Таблица 1 – Функциональное тестирование

№	Действие	Результат	Тест пройден?
1	Регистрация нового пользователя с корректными данными	Пользователь успешно зарегистрирован	Да
2	Регистрация пользователя с существующим в базе данных логином.	Вывод сообщения об ошибке	Да
3	Авторизация зарегистрированного пользователя с правильными учетными данными.	Вход в аккаунт.	Да
4	Попытка авторизации с неверным паролем.	Вывод сообщения об ошибке.	Да
5	Попытка авторизации с несуществующем логином	Сообщение об ошибке.	Да
6	Создание прогноза	Прогноз создан и отображается в «моих ставках»	Да
7	Изменение данных пользователя	Данные успешно изменены	Да
8	Просмотр списка лидеров	Список лидеров отображается	Да
9	Администратор добавляет событие	Событие добавлено и отображается для пользователей	Да
10	Администратор редактирует данные события	Изменения были успешно применены	Да
11	Нажатие кнопки открытия чата	Окно чата открылось	Да
12	Отправка сообщения	Сообщение было отправлено и отображается для других пользователей	Да

Вывод по третьей главе

В третьей главе описаны ключевые этапы создания приложения. Это включает создание базы данных и моделей для хранения и управления данными, разработку модуля авторизации и регистрации. Далее была реализована логика представлений и шаблонов страниц, а также созданы формы для пользовательского ввода. Система прошла функциональное тестирование, результаты которого подтвердили ее работоспособность.

ЗАКЛЮЧЕНИЕ

В рамках данной работы было разработано веб-приложение по прогнозу итогов спортивных встреч. Создание данного веб-приложения осуществлялось на базе фреймворка Django и языка программирования Python с использованием HTML-разметки, а также применением CSS-стилей. Разработанная система предоставляет пользователям удобно совершать прогнозы на итоги различных встреч.

В ходе выполнения работы были успешно решены все поставленные задачи, учитывая как функциональные, так и нефункциональные требования:

1) произведен анализ предметной области;
2) произведен обзор существующих решений;
3) произведено проектирование веб-приложения;
4) произведена реализация и тестирование веб-приложения.

В результате была получена система, которая обеспечивает простоту создания прогнозов и возможность отслеживания своих успехов среди всех пользователей.

Система соответствует заявленному функционалу и обладает потенциалом для дальнейшего его расширения.

ЛИТЕРАТУРА

1. Букмекерская контора «Fonbet». [Электронный ресурс] URL: https://www.fon.bet/ (дата обращения: 10.03.2024 г.).

2. Букмекерская контора «Winline». [Электронный ресурс] URL: https://winline.ru/ (дата обращения: 10.03.2024 г.).

3. Букмекерская компания ПАРИ. [Электронный ресурс] URL: https://www.pari.ru/ (дата обращения: 10.03.2024 г.).

4. Что такое Python? [Электронный ресурс] URL: https://aws.amazon.com/ru/what-is/python/ (дата обращения: 20.03.2024 г.).

5. Что такое JavaScript (JS)? [Электронный ресурс] URL: https://aws.amazon.com/ru/what-is/javascript/ (дата обращения: 22.03.2024 г.).

6. AJAX - Глоссарий MDN Web Docs. [Электронный ресурс] URL: https://developer.mozilla.org/ru/docs/Glossary/AJAX (дата обращения: 22.03.2024 г.).

7. MySQL – система управления базами данных. [Электронный ресурс] URL: https://web-creator.ru/articles/mysql (дата обращения: 23.03.2024 г.).

8. Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя. – 2-е изд. ДМК Пресс, 2006. – С. 47–49.

9. Волков А. С., Волкова К. А. Обзор архитектурных компонентов современного веб-приложения. //Аллея науки, 2019. – Т. 3. – №. 1. – С. 958–961.

10. Руководство по Django часть 2: создание скелета. [Электронный ресурс] URL: https://developer.mozilla.org/ru/docs/Learn/Serverside/Django/skeleton_website (дата обращения: 16.04.2024 г.).

11. Django. Создание моделей. [Электронный ресурс] URL: https://metanit.com/python/django/5.1.php (дата обращения: 17.04.2024 г.).

12. Django ORM и модели. [Электронный ресурс] URL: https://mob25.com/django-orm-i-modeli/ (дата обращения: 17.04.2024 г.).

13. URL dispatcher | Django documentation. [Электронный ресурс] URL: https://docs.djangoproject.com/en/5.0/topics/http/urls/ (дата обращения: 27.04.2024 г.).

14. Представления (Views) | Python: Разработка на фреймворке Django. [Электронный ресурс] URL: https://ru.hexlet.io/courses/pythondjango-basics/lessons/views/theory_unit (дата обращения: 27.04.2024 г.).

15. Request and response objects. [Электронный ресурс] URL: https://docs.djangoproject.com/en/5.0/ref/request-response/ (дата обращения: 27.04.2024 г.).

16. Основы HTML - Изучение веб-разработки. [Электронный ресурс] URL: https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/HTML_basics (дата обращения: 03.05.2024 г.).

17. Introduction Bootstrap v5.0. [Электронный ресурс] URL: https://getbootstrap.com/docs/5.0/getting-started/introduction/ (дата обращения: 03.05.2024 г.).