

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Программное обеспечение средств вычислительной
техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

ВЕБ-ПРИЛОЖЕНИЕ С БАЗОЙ ДАННЫХ ДЛЯ
УЧАСТНИКОВ ИГРОВЫХ ТУРНИРОВ

Студент Б.С. 20.05.2025 Большаков Е. К.
(подпись, дата)

Руководитель Н.В. 20.05.2025 Слепцов Н. В.
(подпись, дата)

Нормоконтролёр А.С. 20.05.2025 А.С. Бычков
(подпись, дата)

Работа допущена к защите (протокол зас. кафедры от 28.05.2025 г.

№ 16)

Заведующий кафедрой

(подпись)

М.А. Митрохин

Работа защищена с отметкой удовлетворительно (протокол заседания ГЭК от

24 июля 2025 г. № 45)

Секретарь ГЭК

(подпись)

Е.И. Гурин

Пенза 2025

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

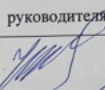
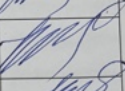
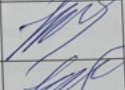
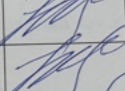
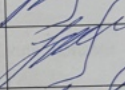
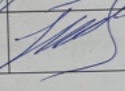
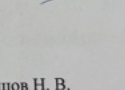
«Утверждаю»
Заведующий кафедрой
М.А. Митрохин

«26» 05 2024 г.

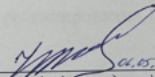
**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Большаков Егор Константинович гр.21ВВП2 факультета вычислительной техники.
2. Тема работы: «Веб-приложение с базой данных для участников игровых турниров»
Тема утверждена приказом ПГУ №601/о от "06" мая 2025г.
3. Руководитель бакалаврской работы к.т.н., доцент кафедры ВТ ПГУ Слепцов Н. В.
4. Задание на работу (назначение разработки, исходные данные и т.п.)
Разработать веб-приложение для организации и управления игровыми турнирами, обеспечивающее: регистрацию участников на турниры через онлайн-интерфейс, автоматическое формирование турнирных сеток, управление турнирами, участниками и матчами через административную панель, контроль статусов турниров
5. Перечень подлежащих разработке вопросов
Требуется произвести анализ предметных областей: изучить потребности организаторов турниров и их участников, проектирование базы данных для хранения информации об участниках и мероприятиях, оптимизировать маршруты Flask для минимизации нагрузки на сервер. Подготовить пользовательские инструкции для каждой роли (участник, организатор, администратор).

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	32	6.05.2025 - 28.05.2025	
2	Анализ требований	32	7.05.2025 - 30.05.2025	
3	Создание макета графического интерфейса	30	9.04.2025 - 3.05.2025	
4	Выбор технологий, применяемых при проектировании и разработке	40	11.05.2025 - 13.05.2025	
5	Реализация графического интерфейса	66	14.05.2025 - 17.05.2025	
6	Разработка основной части программы	70	16.05.2025 - 20.05.2025	
7	Оформление пояснительной записки	54	21.05.2025 - 23.05.2025	

Дата выдачи задания "06" мая 2025 г.

Руководитель бакалаврской работы  04.05.2025 Слепцов Н. В.
(подпись, дата)

Задание к исполнению принял студент  06.05.2025 Большаков Е. К.
(подпись, дата)

Работу к защите допустить
Декан факультета  24.6.2025 Фионова Л.Р.
(подпись, дата)

Реферат

Бакалаврская работа содержит 106 листов, 29 рисунков, 8 источников и 3 приложения.

Цель работы – разработка полнофункционального web-приложения «Платформа для проведения турниров», предоставляющего пользователям возможность участвовать в турнирах, отслеживать статистику, редактировать свой профиль, а администраторам – управлять участниками, модерировать заявки и контролировать процесс проведения турниров. Основная идея проекта – создать удобный онлайн-инструмент для организации соревнований, подходящий как для любительских сообществ, так и для более формальных турниров.

Объектами разработки являются:

- web-приложение с серверной логикой на Flask и HTML-шаблонами на базе Jinja2;
- база данных SQLite, содержащая сведения об участниках, турнирах, заявках и пользовательских профилях;
- система регистрации, входа и редактирования профиля с возможностью загрузки аватара;
- модуль управления турнирами (создание, просмотр, участие и отображение участников);
- панель администратора для модерации заявок, управления пользователями и редактирования информации о турнирах;
- шаблоны страниц для отображения статистики, списка турниров, профилей пользователей и административного интерфейса.

					ПГУ 1.090301.25.001 ПЗ		
Изм.	Лист	№ докум.	Подпис.	Дата	Web-приложение с базой данных для участников игровых турниров. Пояснительная записка		
Разраб.	Большаков Е.К.						
Пров.	Слепцов Н.В.						
Н. Контр.	Бычков А.С.						
Утв.	Митрохин М.А.				Литера	Лист	Листов
						2	107
					ФВТ, каф. ВТ гр. 21ВВП1		

В результате проделанной работы было разработано web-приложение, охватывающее как пользовательскую, так и административную часть.

Реализованы:

- регистрация и вход с хранением данных в базе и возможностью загрузки аватара;
- участие в турнирах, подача заявок, просмотр информации об участниках;
- отображение общей и индивидуальной статистики турниров;
- административная панель с возможностью редактирования данных пользователей и одобрения/отклонения заявок;
- шаблонный интерфейс, адаптированный для различных экранов и обеспечивающий понятную навигацию по сайту.

Проект демонстрирует, как можно создать гибкую и удобную платформу для управления турнирами, объединяющую игроков и организаторов. Система легко расширяется и может быть адаптирована под другие виды мероприятий или соревнований. Полученный результат пригоден как для учебных целей, так и для практического использования в организации событий спортивного или интеллектуального характера.

					ПГУ 1.090301.25.001 ПЗ	Лист
						3
Изм.	Лист	№ докум.	Подпись	Дата		

Содержание

Введение.....	7
1 Web-приложение с базой данных для участников игровых турниров	9
1.1 Анализ предметной области	9
1.1.1 Актуальность разработки.....	9
1.1.2 Анализ существующих аналогов	10
1.2 Постановка задачи	21
1.3 Выбор технологий и средств разработки приложения	24
1.3.1 Выбор технологий для клиентской части	24
1.3.2 Выбор технологий для серверной части	25
1.3.3 Выбор системы управления базами данных	25
1.3.4 Выбор формата обмена данными.....	26
1.4 Архитектура приложения.....	27
1.4.1 Архитектурный паттерн MVC.....	27
1.4.2 Клиент-серверное взаимодействие и REST API.....	27
1.4.3 Основные серверные классы и модели.....	27
1.4.4 Управление доступом по ролям	28
1.4.5 Масштабируемость и расширяемость архитектуры	28
1.5 Разработка приложения.....	30
1.5.1 Разработка клиентской части	30
1.5.2 Разработка серверной части.....	33
1.6 Описание приложения.....	36
1.6.1 Руководство пользователя	37
2 Охрана труда.....	41
2.1 Эргономические требования.....	41
2.2 Организация рабочего места разработчика.....	41
2.3 Правовое соответствие	42
2.4 Требования к тестированию и качеству	43
Заключение	45
Список использованных источников	47

Приложение А-UML-диаграммы	48
Приложение Б-Листинг программы.....	53
Приложение С-Презентация	100

					ПГУ 1.090301.25.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		5

Перечень сокращений и обозначений

Админ-панель – раздел приложения, доступный только администраторам, предназначенный для управления пользователями, турнирами и заявками

API (Application Programming Interface) – интерфейс программирования приложений, предоставляющий способы взаимодействия между различными компонентами программного обеспечения

CORS (Cross-Origin Resource Sharing) – механизм, который позволяет веб-приложениям запрашивать ресурсы с другого домена

ORM (Object-Relational Mapping) – технология, позволяющая работать с базой данных как с объектами программного кода, избавляя от необходимости писать SQL-запросы вручную

JWT (JSON Web Token) – формат токена, применяемый для безопасной передачи информации о пользователе и проверке подлинности доступа

JSON (JavaScript Object Notation) – формат представления структурированных данных в виде текста, широко применяемый при передаче данных между клиентом и сервером

Jinja2 – шаблонизатор, используемый совместно с Flask для генерации HTML-страниц с использованием данных с сервера

Flask – легковесный фреймворк на языке Python, предназначенный для разработки веб-приложений с минимальной структурой, но широкими возможностями расширения

SQLite – встроенная реляционная база данных, хранящая данные в одном файле и не требующая отдельного сервера для работы

Web-приложение – клиент-серверное приложение, где пользователь взаимодействует с сервером через веб-интерфейс (браузер)

Введение

Сегодня, в эпоху стремительного развития цифровых технологий, всё больше процессов переходит в онлайн-формат. Люди уже привыкли к тому, что можно дистанционно учиться, покупать товары, общаться, смотреть фильмы и даже участвовать в спортивных и интеллектуальных соревнованиях. Особенно популярными становятся онлайн-платформы, которые позволяют организовывать мероприятия без необходимости физического присутствия. Это удобно, быстро и доступно для широкой аудитории.

Организация турниров – будь то соревнования по шахматам, киберспорту или интеллектуальным играм – раньше требовала большого количества ресурсов: помещения, оборудования, регистрации участников вручную. Сегодня эти процессы можно автоматизировать и упростить благодаря web-приложениям. Однако, как показывает практика, многие из существующих решений либо перегружены функционалом, либо неудобны в использовании: интерфейсы запутаны, нет нормального отображения статистики, система регистрации сложная, а некоторые и вовсе не предоставляют административного инструментария.

Проектируемое web-приложение «Платформа для проведения турниров» направлено на решение этих проблем. Основная идея – создать удобную, понятную и лёгкую в использовании систему, в которой можно было бы быстро создать турнир, зарегистрироваться как участник, просматривать информацию об игроках и результатах, а также управлять всем процессом через админ-панель.

Целью данной выпускной квалификационной работы является:

- исследование предметной области и анализ аналогичных систем для проведения турниров;
- разработка web-приложения для регистрации участников, создания турниров и отображения результатов;
- реализация системы авторизации с разделением прав доступа между обычными пользователями и администраторами;

- создание простой и логичной структуры базы данных, включающей информацию о турнирах, участниках, заявках и профилях;
- построение интерфейса, который будет удобен как для участников, так и для администраторов;
- обеспечение адаптивности сайта, чтобы им было комфортно пользоваться с разных устройств – от компьютера до телефона.

В ходе работы будет проведён подробный анализ предметной области, сформулированы требования к системе, выбраны подходящие технологии, спроектирована архитектура и реализована функциональная версия приложения. Также будет проведено тестирование, чтобы убедиться, что платформа работает корректно и соответствует заявленным задачам.

					ПГУ 1.090301.25.001 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		

1 Web-приложение с базой данных для участников игровых турниров

1.1 Анализ предметной области

1.1.1 Актуальность разработки

Соревнования, турниры и состязания – это неотъемлемая часть человеческой культуры, будь то спорт, киберспорт, интеллектуальные игры или другие форматы. С развитием интернета и цифровых технологий форма их проведения значительно изменилась. Всё больше турниров переходят в онлайн-среду, поскольку это позволяет расширить охват участников, упростить организацию и снизить затраты на проведение мероприятий.

Однако на практике далеко не все существующие платформы удовлетворяют требованиям пользователей. Одни из них слишком сложны в освоении, другие – перегружены функционалом или, наоборот, обладают крайне ограниченными возможностями. Кроме того, нередко отсутствуют простые инструменты управления участниками, подачи заявок или отображения статистики. Это особенно критично для небольших сообществ, кружков или начинающих организаторов, которые не имеют возможности использовать громоздкие коммерческие решения.

Web-приложение для проведения турниров, разрабатываемое в рамках данной работы, направлено на устранение этих недостатков. В отличие от многих универсальных систем, оно ориентировано на простоту использования, наглядность и удобство для всех участников процесса – как обычных пользователей, так и администраторов. Для пользователей это – возможность быстро зарегистрироваться, подать заявку на участие, отслеживать состав участников и результаты. Для администратора – простой инструмент управления турнирами, участниками и заявками без необходимости погружения в технические детали.

Актуальность подобного проекта особенно велика в условиях роста популярности онлайн-мероприятий и дистанционного участия. Он может быть востребован как в студенческих и школьных клубах, так и среди независимых организаторов киберспортивных или тематических интеллектуальных соревнований. Внедрение такого решения упрощает процесс организации турниров, делает его доступным для широкой аудитории и способствует развитию онлайн-сообществ.

Таким образом, разработка простой и надёжной платформы для проведения турниров отвечает современным требованиям цифровой среды и открывает новые возможности для организации соревнований в самых разных сферах.

1.1.2 Анализ существующих аналогов

Среди платформ, предназначенных для проведения онлайн-турниров, можно выделить несколько популярных решений:

- Toornament – <https://www.toornament.com>[1];
- Battlefy – <https://battlefy.com>[2];
- Cyberscore – <https://cyberscore.live>[3].

Рассмотрим сайт Toornament.

Общий вид данного сервиса представлен на рисунке 1.1.

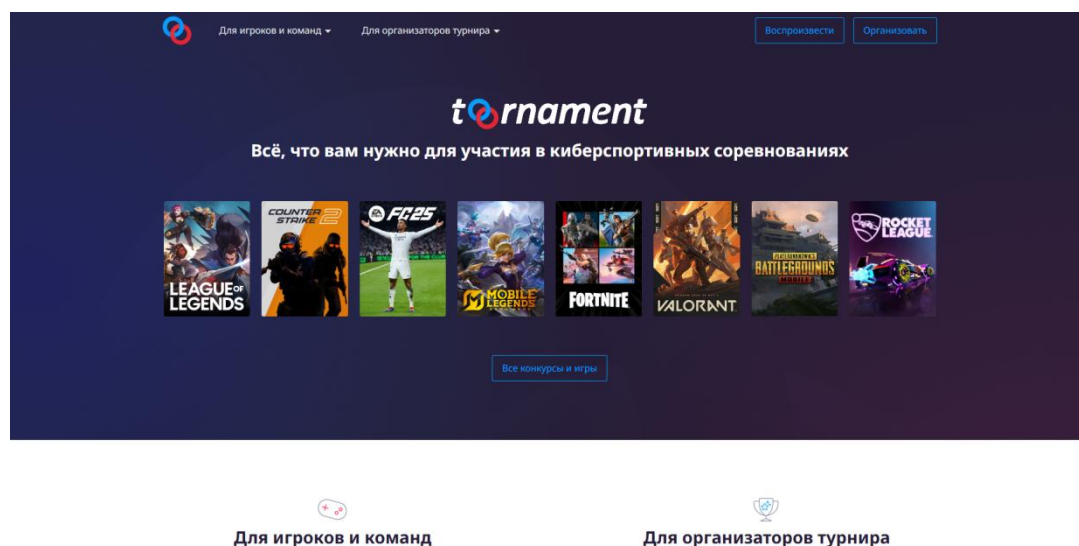


Рисунок 1.1– Главная страница Battlefy

Для первичного ознакомления с турнирами регистрация не требуется. Пользователь может просматривать список доступных соревнований, переходить на страницы отдельных турниров и изучать информацию об их дате, месте проведения и участниках.

Однако для подачи заявки на участие, доступа к личному кабинету и возможности редактировать профиль необходимо авторизоваться. При попытке совершить любое действие, требующее регистрации, пользователю будет предложено войти в систему или создать аккаунт. Возможности просмотра турниров без регистрации показаны на рисунке 1.2, форма регистрации – на рисунке 1.3.

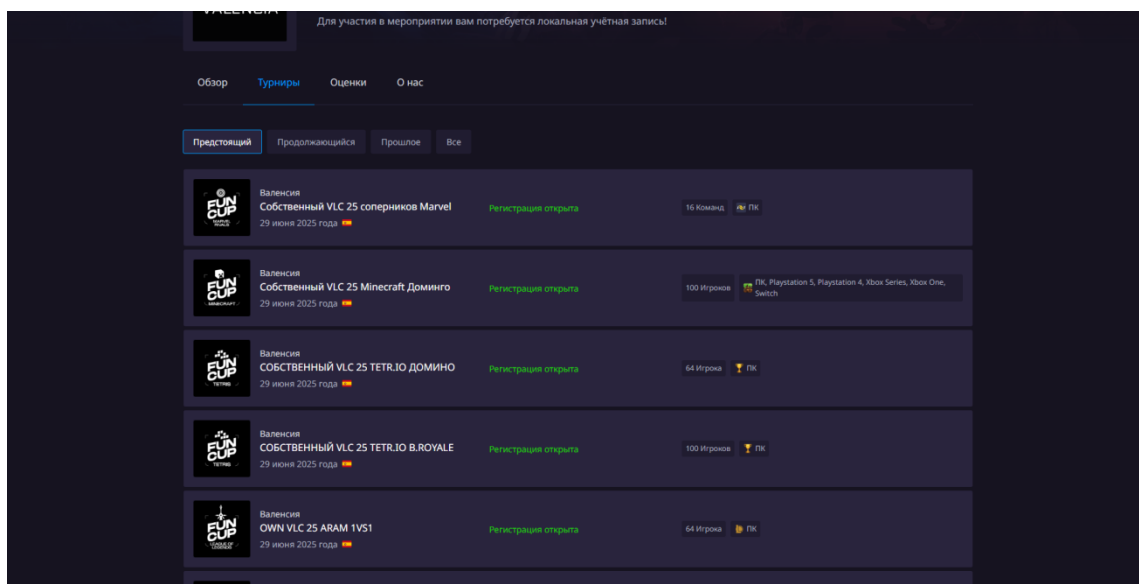


Рисунок 1.2 – Просмотр турнира без регистрации

Рисунок 1.3 –Форма регистрации

После регистрации пользователю становятся доступны все функции платформы. В частности, он может подавать заявки на участие в турнирах, просматривать статус своих заявок, а также видеть личную статистику – количество побед, поражений и общее участие.

Вся эта информация находится в разделе «Мой профиль», где также можно изменить данные аккаунта и загрузить аватар.

Интерфейс личного кабинета пользователя показан на рисунке 1.4.

Рисунок 1.4 – Интерфейс личного кабинета

					ПГУ 1.090301.25.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		12

Для подачи заявки на участие в турнире требуется авторизация. Только зарегистрированный пользователь может подтвердить своё участие в конкретном событии.

После нажатия кнопки подачи заявки система проверяет, вошёл ли пользователь в аккаунт. Если да – заявка сохраняется в базе, и пользователь получает уведомление об успешной отправке. Если нет – происходит перенаправление на страницу входа.

Процесс подачи заявки на турнир и проверка регистрации 1.5 и 1.6.

Регистрация на турнир Открыто до 27 июня 2025 года, 21:30 [Правила](#)

Дополнительная информация об игроке

País
Россия

Город (Частный)
пенза

Edad (необязательно) (Личное)
egorbolnoy1337@gmail.com

Идентификатор Discord (Приватный)
432432

КОД ДЛЯ ВХОДА В LAN (Частный)
fd

Вы играете в Java или Bedrock? (Личное)
Скальная порода

Ник из Minecraft (Частный)
прошувывживи

[Назад](#) [Далее](#)

Рисунок 1.5 – Процесс подачи заявки

Рисунок 1.6 – Проверка регистрации

После того , как пользователь оставил заявку на турнир , он может следить в своем личном профиле за статусом этой заявки. Статус всех регистраций в профиле пользователя показан рисунке1.7.

Рисунок 1.7 – Регистрации в профиле

Одна из удобных функций на платформе Toornament – это выбор игры при создании турнира. Организатору не нужно настраивать всё вручную: достаточно выбрать нужную игру из списка, и сайт автоматически подстроит формат турнира под особенности этой дисциплины.

Список игр довольно большой – от популярных вроде CS:GO, Dota 2 и FIFA до менее известных или нишевых проектов. Это упрощает настройку и делает интерфейс более понятным. Также выбранная игра влияет на оформление страницы турнира и доступные поля – например, можно сразу указать количество игроков в команде или игровые режимы.

Как выглядит процесс выбора игры при создании турнира, показано на рисунке 1.8.

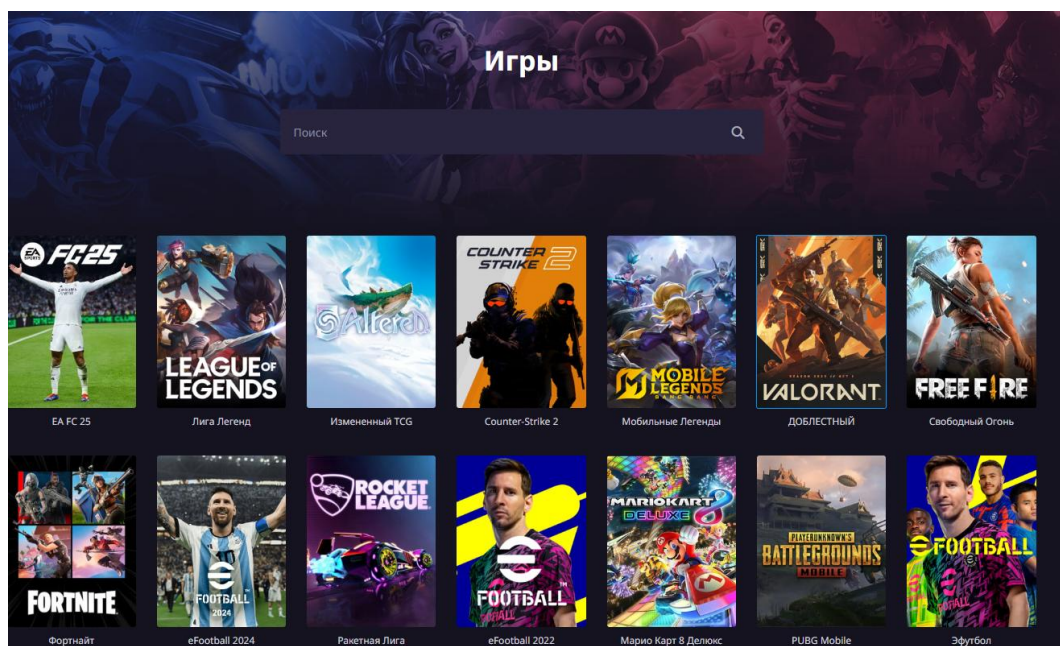


Рисунок 1.8 – Процесс выбора игры

На платформе Toornament есть отдельный раздел под названием Circuits. Это своего рода витрина, где организаторы могут объединять несколько турниров в одну общую серию. Например, можно создать лигу или чемпионат, который длится несколько месяцев и состоит из этапов. Каждый этап – это отдельный турнир, но все они связаны между собой общим зачётом.

Внутри Circuits удобно отслеживать рейтинг участников, общий прогресс по этапам и финальные результаты. Участникам не нужно искать каждый турнир по отдельности – всё собрано в одном месте. Это удобно как для игроков, так и для зрителей, которые следят за ходом всей серии.

Такой подход особенно полезен для крупных организаций, которые проводят регулярные турниры по одной игре. Раздел Circuits и пример отображения серии турниров в разделе Circuits показан на рисунке 1.9. и 1.10.

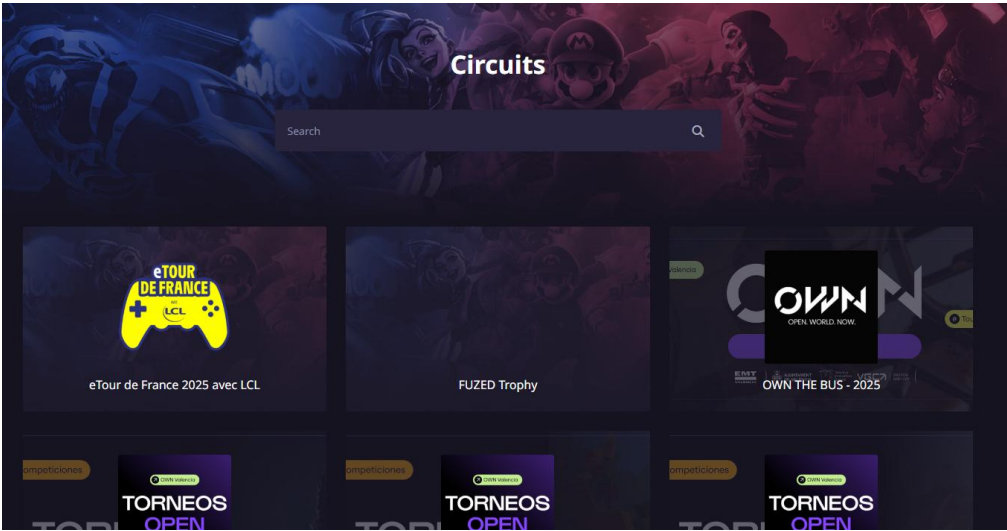


Рисунок 1.9 – раздел Circuits

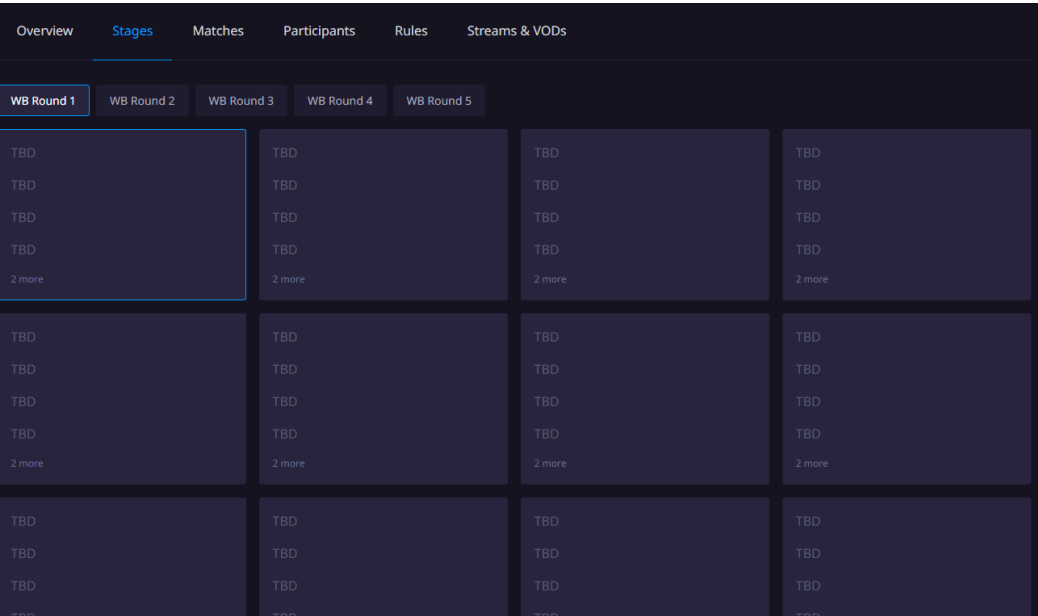


Рисунок 1.10 – пример отображения

На платформе Toornament предусмотрена функция создания команд. Это удобно в тех случаях, когда турниры проводятся не для одиночных участников, а для коллективов – например, в дисциплинах вроде CS:GO, Dota 2 или Valorant.

Пользователь может создать собственную команду, указать её название, тег, логотип и пригласить других игроков. У каждого члена команды будет свой профиль, но участие в турнирах будет происходить от имени команды. Это позволяет легко управлять составом, видеть общую статистику и подавать заявки сразу от всей команды, а не по отдельности.

Такая возможность особенно полезна для организаторов, которым нужно отслеживать не только игроков, но и их принадлежность к определённым коллективам.

Интерфейс создания команды и управления участниками представлен на рисунках 1.11 и 1.12.

Рисунок 1.11 – Форма создания команды

Рисунок 1.12 – Управление командой

Чтобы организовать свой турнир на платформе Toornament, нужно пройти несколько шагов. Всё начинается с обычной регистрации на сайте. После входа пользователь получает доступ к панели организатора, где можно создать новый турнир.

На первом этапе указывается базовая информация: название турнира, игра, формат проведения (например, олимпийка, круговой, швейцарская система), дата и описание. Далее можно настроить правила, этапы, расписание, форму регистрации и другие детали.

Платформа позволяет детально настроить турнир под себя – вплоть до внешнего оформления страницы и логотипа события. Также можно указать, будет ли турнир открытым для всех или только по приглашениям.

После настройки создаётся ссылка на турнир, которую можно распространять среди участников. Они смогут зарегистрироваться, а организатор – отслеживать заявки, формировать пары и публиковать результаты. Страница с организацией собственного турнира панель создания турнира и основные поля представлены на рисунках 1.13. и 1.14.

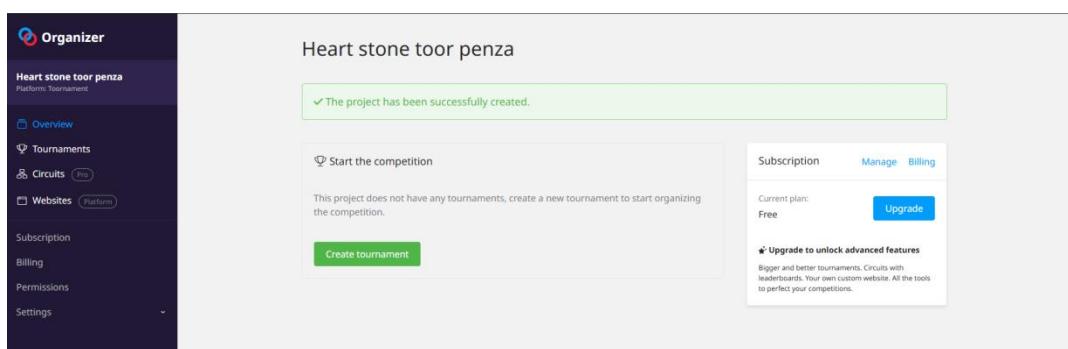


Рисунок 1.13 – страница с организацией собственного турнира

Create new tournament

Tournament name (maximum 30 characters)

Need a hand?
Check our Organizer Starter Guide.

Discipline

Platform(s)

PC
Playstation 4
Playstation 5
Xbox One
Xbox Series
Switch
Switch 2
+ Others

Size
Participants
☐ Players ☐ Teams

Want to organize a big tournament with more than 128 participants? Check the plans on the pricing page.

Timezone

+ Create

Рисунок 1.14 – панель создания турнира

Данный сервис обладает следующими преимуществами:

- удобная и понятная навигация, даже несмотря на большое количество функций;
- широкий выбор форматов турниров, включая сетки, круговые и швейцарские системы;
- поддержка множества игр с готовыми шаблонами настроек;
- возможность объединения турниров в серии (раздел Circuits);
- адаптивный дизайн, сайт удобно использовать как на компьютере, так и с телефона;
- продвинутые функции для организаторов: аналитика, брендинг, настройки доступа.

Но есть и свои минусы, которые стоит учитывать:

- перегруженный интерфейс для новичков, особенно в части настроек;
- почти всё завязано на обязательной регистрации;

					ПГУ 1.090301.25.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		19

- многие функции доступны только в платных тарифах;
- сайт в первую очередь ориентирован на киберспорт и командные соревнования, что не всегда удобно для локальных или учебных турниров;
- интерфейс и поддержка на русском языке отсутствуют, что может затруднить использование для русскоязычных пользователей.

В целом, Toornament – мощный инструмент для опытных организаторов, но для простых пользователей или небольших сообществ может оказаться немного сложным. Этот опыт был учтён при разработке собственного приложения.

					ПГУ 1.090301.25.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		20

1.2 Постановка задачи

Необходимо разработать web-приложение, предназначенное для организации и проведения турниров, в котором будут реализованы следующие функции:

- регистрация и вход в систему для пользователей, возможность создания личного кабинета;
- разграничение ролей (обычный пользователь и администратор);
- создание и отображение турниров (список турниров, подробная информация о каждом (название, описание, дата, статус, список участников);
- подача заявки на участие в турнире;
- просмотр списка участников и результатов;
- отображение личной статистики участника;
- редактирование профиля: смена имени, загрузка аватара, смена пароля;
- для администратора: возможность подтверждения или отклонения заявок на участие, управление пользователями, редактирование информации о турнирах;
- адаптивный интерфейс, чтобы приложение одинаково удобно работало как на компьютере, так и на смартфоне.

Разрабатываемый онлайн сервис должен соответствовать следующим техническим требованиям:

- простой и понятный интерфейс с логичной навигацией;
- оформление всех страниц в едином стиле с акцентом на удобство чтения;
- поддержка разных разрешений экранов благодаря адаптивной вёрстке;
- использование клиент-серверной архитектуры;
- корректная работа во всех популярных браузерах: Google Chrome, Mozilla Firefox, Microsoft Edge.

На рисунках 1.15 и 1.16 изображены диаграммы использования web-приложения с учетом ролевой модели доступа

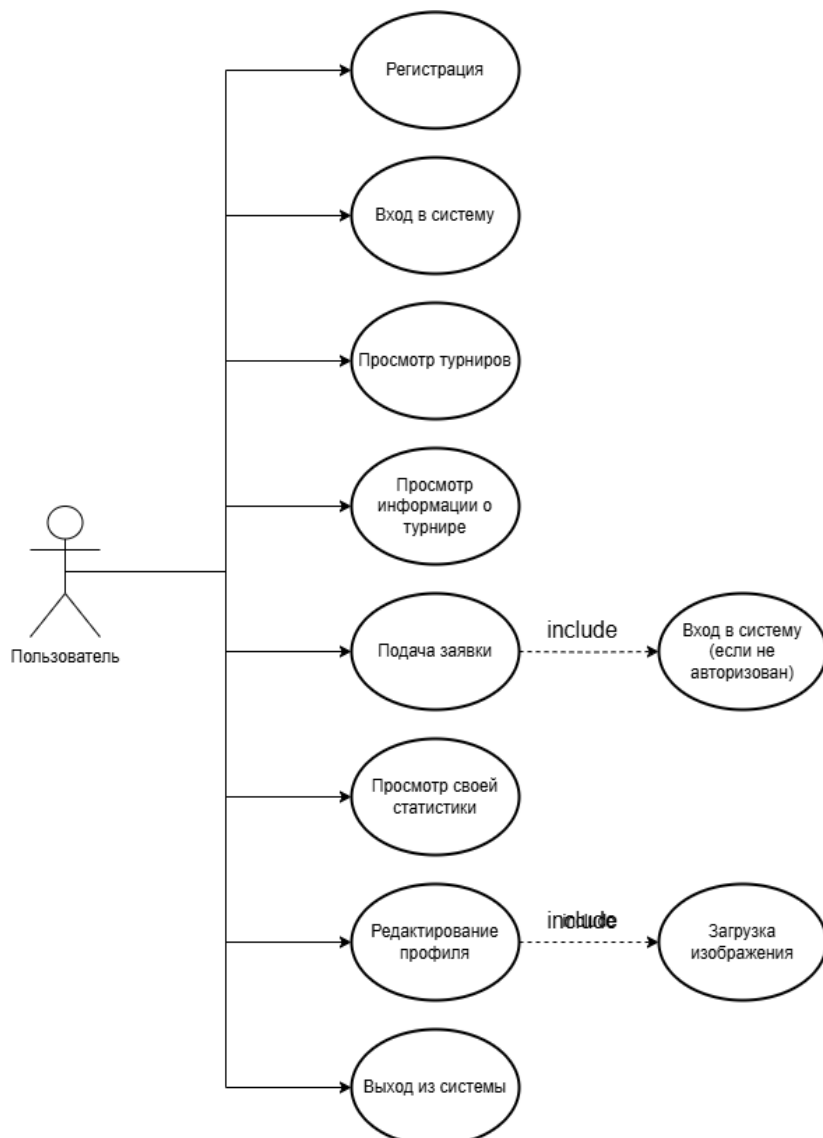


Рисунок 1.15—Диаграмма использования web-приложения пользователя без прав администратора

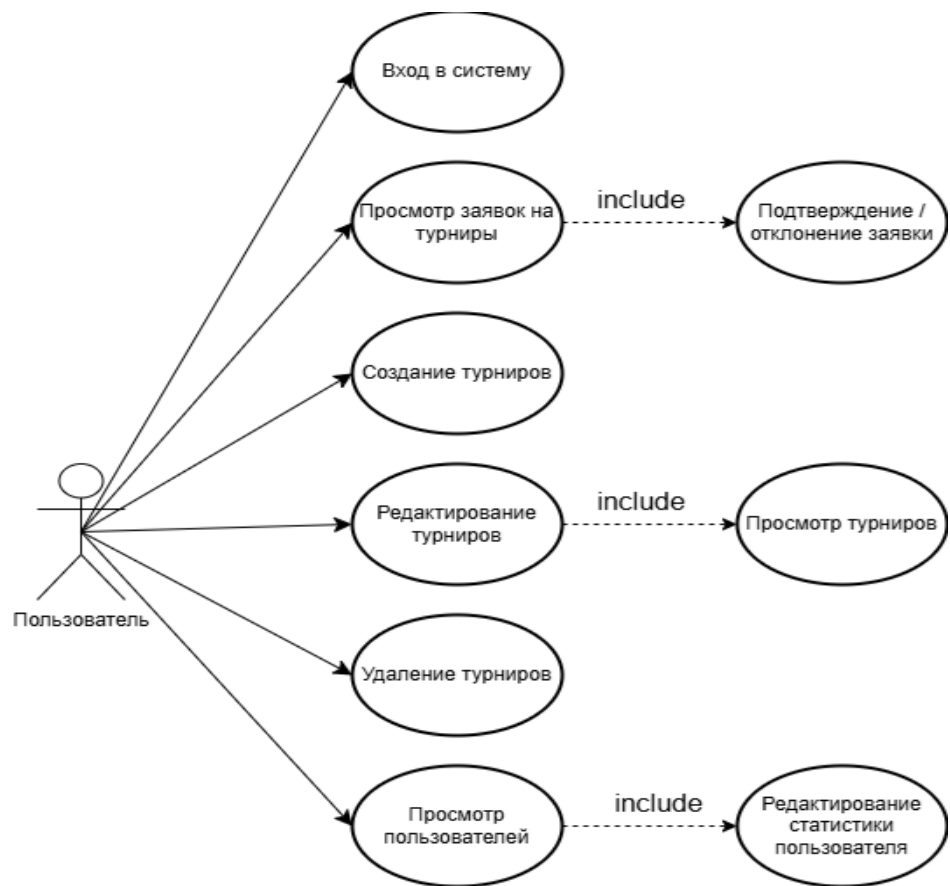


Рисунок 1.16 – Диаграмма использования web-приложения пользователя с правами администратора

1.3 Выбор технологий и средств разработки приложения

В этом разделе рассматриваются технологии и инструменты, которые были использованы при создании web-приложения для проведения турниров. Подбор средств был основан на простоте интеграции, стабильности, а также возможностях быстрой разработки и поддержки проекта.

Вот список основных технологий, применённых в ходе работы:

- HTML5 и CSS3;
- Jinja2;
- Python;
- Flask;
- SQLite;
- JSON.

1.3.1 Выбор технологий для клиентской части

Для отображения интерфейса пользователя использовались HTML5 и CSS3. Эти технологии позволяют формировать структуру страниц и придавать им современный внешний вид. HTML5 обеспечивает семантически корректную разметку, что упрощает поддержку и развитие проекта, а CSS3 помогает оформить интерфейс удобно и адаптивно, чтобы сайт корректно отображался как на компьютерах, так и на мобильных устройствах.

В качестве шаблонизатора применялась система Jinja2, встроенная во Flask. Это дало возможность динамически формировать страницы на основе данных, поступающих с сервера, без необходимости внедрять отдельный JavaScript-фреймворк вроде React.

Ядро всего приложения написано на языке Python. Это простой в освоении, но в то же время мощный язык, который хорошо подходит для разработки веб-приложений. Его читаемый синтаксис и большое количество готовых библиотек позволяют быстро и удобно создавать рабочие решения.

В качестве фреймворка для серверной части выбран Flask – минималистичный веб-фреймворк, который даёт возможность собрать всю логику

сервера буквально с нуля, подключая только нужные компоненты. Он не перегружен лишними зависимостями, хорошо подходит для небольших и средних проектов, таких как платформа для проведения турниров. Flask позволяет быстро настроить маршруты, обрабатывать запросы пользователей, подключать базу данных и рендерить HTML-шаблоны через встроенный шаблонизатор Jinja2.

Основные плюсы Flask:

- лёгкость в освоении и гибкость в использовании;
- не навязывает структуру проекта, давая свободу в архитектуре;
- отличная документация и большое русскоязычное сообщество.

1.3.2 Выбор технологий для серверной части

Выбор серверной части для проекта с турнирами был основан на простоте, надёжности и возможности быстро реализовать весь необходимый функционал. В качестве языка программирования использован Python — он понятен, читаем и отлично подходит для веб-разработки.

Для построения серверной логики выбран фреймворк Flask. Он не перегружен лишними компонентами, позволяет гибко настраивать маршруты и быстро запускать приложение даже без сложной конфигурации. Flask отлично справляется с задачами обработки запросов, авторизации, работы с базой данных и взаимодействия с клиентской частью.

В качестве хранилища данных используется SQLite. Это лёгкая и встроенная база, которая не требует отдельного сервера, что особенно удобно на этапе разработки. Все данные о пользователях, турнирах, заявках и результатах записываются в неё. Благодаря ORM-библиотеке SQLAlchemy работа с базой данных становится простой и наглядной.

Такой выбор технологий позволил создать надёжную серверную часть, легко масштабируемую и удобную для сопровождения.

1.3.3 Выбор системы управления базами данных

Для хранения информации о пользователях, турнирах, заявках и другой важной информации используется база данных SQLite. Это встроенная

реляционная СУБД, которая работает прямо в виде одного файла на сервере, без необходимости настраивать отдельный сервер баз данных.

SQLite отлично подходит для проектов с небольшим числом пользователей, где важны простота, надёжность и удобная интеграция. Она работает стабильно, поддерживает транзакции и легко подключается к Python-приложению через стандартную библиотеку.

Почему выбран именно SQLite:

- не требует отдельного сервера, достаточно одного файла;
- легко подключается к приложению на Flask;
- работает быстро и стабильно, особенно при умеренной нагрузке;
- идеально подходит для разработки и тестирования.

1.3.4 Выбор формата обмена данными

Для обмена данными между клиентом и сервером в проекте используется формат JSON. Это текстовый формат, понятный как человеку, так и компьютеру. Он особенно хорошо подходит для взаимодействия между фронтендом и бэкендом в веб-приложениях.

JSON применяется в следующих сценариях:

- передача данных при регистрации и авторизации;
- отправка заявок на участие в турнире;
- получение списков турниров, участников, статистики;
- передача уведомлений, ошибок и системных сообщений.

Преимущества использования JSON:

- легко читается и быстро обрабатывается;
- поддерживается во всех современных языках программирования;
- хорошо работает с вложенными структурами (например, списки заявок или турниров);
- отлично сочетается с Flask и позволяет быстро создавать REST-интерфейсы.

1.4 Архитектура приложения

1.4.1 Архитектурный паттерн MVC

В разработке веб-приложения для турниров используется классическая клиент-серверная архитектура. Основной упор сделан на простоту, читаемость кода и разделение логики между сервером и представлением. Интерфейс страниц формируется с помощью HTML-шаблонов Jinja2 прямо на сервере, что исключает необходимость в отдельном фронтенд-фреймворке, таком как React.

Это позволило упростить структуру и сделать приложение более доступным для поддержки, так как вся логика сосредоточена в одном месте – на сервере. Разметка, логика отображения и данные объединены в шаблонах, что особенно удобно при небольшом числе страниц и функциональности, не требующей сложных динамических интерфейсов.

1.4.2 Клиент-серверное взаимодействие и REST API

Клиентская часть (браузер пользователя) взаимодействует с сервером по протоколу HTTP. Большая часть страниц формируется на сервере и отправляется в готовом виде. Однако при необходимости – например, при отправке заявок, обновлении профиля или загрузке аватара – данные передаются в формате JSON через специальные маршруты во Flask.

Сервер принимает запросы, обрабатывает данные, обращается к базе данных SQLite, а затем возвращает результат в виде HTML-страниц или JSON-ответов. Это позволяет комбинировать классический подход с элементами REST API.

1.4.3 Основные серверные классы и модели

Серверная часть приложения построена на Flask и использует ORM SQLAlchemy для работы с базой данных. Внутри реализованы следующие ключевые модели:

- модель User – представляет пользователя системы, содержит ID, имя пользователя, e-mail, пароль и роль (обычный пользователь или администратор);
- модель Tournament – хранит информацию о турнирах (название, описание, дата, статус и список участников);

- модель Application – отражает заявки пользователей на участие в турнире;
- модель Stats – отвечает за хранение статистики по каждому пользователю (например, количество турниров, побед и т.д.);
- модель Avatar – дополнительная сущность для загрузки и хранения пользовательских аватаров.

Эти классы отражают логику работы системы и обеспечивают удобную структуру хранения всех необходимых данных.

1.4.4 Управление доступом по ролям

В приложении предусмотрено два типа пользователей – обычные участники и администраторы:

- обычный пользователь может регистрироваться, входить в систему, просматривать турниры, подавать заявки и редактировать свой профиль;
- администратор имеет расширенные права: он может просматривать все заявки, подтверждать или отклонять участие, управлять списками пользователей и редактировать турниры.

Роль пользователя определяется при входе и сохраняется в текущем сеансе. Все действия, требующие прав администратора, защищены и проверяются на стороне сервера. Таким образом обеспечивается надёжная система доступа, которая исключает возможность несанкционированного вмешательства в управление.

1.4.5 Масштабируемость и расширяемость архитектуры

Архитектура приложения построена с учётом будущего расширения. При необходимости можно легко добавить новые функции – например, роли судьи, модератора или организатора – без кардинальной переработки системы.

Новая модель данных или страница может быть внедрена в рамках существующей структуры с минимальными изменениями. Это достигается благодаря модульной организации кода, разделению ответственности между частями приложения и использованию гибкого фреймворка Flask. Схема хранения загружаемых файлов показана на рисунке 1.17

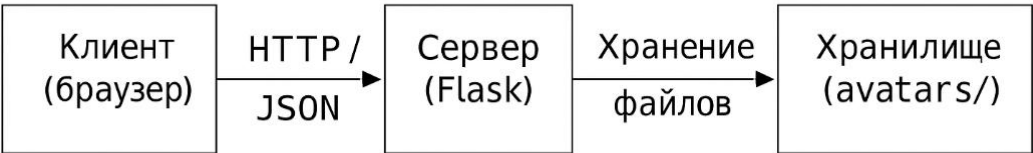


Рисунок 1.17–Схема хранения загружаемых файлов

1.5 Разработка приложения

1.5.1 Разработка клиентской части

Клиентская часть платформы для проведения турниров реализована с помощью HTML-шаблонов, встроенных в Flask. Вместо использования отдельного фронтенд-фреймворка вроде React, приложение использует серверный рендеринг через шаблонизатор Jinja2. Это позволило упростить структуру, не теряя в функциональности.

В основе всех страниц лежит шаблон `base.html`, который содержит общие элементы интерфейса: шапку, подвал, подключение стилей и блок контента. Через блоки `{% block content %}` в него вставляются конкретные шаблоны страниц – например, главная, личный кабинет, список турниров, страница заявки и другие.

Навигация динамически адаптируется в зависимости от роли пользователя. Для неавторизованных пользователей отображаются только кнопки входа и регистрации. Если пользователь авторизован – появляется доступ к личному кабинету, заявкам и выходу из системы. У администратора, помимо этого, есть ссылка на административную панель.

На главной странице отображаются ближайшие турниры. Список формируется в маршруте Flask, и данные передаются в шаблон:

```
@app.route('/')
def home():
    tournaments =
    Tournament.query.order_by(Tournament.date.desc()).limit(3).all()
    return render_template('index.html', tournaments=tournaments).
```

В шаблоне `index.html` список выводится через простой цикл:

```
{% for tournament in tournaments %}
    <div class="card">
    <h3>{{ tournament.name }}</h3>
    <p>Дата: {{ tournament.date.strftime('%d.%m.%Y') }}</p>
    <p>Место: {{ tournament.location }}</p>
    </div>
{% endfor %}.
```

					ПГУ 1.090301.25.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		30

Страница профиля позволяет пользователю загрузить новый аватар. Для этого используется форма с типом multipart/form-data. После отправки файл сохраняется на сервере, и путь к аватару обновляется в базе данных. На клиенте загруженный аватар сразу отображается.

Форма профиля:

```
<formmethod="POST" enctype="multipart/form-data">
<inputtype="file" name="avatar">
<buttontype="submit">Обновить</button></form>.
```

Отдельный шаблон отображает список всех доступных турниров. У каждого турнира есть кнопка «Подать заявку». При нажатии на кнопку отправляется POST-запрос на соответствующий маршрут, который сохраняет заявку в базе данных.

```
{% fortintournaments %}
    <formmethod="POST" action="{{ url_for('apply', id=t.id) }}">
<h3>{{ t.name }}</h3>
<p>{{ t.date.strftime('%d.%m.%Y') }} - {{ t.location }}</p>
<buttontype="submit">Подать заявку</button>
    </form>
{% endfor %}.
```

Если пользователь уже подавал заявку, кнопка заменяется на надпись «Заявка отправлена». Это определяется логикой на сервере.

В административной панели администратор может просматривать и подтверждать заявки участников, а также управлять турнирами и статистикой. Интерфейс простой и текстовый, но полностью покрывает нужный функционал. Шаблоны также построены на Jinja2 и используют таблицы и формы.

Каждая страница сайта адаптивна: базовая вёрстка CSS построена с учётом разных размеров экранов. Сайт корректно отображается как на десктопе, так и на мобильных устройствах.

Таким образом, клиентская часть приложения полностью построена на шаблонах Flask и HTML. Это позволило добиться стабильной и предсказуемой

работы интерфейса без необходимости использовать JavaScript или сторонние библиотеки. Такой подход отлично подходит для проектов средней сложности, где важна простота и скорость разработки. Диаграмма классов пользователей показана на рисунке 1.18.

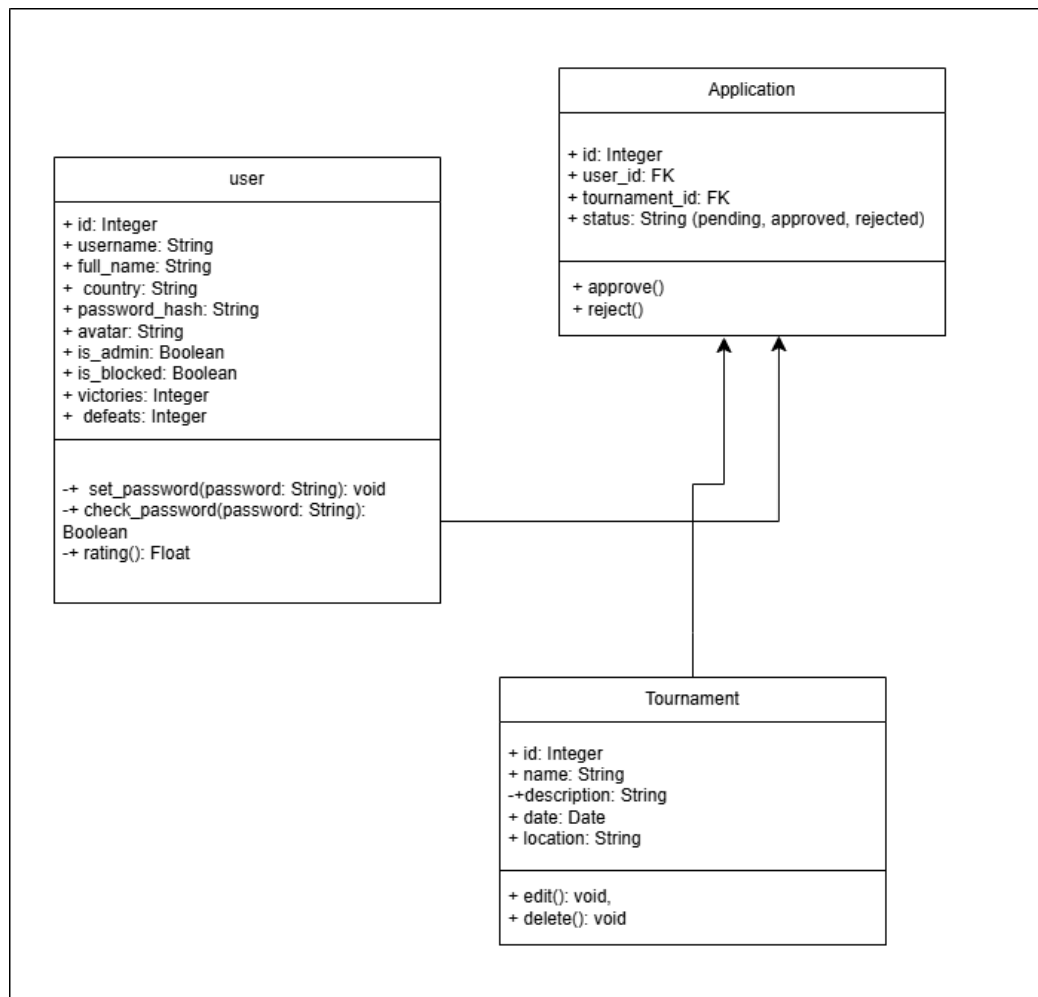


Рисунок 1.18–Диаграмма классовпользователей

Панель администратора представлена в `AdminDashboard.jsx`. Внутри используются компоненты `NewBooksTab` (заявки на турниры), `RequestsTab` (заявки на статус автора) и `UsersListTab` (список зарегистрированных пользователей). Для каждого типа запроса реализованы кнопки "одобрить" и "отклонить". Администратор может просматривать список пользователей, блокировать и удалять их.. Диаграмма классов пользователей с правами администратора показана на рисунке 1.19.

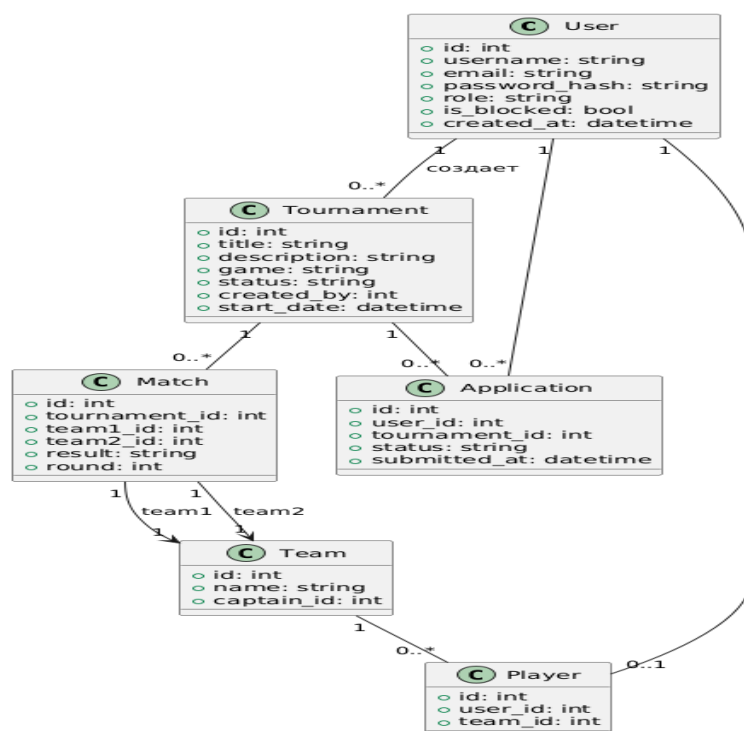


Рисунок 1.19 – Диаграмма классов пользователя с правами администратора

1.5.2 Разработка серверной части

Серверная часть веб-приложения написана на Python с использованием фреймворка Flask. В основном файле app.py настраивается всё необходимое: создаётся приложение, подключается база данных SQLite, настраивается директория для загрузки аватаров, и регистрируются маршруты.

Основная логика приложения реализована через декораторы `@app.route`. Внутри этих маршрутов обрабатываются все ключевые действия: регистрация, вход в систему, подача заявок, отображение турниров, а также административные действия – подтверждение заявок и редактирование профилей пользователей.

Например, регистрация нового пользователя реализована с помощью формы `RegistrationForm`. Когда пользователь заполняет данные и отправляет форму, создаётся новый объект `User` и сохраняется в базу данных. Вот как это работает:

```

@app.route('/register', methods=['GET', 'POST'])
def register():
    form = RegistrationForm()
    if form.validate_on_submit():

```

```

user = User(
username=form.username.data,
full_name=form.full_name.data,
country=form.country.data
)
user.set_password(form.password.data)
db.session.add(user)
db.session.commit()
flash('Регистрация прошла успешно')
return redirect(url_for('login'))
return render_template('register.html', form=form) .

```

Также реализована подача заявки на участие в турнире. Если пользователь нажимает кнопку «Подать заявку», сервер проверяет, нет ли уже существующей заявки, и если всё в порядке, сохраняет новую:

```

@app.route('/tournament/int:id/apply', methods=['POST'])
@login_required
def apply(id):
existing = Application.query.filter_by(user_id=current_user.id,
tournament_id=id).first()
if existing:
flash('Вы уже подали заявку')
else:
application = Application(user_id=current_user.id, tournament_id=id)
db.session.add(application)
db.session.commit()
flash('Заявка отправлена')
return redirect(url_for('tournament_detail', id=id)) .

```

Всё, что касается базы данных, реализовано через SQLAlchemy. Внутри app.py определены модели пользователей, турниров и заявок. Например, вот как выглядит модель User:

```

class User(UserMixin, db.Model):
id = db.Column(db.Integer, primary_key=True)
username = db.Column(db.String(50), unique=True, nullable=False)
full_name = db.Column(db.String(100), nullable=False)

```

```
country = db.Column(db.String(50), nullable=False)
password_hash = db.Column(db.String(128))
is_admin = db.Column(db.Boolean, default=False)
avatar = db.Column(db.String(100), default='default.jpg')
victories = db.Column(db.Integer, default=0)
defeats = db.Column(db.Integer, default=0) .
```

В модели также реализованы функции установки и проверки пароля. Расчёт рейтинга строится на соотношении побед и поражений.

Конфигурация приложения (включая путь к базе данных и размер загружаемых файлов) задаётся прямо в app.py. Например:

```
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///tournament.db'
app.config['UPLOAD_FOLDER'] = 'static/uploads/avatars'
app.config['MAX_CONTENT_LENGTH'] = 2 * 1024 * 1024.
```

Для авторизации и разграничения прав используется Flask-Login. При входе в систему создаётся сессия, в которой сохраняются данные пользователя. В зависимости от роли (обычный пользователь или администратор) отображаются разные элементы интерфейса и доступны разные функции.

Большинство действий, особенно связанных с изменением данных, защищены декоратором @login_required. Это значит, что только авторизованные пользователи могут, например, подать заявку или изменить аватар.

Сервер возвращает либо готовые HTML-страницы, либо сообщения через flash. JSON-ответы используются в отдельных случаях, например, при загрузке аватара или асинхронной обработке форм.

В целом, серверная часть обеспечивает всю основную логику проекта, включая регистрацию, авторизацию, управление турнирами и ролями пользователей. Всё построено в рамках одного Flask-приложения, что делает структуру простой и понятной.

1.6 Описание приложения

1.6.1 Руководство пользователя

После перехода на сайт пользователь попадает на главную страницу, где отображаются ближайшие турниры. Каждый турнир представлен в виде карточки с названием, датой проведения и местом. Отсюда пользователь может перейти к подробному описанию турнира или подать заявку на участие, если он авторизован.

Незарегистрированный пользователь может просматривать список турниров и состав участников, но не может подавать заявки и просматривать личную статистику. Если он попытается воспользоваться функцией, требующей входа, система автоматически предложит авторизоваться или зарегистрироваться.

Регистрация проходит через простую форму, где необходимо указать никнейм, полное имя, страну и пароль. При успешной регистрации создаётся личный кабинет. Вход в систему осуществляется через отдельную форму с проверкой правильности введённых данных. В случае ошибки появляется уведомление, что логин или пароль указаны неверно. Интерфейс формы входа представлен на рисунке 1.20.

Вход в систему

Никнейм

admin

Пароль

.....

Войти

Нет аккаунта? [Зарегистрироваться](#)

Рисунок 1.20–Форма входа в профиль

Главная страница отображает ближайшие турниры в виде карточек с названием, датой и местом проведения. В верхней части страницы находится

навигационное меню, откуда можно перейти в профиль, список всех турниров или выполнить вход в систему. Главная страница сайта показана на рисунке 1.21.

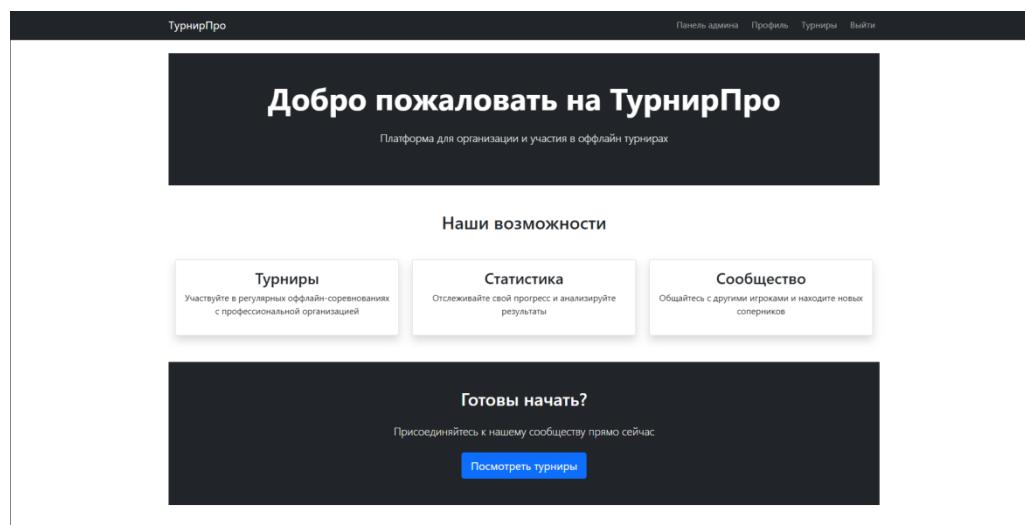


Рисунок 1.21 – Главная страница сайта

На этой странице отображается полный список турниров, доступных на платформе. Для каждого турнира указано его название, дата и место проведения. Пользователь может нажать на любой из них, чтобы перейти к подробному описанию и подаче заявки. Список всех турниров показан на рисунке 1.22.

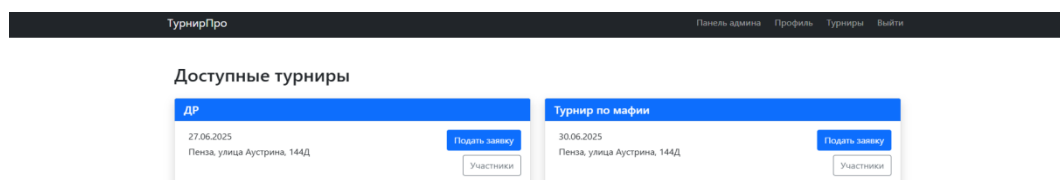


Рисунок 1.22–Просмотр турнира и подача заявки

На странице турнира отображаются его название, дата, описание и список участников. Авторизованный пользователь может нажать кнопку «Подать заявку». Если заявка уже была подана – выводится соответствующее сообщение. Страница турнира с о списком участников показана на рисунке 1.23. Форма ближайших турниров показана на картинке 1.24.

Участники турнира "ДР"

Список участников

Рисунок 1.23 – Просмотр списка участников

Ближайшие турниры

Турнир по мафии	ДР
30.06.2025	27.06.2025
Пенза, улица Аустрина, 144Д	Пенза, улица Аустрина, 144Д
Подать заявку	Подать заявку

Рисунок 1.24 – Форма ближайших турниров на главной страницы

Регистрационная форма позволяет создать аккаунт, указав никнейм, полное имя, страну и пароль. При ошибке система сообщает о необходимости корректного ввода данных. Интерфейс регистрации показан на рисунке 1.25.

Регистрация участника

Никнейм

Полное имя

Страна

Пароль

Подтверждение пароля

[Зарегистрироваться](#)

Уже зарегистрированы? [Войти](#)

Рисунок 1.25 – Интерфейс регистрации

Системой предусмотрен пользователь с правами администратора, в аккаунт которого можно зайти с помощью специального логина и пароля. На рисунке 1.26 показан профиль администратора.

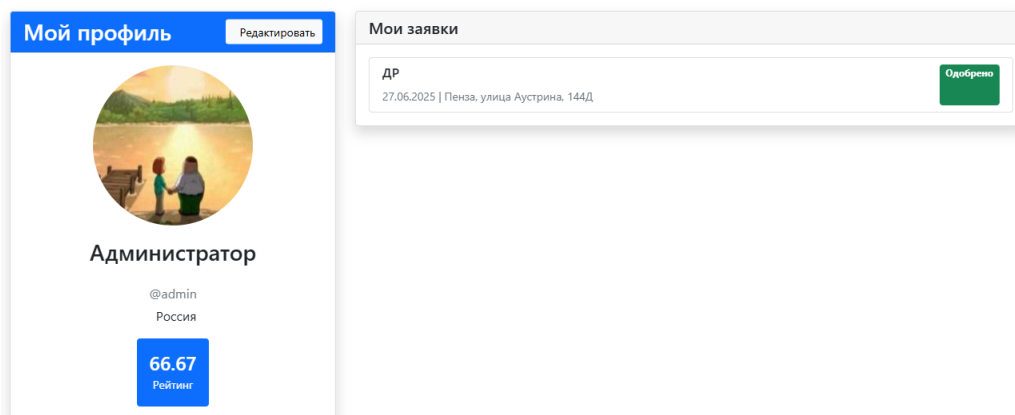


Рисунок 1.26 – Профиль администратора

В панели администратора отображаются заявки на участие. Администратор может их подтвердить или отклонить, а также просмотреть, кто и в какой турнир подал заявку. Интерфейс управления заявками показан на рисунке 1.33. Раздел администратора с возможностью редактировать статистику пользователей, блокировать или удалять аккаунты. Упрощает контроль за честностью и поведением участников. Полный вид панели администратора и интерфейс управления пользователями показаны на рисунках 1.27 и 1.28.

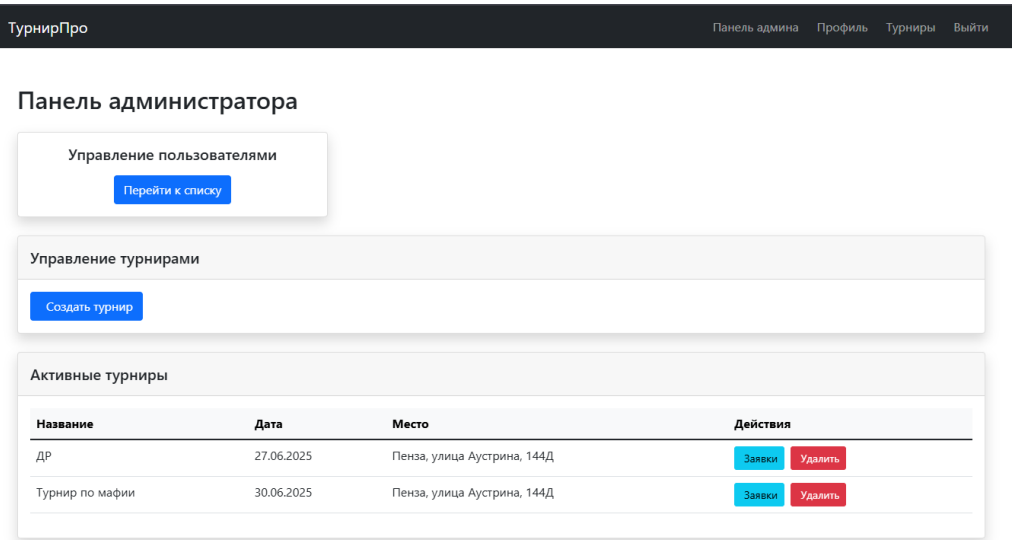


Рисунок 1.27–Общий вид панели администратора

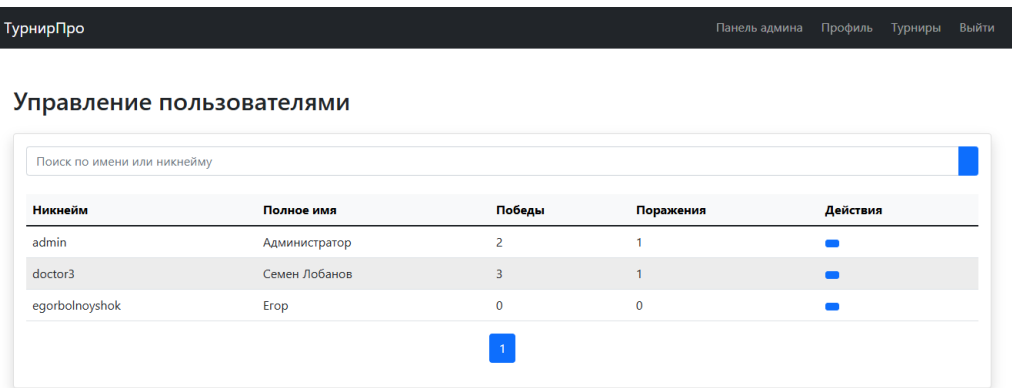


Рисунок 1.28– Форма управление пользователями на панели администратора

2 Охранатруда

2.1 Эргономическиетребования

При разработке веб-приложения для проведения турниров большое внимание уделялось удобству интерфейса. От того, насколько сайт понятен и комфортен в использовании, напрямую зависит, как часто им будут пользоваться и с каким настроением.

Всё оформление построено на простых и нейтральных цветах, без лишней пестроты. Шрифты подобраны такие, чтобы текст было легко читать даже на небольших экранах. Кнопки и поля расположены последовательно и логично – ничего не сбивает с толку и не отвлекает. Благодаря адаптивной вёрстке сайт одинаково хорошо выглядит и на компьютере, и на телефоне.

Особое внимание уделялось тому, чтобы пользователю не приходилось тратить лишние усилия на поиск нужных разделов. Навигация построена просто и понятно. Нет навязчивых всплывающих окон, раздражающих баннеров и других визуальных перегрузок – всё сделано для того, чтобы можно было сосредоточиться на сути: участии в турнирах и работе с личным кабинетом.

2.2 Организация рабочего места разработчика

Разработка велась за обычным персональным компьютером, и здесь тоже важно было учитывать комфорт и безопасность. Долгая работа за экраном требует внимания к деталям, которые на первый взгляд кажутся мелочами.

Монитор должен быть установлен на уровне глаз, на расстоянии около 60–70 сантиметров – так снижается нагрузка на зрение и шею. Рабочее место должно быть хорошо освещено, но без бликов на экране. Кресло желательно с регулировкой и поддержкой поясницы – это важно, особенно при длительном сидении.

Разработчику нужно делать короткие перерывы каждый час – хотя бы 5–10 минут, чтобы размять руки, отвести глаза от экрана, немного пройтись. Это помогает избежать переутомления и сохранять работоспособность дольше. Также

имеет значение температура в помещении, наличие свежего воздуха и тишина – всё это влияет на концентрацию и общее самочувствие во время работы.

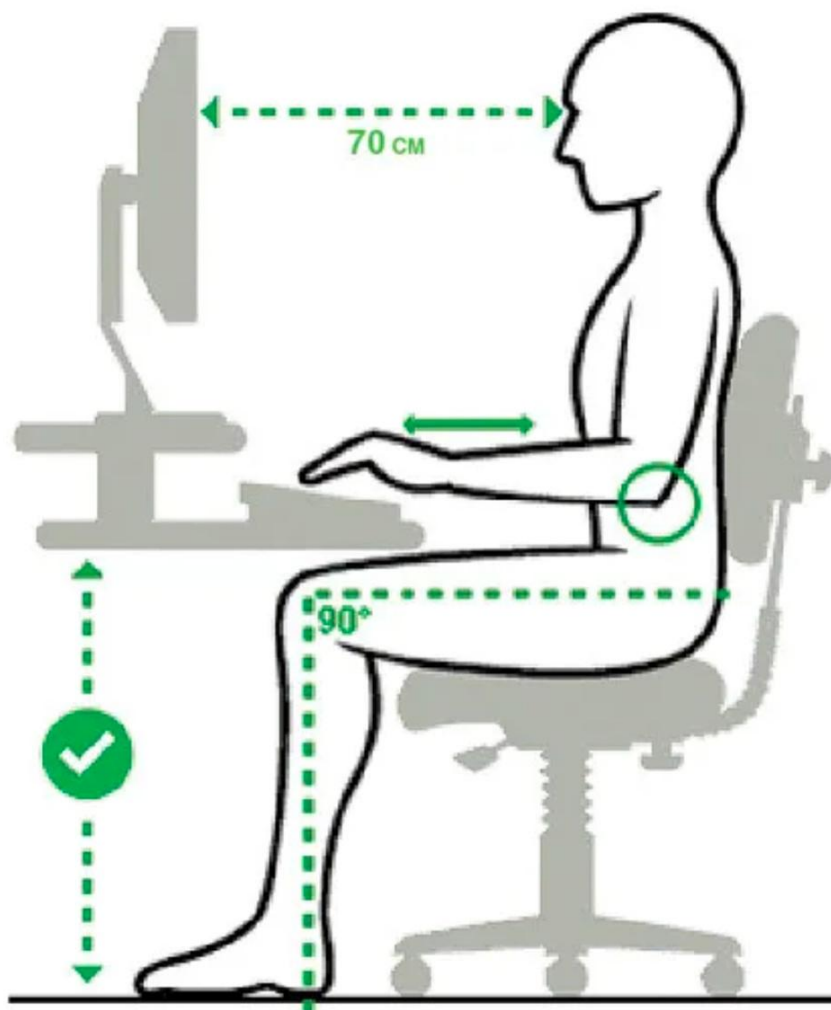


Рисунок 2.1–Правильная посадка за ПК

2.3 Правовое соответствие

При разработке веб-приложения учитывались основные положения закона №152-ФЗ «О персональных данных». В приложении действительно обрабатываются некоторые личные данные пользователей – например, имя, никнейм, страна и аватар. Эти данные нужны, чтобы пользователь мог зарегистрироваться, войти в систему и полноценно использовать функционал платформы.

Так как проект пока работает в тестовом режиме и не размещён на публичном сервере, подключение HTTPS пока не реализовано. Это задача на будущее – при переходе на полноценный хостинг и развёртывании в продакшн-режиме обязательно будет добавлен защищённый протокол и сертификат безопасности.

Тем не менее, базовые меры защиты и разделения прав доступа уже учтены:

- реализована система авторизации, где обычные пользователи и администраторы имеют разные уровни доступа;
- действия, которые выполняются в системе, проверяются на стороне сервера;
- предусмотрена возможность добавить пользовательское соглашение и политику конфиденциальности, которые в будущем можно будет показывать и подтверждать при регистрации;

Таким образом, даже на этапе локальной разработки вопрос безопасности и соответствия требованиям законодательства не остался без внимания.

2.4 Требования к тестированию и качеству

Важно не только, чтобы сайт работал – важно, чтобы он работал стабильно и не раздражал пользователей. Ошибки, зависания и потеря данных могут вызывать стресс как у обычных участников, так и у тех, кто поддерживает проект.

Поэтому в процессе разработки были предприняты меры, чтобы сделать платформу максимально устойчивой и надёжной:

- интерфейс тестировался на разных устройствах и в популярных браузерах (чтобы не было проблем с отображением);
- формы ввода защищены от пустых или некорректных данных, а ошибки выводятся понятно и по делу;
- при сбоях предусмотрены простые действия для восстановления – например, повторная загрузка формы или возврат к безопасному экрану;
- специально проверялись ситуации, когда пользователь вводит что-то не так, закрывает страницу, обновляет её или теряет подключение;

– уведомления о результатах действий (успешно/ошибка) выводятся прямо на экран, чтобы человек понимал, что происходит.

Эти шаги не только упрощают работу пользователей, но и снижают вероятность утомления и стресса от общения с системой. Это тоже часть охраны труда – просто в цифровом смысле.

					ПГУ 1.090301.25.001 ПЗ	Лист
						44
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

В рамках выпускной квалификационной работы было разработано веб-приложение для проведения и администрирования турниров. Основной упор сделан на создание понятного и удобного интерфейса, который позволял бы участникам легко находить интересующие турниры и подавать заявки, а администраторам – управлять заявками и участниками.

Клиентская и серверная часть реализованы на базе Flask, что позволило объединить всю логику в едином приложении. Данные хранятся в базе SQLite, а визуальная часть формируется с помощью HTML-шаблонов Jinja2. Такой подход упростил разработку и сделал систему понятной для поддержки и расширения. Реализована система авторизации с разграничением ролей: обычный пользователь и администратор. Также предусмотрена возможность загрузки аватара, просмотра личной статистики, подачи и обработки заявок на участие в турнирах.

Пользователь получает простой доступ к функционалу: просмотр турниров, подача заявки, редактирование профиля. Администратор – к панели управления, где можно модерировать участников, редактировать информацию о турнирах и принимать решения по заявкам. Все действия логически разделены, а навигация организована так, чтобы работать с системой было удобно и понятно.

С технической стороны проект выполнен в модульной структуре: база моделей, маршруты, шаблоны, формы и конфигурация разделены по логике, что позволяет легко дорабатывать и масштабировать приложение. Проведённое тестирование показало стабильную работу на разных устройствах и в популярных браузерах. При разработке учитывались и базовые нормы охраны труда, что особенно важно при длительной работе за компьютером.

В дальнейшем проект можно развивать в следующих направлениях:

- подключение внешнего хостинга и настройка защищённого HTTPS-соединения;
- добавление роли организатора турниров и расширение прав доступа;
- реализация системы уведомлений о результатах;
- подключение панели аналитики с данными по участникам и турнирам;

– визуальное оформление турнирных сеток и автоматическая генерация матчей.

В целом работа над этим проектом позволила закрепить и применить на практике полученные знания в области веб-разработки, проектирования интерфейсов и работы с базами данных. Результатом стала полноценная система, которую можно использовать в образовательных, спортивных и игровых сообществах для организации и ведения турниров.

					ПГУ 1.090301.25.001 ПЗ	Лист
						46
Изм.	Лист	№ докум.	Подпись	Дата		

Список использованных источников

1. Toornament – популярный онлайн-сервис для проведения турниров. Использовался как пример готовой платформы. Дата обращения: 02.05.2025.
<https://www.toornament.com>

2. Battlefy – ещё один крупный сайт для организации онлайн-турниров. Послужил источником идей для интерфейса. Дата обращения: 02.05.2025.
<https://battlefy.com>

3. Cyberscore – русскоязычная платформа для проведения соревнований. Анализировался как локальный аналог. Дата обращения: 02.05.2025.
<https://cyberscore.live>

4. Хабр, Алексей Мерсер – подробная статья по Flask, помогла разобраться в структуре приложения и маршрутах. Дата обращения: 10.05.2025.
<https://habr.com/ru/articles/817369/>

5. SkillFactory – краткое объяснение, что такое REST API и как его применять. Дата обращения: 10.05.2025.
<https://blog.skillfactory.ru/glossary/rest-api/>

6. Документация Flask -официальный сайт, часто использовался для уточнения синтаксиса и примеров. Дата обращения: 11.05.2025. (дата обращения 14.05.2025).

7. GitVerse - статья о шаблонизаторе Jinja2, помогла понять, как вставлять данные в HTML. Дата обращения: 13.05.2025.
<https://gitverse.ru/blog/articles/development/jinja2-basics> (дата обращения 15.05.2025).

8. Flask-WTF – официальная документация по формам и валидации данных в Flask. Дата обращения: 14.05.2025. <https://flask-wtf.readthedocs.io>

Приложение А

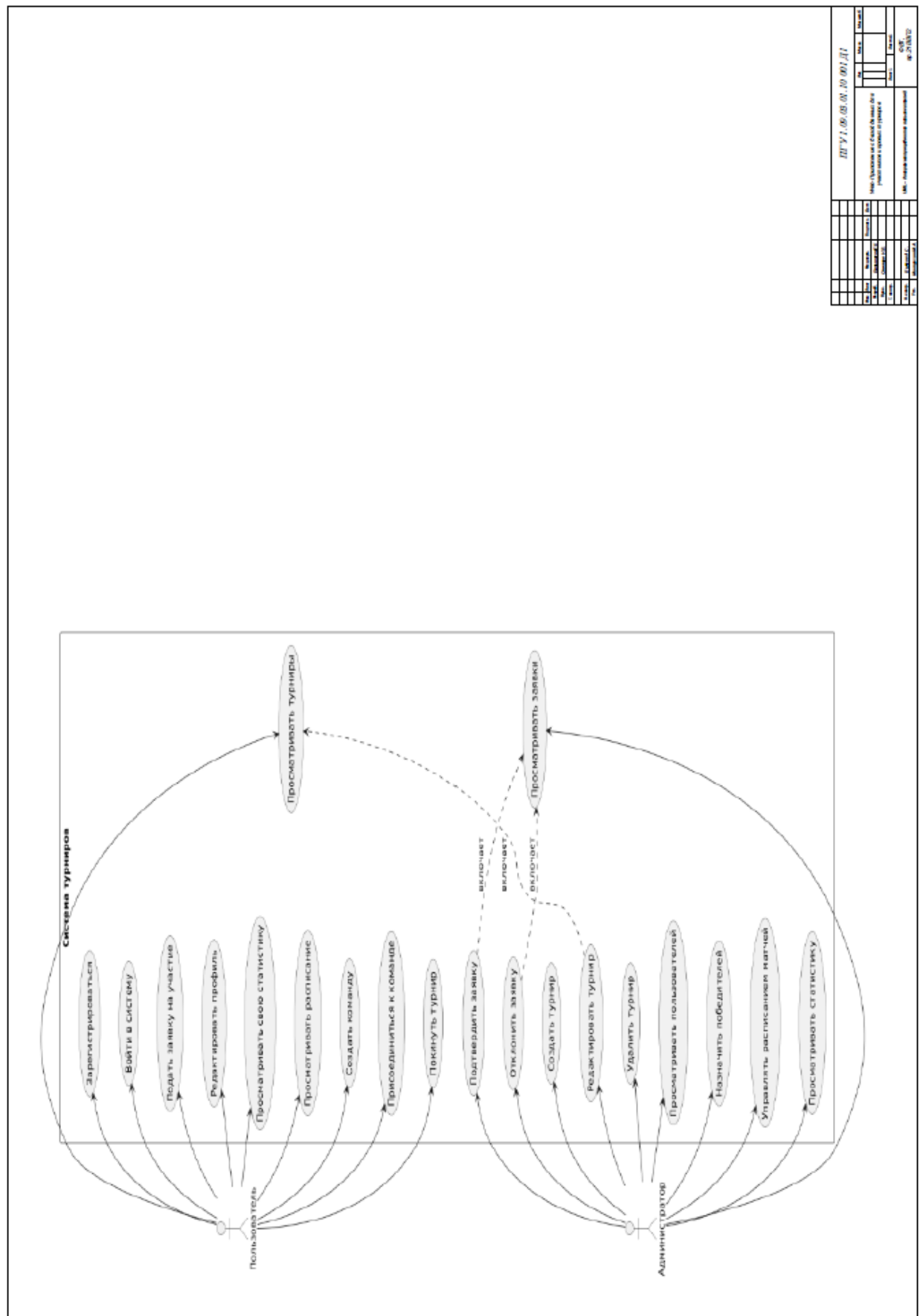
(обязательное)

Web-приложение с базой данных для участников игровых турниров

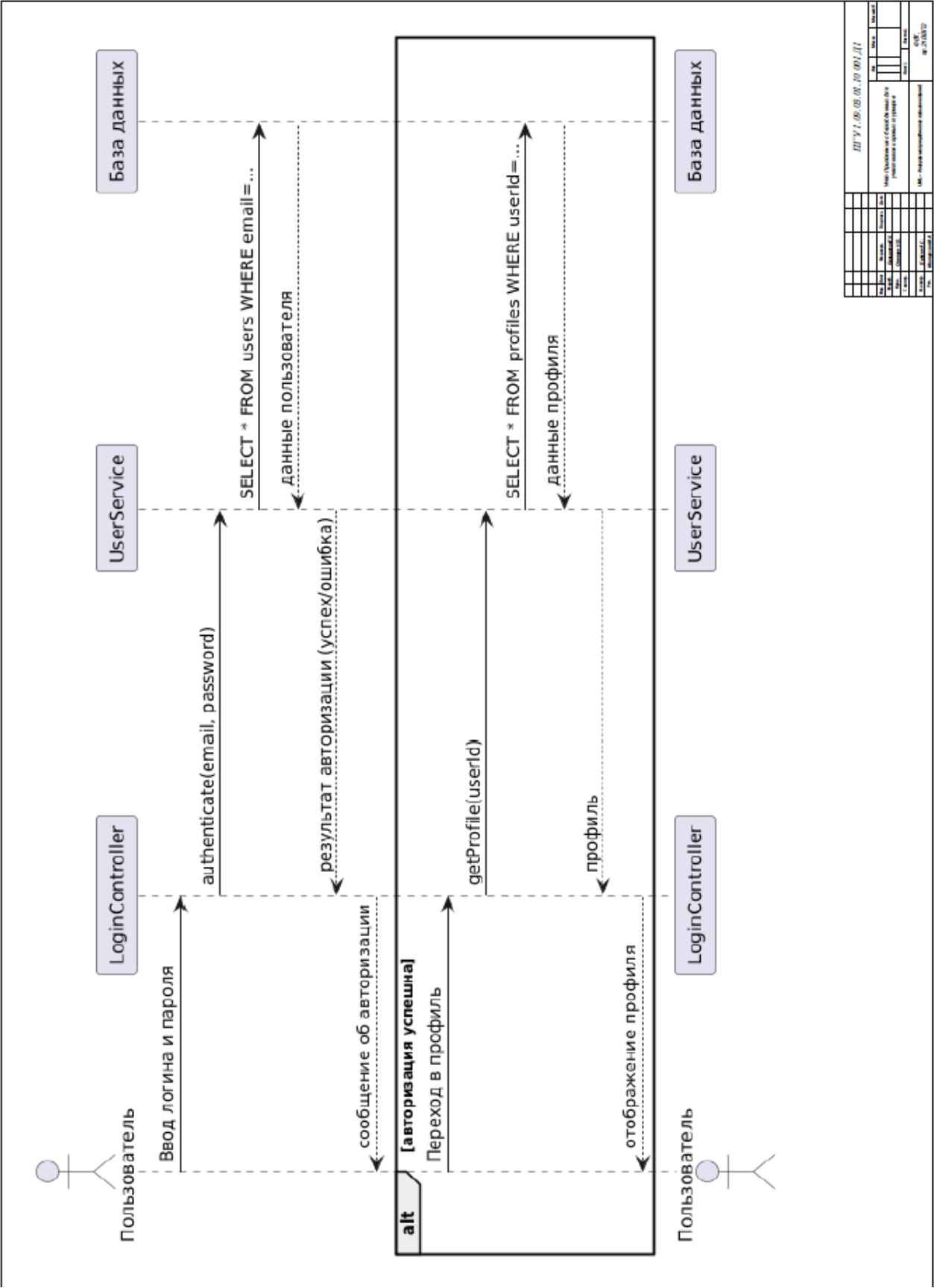
UML-диаграммы

					ПГУ 1.090301.25.001 ПЗ	Лист
						48
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А.1 – UML – диаграмма прецедентов пользователей

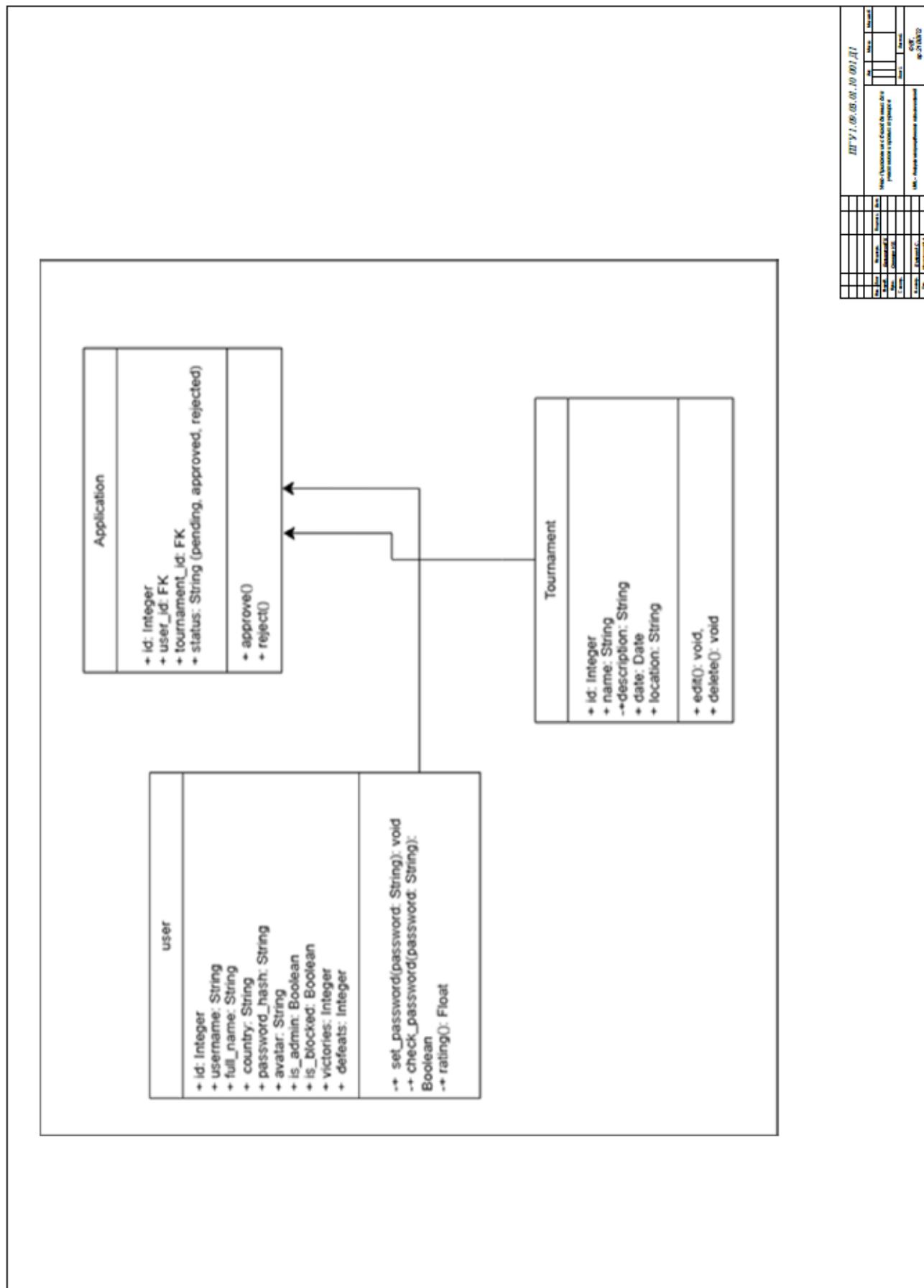


Приложение А.2 – UML – диаграмма последовательности оформления
заказа

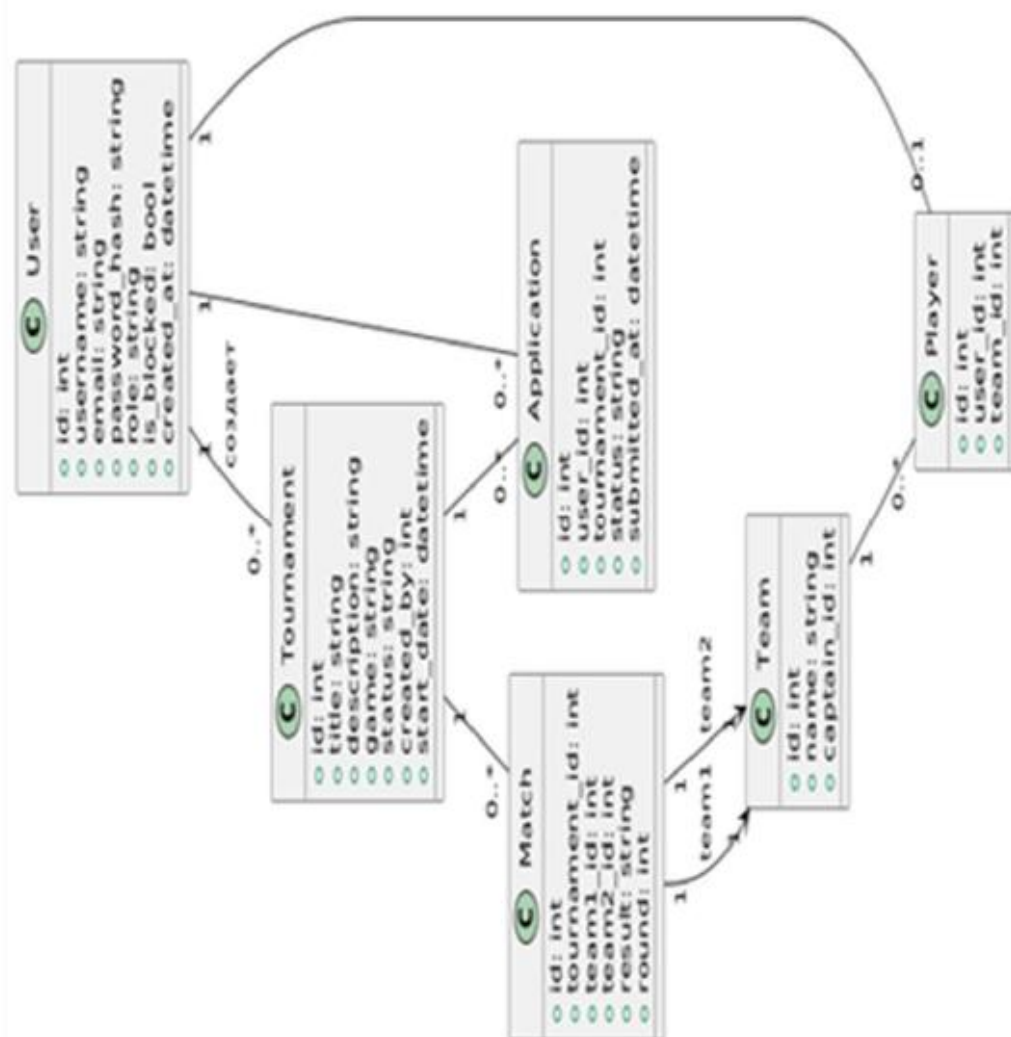


ПГУ 1.09.03.01.25.001 ПЗ									
№ докум.	Дата	Изм.	Исполн.	Провер.	Соглас.	Соглас.	Соглас.	Соглас.	Соглас.
1.0	2020.08.10	1	И.И.И.	И.И.И.	И.И.И.	И.И.И.	И.И.И.	И.И.И.	И.И.И.
Итого: 10 листов, в том числе 10 листов с рисунками									
Итого: 10 листов, в том числе 10 листов с рисунками									
Итого: 10 листов, в том числе 10 листов с рисунками									
Итого: 10 листов, в том числе 10 листов с рисунками									
Итого: 10 листов, в том числе 10 листов с рисунками									
Итого: 10 листов, в том числе 10 листов с рисунками									
Итого: 10 листов, в том числе 10 листов с рисунками									

Приложение А.3 – UML – диаграмма классовклиентской части



Приложение А.4 – UML – диаграмма классов серверной части

[illegible]

Приложение Б

(обязательное)

Web-приложение с базой данных для участников игровых турниров

Листинг программы

					ПГУ 1.090301.25.001 ПЗ	Лист
						53
Изм.	Лист	№ докум.	Подпись	Дата		

Файл«app.py»

```
from flask import Flask, render_template, redirect, url_for,
request, flash
from flask_sqlalchemy import SQLAlchemy
from flask_login import LoginManager, UserMixin, login_user,
login_required, logout_user, current_user
from flask_wtf import FlaskForm
from wtforms import (
    StringField,
    PasswordField,
    SubmitField,
    DateField,
    FileField,
    IntegerField
)
from wtforms.validators import DataRequired, Length, EqualTo
from werkzeug.security import generate_password_hash,
check_password_hash
from wtforms.validators import InputRequired, NumberRange
from datetime import datetime
import os
from werkzeug.utils import secure_filename
from PIL import Image
from wtforms import FileField
app = Flask(__name__)
app.config['SECRET_KEY'] = 'your-secret-key-here'
app.config['SQLALCHEMY_DATABASE_URI'] =
'sqlite:///tournament.db'
app.config['UPLOAD_FOLDER'] = 'static/uploads/avatars'
app.config['ALLOWED_EXTENSIONS'] = {'png', 'jpg', 'jpeg',
'gif'}
app.config['MAX_CONTENT_LENGTH'] = 2 * 1024 * 1024 # 2MB

# Создаем папку для загрузок если ее нет
os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True)
```

					ПГУ 1.090301.25.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		54

```

db = SQLAlchemy(app)
login_manager = LoginManager(app)
login_manager.login_view = 'login'

def allowed_file(filename):
    return '.' in filename and \
        filename.rsplit('.', 1)[1].lower() in
app.config['ALLOWED_EXTENSIONS']

def resize_image(image_path, output_size=(200, 200)):
    img = Image.open(image_path)
    img.thumbnail(output_size)
    img.save(image_path)

# Модели базы данных
class User(UserMixin, db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(50), unique=True,
nullable=False)
    full_name = db.Column(db.String(100), nullable=False)
    country = db.Column(db.String(50), nullable=False)
    rating = db.Column(db.Integer, default=1000)
    password_hash = db.Column(db.String(128))
    is_admin = db.Column(db.Boolean, default=False)
    avatar = db.Column(db.String(100), default='default.jpg')
    victories = db.Column(db.Integer, default=0)
    defeats = db.Column(db.Integer, default=0)
    @property
    def rating(self):
        total = self.victories + self.defeats
        return round((self.victories / total * 100), 2) if
total > 0 else 0
    def set_password(self, password):
        self.password_hash = generate_password_hash(password)

    def check_password(self, password):

```

					ПГУ 1.090301.25.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		55

```

        return check_password_hash(self.password_hash,
password)

class Tournament(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    date = db.Column(db.DateTime, nullable=False)
    location = db.Column(db.String(100), nullable=False)
    status = db.Column(db.String(20), default='open')

class Application(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'),
nullable=False)
    tournament_id = db.Column(db.Integer,
db.ForeignKey('tournament.id'), nullable=False)
    status = db.Column(db.String(20), default='pending')
    user = db.relationship('User', backref='applications')
    tournament = db.relationship('Tournament',
backref='applications')
    user = db.relationship('User', backref='applications')
    tournament = db.relationship('Tournament',
backref='applications')
# Формы
class RegistrationForm(FlaskForm):
    username = StringField('Никнейм', validators=[
        DataRequired(message='Введите никнейм'),
        Length(min=4, max=25)
    ])
    full_name = StringField('Полное имя', validators=[
        DataRequired(message='Введите ФИО')
    ])
    country = StringField('Страна', validators=[
        DataRequired(message='Введите страну')
    ])
    password = PasswordField('Пароль', validators=[

```

```

        DataRequired(message='Введите пароль')
    ])
    confirm_password = PasswordField('Подтверждение пароля',
validators=[
        DataRequired(message='Подтвердите пароль'),
        EqualTo('password', message='Пароли должны совпадать')
    ])
    submit = SubmitField('Зарегистрироваться')

class LoginForm(FlaskForm):
    username = StringField('Никнейм', validators=[
        DataRequired(message='Введите никнейм')
    ])
    password = PasswordField('Пароль', validators=[
        DataRequired(message='Введите пароль')
    ])
    submit = SubmitField('Войти')

class AdminTournamentForm(FlaskForm):
    name = StringField('Название турнира', validators=[
        DataRequired(message='Введите название')
    ])
    date = DateField('Дата', format='%Y-%m-%d', validators=[
        DataRequired(message='Выберите дату')
    ])
    location = StringField('Место проведения', validators=[
        DataRequired(message='Введите место')
    ])
    submit = SubmitField('Создать турнир')

class UpdateProfileForm(FlaskForm):
    avatar = FileField('Аватар профиля')
    submit = SubmitField('Обновить')

class AdminEditUserForm(FlaskForm):
    victories = IntegerField('Победы', validators=[

```

```

        InputRequired(message="Поле обязательно"),
        NumberRange(min=0, message="Не может быть отрицательным")
    ])
    defeats = IntegerField('Поражения', validators=[
        InputRequired(message="Поле обязательно"),
        NumberRange(min=0, message="Не может быть
отрицательным")
    ])
    submit = SubmitField('Обновить')

# Загрузчик пользователя
@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))

# Маршруты
@app.route('/')
def home():
    tournaments =
Tournament.query.order_by(Tournament.date.desc()).limit(3).all()
    return render_template('index.html',
tournaments=tournaments)

@app.route('/register', methods=['GET', 'POST'])
def register():
    form = RegistrationForm()
    if form.validate_on_submit():
        if
User.query.filter_by(username=form.username.data).first():
            flash('Этот никнейм уже занят', 'danger')
    return redirect(url_for('register'))

    user = User(
        username=form.username.data,
        full_name=form.full_name.data,
        country=form.country.data,

```

```

        is_admin=False
    )
    user.set_password(form.password.data)
    db.session.add(user)
    db.session.commit()

    flash('Регистрация прошла успешно! Теперь вы можете
войти', 'success')
    return redirect(url_for('login'))
    return render_template('register.html', form=form)

@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        user =
User.query.filter_by(username=form.username.data).first()
        if user and user.check_password(form.password.data):
            login_user(user)
            next_page = request.args.get('next')
            return redirect(next_page or url_for('profile'))
            flash('Неверные учетные данные', 'danger')
    return render_template('login.html', form=form)

@app.route('/logout')
@login_required
def logout():
    logout_user()
    return redirect(url_for('home'))

@app.route('/profile')
@login_required
def profile():
    applications =
Application.query.filter_by(user_id=current_user.id).all()
    return render_template('profile.html', user=current_user,
applications=applications)

```

```

@app.route('/tournaments')
@login_required
def tournaments():
    tournaments = Tournament.query.all()
    return render_template('tournaments.html',
tournaments=tournaments)

@app.route('/apply/<int:tournament_id>')
@login_required
def apply(tournament_id):
    existing_application = Application.query.filter_by(
        user_id=current_user.id,
        tournament_id=tournament_id
    ).first()

    if not existing_application:
        application = Application(
            user_id=current_user.id,
            tournament_id=tournament_id
        )
        db.session.add(application)
        db.session.commit()
        flash('Заявка подана успешно!', 'success')
    return redirect(url_for('tournaments'))

@app.route('/stats')
@login_required
def stats():
    players = User.query.order_by(User.rating.desc()).all()
    return render_template('stats.html', players=players)

# Админские маршруты
@app.route('/admin/dashboard')
@login_required

```

```

def admin_dashboard():
    if not current_user.is_admin:
        flash('Доступ запрещен', 'danger')
        return redirect(url_for('home'))
    tournaments = Tournament.query.all()
    return render_template('admin_dashboard.html',
tournaments=tournaments)

@app.route('/admin/create-tournament', methods=['GET', 'POST'])
@login_required
def create_tournament():
    if not current_user.is_admin:
        return redirect(url_for('home'))
    form = AdminTournamentForm()
    if form.validate_on_submit():
        tournament = Tournament(
            name=form.name.data,
            date=form.date.data,
            location=form.location.data
        )
        db.session.add(tournament)
        db.session.commit()
        flash('Турнир успешно создан!', 'success')
    return redirect(url_for('admin_dashboard'))
    return render_template('create_tournament.html', form=form)

@app.route('/admin/applications/<int:tournament_id>')
@login_required
def view_applications(tournament_id):
    if not current_user.is_admin:
        return redirect(url_for('home'))
    applications =
Application.query.filter_by(tournament_id=tournament_id).all()
    tournament = Tournament.query.get_or_404(tournament_id)
    return render_template('applications.html',
        applications=applications,

```

```

tournament=tournament)

@app.route('/admin/application/<int:app_id>/<status>')
@login_required
def update_application(app_id, status):
    if not current_user.is_admin:
        return redirect(url_for('home'))
    application = Application.query.get(app_id)
    application.status = status
    db.session.commit()
    flash('Статусзаявкиобновлен', 'success')
    return redirect(url_for('view_applications',
tournament_id=application.tournament_id))

@app.route('/admin/users')
@login_required
def admin_users():
    if not current_user.is_admin:
        flash('Доступ запрещен', 'danger')
        return redirect(url_for('home'))

    search = request.args.get('search', '')
    page = request.args.get('page', 1, type=int)

    query = User.query
    if search:
        query = query.filter(db.or_(
            User.username.ilike(f'%{search}%'),
            User.full_name.ilike(f'%{search}%')
        ))

    users = query.paginate(page=page, per_page=10)
    return render_template('admin_users.html', users=users,
search=search)

```

```

@app.route('/admin/edit-user/<int:user_id>', methods=['GET',
'POST'])
@login_required
def admin_edit_user(user_id):
    if not current_user.is_admin:
        flash('Доступ запрещен', 'danger')
        return redirect(url_for('home'))

    user = User.query.get_or_404(user_id)
    form = AdminEditUserForm(obj=user)

    if form.validate_on_submit():
        form.populate_obj(user)
        db.session.commit()
        flash('Статистика обновлена!', 'success')
    return redirect(url_for('admin_users'))

    return render_template('admin_edit_user.html', form=form,
user=user)

@app.route('/admin/delete-tournament/<int:tournament_id>',
methods=['POST'])
@login_required
def delete_tournament(tournament_id):
    if not current_user.is_admin:
        flash('Доступ запрещен', 'danger')
        return redirect(url_for('home'))

    tournament = Tournament.query.get_or_404(tournament_id)

    # Удаляем все связанные заявки

Application.query.filter_by(tournament_id=tournament_id).delete()

db.session.delete(tournament)
db.session.commit()

```

```

        flash('Турнир успешно удален', 'success')
    return redirect(url_for('admin_dashboard'))

# Просмотр участников турнира
@app.route('/tournament/<int:tournament_id>/participants')
def tournament_participants(tournament_id):
    tournament = Tournament.query.get_or_404(tournament_id)
    participants = Application.query.filter_by(
        tournament_id=tournament_id,
        status='approved'
    ).join(User).all()
    return render_template('participants.html',
                           tournament=tournament,
                           participants=participants)

@app.route('/edit_profile', methods=['GET', 'POST'])
@login_required
def edit_profile():
    form = UpdateProfileForm()
    if form.validate_on_submit():
        if form.avatar.data:
            file = form.avatar.data
            if file and allowed_file(file.filename):
                filename =
secure_filename(f"user_{current_user.id}.{file.filename.split('.')[-
1]}")

                file_path =
os.path.join(app.config['UPLOAD_FOLDER'], filename)
                file.save(file_path)
                resize_image(file_path)
                current_user.avatar = filename
                db.session.commit()
                flash('Профиль обновлен!', 'success')
    return redirect(url_for('public_profile',
username=current_user.username))

```

					ПГУ 1.090301.25.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		64

```

        return render_template('edit_profile.html', form=form)

# Публичный профиль участника
@app.route('/user/<username>')
def public_profile(username):
    user =
User.query.filter_by(username=username).first_or_404()
    participated_tournaments = Application.query.filter_by(
        user_id=user.id,
        status='approved'
    ).join(Tournament).all()
    return render_template('public_profile.html',
                           user=user,
                           tournaments=participated_tournaments)

@app.template_filter('ru_status')
def ru_status_filter(status):
    status_map = {
'pending': 'На рассмотрении',
        'approved': 'Одобрено',
'rejected': 'Отклонено'
    }
    return status_map.get(status, status)

if __name__ == '__main__':
    with app.app_context():
        db.create_all()
        app.run(debug=True)

```

Файл «Profile.jsx»

```

import React, { useContext, useRef, useState } from 'react';
import { UserContext } from '../context/UserContext';
import { FaQuestionCircle, FaPlus } from 'react-icons/fa';
import { useNavigate } from 'react-router-dom';
import './Profile.css';

```

```

export default function Profile() {
  const { user, token, login, updateUser } =
useContext(UserContext);

  const [hovered, setHovered]      = useState(false);
  const [uploading, setUploading] = useState(false);
  const inputRef                    = useRef();
  const navigate                    = useNavigate();

  if (!user) return <div className="container mt-
4">Вы не авторизованы</div>;

  // Обработчик выбора файла
  const onFileChange = async (e) => {
    const file = e.target.files[0];
    if (!file) return;

    const fd = new FormData();
    fd.append('avatar', file);
    setUploading(true);

    const res = await fetch('http://localhost:5000/upload-
avatar', {
      method: 'POST',
      mode: 'cors',                // разрешаем CORS
      credentials: 'include',      // на всякий случай,
если будут куки
      headers: {
        Authorization: `Bearer ${token}`
      },
      body: fd
    });

    setUploading(false);

    if (!res.ok) {
      const err = await res.json().catch(() => ({}));

```

					ПГУ 1.090301.25.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		66

```

        alert(err.error || 'Ошибка загрузки');
        return;
    }

    const data = await res.json();
    // обновляем токен и URL аватарки
    login(data.token);
    const fullUrl = `http://localhost:5000${data.avatar_url}`;
    updateUser({ avatar: fullUrl });
};

const backend = 'http://localhost:5000';
const avatarSrc = user.avatar
    ? (user.avatar.startsWith('http') ? user.avatar :
`${backend}${user.avatar}`)
    : 'https://via.placeholder.com/100';

return (
    <div className="container mt-4">
        <div className="d-flex align-items-center mb-4">
            <div
                className="avatar-container"
                onMouseEnter={() => setHovered(true)}
                onMouseLeave={() => setHovered(false)}
                onClick={() => inputRef.current.click()}
            >
                <img src={avatarSrc} alt="" />

                {(hovered || uploading) && (
                    <>
                        <div className="avatar-overlay">
                            {uploading ? '...' : <FaPlus size={24} />}
                        </div>
                        <div className="avatar-tooltip">
                            {uploading ? 'Загрузка...' : 'Изменить фото
профиля'}

```

```

</div>
    </>
  )}

  <input
    type="file"
    accept=".jpg, .jpeg, .png"
    className="d-none"
    ref={inputRef}
    onChange={onFileChange}
  />
</div>

<div className="ms-4">
  <h4 className="mb-1">{user.username}</h4>
  <p className="text-muted mb-2">
    {{
user:    'Пользователь',
author: 'Автор',
admin:   'Администратор'
}}[user.role]}
  </p>
  {user.role === 'user' && (
    <div className="d-flex align-items-center">
      <button
        className="btn btn-outline-primary btn-sm"
        onClick={() => navigate('/request-author')}
      >
        Стать Автором
      </button>
      <FaQuestionCircle
        title="Нажмите, чтобы подать заявку на статус автора и
получить возможность публиковать свои книги после модерации"
      />
    </div>
  )}

```

```

        </div>
    </div>

    {user.role === 'admin' ? (
        <button
            className="btn btn-primary"
            onClick={() => navigate('/admin')}
        >
            Перейти в панель администратора
        </button>
    ) : (
        <>
            <hr />

            {user.role === 'author' && (
                <>
                    <button
                        className="btn btn-primary mb-3"
                        onClick={() => navigate('/author/books')}
                    >
                        Загрузить свою книгу
                    </button>

                    <ul className="list-group mb-3">
                        <li
                            className="list-group-item bg-light text-
primary"
                            onClick={() => navigate('/author/edit-
books')}
                            style={{ cursor: 'pointer' }}
                        >
                            Редактировать свои книги
                        </li>
                    </ul>
                </>
            )}
        </>
    )}

```

```

        <ul className="list-group">
            {[
                ['Моикниги',    '/my-books'],
                ['Заказы',      '/orders'],
                ['Отложенные',   '/favorites'],
                ['Корзина',      '/cart'],
                ['Настройки',    '/settings']
            ]}.map(([title, url]) => (
                <li key={url} className="list-group-item">
                    <a href={url} className="text-decoration-none">
                        {title}
                    </a>
                </li>
            )))
        </ul>
    </>
    )}
</div>
);
}

```

Файл «admin_dashboard.html»

```

{% extends "base.html" %}

{% block content %}
<div class="container mt-5 animate__animated animate__fadeIn">
<h2 class="mb-4">Панель администратора</h2>

<div class="row mb-4">
<div class="col-md-4">
<div class="card shadow">
<div class="card-body text-center">
<i class="fas fa-users fa-3x text-primary mb-3"></i>
<h5>Управление пользователями</h5>

```

```

        <a href="{{ url_for('admin_users') }}" class="btn btn-primary
mt-2">
        Перейти к списку
    </a>
</div>
</div>
</div>
</div>

<div class="card shadow mb-4">
<div class="card-header py-3">
<h5 class="m-0">Управление турнирами</h5>
</div>
<div class="card-body">
    <a href="{{ url_for('create_tournament') }}" class="btn btn-
primary">
        <i class="fas fa-plus-circle me-2"></i>Создать турнир
    </a>
</div>
</div>

<div class="card shadow">
<div class="card-header py-3">
<h5 class="m-0">Активные турниры</h5>
</div>
<div class="card-body">
<div class="table-responsive">
<table class="table table-hover">
<thead class="table-light">
<tr>
<th>Название</th>
<th>Дата</th>
<th>Место</th>
<th>Действия</th>
</tr>
</thead>

```

```

<tbody>

                                {% for tournament in tournaments %}

<tr>
<td>{{ tournament.name }}</td>
<td>{{ tournament.date.strftime('%d.%m.%Y') }}</td>
<td>{{ tournament.location }}</td>
<td>
<div class="d-flex gap-2">
<a href="{{ url_for('view_applications',
tournament_id=tournament.id) }}"
                                class="btn btn-sm btn-
info">
<i class="fas fa-list me-1"></i>Заявки
</a>
<!-- Добавленная кнопка удаления -->
<form action="{{ url_for('delete_tournament',
tournament_id=tournament.id) }}" method="POST">
<button type="submit"
                                class="btn btn-
sm btn-danger"
onclick="returnconfirm('Вы уверены, что хотите удалить этот турнир?
Все связанные заявки также будут удалены.');">
<i class="fas fa-trash-alt me-1"></i>Удалить
</button>
</form>
</div>
</td>
</tr>

                                {% endfor %}

</tbody>
</table>
</div>
</div>
</div>
</div>

```

```
{% endblock %}
```

Файл «admin_edit_user.html»

```
{% extends "base.html" %}
```

```
{% block content %}
```

```
<div class="container mt-5">
```

```
<h2 class="mb-4">Редактирование статистики: {{ user.username }}</h2>
```

```
<div class="card shadow">
```

```
<div class="card-body">
```

```
<form method="POST">
```

```
    {{ form.hidden_tag() }}
```

```
<div class="row">
```

```
<div class="col-md-6">
```

```
<div class="mb-3">
```

```
    {{
```

```
form.victories.label(class="form-label") }}
```

```
    {{ form.victories(class="form-control", min="0") }}
```

```
    {% for error in form.victories.errors %}
```

```
<div class="text-danger">{{ error }}</div>
```

```
    {% endfor %}
```

```
</div>
```

```
<div class="mb-3">
```

```
    {{ form.defeats.label(class="form-label") }}
```

```
    {{ form.defeats(class="form-control", min="0") }}
```

```
    {% for error in form.defeats.errors %}
```

```
<div class="text-danger">{{ error }}</div>
```

```
    {% endfor %}
```

					ПГУ 1.090301.25.001 ПЗ	Лист
						73
Изм.	Лист	№ докум.	Подпись	Дата		

```

</div>
</div>
</div>
<div class="d-grid">
    {{ form.submit(class="btn btn-primary") }}
</div>
</form>
</div>
</div>
</div>
{% endblock %}

```

Файл «admin_users.html»

```

{% extends "base.html" %}

{% block content %}
<divclass="containermt-5">
<h2 class="mb-4">Управление пользователями</h2>

<div class="card shadow">
<div class="card-body">
<div class="table-responsive">
<div class="mb-4">
<form method="GET" action="{{ url_for('admin_users') }}">
<div class="input-group">
<input type="text"

class="form-control"
placeholder="Поиск по имени или
никнейму"

name="search"
value="{{
request.args.get('search', '') }}">
<button class="btn btn-primary" type="submit">
<i class="fas fa-search"></i>
</button>
</div>

```

```

</form>
</div>
<table class="table table-hover">
<thead class="table-light">
<tr>
<th>Никнейм</th>
<th>Полное имя</th>
<th>Победы</th>
<th>Поражения</th>
<th>Действия</th>
</tr>
</thead>
<tbody>
                                {% for user in users %}

<tr>
<td>{{ user.username }}</td>
<td>{{ user.full_name }}</td>
<td>{{ user.victories }}</td>
<td>{{ user.defeats }}</td>
<td>
<a href="{{ url_for('admin_edit_user', user_id=user.id) }}"
                                class="btn btn-sm btn-
primary">
<i class="fas fa-edit"></i>
</a>
</td>
</tr>
                                {% endfor %}

</tbody>
</table>
<nav aria-label="Page navigation">
<ul class="pagination justify-content-center">
                                {% if users.has_prev %}

<li class="page-item">
<a class="page-link" href="{{ url_for('admin_users',
page=users.prev_num) }}">Назад</a>

```

```

</li>

{% endif %}

{% for page_num in users.iter_pages()
%}

<li class="page-item {% if page_num == users.page %}active{%
endif %}">

<a class="page-link" href="{{ url_for('admin_users',
page=page_num) }}">{{ page_num }}</a>

</li>

{% endfor %}

{% if users.has_next %}

<li class="page-item">

<a class="page-link" href="{{ url_for('admin_users',
page=users.next_num) }}">Вперед</a>

</li>

{% endif %}

</ul>
</nav>
</div>
</div>
</div>
</div>
{% endblock %}

```

Файл «applications.html»

```

{% extends "base.html" %}

{% block content %}

<div class="container mt-5 animate__animated animate__fadeIn">
<h2 class="mb-4">Заявки на турнир "{{ tournament.name }}"</h2>

<div class="card shadow">
<div class="card-header py-3">
<h5 class="m-0">Список заявок</h5>

```

```

</div>
<div class="card-body">
<div class="table-responsive">
<table class="table table-hover">
<thead class="table-light">
<tr>
<th>Участник</th>
<th>Статус</th>
<th>Действия</th>
</tr>
</thead>
<tbody>

                                {% for application in applications %}

<tr>
<td>{{ application.user.full_name }} ({{
application.user.username }})</td>
<td>

                                {% if application.status ==
'pending' %}
<span class="badge bg-warning">На рассмотрении</span>
                                {% elif application.status ==
'approved' %}
<span class="badge bg-success">Одобрено</span>
                                {% else %}
<span class="badge bg-danger">Отклонено</span>
                                {% endif %}

</td>
<td>
<div class="btn-group">

                                {% if application.status !=
'approved' %}
<a href="{{ url_for('update_application',
app_id=application.id, status='approved') }}"
                                class="btn btn-sm btn-
success">
<i class="fas fa-check"></i>

```

```

</a>

                                {% endif %}
                                {% if application.status !=
'rejected' %}
        <a href="{{ url_for('update_application',
app_id=application.id, status='rejected') }}"
                                class="btn btn-sm btn-
danger">
        <i class="fas fa-times"></i>
</a>

                                {% endif %}

</div>
</td>
</tr>

                                {% endfor %}

</tbody>
</table>
</div>
</div>
</div>
</div>
{% endblock %}

```

Файл «base.html»

```

<!DOCTYPE html>
<html lang="ru">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<title>Турнирный менеджер</title>
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstra
p.min.css" rel="stylesheet">

```

```

<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css">

<link rel="stylesheet" href="{{ url_for('static',
filename='styles.css') }}">

<script src="https://kit.fontawesome.com/ваш-код-скрипта.js"
crossorigin="anonymous"></script>

</head>
<body>
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
<div class="container">
<a class="navbar-brand" href="/">ТурнирПро</a>
<button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target="#navbarNav">
<span class="navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse" id="navbarNav">
<div class="navbar-nav ms-auto">
                {% if current_user.is_authenticated %}
                {% if current_user.is_admin %}
                <a class="nav-item nav-link" href="{{
url_for('admin_dashboard') }}">
<i class="fas fa-tachometer-alt me-2"></i>Панель админа
</a>

                {% endif %}
                <a class="nav-item nav-link" href="{{ url_for('profile') }}">
<i class="fas fa-user me-2"></i>Профиль
</a>
                <a class="nav-item nav-link" href="{{ url_for('tournaments')
}}">
<i class="fas fa-trophy me-2"></i>Турниры
</a>
                <a class="nav-item nav-link" href="{{ url_for('logout') }}">
<i class="fas fa-sign-out-alt me-2"></i>Выйти
</a>

                {% else %}

```

```

<a class="nav-item nav-link" href="{{ url_for('login') }}">
<i class="fas fa-sign-in-alt me-2"></i>Войти
</a>
<a class="nav-item nav-link" href="{{ url_for('register') }}">
<i class="fas fa-user-plus me-2"></i>Регистрация
</a>

        {% endif %}

</div>
</div>
</div>
</nav>

<div class="container mt-4">
    {% with messages =
get_flashed_messages(with_categories=true) %}
        {% if messages %}
            {% for category, message in messages %}
                <div class="alert alert-{{ category }} alert-dismissible fade
show animate__animated animate__bounceIn">
                    {{ message }}
                <button type="button" class="btn-close" data-bs-
dismiss="alert"></button>
            </div>
            {% endfor %}
        {% endif %}
    {% endwith %}

    {% block content %}{% endblock %}
</div>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.
bundle.min.js"></script>
</body>
</html>

```

Файл «create_tournament.html»

```
{% extends "base.html" %}

{% block content %}
<div class="container mt-5 animate__animated animate__fadeIn">
<div class="row justify-content-center">
<div class="col-md-8">
<div class="card shadow">
<div class="card-header bg-primary text-white">
<h3 class="mb-0 text-center">Создание нового турнира</h3>
</div>
<div class="card-body">
<form method="POST">

                                {{ form.hidden_tag() }}

<div class="mb-4">

                                {{ form.name.label(class="form-
label") }}

                                {{ form.name(class="form-control")
}}

                                {% for error in form.name.errors %}
<div class="text-danger">{{ error }}</div>
                                {% endfor %}

</div>
<div class="mb-4">

                                {{ form.date.label(class="form-
label") }}

                                {{ form.date(class="form-control")
}}

                                {% for error in form.date.errors %}
<div class="text-danger">{{ error }}</div>
                                {% endfor %}

</div>
<div class="mb-4">

                                {{ form.location.label(class="form-
label") }}
```

```

        {{ form.location(class="form-
control") }}

        {% for error in
form.location.errors %}
            <div class="text-danger">{{ error }}</div>

            {% endfor %}

        </div>
        <div class="d-grid">

            {{ form.submit(class="btn btn-
primary btn-lg") }}

        </div>
    </form>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
{% endblock %}

```

Файл «edit_profile.html»

```

{% extends "base.html" %}

{% block content %}
<div class="container mt-5">
<div class="row justify-content-center">
<div class="col-md-6">
<div class="card shadow">
<div class="card-header bg-primary text-white">
<h3 class="mb-0 text-center">Редактирование профиля</h3>
</div>
<div class="card-body">
<form method="POST" enctype="multipart/form-data">
        {{ form.hidden_tag() }}

<div class="mb-4 text-center">
    
{{ form.avatar(class="form-
control") }}
</div>
<div class="d-grid">
{{ form.submit(class="btn btn-
primary btn-lg") }}
</div>
</form>
</div>
</div>
</div>
</div>
</div>
</div>
{% endblock %}

```

Файл «index.html»

```

{% extends "base.html" %}

{% block content %}
<!-- Hero Section -->
<section class="hero bg-dark text-white py-5">
<div class="container text-center">
<h1 class="display-4 fw-bold mb-4 animate__animated
animate__fadeInDown">
Добро пожаловать на ТурнирПро
</h1>
<p class="lead mb-4 animate__animated animate__fadeInUp
animate__delay-1s">
Платформа для организации и участия в оффлайн турнирах
</p>

{% if not current_user.is_authenticated %}

```

```

        <div class="animate__animated animate__fadeInUp animate__delay-
2s">
            <a href="{{ url_for('register') }}" class="btn btn-primary btn-
lg mx-2">
                Присоединиться сейчас
            </a>
            <a href="{{ url_for('login') }}" class="btn btn-outline-light
btn-lg mx-2">
                Войти
            </a>
        </div>
    {% endif %}
</div>
</section>

<!-- Features Section -->
<section class="py-5">
    <div class="container">
        <h2 class="text-center mb-5">Наши возможности</h2>
        <div class="row g-4">
            <div class="col-md-4">
                <div class="card h-100 shadow animate__animated
animate__fadeIn">
                    <div class="card-body text-center">
                        <i class="fas fa-trophy fa-3x text-primary mb-3"></i>
                        <h3>Турниры</h3>
                        <p>Участвуйте в регулярных оффлайн-соревнованиях с
профессиональной организацией</p>
                    </div>
                </div>
            </div>
            <div class="col-md-4">
                <div class="card h-100 shadow animate__animated animate__fadeIn
animate__delay-1s">
                    <div class="card-body text-center">
                        <i class="fas fa-chart-line fa-3x text-primary mb-3"></i>

```

```

<h3>Статистика</h3>
<p>Отслеживайте свой прогресс и анализируйте результаты</p>
</div>
</div>
</div>
<div class="col-md-4">
  <div class="card h-100 shadow animate__animated animate__fadeIn
animate__delay-2s">
    <div class="card-body text-center">
      <i class="fas fa-users fa-3x text-primary mb-3"></i>
      <h3>Сообщество</h3>
      <p>Общайтесь с другими игроками и находите новых соперников</p>
    </div>
  </div>
</div>
</div>
</div>
</div>
</section>

<!-- Latest Tournaments -->
{% if tournaments %}
<section class="bg-light py-5">
  <div class="container">
    <h2 class="text-center mb-5">Ближайшие турниры</h2>
    <div class="row g-4">
      {% for tournament in tournaments %}
        <div class="col-md-4">
          <div class="card h-100 shadow">
            <div class="card-header bg-primary text-white">
              <h5>{{ tournament.name }}</h5>
            </div>
            <div class="card-body">
              <p class="card-text">
                <i class="fas fa-calendar-alt me-2"></i>
                {{
tournament.date.strftime('%d.%m.%Y') }}

```

```

</p>
<p class="card-text">
<i class="fas fa-map-marker-alt me-2"></i>
{{ tournament.location }}

</p>

{% if current_user.is_authenticated %}
<a href="{{ url_for('apply', tournament_id=tournament.id) }}"
class="btn btn-primary">
    Подать заявку

</a>

{% endif %}

</div>
</div>
</div>

{% endfor %}

</div>
</div>
</section>
{% endif %}

<!-- CTA Section -->
<section class="py-5 bg-dark text-white">
<div class="container text-center">
<h2 class="mb-4">Готовы начать?</h2>
<pclass="leadmb-4">Присоединяйтесь к нашему сообществу прямо
сейчас</p>
<div class="d-flex justify-content-center gap-3">
    {% if current_user.is_authenticated %}
    <a href="{{ url_for('tournaments') }}" class="btn btn-primary
btn-lg">
        Посмотреть турниры

    </a>

    {% else %}
    <a href="{{ url_for('register') }}" class="btn btn-primary btn-
lg">
        Зарегистрироваться

```

```

        </a>
        <a href="{{ url_for('login') }}" class="btn btn-outline-light
btn-lg">
            Войти
        </a>
    {% endif %}
</div>
</div>
</section>
{% endblock %}

```

Файл «login.html»

```

{% extends "base.html" %}

{% block content %}
<div class="container mt-5 animate__animated animate__fadeIn">
<div class="row justify-content-center">
<div class="col-md-6">
<div class="card shadow">
<div class="card-header bg-primary text-white">
<h3 class="mb-0 text-center">Вход в систему</h3>
</div>
<div class="card-body">
<form method="POST">
            {{ form.hidden_tag() }}

<div class="mb-4">
                {{ form.username.label(class="form-
label") }}
                {{ form.username(class="form-
control") }}
                {% for error in
form.username.errors %}
<div class="text-danger">{{ error }}</div>
                {% endfor %}
            </div>
<div class="mb-4">

```

```

{{ form.password.label(class="form-
label") }}

{{ form.password(class="form-
control") }}

{% for error in
form.password.errors %}
    <div class="text-danger">{{ error }}</div>
{% endfor %}

</div>
<div class="d-grid">

    {{ form.submit(class="btn btn-
primary btn-lg") }}

</div>
</form>
</div>
<div class="card-footer text-center">
    Нет аккаунта? <a href="{{
url_for('register') }}">Зарегистрироваться</a>
</div>
</div>
</div>
</div>
</div>
</div>
{% endblock %}

```

Файл «participants.html»

```

{% extends "base.html" %}

{% block content %}
<div class="container mt-5 animate__animated animate__fadeIn">
<h2 class="mb-4">Участники турнира "{{ tournament.name }}"</h2>

<div class="card shadow">
<div class="card-header">
<h5 class="mb-0">Список участников</h5>
</div>

```

```

<div class="card-body">
<div class="row row-cols-1 row-cols-md-3 g-4">
    {% for participant in participants %}
<div class="col">
<div class="card h-100 shadow-sm">
<div class="card-body text-center">
    

    <h5 class="card-title">
    <a href="{{ url_for('public_profile',
username=participant.user.username) }}"
                                class="text-decoration-
none">

                                {{
participant.user.full_name }}
    </a>
    </h5>
    <p class="text-muted mb-1">@{{ participant.user.username }}</p>
    <p class="mb-0">
    <i class="fas fa-map-marker-alt me-2"></i>
                                {{ participant.user.country }}

    </p>
    </div>
    <div class="card-footer bg-transparent">
    <small class="text-muted">
                                Рейтинг: {{
participant.user.rating }}
    </small>
    </div>
    </div>
    </div>
    </div>
    {% endfor %}

```

```

</div>
</div>
</div>
</div>
{% endblock %}

```

Файл «profile.html»

```

{% extends "base.html" %}

{% block content %}
<div class="container mt-5 animate__animated animate__fadeIn">
<div class="row">
<div class="col-md-4">
<div class="card shadow profile-card">
<div class="card-header bg-primary text-white position-
relative">
<h3 class="mb-0">Мой профиль</h3>
{% if current_user.id == user.id %}
<a href="{{ url_for('edit_profile') }}"
class="btn btn-light btn-sm position-absolute
top-50 end-0 translate-middle-y me-3">
<i class="fas fa-edit me-2"></i>Редактировать
</a>
{% endif %}
</div>
<div class="card-body text-center">

<h3 class="card-title mb-4">{{ user.full_name }}</h3>
<p class="text-muted mb-1">@{{ user.username }}</p>

```

```

<p class="mb-3">
<i class="fas fa-map-marker-alt me-2"></i>{{ user.country }}
</p>
<div class="d-flex justify-content-center">
<div class="rating-box bg-primary text-white p-3 rounded">
<h4 class="mb-0">{{ user.rating }}</h4>
<small>Рейтинг</small>
</div>
</div>
</div>
</div>
</div>
</div>

<div class="col-md-8">
<div class="card shadow">
<div class="card-header">
<h5 class="mb-0">Мои заявки</h5>
</div>
<div class="card-body">
                {% if applications %}
<div class="list-group">
                                {% for app in applications %}
<div class="list-group-item">
<div class="d-flex justify-content-between">
<div>
<h6>{{ app.tournament.name }}</h6>
<small class="text-muted">
                                {{
app.tournament.date.strftime('%d.%m.%Y') }} |
                                {{
app.tournament.location }}
</small>
</div>
<span class="badge bg-{% if app.status == 'approved' %}success
                                {% elif app.status ==
'rejected' %}danger

```

```

                                                                    {% else %}warning{%
endif %}">

                                                                    {{ app.status|ru_status
}}

    </span>
  </div>
</div>

                                                                    {% endfor %}

</div>

                                                                    {% else %}
<p class="text-muted text-center">Нет активных заявок</p>
                                                                    {% endif %}

</div>
</div>
</div>
</div>
</div>
{% endblock %}

```

Файл «public_profile.html»

```

{% extends "base.html" %}

{% block content %}
<div class="container mt-5 animate__animated animate__fadeIn">
<div class="row">
<div class="col-md-4">
<div class="card shadow profile-card">
<div class="card-body text-center">
<h3 class="card-title mb-4">{{ user.full_name }}</h3>


<p class="text-muted mb-1">@{{ user.username }}</p>

```

```

<p class="mb-3">
<i class="fas fa-map-marker-alt me-2"></i>{{ user.country }}
</p>
<div class="d-flex justify-content-center">
<div class="rating-box bg-primary text-white p-3 rounded">
<h4 class="mb-0">{{ user.rating }}</h4>
<small>Рейтинг</small>
</div>
</div>
</div>
</div>
</div>

<div class="col-md-8">
<div class="card shadow">
<div class="card-header">
<h5 class="mb-0">Участие в турнирах</h5>
</div>
<div class="card-body">
                {% if tournaments %}
<div class="list-group">
                                {% for app in tournaments %}
<div class="list-group-item">
<div class="d-flex justify-content-between align-items-center">
<div>
<h6>{{ app.tournament.name }}</h6>
<small class="text-muted">
                                {{
app.tournament.date.strftime('%d.%m.%Y') }} |
                                {{
app.tournament.location }}
                                </small>
                                </div>
                                <a href="{{ url_for('tournament_participants',
tournament_id=app.tournament.id) }}"

```

```

class="btn btn-sm btn-
outline-primary">
    Участники
</a>
</div>
</div>

{% endfor %}

</div>

{% else %}
<pclass="text-muted text-center">Пока не участвовал в
турнирах</p>

{% endif %}

</div>
</div>
</div>
</div>
</div>
{% endblock %}

```

Файл «register.html»

```

{% extends "base.html" %}

{% block content %}
<div class="container mt-5 animate__animated animate__fadeIn">
<div class="row justify-content-center">
<div class="col-md-6">
<div class="card shadow">
<div class="card-header bg-primary text-white">
<h3 class="mb-0 text-center">Регистрация участника</h3>
</div>
<div class="card-body">
<form method="POST">

{{ form.hidden_tag() }}

<div class="mb-3">

{{ form.username.label(class="form-
label") }}

```

```

{{ form.username(class="form-
control") }}

{% for error in
form.username.errors %}
    <div class="text-danger">{{ error }}</div>
    {% endfor %}

</div>
<div class="mb-3">

    {{
form.full_name.label(class="form-label") }}

    {{ form.full_name(class="form-
control") }}

    {% for error in
form.full_name.errors %}
        <div class="text-danger">{{ error }}</div>
        {% endfor %}

    </div>
    <div class="mb-3">

        {{ form.country.label(class="form-
label") }}

        {{ form.country(class="form-
control") }}

        {% for error in form.country.errors
%}
            <div class="text-danger">{{ error }}</div>
            {% endfor %}

        </div>
        <div class="mb-3">

            {{ form.password.label(class="form-
label") }}

            {{ form.password(class="form-
control") }}

            {% for error in
form.password.errors %}
                <div class="text-danger">{{ error }}</div>
                {% endfor %}

```

```

</div>
<div class="mb-4">
    {{
form.confirm_password.label(class="form-label") }}
    {{
form.confirm_password(class="form-control") }}
    {% for error in
form.confirm_password.errors %}
    <div class="text-danger">{{ error }}</div>
    {% endfor %}

</div>
<div class="d-grid">
    {{ form.submit(class="btn btn-
primary btn-lg") }}
</div>
</form>
</div>
<div class="card-footer text-center">
    Уже зарегистрированы? <a href="{{
url_for('login') }}">Войти</a>
</div>
</div>
</div>
</div>
</div>
</div>
{% endblock %}

```

Файл «stats.html»

```

{% extends "base.html" %}

{% block content %}
<div class="container mt-5 animate__animated animate__fadeIn">
<h2 class="mb-4">Рейтинг игроков</h2>

<div class="card shadow">
<div class="card-body">

```

```

<div class="table-responsive">
<table class="table table-hover">
<thead class="table-light">
<tr>
<th>#</th>
<th>Игрок</th>
<th>Страна</th>
<th>Рейтинг</th>
</tr>
</thead>
<tbody>

        {% for player in players %}

<tr>
<td>{{ loop.index }}</td>
<td>
<strong>{{ player.full_name }}</strong>
<br><small class="text-muted">@{{ player.username }}</small>
</td>
<td>{{ player.country }}</td>
<td class="text-primary">{{ player.rating }}</td>
</tr>

        {% endfor %}

</tbody>
</table>
</div>
</div>
</div>
</div>
{% endblock %}

```

Файл «tournaments.html»

```

{% extends "base.html" %}

{% block content %}
<div class="container mt-5 animate__animated animate__fadeIn">
<h2 class="mb-4">Доступные турниры</h2>

```

```

<div class="row">
    {% for tournament in tournaments %}
    <div class="col-md-6 mb-4">
    <div class="card shadow h-100">
    <div class="card-header bg-primary text-white">
    <h5 class="mb-0">{{ tournament.name }}</h5>
    </div>
    <div class="card-body">
    <div class="row">
    <div class="col-8">
    <p class="mb-1">
    <i class="fas fa-calendar-alt me-2"></i>
        {{
tournament.date.strftime('%d.%m.%Y') }}
    </p>
    <p class="mb-0">
    <i class="fas fa-map-marker-alt me-2"></i>
        {{ tournament.location }}
    </p>
    </div>
    <div class="col-4 text-end">
    <a href="{{ url_for('apply', tournament_id=tournament.id) }}"
        class="btn btn-primary">
            Подать заявку
    </a>
    <a href="{{ url_for('tournament_participants',
tournament_id=tournament.id) }}"
        class="btn btn-outline-secondary
mt-2">
        Участники
    </a>
    </div>
    </div>
    </div>
    </div>
    </div>
    </div>

```

```
</div>
        {% endfor %}
</div>
</div>
{% endblock %}
```

					ПГУ 1.090301.25.001 ПЗ	Лист
						99
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение С

(обязательное)

Web-приложение с базой данных для участников игровых турниров

Презентация

					ПГУ 1.090301.25.001 ПЗ	Лист
						100
Изм.	Лист	№ докум.	Подпись	Дата		

Тема работы : Web-приложения с базой данных для участников игровых турниров

Автор работы:
Студент группы 21ВВП2 Большаков Е.К.

Научный руководитель:
к.т.н. доцент кафедры ВТ Пензенский государственный университет
Слепцов Н.В.

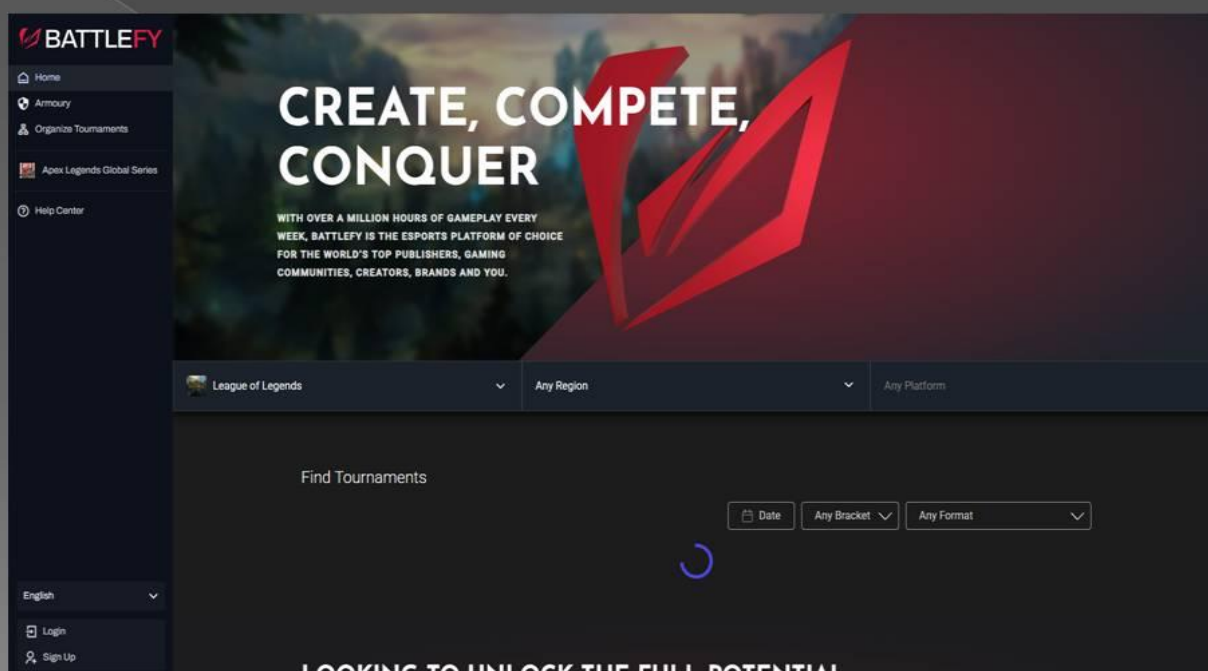
Цель проекта

- Создать удобное и надёжное веб-приложение для управления спортивными и игровыми турнирами.
- Автоматизировать процесс подачи заявок, проведения соревнований и управления участниками.
- Предоставить администраторам гибкий интерфейс для контроля, а участникам — доступный функционал участия.

Функциональные возможности

- Регистрация и авторизация пользователей.
- Просмотр доступных турниров и подача заявок.
- Личный кабинет с отображением статистики участника.
- Панель администратора для управления турнирами и пользователями.
- Подтверждение заявок, редактирование результатов.

Аналоги



Аналоги



Сравнение с аналогами

- ❖ Toornament — профессиональный интерфейс, требует обучения.
- ❖ Battlefy — удобен, но не предоставляет расширенной модерации.
- ❖ Мой проект — упрощённый и интуитивный, подходит для локальных сообществ и небольших клубов.

					ПГУ 1.090301.25.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		103

Диаграммы и модели

- ❶ Диаграмма вариантов использования (Use-case).
- ❷ Диаграмма классов (Class diagram).
- ❸ Структура базы данных (ER-диаграмма).

Диаграмма прав администратора

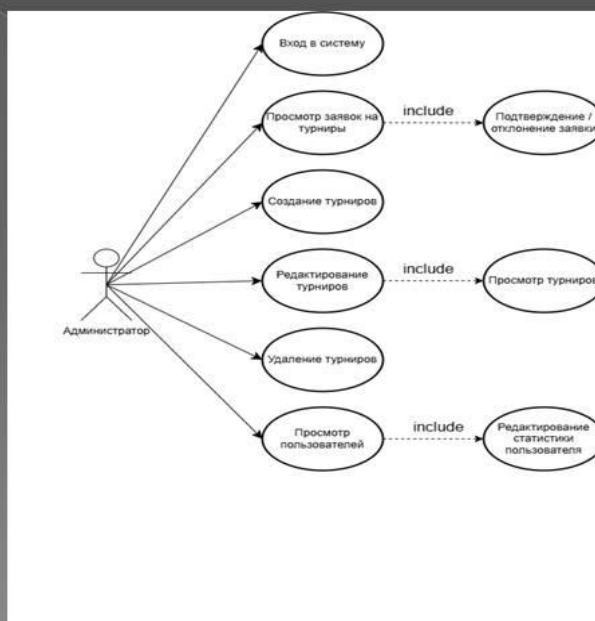
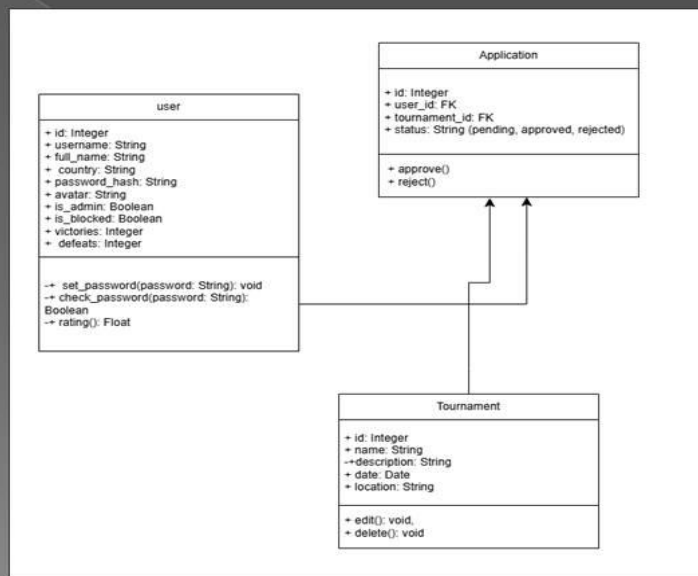


Диаграмма классов пользователей



Заключение

- ❶ Разработанное приложение охватывает полный цикл управления турнирами.
- ❷ Обеспечивает удобство как для участников, так и для администраторов.
- ❸ Проект может быть дополнен аналитикой, интеграцией оплаты и международной поддержкой.
- ❹ Система может быть внедрена в клубы, школы, сообщества и даже в профессиональные соревнования.

Спасибо за внимание

					ПГУ 1.090301.25.001 ПЗ	Лист
						106
Изм.	Лист	№ докум.	Подпись	Дата		