МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ КАЗАХСТАН

Казахский национальный исследовательский технический университет имени К.И.Сатпаева

Институт цифровых технологий и профессионального развития

Кафедра «Программная инженерия»

Кенжегулов Мадияр Газизжанович

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к дипломному проекту

Образовательная программа: 6B06102 Computer Science

Алматы 2025

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ КАЗАХСТАН

Казахский национальный исследовательский технический университет имени К.И.Сатпаева

Институт цифровых технологий и профессионального развития

Кафедра «Программная инженерия»

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой ПИ
канд. тех. наук, ассоц.профессор
_____Ф.Н. Абдолдина
«____» _____2025 г

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к дипломному проекту

На тему: «Web-платформа для проведения турниров по настольному теннису»

Образовательная программа: 6B06102 Computer Science

Выполнил Кенжегулов М.Г.

Рецензент
Доктор PhD
_____ М.Е. Мансурова
"____"_____ 2025 г.

Научный руководитель
Магистр, старший преподаватель
_____ Р.Р. Абдикаров
"____"_____ 2025 г.

Алматы 2025

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ КАЗАХСТАН

Казахский национальный исследовательский технический университет имени К.И.Сатпаева

Институт цифровых технологий и профессионального развития

Кафедра «Программная инженерия»

УТВЕРЖДАЮ
Заведующий кафедрой ПИ
канд. тех. наук, ассоц.профессор
_____Ф.Н. Абдолдина
«____» _____2025 г.

ЗАДАНИЕ
на выполнение дипломного проекта

Обучающемуся: Кенжегулову Мадияру Газизжановичу
Тема: Web-платформа для проведения турниров по настольному теннису
Утверждена приказом проректора по академической работе:
№ 1804-ДО_____

	от «25» ноября 2024 г.
Срок сдачи законченного проекта	*«____» _____ 2025 г.*

Исходные данные к дипломному проекту:
А) Анализ предметной области, анализ аналогичных проектов;
Б) Разработка технического задания;
В) Разработка архитектуры ПО;
Г) Проектирование и разработка понятного и интерактивного дизайна;
Д) Разработка и тестирование ПО;
Перечень подлежащих разработке в дипломном проекте вопросов: (с точным указанием обязательных чертежей): представлены ____ слайда презентации.
Рекомендуемая основная литература: из ___ наименований.

Алматы 2025

ГРАФИК

подготовки дипломного проекта

Наименование разделов, перечень разрабатываемых вопросов	**Сроки представления научному руководителю и консультантам**	**Примечание**
1. Анализ предметной области, разработка технического задания	01.12.2024	Выполнено
2. Выбор технологий для разработки	05.12.2024	Выполнено
3. Разработка функционала системы	04.01.2025	Выполнено
4. Тестирование и оптимизация	09.02.2025	Выполнено
5. Написание пояснительной записки к дипломному проекту	29.03.2025	Выполнено

Подписи

консультантов и нормоконтролера на законченный дипломный проект с указанием относящихся к ним разделов проекта

Наименования разделов	**Консультанты, И.О.Ф. (уч. степень, звание)**	**Дата подписания**	**Подпись**	**Оценка**
Программное обеспечение	**Шаханов Ә.Р. преподаватель, магистр**			
Нормоконтролер	**Имаматдинова К.Ф. ст. преподаватель, магистр.**			

Научный руководитель _____ Абдикаров Р.Р.

Задание принял к исполнению обучающийся Кенжегулов М. Г.

Дата «___» _____2025 г.

Алматы 2025

АҢДАТПА

Бұл дипломдық жобаның мақсаты – Қазақстандағы әуесқой үстел теннисі турнирлерін ұйымдастыруды жеңілдететін веб-платформа әзірлеу. Платформа клуб менеджерлеріне турнирлер құруға, олардың сипаттамаларын өзгертуге және турнирлік торларды басқаруға мүмкіндік береді. Пайдаланушылар тіркеліп, турнирлерге қатысуға өтінім беріп, төлем чектерін тіркей алады.

Жүйеде әртүрлі рөлдерге сәйкес функциялар қарастырылған: қарапайым пайдаланушылар, клуб менеджерлері және әкімшілер. Турнир нәтижелері негізінде ойыншылардың рейтингі ELO жүйесі бойынша есептеледі. Сондай-ақ клубтар картасы мен әр қатысушыға арналған жеке кабинет қарастырылған.

Платформа заманауи технологиялар негізінде әзірленіп, сенімді серверлік және инфрақұрылымдық шешімдермен қамтамасыз етілген.

АННОТАЦИЯ

Цель данного дипломного проекта — разработка веб-платформы, упрощающей проведение любительских турниров по настольному теннису в Казахстане. Система позволяет менеджерам клубов создавать турниры, управлять их параметрами, редактировать турнирные сетки и отслеживать ход соревнований. Участники могут регистрироваться, подавать заявки на участие и прикреплять чеки об оплате.

Платформа реализует ролевую модель, в рамках которой различаются права пользователей, менеджеров и администраторов. Игрокам начисляются рейтинговые очки по системе ELO в зависимости от результатов матчей. Также доступны карта клубов и личные кабинеты с историей участий.

Проект использует современные веб-технологии и размещён на VPS с поддержкой облачных сервисов для хранения данных, отправки писем и ускоренной доставки медиафайлов.

ABSTRACT

This diploma project presents the development of a web platform designed to simplify the organization of amateur table tennis tournaments in Kazakhstan. The system enables club managers to create tournaments, configure details, manage brackets, and monitor tournament progress. Players can register, submit participation requests, and upload payment receipts.

The platform features a role-based access model that distinguishes between regular users, club managers, and administrators. Player ratings are updated using the ELO system based on match outcomes. Users can view club locations on an interactive map and access their personal dashboard with tournament history.

Built with modern technologies and deployed on a VPS, the platform integrates cloud services for media storage, email delivery, and fast content distribution.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Настольный теннис в Казахстане становится всё более популярным видом спорта. По данным местных спортивных федераций, число любителей настольного тенниса за последние пять лет выросло на 45%, а количество клубов увеличилось с 50 до более чем 120. Открытие новых клубов, секций и специализированных центров способствует активному вовлечению как молодёжи, так и взрослых. Тем не менее, несмотря на рост интереса и расширение инфраструктуры, проведение любительских турниров сталкивается с рядом организационных проблем.

Согласно опросам среди организаторов, более 75% из них заявляют, что основной сложностью является отсутствие автоматизированной системы для управления турнирами. В большинстве случаев регистрация участников осуществляется через мессенджеры, подтверждение оплаты производится отправкой чеков, а турнирные сетки формируются вручную. Такой подход требует значительных усилий, что снижает эффективность управления соревнованиями и ограничивает их масштаб.

Некоторые клубы пытаются использовать международные решения, такие как RTTF.ru, TTScore или Challonge, однако эти платформы не адаптированы к специфике Казахстана. Они не поддерживают казахский язык, не интегрированы с местными клубами и часто содержат избыточный функционал, сложный для большинства организаторов. В результате около 60% мероприятий остаются полностью неструктурированными, что затрудняет их проведение и участие в них.

Цель данного дипломного проекта — разработка web-платформы, которая автоматизирует ключевые процессы организации любительских турниров по настольному теннису: онлайн-регистрацию, подтверждение оплаты, формирование турнирных сеток, отображение событий на карте и ведение учёта результатов. В системе будет реализована ролевая модель с разграничением доступа для пользователей, менеджеров клубов и администраторов.

Таким образом, предлагаемый проект нацелен на решение актуальной проблемы, препятствующей развитию любительского настольного тенниса в Казахстане, и создание удобного, локализованного инструмента, который соответствует требованиям организаторов и игроков.

1 Анализ предметной области и постановка задачи

1.1 Состояние любительского настольного тенниса в Казахстане

Настольный теннис в Казахстане за последнее десятилетие приобрёл статус одного из наиболее массовых и доступных видов спорта. В период с 2015 по 2025 годы наблюдается устойчивый рост числа как проводимых любительских турниров, так и их участников.

Если в 2015 году по стране проходило всего несколько чемпионатов с участием порядка 300 человек, то к 2025 году проводится более 20 любительских турниров различного уровня, а число участников превышает 2 300 человек в год. Такой рост обеспечен за счёт активного развития школьных и студенческих лиг, поддержки со стороны Федерации настольного тенниса РК и расширения инфраструктуры.

Особенно значимую роль сыграли:

– национальный проект SportFEST Kazakhstan, охвативший более 21 000 школьников;

– регулярные универсиады и студенческие чемпионаты;

– открытие 11 специализированных центров настольного тенниса в 10 городах, включая Алматы, Астану и Караганду;

– проведение открытых городских турниров, спартакиад и праздничных соревнований.

По данным inbusiness.kz, в 2025 году настольным теннисом систематически занимаются более 260 000 человек — что отражает высокий уровень вовлечённости населения в этот вид спорта.

Для наглядности представлены два графика, демонстрирующие положительную динамику:

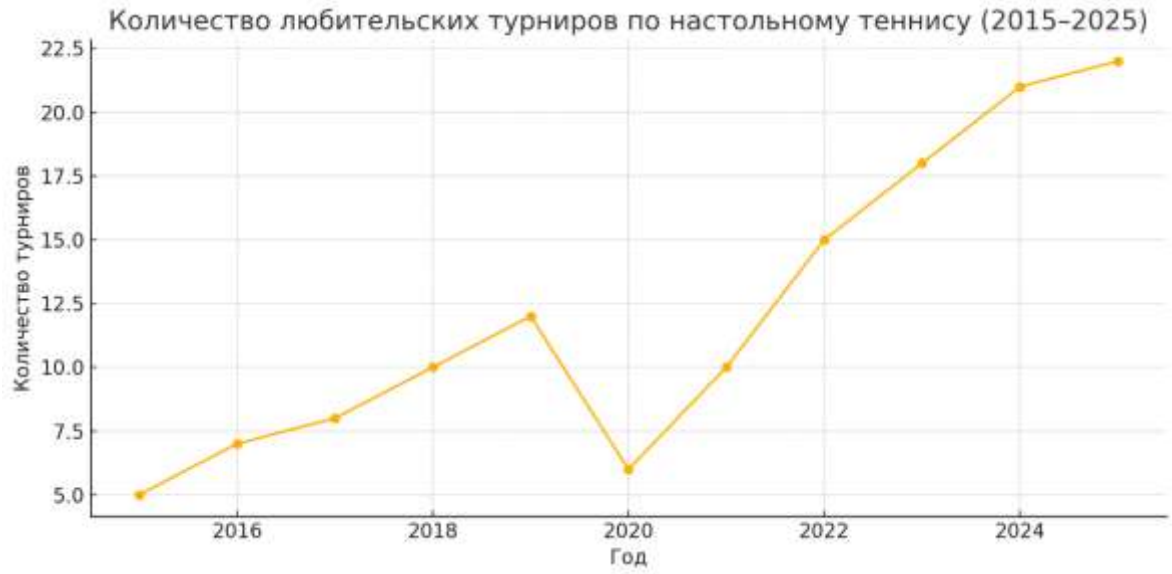

Рисунок 1.1.1 – Динамика количества любительских турниров в Казахстане (2015–2025)

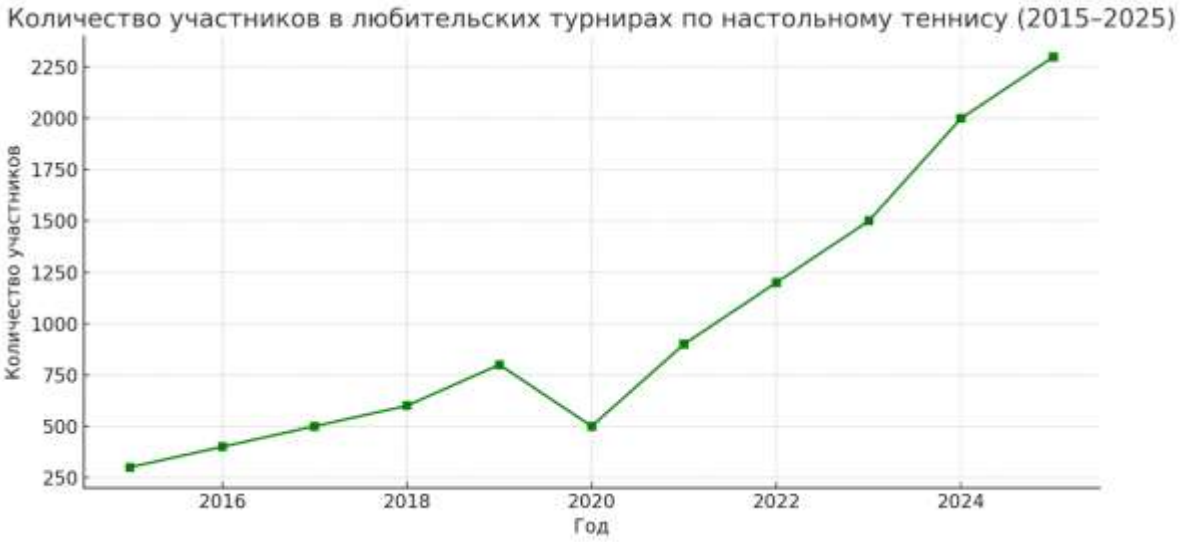

Рисунок 1.1.2 – Рост количества участников любительских турниров (2015–2025)

Таким образом, любительский настольный теннис в Казахстане перешёл от малочисленных мероприятий к устойчивой системе массовых соревнований, охватывающих все возрастные группы и регионы страны.

1.2 Проблемы организации турниров

На практике проведение любительского турнира в Казахстане связано с рядом организационных трудностей, особенно в условиях отсутствия цифровой поддержки:

– Список участников и контроль оплаты ведутся вручную, что увеличивает риск ошибок, дублирования данных и затрудняет быструю проверку статуса регистрации. При регистрации в мессенджерах организаторы вынуждены вручную сопоставлять игроков с платежами, что неэффективно при большом количестве участников.

– Формирование турнирных сеток зачастую выполняется в Excel или на бумаге, без поддержки автоматических алгоритмов жеребьёвки и построения сетки. Это приводит к задержкам, особенно при необходимости пересчётов и корректировок.

– Подсчёт очков и переходов между раундами требует ручной верификации, а значит — дополнительного времени, возможности ошибок и задержек в публикации результатов.

– Расписание матчей, уведомления участников и публикация результатов зависят исключительно от инициативы менеджера турнира, так как не существует централизованной системы рассылки и автоматизации расписания.

– Отсутствие унифицированной системы учёта рейтинга игроков делает невозможным объективную оценку уровня участников и построение справедливой системы посева.

– Невозможность публичного отслеживания турнира ограничивает интерес аудитории и снижает вовлечённость — как со стороны зрителей, так и самих игроков.

Эти проблемы становятся особенно острыми при:

– увеличении числа участников (от 32 и выше);

– проведении многоэтапных турниров (группы + плей-офф);

– необходимости учитывать платное участие.

В совокупности ручной подход не только усложняет работу организаторов, но и сдерживает развитие любительской сцены, снижая привлекательность и масштабируемость мероприятий. Это подчёркивает актуальность внедрения автоматизированных цифровых решений.

1.3 Обзор существующих платформ

1.3.1 Анализ существующих решений

Для оценки перспектив и постановки задач разработки была проведена оценка аналогичных платформ, используемых для организации и учёта турниров по настольному теннису как в Казахстане, так и за рубежом.

RTTF.ru — основная платформа для проведения как профессиональных, так и любительских турниров по настольному теннису в России и странах СНГ. Она активно используется организаторами для регистрации участников, генерации турнирных сеток, ведения рейтингов и отображения результатов в реальном времени. Также доступны функции отображения расписаний, база клубов и тренеров.

Платформа обладает широким функционалом, но её интерфейс выглядит перегруженным: на главной странице одновременно представлены расписание турниров, результаты матчей, рекламные баннеры, списки клубов и новостные блоки. Это может затруднять восприятие и навигацию, особенно для новых пользователей.

Рисунок 1.3.1 – Главная страница платформы RTTF.ru с расписанием турниров, клубами и результатами

TTFRK.kz — официальный сайт Федерации настольного тенниса Республики Казахстан. Платформа выполняет преимущественно информационную функцию: публикует календарь соревнований, новости и результаты матчей. Однако все данные представлены в виде статичных документов (PDF и Excel), без динамической обработки и автоматизации. Например, рейтинги игроков публикуются как ссылки на Excel-файлы, которые нужно скачивать вручную, что неудобно для регулярного анализа и не позволяет интегрировать данные с внешними сервисами. Также отсутствует возможность онлайн-регистрации, формирования турнирных сеток и отслеживания хода матчей в реальном времени.

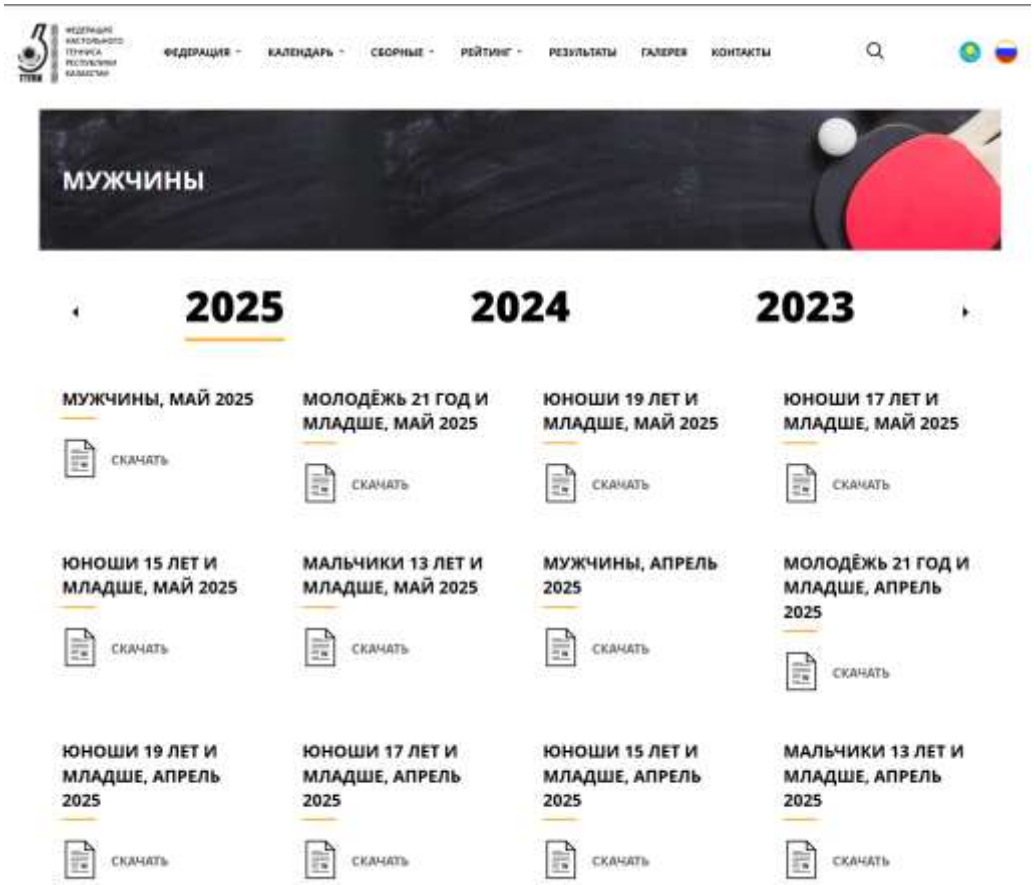

Рисунок 1.3.2 – Страница с рейтингами с разбивкой по годам и месяцам TTFRK.kz

TTScore — международная платформа, ориентированная на гибкую настройку турнирных форматов, автоматизацию жеребьёвки и визуализацию сеток. Платформа поддерживает интеграцию через API, однако требует технической подготовки и не адаптирована под массовое использование на

уровне любительских турниров. В Казахстане сервис практически не применяется.

Challonge — онлайн-сервис с простым интерфейсом и поддержкой различных турнирных форматов: single elimination, double elimination, round-robin и других. Позволяет генерировать сетки и делиться ими через публичные ссылки. Несмотря на удобство, Challonge также не получил широкого распространения в Казахстане из-за своей универсальности и отсутствия локальной специфики.

Таким образом, существующие решения либо не учитывают особенности казахстанской аудитории, либо обладают ограниченным функционалом. Это подтверждает необходимость создания специализированной платформы, ориентированной на местные реалии и потребности организаторов любительских турниров по настольному теннису.

1.3.2 Сравнительная характеристика платформ

Для наглядного сопоставления возможностей существующих решений была проведена сравнительная оценка ключевых характеристик четырёх платформ: RTTF.ru, TTFRK.kz, TTScore и Challonge.

Сильные стороны:

– RTTF.ru — наличие рейтинговой системы, поддержка live-результатов, удобная онлайн-регистрация.

– TTScore — широкие возможности автоматизации, API-интеграция, адаптация под различные форматы турниров.

– Challonge — универсальность, простота создания сеток и поддержки команд, удобный интерфейс.

Слабые стороны:

– TTFRK.kz — отсутствует онлайн-регистрация и автоматизация; результаты выкладываются вручную; платформа служит скорее архивом, чем инструментом.

– RTTF.ru, TTScore, Challonge — ориентированы в первую очередь на другие страны, не интегрированы с клубной системой Казахстана и не учитывают локальные особенности.

Для удобства сравнительные характеристики сведены в таблицу:

Таблица 1 – Сравнение платформ по основным критериям

Платформа	RTTF.ru	TTFRK.kz	TTScore	Challonge
Страна	РФ	РК	Греция	США

Онлайн-регистрация	Да	Нет	Да	Да
Генерация сеток	Да	Нет	Да	Да
Рейтинг игроков	Да	Да	Да	Да
Live-результаты	Да	Нет	Да	Нет
Календарь турниров	Да	Да	Да	Нет
Интеграция через API	Нет	Нет	Да	Да
Мобильные решения	Нет	Нет	Web	Web
Уровень	СНГ	РК	Международный	Международный
Любительские турниры	Да	Да	Да	Да

1.4 Цель разработки информационной системы

На основании анализа текущего состояния в сфере организации любительских турниров по настольному теннису в Казахстане, а также обзора существующих цифровых решений, была выявлена необходимость создания специализированной информационной системы. Существующие подходы характеризуются такими недостатками, как ручная регистрация участников, отсутствие цифрового учёта результатов и рейтингов, а также разрозненное представление информации о клубах и турнирах.

Целью дипломного проекта является разработка web-платформы, предназначенной для автоматизации ключевых процессов, связанных с проведением и администрированием турниров, а также для обеспечения прозрачного и удобного взаимодействия всех участников.

Разрабатываемая система ориентирована на простоту использования, адаптацию под локальные условия, гибкость и масштабируемость. В рамках проекта предполагается реализация следующих задач для разных категорий пользователей:

Для участников:

регистрация и участие в турнирах через личный кабинет;
подача заявок с возможностью прикрепления подтверждающих документов (чеков);

просмотр турнирной сетки и расписания матчей; отслеживание собственного рейтинга и истории участия.

Для менеджеров клубов:

– создание и редактирование информации о клубах;

– организация турниров с настройкой даты проведения, правил, статусов и стоимости участия;

– управление участниками и ввод результатов матчей;

– редактирование и контроль за заполнением турнирной сетки.

Для администраторов:

– верификация новых менеджеров клубов;

– модерация клубов и турниров;

Полный перечень функциональных и нефункциональных требований изложен в Приложении А — техническом задании. В последующих разделах дипломной работы рассмотрены архитектурные и проектные решения, обеспечивающие реализацию поставленных задач.

Таким образом, разрабатываемая платформа призвана повысить качество и прозрачность организации турниров, создать цифровую инфраструктуру для настольного тенниса и улучшить взаимодействие между всеми участниками процесса.

1.5 Обоснование выбора технологий и инструментов

Для реализации информационной системы был выбран современный технологический стек, который обеспечивает модульность, масштабируемость, отказоустойчивость и удобство сопровождения. Выбор инструментов обусловлен как функциональными требованиями к платформе, так и особенностями архитектуры web-приложений.

Фронтенд реализован с использованием Next.js — фреймворка для React с поддержкой серверного рендеринга (SSR), что обеспечивает высокую производительность и улучшенную индексацию в поисковых системах (SEO). Архитектура Next.js упрощает создание масштабируемых и адаптивных интерфейсов, а встроенная маршрутизация и поддержка API-роутов делает его особенно удобным для проектов с ограниченными ресурсами.

Бэкенд построен на NestJS — серверном фреймворке для Node.js, обладающем модульной архитектурой и полной поддержкой TypeScript. NestJS позволяет структурировать код по лучшим практикам, обеспечивая высокую читаемость и надёжность. Он также упрощает реализацию безопасной авторизации, валидации входных данных и взаимодействия с базой данных.

База данных — PostgreSQL, надёжная реляционная СУБД с поддержкой ACID-транзакций, мощным SQL-движком и возможностями масштабирования.

Она оптимально подходит для хранения информации о пользователях, клубах, турнирах и результатах матчей. В качестве ORM используется Prisma, что позволяет описывать структуру данных декларативно и работать с БД через типизированные запросы.

Redis используется как хранилище данных в оперативной памяти, преимущественно для временных сценариев:

– хранения данных до подтверждения регистрации;

– ограничения количества попыток верификации OTP-кодов.

Это позволяет сократить задержки и повысить безопасность при обработке чувствительных операций.

Docker и Docker Compose применяются для контейнеризации всех компонентов системы. Описание инфраструктуры в виде docker-compose.yml обеспечивает воспроизводимость, автоматизированный запуск сервисов (Next.js, NestJS, PostgreSQL, Redis) и упрощает сопровождение проекта.

– Облачная инфраструктура размещена на Yandex Cloud, где используются следующие сервисы:

– Object Storage (S3): для загрузки и хранения изображений, аватаров, квитанций;

– CDN: для ускоренной доставки медиафайлов;

– DNS: для управления доменами;

– Yandex Postbox: для надёжной отправки email-уведомлений (OTP-коды, восстановление пароля).

Выбранная технологическая связка обеспечивает надёжную и производительную работу системы, соответствует современным требованиям к безопасности и расширяемости, а также упрощает интеграцию и сопровождение платформы в условиях ограниченных ресурсов.

2 Проектирование информационной системы

2.1 Проектирование архитектуры ИС

Архитектура разрабатываемой информационной системы основана на клиент-серверной модели. Фронтенд-часть реализуется с использованием Next.js, обеспечивая SSR (Server Side Rendering) и адаптивную верстку. Бэкенд построен на NestJS, что обеспечивает модульность и масштабируемость. Коммуникация между клиентом и сервером осуществляется по REST API.

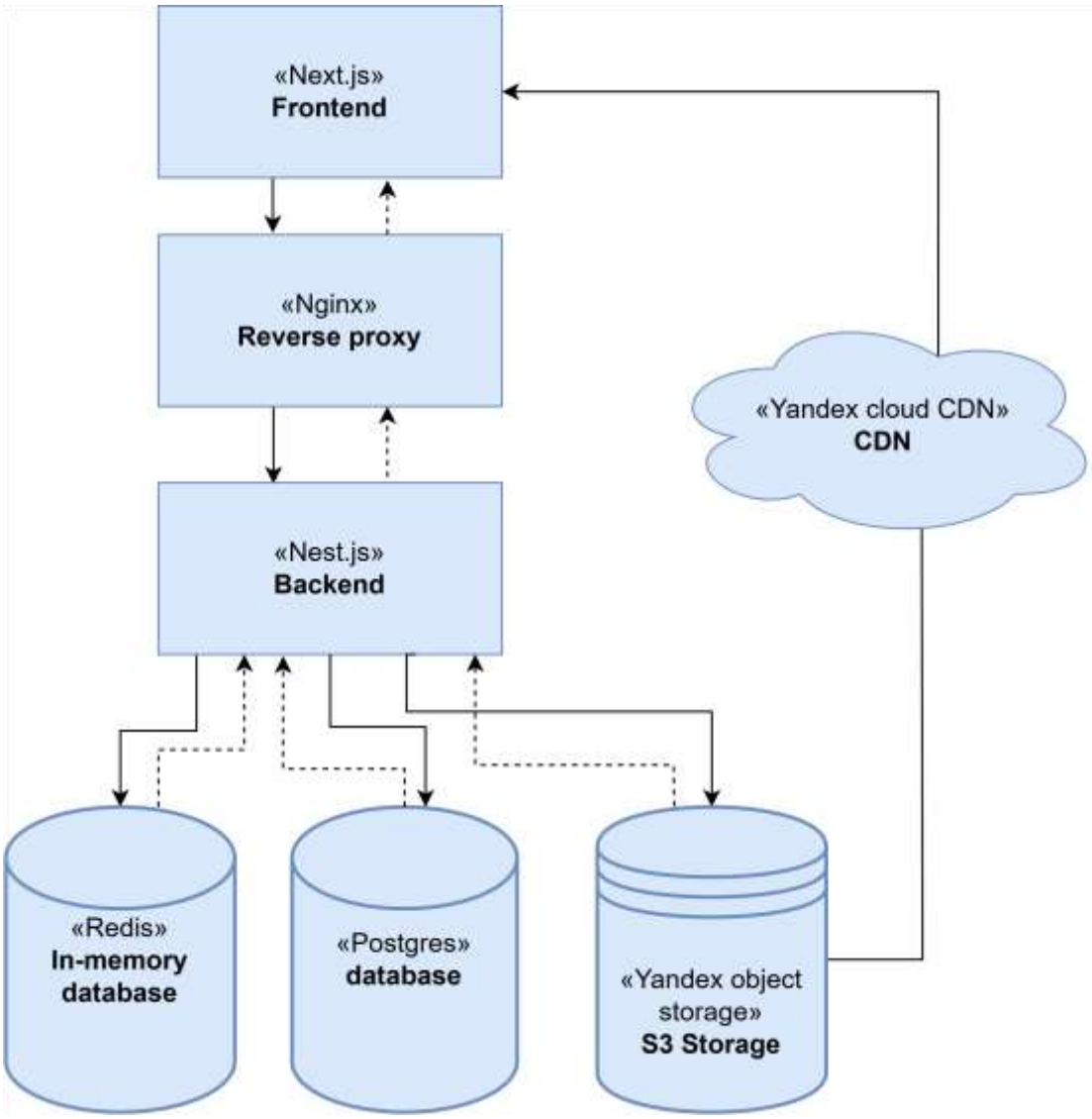

Рисунок 2.1.1 – Архитектура ИС

2.2 Структура базы данных

База данных разработана на основе реляционной модели с использованием системы управления базами данных PostgreSQL. Структура включает ключевые сущности: пользователи, клубы, турниры, участники, матчи и города. В модели реализованы связи типа «один ко многим» и «многие ко многим», что обеспечивает логическую связанность данных, целостность при обновлениях и эффективную организацию запросов. Данная схема позволяет точно отражать реальные процессы участия, регистрации и проведения турниров.

Рисунок 2.2.1 – Диаграмма базы данных

2.3 Разработка бэкенда

2.3.1 Архитектура серверной части проекта

Архитектура серверной части проекта построена на основе модульного подхода NestJS, который позволяет разделить функциональность приложения на изолированные, переиспользуемые и масштабируемые компоненты. Это обеспечивает строгое логическое разделение, упрощает поддержку кода и способствует расширению системы без существенного изменения существующих частей.

В центре архитектуры находится AppModule, который агрегирует остальные модули и отвечает за инициализацию приложения. Каждый функциональный блок реализован в виде отдельного модуля, зарегистрированного в AppModule:

– AuthModule — отвечает за процессы аутентификации и авторизации, интегрируется с JwtModule и PassportModule для обеспечения безопасности;

– UsersModule — управляет данными пользователей и связан с AuthModule, AdminModule и RedisModule;

– ProfileModule, TournamentsModule, ClubsModule, CityModule — реализуют предметно-ориентированные модули, связанные с ключевыми сущностями системы;

– MailerModule — обеспечивает отправку уведомлений и OTP-кодов;

– RedisModule — используется для кэширования временных данных и ограничения частоты запросов (rate limiting);

– ConfigModule — централизованно управляет переменными окружения и конфигурацией;

– PrismaModule — инкапсулирует доступ к базе данных через Prisma ORM;

– S3Module — предоставляет интерфейс для работы с объектным хранилищем (загрузка квитанций, аватаров и т.п.);

– AdminModule — реализует административные функции, включая модерацию и управление доступом.

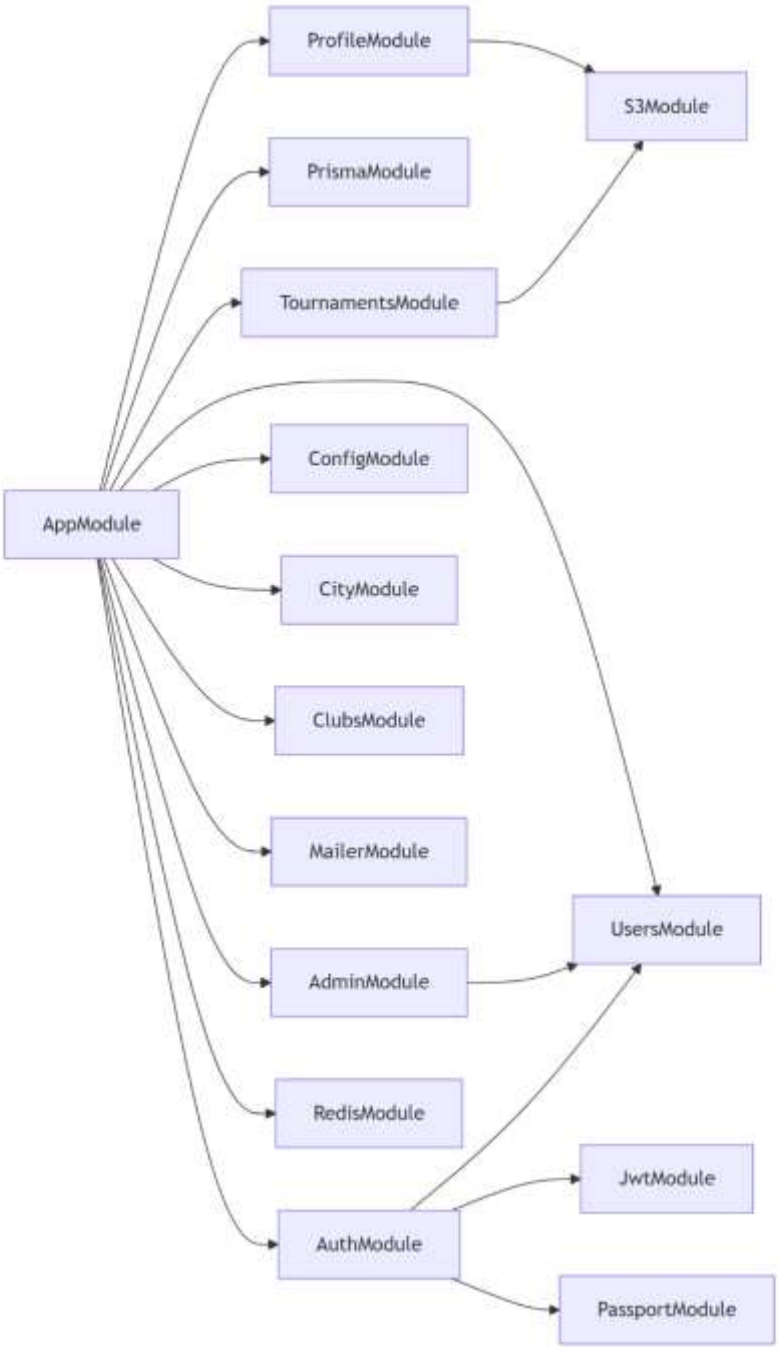

Рисунок 2.3.1.1 – Схема взаимодействия модулей NestJS

Такой подход обеспечивает слабую связанность и высокую согласованность компонентов, позволяя независимо развивать функциональные блоки и эффективно управлять зависимостями.

2.4 Разработка фронтенда

Разработка пользовательского интерфейса осуществлялась с использованием библиотеки Shadcn/ui, основанной на React и TailwindCSS. Выбор данной библиотеки был обусловлен её гибкостью, визуальной строгостью и соответствием современным UI-стандартам.

2.4.1 Архитектура фронтенда

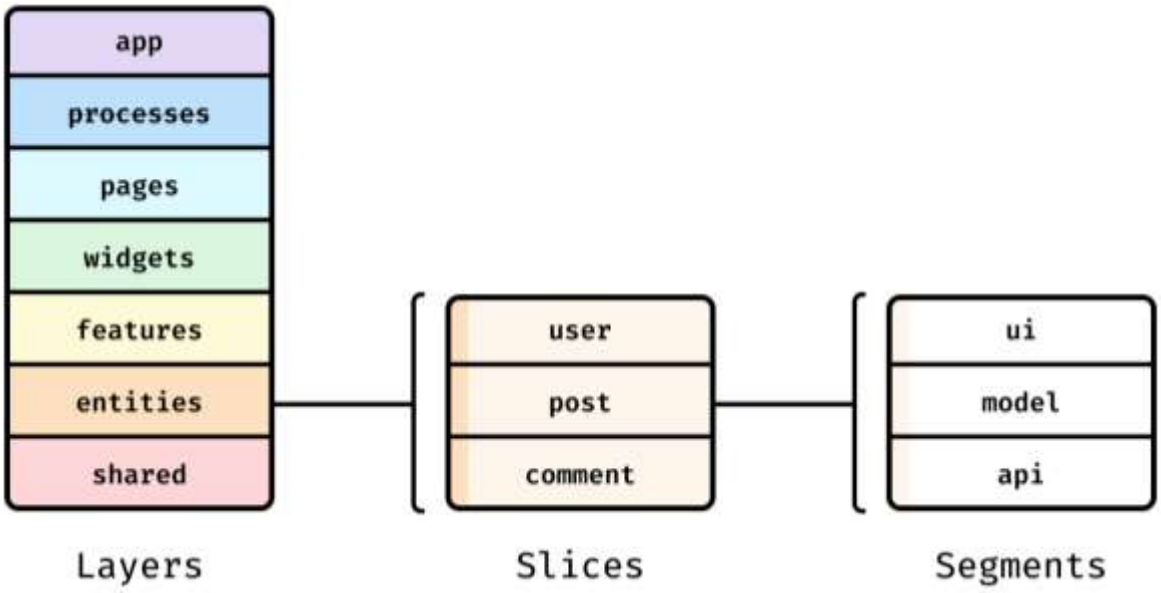

Рисунок 2.4.1.1 – Схема из документации FSD

Для организации клиентской части проекта применялась архитектурная методология Feature-Sliced Design (FSD), обеспечивающая структурное разделение логики приложения по слоям. Это позволяет масштабировать фронтенд и сохранять читаемость кода даже при увеличении функциональности.

На практике структура проекта была реализована в соответствии с подходом Feature-Sliced Design (FSD) и представлена следующим образом:

– app/ — точка входа и глобальные стили приложения;

– entities/ — доменные сущности, объединяющие бизнес-логику, UI и данные для клубов, профилей и турниров;

– features/ — независимые функциональные единицы, такие как авторизация (auth), включающие model и ui;

– pages/ — маршруты приложения, соответствующие страницам интерфейса (home, clubs, profile, tournaments и др.);

– shared/ — переиспользуемые модули: API-клиенты, конфигурации, хуки, типы, компоненты и утилиты;

– widgets/ — готовые композиционные элементы, такие как main-layout, profile-layout, предназначенные для объединения entities и features в законченную UI-структуру.

Такое разделение кода обеспечивает модульность, переиспользуемость и упрощает сопровождение проекта. Благодаря выделению слоёв entities, features, shared и widgets, код остаётся структурированным и легко расширяемым по мере роста системы.

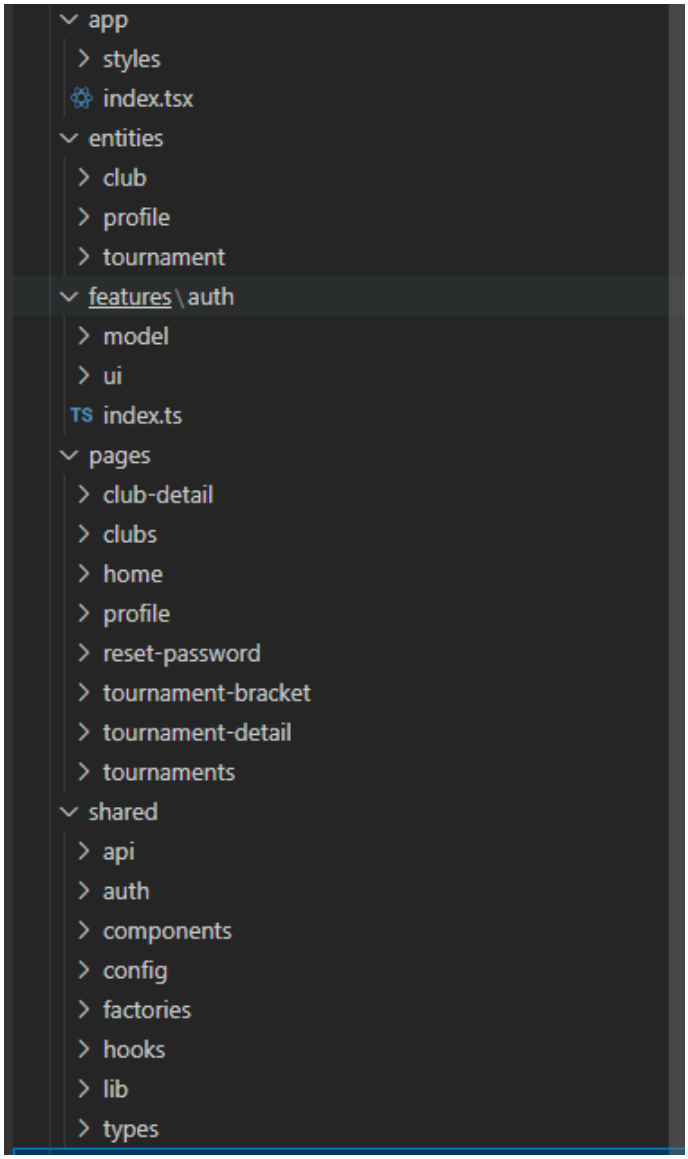

Рисунок 2.4.1.2 – Структура проекта по FSD

2.4.2 Использование библиотеки компонентов Shadcn/ui

Для построения пользовательского интерфейса веб-приложения использовалась библиотека готовых компонентов Shadcn/ui, включающая:

– кнопки и формы ввода;

– модальные окна и диалоги;

– выпадающие списки (dropdown);

– таблицы, вкладки, шаги (steps) и другие элементы управления.

Компоненты применялись в стандартной реализации без кастомизации, что позволило ускорить разработку интерфейса и обеспечить единообразный внешний вид на всех страницах платформы. Библиотека построена на базе TailwindCSS, благодаря чему её элементы изначально адаптированы под современные UX/UI-стандарты и корректно отображаются на различных устройствах.

2.4.3 Использование библиотеки Effector для разделения бизнес-логики и пользовательского интерфейса

В рамках клиентской части веб-приложения для управления состоянием и построения архитектуры приложения была выбрана библиотека Effector. Это реактивная библиотека, предоставляющая декларативный подход к управлению состоянием, событиями и эффектами в приложениях.

Главным преимуществом использования Effector является жесткое разделение бизнес-логики и пользовательского интерфейса (UI). Компоненты пользовательского интерфейса не содержат в себе бизнес-логику, а лишь подписываются на реактивные источники данных и отображают состояние. Таким образом достигается высокая модульность и повторное использование кода.

Бизнес-логика оформляется в виде юнитов (stores, events, effects), которые описывают поведение приложения независимо от конкретного фреймворка или среды выполнения. Это позволяет:

– минимизировать связанность компонентов;

– облегчить тестирование логики без запуска UI;

– сократить побочные эффекты за счет централизованного управления асинхронными вызовами.

Например, действия, связанные с регистрацией и авторизацией пользователя, реализуются в виде эффектов (createEffect), а состояния форм — в виде createStore. UI-компоненты подключаются к этим хранилищам и отображают их текущие значения.

Такой подход соответствует принципам чистой архитектуры и способствует масштабируемости проекта за счёт чёткого разграничения ответственности между слоями приложения.

2.5 Формула расчета рейтинга

Для отражения относительного уровня игроков в платформе реализована система рейтинга, основанная на алгоритме ELO — широко используемой модели, применяемой в спортивных соревнованиях и играх с двумя участниками. После завершения каждого матча рейтинг обоих игроков пересчитывается с учётом результата.

Для игрока A: $RA' = RA + K \cdot (SA - EA)$

Для игрока B: $RB' = RB + K \cdot (SB - EB)$

Где:

– RA, RB — текущие рейтинги;

– RA', RB' — новые рейтинги;

– $SA \in \{0,1\}$ — фактический результат матча для игрока A;

– EA — ожидаемый результат игрока A;

– $SB \in \{0,1\}$ — фактический результат матча для игрока B;

– EB — ожидаемый результат игрока B;

– K — коэффициент чувствительности рейтинга (обычно 8, 16 , 32 или тд).

3 Разработка ИС

3.1 Организация процесса разработки

Разработка информационной системы велась индивидуально и охватывала полный цикл — от формализации требований до финального развёртывания и тестирования. Все архитектурные решения и сценарии использования были сформированы самостоятельно на основе анализа аналогичных платформ, личного опыта участия в турнирах и понимания потребностей конечных пользователей.

В ходе выполнения проекта регулярно проводились консультации с научным руководителем, в рамках которых обсуждались ключевые проектные решения, структура дипломной работы, а также актуальные проблемы на этапах анализа и реализации.

Процесс разработки организован по итерационной модели с использованием элементов методологии Kanban. Задачи фиксировались и отслеживались в Notion с применением визуальной доски с колонками: «Не начато», «В процессе», «Готово».

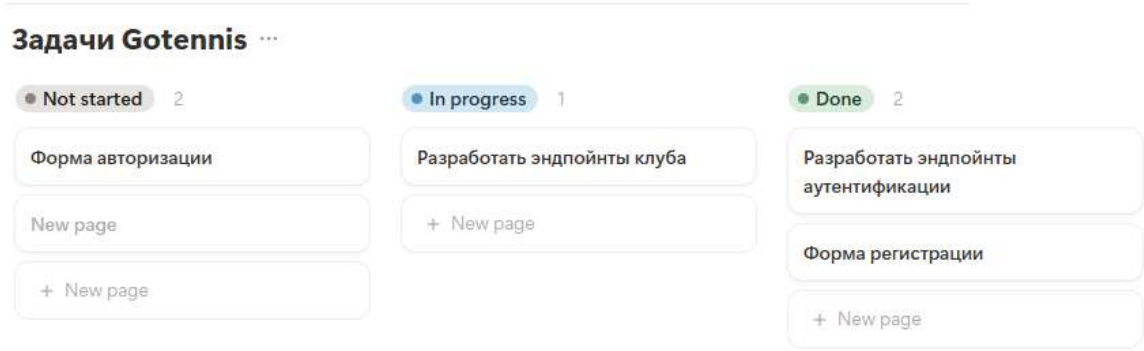

Рисунок 3.1.1 - Доска Kanban с задачами

На начальном этапе были определены основные пользовательские роли и сценарии, после чего сформирован минимальный жизнеспособный продукт (MVP), включающий ключевые функции: регистрацию, участие в турнирах, управление клубами, учёт рейтингов и отображение сетки.

Разработка велась поэтапно. Сначала реализовывались основные модули и пользовательский интерфейс. По завершении каждого этапа проводилось модульное и ручное тестирование. Были реализованы базовые проверки, тестовые сценарии и ручная отладка типичных пользовательских действий.

После тестирования вносились правки в поведение форм, добавлялись уведомления, а также заглушки для нестабильных сетевых ситуаций.

Весь исходный код размещён в приватном репозитории с сохранением истории изменений. Такой подход позволил обеспечить гибкость, управляемость и надёжность процесса разработки.

3.2 Подготовка серверной инфраструктуры для развёртывания

Для размещения и запуска информационной системы был предварительно подготовлен сервер и сопутствующая инфраструктура. На этом этапе выполнены действия, обеспечивающие возможность последующего развёртывания и автоматической доставки приложения.

Был арендован домен gotennis.kz, предназначенный для использования в публичной части платформы.

В качестве хостинга выбран облачный VPS-сервер, соответствующий минимальным требованиям проекта:

– 1 виртуальное ядро (vCPU);

– 1 ГБ оперативной памяти;

– 20 ГБ SSD-диска;

– операционная система — Ubuntu 24.04 LTS.

После развёртывания сервера были выполнены следующие действия:

– установлен и настроен Docker — основной инструмент для контейнеризации приложения и его компонентов;

– сгенерированы SSH-ключи, необходимые для безопасной передачи файлов и удалённого выполнения команд;

– публичный ключ был добавлен на VPS, а приватный — зашифрован и сохранён в GitHub Secrets, что позволяет запускать деплой через GitHub Actions;

– настроен ограниченный доступ по SSH и базовая защита сервера.

Таким образом, серверная среда была полностью подготовлена для приёма Docker-контейнеров и автоматического развёртывания приложения с использованием средств CI/CD.

3.3 Настройка сервисов Yandex Cloud

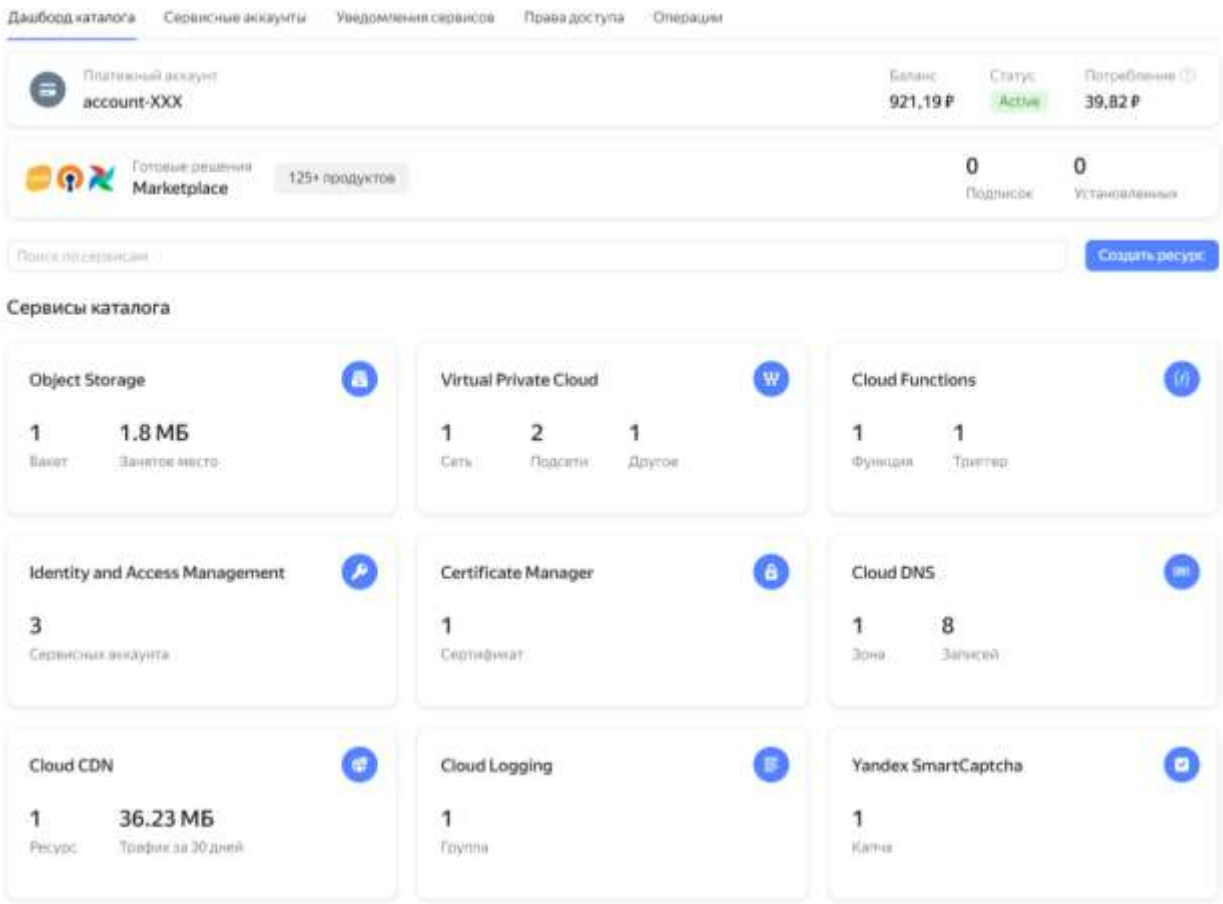

Рисунок 3.3.1 – Интерфейс организации в Yandex Cloud со списком используемых сервисов

Для размещения и поддержки инфраструктуры платформы использовались облачные сервисы Yandex Cloud, что позволило обеспечить надёжную работу компонентов, масштабируемость и удобное управление внешними ресурсами. Ниже представлены основные шаги, выполненные при настройке облачной среды.

Первоначально был делегирован домен gotennis.kz в систему Cloud DNS. Это позволило централизованно управлять записями, включая основной домен и поддомены, используемые различными компонентами системы.

Для хранения статических файлов (аватаров пользователей и квитанций об оплате) был настроен сервис Yandex Object Storage. Внутри него создан отдельный бакет с разграничением доступа. Для безопасного взаимодействия

backend-приложения с бакетом использовался сервис Identity and Access Management (IAM) — в рамках которого был создан сервисный аккаунт с правами на чтение и запись в Object Storage. Доступ к бакету осуществляется с использованием ключей доступа, привязанных к данному аккаунту.

С целью ускорения доставки изображений и других файлов пользователям по Казахстану был настроен сервис Cloud CDN. Для него был выделен поддомен cdn.gotennis.kz, через который осуществляется раздача контента из S3-совместимого бакета. Сертификат безопасности для данного поддомена был сгенерирован с помощью сервиса Certificate Manager, где использовался автоматический выпуск SSL-сертификатов от Let’s Encrypt.

Для отправки email-сообщений (в частности, одноразовых кодов подтверждения регистрации и ссылок на восстановление пароля) был настроен сервис Cloud Postbox. Он обеспечивает надёжную доставку писем и простую SMTP-интеграцию с backend-приложением.

Вся настройка сервисов производилась через Yandex Cloud Console, с использованием рекомендованных политик безопасности и минимальных необходимых прав доступа.

3.4 Разработка бэкенда

Серверная часть информационной системы была реализована с использованием фреймворка NestJS, обладающего модульной архитектурой и полной поддержкой TypeScript. Такой подход позволил обеспечить логичное разделение бизнес-логики по функциональным модулям, упростить тестирование и дальнейшее расширение системы.

Для взаимодействия с базой данных применялась объектно-реляционная модель Prisma ORM, обеспечивающая строгую типизацию, автоматическую генерацию клиентского кода и надёжную миграцию структуры базы данных. Доступ к данным пользователей инкапсулирован в отдельный UsersModule, предоставляющий сервис UsersService. Этот сервис реализует все основные действия с пользовательскими данными — создание, поиск по различным параметрам, обновление профиля и т.д. Все обращения к базе данных выполняются централизованно через UsersService, что упрощает поддержку и повторное использование логики.

Первым реализованным компонентом стал AuthModule, отвечающий за процессы аутентификации и регистрации пользователей. В нём применяется Redis для хранения одноразовых кодов подтверждения (OTP) и ограничения частоты запросов. Для всех операций с данными пользователей AuthModule взаимодействует с UsersService, предоставляемым UsersModule, что

обеспечивает строгое разделение ответственности и повышает модульность системы.

Следующим этапом был разработан ProfileModule, включающий функции управления профилем. В частности, реализована возможность смены аватара: загруженный файл сохраняется в Yandex Object Storage (S3), после чего сгенерированная ссылка записывается в базу данных через UsersService.

В процессе разработки API использовались аннотации из пакета @nestjs/swagger, что позволило автоматически формировать спецификацию OpenAPI. Эта спецификация использовалась для генерации типизированного клиента на frontend-стороне и обеспечивала строгую синхронизацию между слоями приложения.

Остальные модули (работа с клубами, турнирами, регистрациями, матчами, рейтингами) разрабатывались поэтапно, синхронно с подключением frontend-интерфейса. Такой подход позволил гибко тестировать каждую функцию в процессе интеграции и своевременно адаптировать API под реальные пользовательские сценарии.

Рисунок 3.4.1 – Документация API в Swagger

3.5 Разработка фронтенда

Клиентская часть информационной системы была реализована с использованием фреймворка Next.js, что обеспечило поддержку серверного рендеринга, удобную маршрутизацию и масштабируемую архитектуру. Основной акцент при разработке был сделан на модульность, удобство взаимодействия с API и пользовательский опыт.

В качестве основы интерфейса использовался компонентный UI Kit Shadcn, предоставляющий широкий набор готовых компонентов, соответствующих современным стандартам дизайна. Благодаря этому удалось быстро собрать функциональный и визуально согласованный пользовательский интерфейс.

Для взаимодействия с серверной частью применялась автоматическая генерация HTTP-клиента на основе OpenAPI-спецификации, экспортируемой с backend-приложения. В проекте использовался пакет @hey-api/openapi-ts, который на основании актуального описания API генерирует типобезопасный HTTP-клиент на TypeScript. Такой подход обеспечивает строгое соответствие контрактов между клиентом и сервером и минимизирует ошибки при интеграции.

Клиентская валидация форм реализована с помощью библиотеки Zod, которая обеспечивает декларативное определение схем данных и выдачу информативных ошибок. Это позволяет проверять ввод пользователя ещё до отправки запроса на сервер, повышая удобство и надёжность интерфейса.

Бизнес-логика клиентского приложения организована с использованием реактивной библиотеки Effector, что позволило реализовать управляемое состояние, логику событий и потоков данных без излишней сложности. Такой подход улучшает читаемость кода и упрощает поддержку.

```
sample({
  clock: refreshed,
  source: $$tournament.$tournament,
  filter: Boolean,
  fn: (tournament) => tournament!.id,
  target: [
    $$matches.fetchMatchesFx,
    $$tournament.fetchTournamentFx,
    $$participants.fetchParticipantsFx,
  ],
});

sample({
  clock: matchPicked,
  fn: (match) => {
    return {
      openModal: true,
      editingMatch: match,
      selectedPlayer1Id: match.player1Id ?? null,
      selectedPlayer2Id: match.player2Id ?? null,
      winnerId: match.winnerId ?? null,
      currentRound: match.round,
    };
  },
  target: spread({
    openModal: $$playerSetup.$$modal.changed,
    editingMatch: $$playerSetup.$editingMatch,
    selectedPlayer1Id: $$playerSetup.$selectedPlayer1Id,
    selectedPlayer2Id: $$playerSetup.$selectedPlayer2Id,
    winnerId: $$playerSetup.$winnerId,
    currentRound: $$playerSetup.$currentRound,
  }),
});
```

Рисунок 3.5.1 – Бизнес-логика написанная с помощью effector

Для оформления интерфейса использовалась утилитарная CSS-библиотека Tailwind CSS, обеспечивающая адаптивную верстку и быстрое прототипирование. Комбинация Tailwind и Shadcn позволила добиться визуальной целостности и гибкой настройки компонентов.

Таким образом, frontend-приложение построено на современном и согласованном технологическом стеке, обеспечивающем высокую производительность, строгую типизацию, быстрый отклик интерфейса и удобство интеграции с backend-сервисами.

3.6 Тестирование ИС

3.6.1 Тестирование бэкенда

Тестирование серверной части платформы проводилось с помощью фреймворка Vitest. Данный инструмент был выбран за скорость выполнения тестов, удобство настройки и поддержку TypeScript, что позволило эффективно проверять корректность бизнес-логики, API-контроллеров и взаимодействие с базой данных.

В процессе тестирования были охвачены следующие компоненты системы:

– Бизнес-логика**:** корректность обработки данных при регистрации, авторизации, создании турниров и клубов.

– Контроллеры**:** правильность HTTP-ответов, соответствие статусов, обработка ошибок и исключений.

– Интеграция с БД (Prisma ORM): проверка взаимодействия с PostgreSQL, корректность запросов и транзакций.

– Обработка исключений**:** проверка поведения системы при некорректных входных данных и непредвиденных ситуациях.

– При тестировании особое внимание было уделено:

– Валидации входных данных и граничным условиям.

– Проверке безопасности, включая разграничение прав доступа по ролям и предотвращение несанкционированного доступа.

– Надёжности и устойчивости системы при некорректных или вредоносных запросах.

Благодаря модульной архитектуре, реализованной в NestJS, тесты были изолированы и легко поддерживались. Использование Vitest обеспечило высокую скорость и подробность отчётности по результатам тестирования.

Результаты тестирования подтвердили, что backend-система соответствует заявленным требованиям и демонстрирует стабильность и надёжность работы.

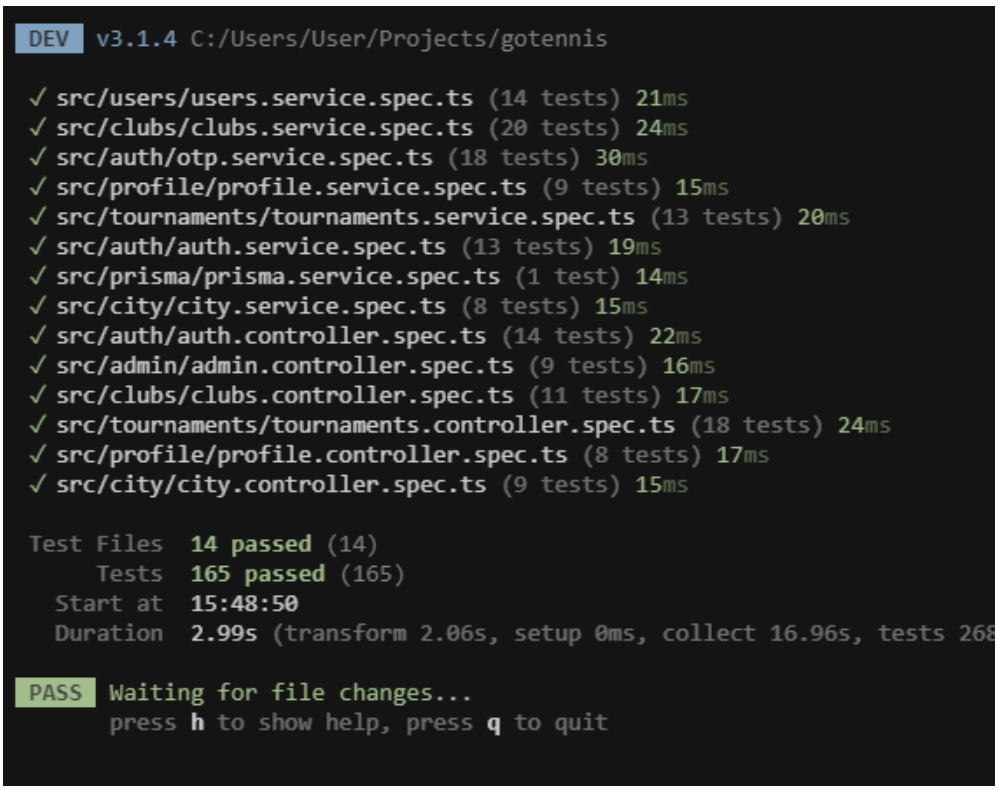

Рисунок 3.6.1.1 – Успешное выполнение тестов на Vitest.

3.6.2 Тестирование фронтенда

Для оценки качества пользовательского интерфейса было проведено ручное тестирование по заранее разработанным сценариям. Проверялась работоспособность таких ключевых функций, как регистрация и авторизация пользователей, создание и управление турнирами и клубами, верификация и управление профилями пользователей, генерация и редактирование турнирных сеток.

Основные цели ручного тестирования:

– Подтверждение корректности работы интерфейса для пользователей различных ролей (администратор, менеджер клуба, участник).

– Проверка удобства использования и стабильности интерфейса в разных сценариях взаимодействия.

– Выявление и оперативное устранение недочётов.

По результатам тестирования платформа подтвердила своё соответствие основным функциональным требованиям. Выявленные проблемы были исправлены в процессе разработки.

Полная таблица тест-кейсов ручного тестирования интерфейса представлена в Приложении Б**.**

3.7 Развертывание ИС

Развертывание проекта автоматизировано с использованием инфраструктуры на базе GitHub Actions, Docker и GitHub Container Registry (GHCR). Это обеспечивает стабильность и повторяемость всех этапов — от сборки до публикации и миграции базы данных.

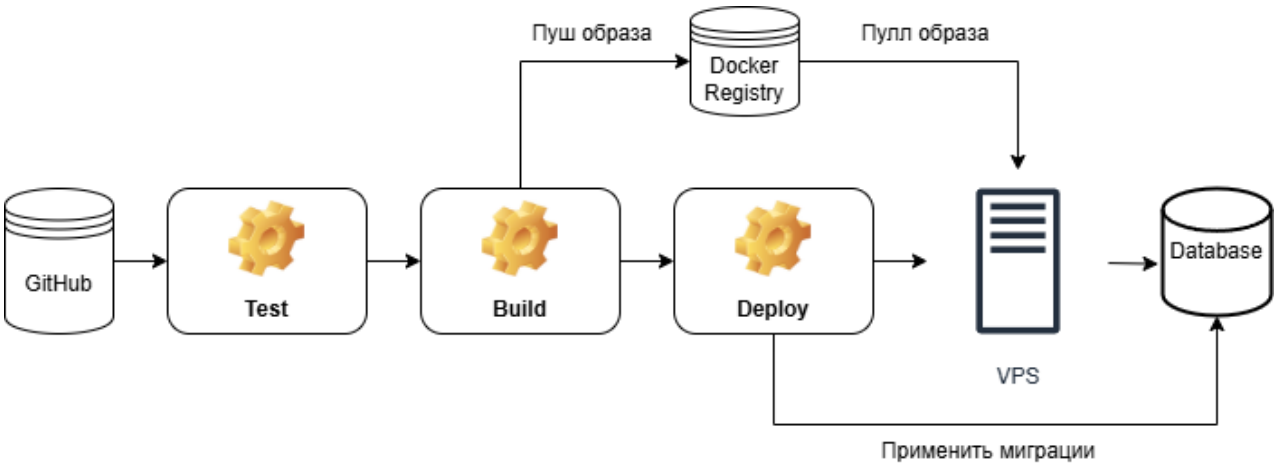

Рисунок 3.7.1 - Схема развертывания приложения

Цепочка сборки и развёртывания включает следующие этапы:

1. Отправка изменений в репозиторий GitHub
   Разработка ведётся с использованием системы контроля версий Git. Каждый коммит в основной репозиторий инициирует запуск CI/CD-процесса.
2. Автоматическая сборка Docker-образа
   С помощью GitHub Actions создаётся новый Docker-образ, включающий в себя все необходимые зависимости и настройки проекта. Этот образ затем публикуется в GitHub Container Registry (GHCR).
3. Деплой на удалённый сервер по SSH
   После сборки происходит 9втоматическое подключение к серверу через SSH, где выполняются:
   – загрузка свежего Docker-образа из GHCR;
   – остановка старого и запуск нового контейнера с актуальной версией приложения;
   – применение миграций к базе данных с использованием ORM Prisma (через команду npx prisma migrate deploy), что обеспечивает актуальное состояние схемы базы данных.

ЗАКЛЮЧЕНИЕ

В рамках дипломного проекта была разработана web-платформа для автоматизации проведения любительских турниров по настольному теннису в Казахстане. Проанализированы существующие проблемы в организации турниров, проведено исследование аналогичных решений, определены требования к системе и обоснован выбор технологий. Разработано приложение, включающее функционал регистрации пользователей, создания и управления турнирами, отображения турнирной сетки и работы с картой клубов. Платформа реализована с использованием современных инструментов и технологий, включая Next.js, NestJS, PostgreSQL, Redis, Docker и Yandex Cloud. Результаты тестирования показали корректную работу системы и её соответствие заявленным требованиям. В дальнейшем проект может быть расширен за счёт внедрения онлайн-трансляций, автоматического подсчёта рейтингов, поддержки мобильного приложения и интеграции с официальными структурами.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1 Лемарк Т. Docker. Простой способ развертывания веб-приложений. — СПб.: Питер, 2021. — 240 с.
2 Иванов А. Н. NGINX: от начинающего до профессионала. — М.: ДМК Пресс, 2022. — 352 с.
3 Краснов А. Redis на практике. — М.: БХВ-Петербург, 2020. — 288 с.
4 Гольдштейн Б. PostgreSQL. Основы администрирования и программирования. — М.: ДМК Пресс, 2021. — 432 с.
5 Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. — М.: Питер, 2020. — 384 с.
6 Макдональд Д. DevOps для разработчиков. — СПб.: Питер, 2021. — 352 с.
7 Белл Дж. Docker и Kubernetes. Полное руководство. — М.: Вильямс, 2022. — 416 с.
8 Вадим Макеев, Алексей Симоненко. Практика современного фронтенда. — М.: Питер, 2023. — 320 с.
9 Шмидт Н. JavaScript и TypeScript для начинающих. — СПб.: БХВ-Петербург, 2023. — 368 с.
10 Документация NestJS. - https://docs.nestjs.com
11 Документация Next.js. - https://nextjs.org/docs
12 Генерация DBML из схемы Prisma - https://github.com/notiz-dev/prisma-dbml-generator
13 Документация по FSD - https://feature-sliced.github.io/documentation/

Приложение А
(обязательное)

Техническое задание

А.1.1 Техническое задание на разработку системы управления платформой для турниров и клубов

Настоящее техническое задание распространяется на разработку программного обеспечения для управления платформой, предназначенной для организации мероприятий и взаимодействия в области настольного тенниса. Система предназначена для регистрации участников на турниры, управления клубами и предоставления пользователям удобного способа взаимодействия. Она позволит участникам легко регистрироваться на турниры, вносить данные и загружать необходимые документы; менеджерам клубов — управлять информацией о клубах, проверять их данные и вести взаимодействие с пользователями; администраторам — анализировать статистику, модерировать контент и поддерживать порядок в системе.

Платформа будет адаптирована для настольного тенниса, обеспечивая необходимые инструменты для организации и управления спортивными событиями в этой области. Такой подход сделает систему удобной и эффективной для взаимодействия между игроками, клубами и организаторами турниров, способствуя развитию настольного тенниса.

А.1.2 Основание для разработки

Платформа разрабатывается на основании согласия дипломного руководителя выбранной дипломником темой.

А.1.3 Назначение

Платформа предназначена для организации и управления турнирами, регистрации клубов и участников, а также предоставления пользователям удобного и интуитивного интерфейса взаимодействия с платформой. Основная цель — повысить эффективность и прозрачность управления спортивными событиями.

А.1.4 Требования к функциональным характеристикам

Платформа предлагает функционал, предназначенный для разных ролей пользователей: участников, менеджеров клубов и администраторов. Для каждой роли предусмотрены свои функции, разработанные с учетом требований и задач пользователей. Так, участники получают возможность быстро и легко регистрироваться на турниры, вносить необходимые данные и загружать подтверждающие документы. Менеджеры клубов имеют доступ к инструментам управления, позволяющим добавлять или редактировать информацию о клубах. Верификация менеджеров клубов производится администраторами, чтобы обеспечить достоверность информации и предотвратить мошенничество. Администраторы, в свою очередь, могут модерировать контент, что обеспечивает общий порядок и прозрачность в системе.

А.1.4.1 Входные данные и способы их ввода

Для пользователей:

– Регистрация и аутентификация: ввод имени пользователя, email, пароля.

– Редактирование профиля: имя, фамилия, никнейм, аватар.

– Регистрация на турниры: загрузка квитанции об оплате (если требуется).

Для менеджеров:

– Управление клубами: добавление/редактирование информации о клубах.

– Управление турнирами: добавление/редактирование информации о турнирах, смена статусов, настройка турнирной сетки.

Для администраторов:

– Модерация контента: верификация менеджеров, управление клубами, управление турнирами.

А.1.4.2 Выходные данные

Система должна предоставлять следующие данные:

– Профиль пользователя

– Список доступных турниров с возможностью фильтрации по дате, статусу и типу мероприятия(платное/бесплатное).

– Список доступных клубов, отображение клубов на карте.

– Информация о статусе турнира, включая подробности этапов (предстоящий, в процессе, завершённый, отмененный) и список участников.

– Информация о клубе (расположение, контактные данные), список предстоящих турниров.

– Сообщения об ошибках:

– Некорректный ввод данных, например, неверный формат email или пароля.

– Превышение лимита загрузки изображений или других файлов.

– Попытка доступа к закрытым или удаленным данным.

В случае успешного выполнения операций система должна уведомлять пользователя, отображая соответствующее сообщение о подтверждении, а также предоставлять ссылки на связанные действия или разделы.

А.1.5 Требования к составу и параметрам технических средств

Для обеспечения эффективной работы веб-приложения требуется соответствие следующим аппаратным и программным требованиям:

Клиентская сторона (пользователь)

– Устройство: персональный компьютер, ноутбук, планшет или смартфон с доступом к сети Интернет.

– Операционная система: Windows 10+, macOS 11+, Android 9+, iOS 13+.

– Браузер: современный с поддержкой HTML5 и JavaScript (Chrome 110+, Firefox 100+, Safari 13+, Edge 110+).

– Разрешение экрана: от 360×640 пикселей (адаптивность поддерживается).

Серверная сторона (VPS)

– Процессор: 1 vCPU.

– Оперативная память: 1 ГБ.

– Дисковое пространство: от 10 ГБ SSD.

– Подключение к сети с выходом в Интернет.

– Поддержка Docker и Docker Compose.

– Операционная система сервера: Ubuntu 20.04 LTS или совместимая.

А.1.6 Требования к надежности

Система должна обеспечивать валидацию вводимых данных, защищённое хранение информации и минимизацию ошибок при выполнении операций. Разграничение доступа между ролями пользователей является обязательным.

А.1.7 Требования к безопасности

Для обеспечения безопасности данных система должна использовать следующие механизмы:

– Передача данных исключительно по защищённым протоколам HTTPS с использованием TLS 1.2/1.3.

– Реализация защищённой аутентификации пользователей.

– Шифрование данных, хранимых на сервере и передаваемых между клиентом и сервером.

Приложение Б

Тест-кейсы для ручного тестирования

Таблица Б.1 – Тест-кейсы ручного тестирования интерфейса

№	Название теста	Действия	Ожидаемый результат
1	Регистрация пользователя (валидные данные)	Заполнить форму регистрации правильно	Успешная регистрация, отправлен OTP
2	Регистрация (неверный email)	Ввести некорректный email	Ошибка валидации email
3	Подтверждение OTP (правильный код)	Ввести корректный код	Аккаунт подтверждён
4	Авторизация (правильные данные)	Войти с верным email и паролем	Установка HttpOnly куки, переход в профиль
5	Авторизация (неверный пароль)	Ввести неправильный пароль	Ошибка авторизации
6	Создание клуба менеджером	Создать клуб с необходимыми полями	Клуб успешно создан, отображается в списке клубов
7	Редактирование информации о клубе	Изменить название и контактные данные клуба	Изменения сохранены и видны другим пользователям
8	Удаление клуба менеджером	Удалить созданный клуб	Клуб удалён, не отображается на платформе
9	Активация и деактивация клуба	Менеджер меняет статус клуба (активен/неактивен)	Неактивный клуб не отображается в общем списке клубов

10	Создание турнира	Создание турнира через интерфейс менеджера	Турнир появляется в списке турниров со статусом «Создан»
11	Регистрация на платный турнир	Пользователь регистрируется и загружает чек	Заявка ожидает подтверждения оплаты
12	Регистрация на бесплатный турнир	Пользователь подаёт заявку на участие в бесплатном турнире	Заявка сразу подтверждена
13	Фильтрация турниров по статусу	Использовать фильтр «Предстоящие»	Отображаются только турниры со статусом «Предстоящие»
14	Фильтрация турниров по дате	Выбрать конкретную дату проведения турниров	Показаны турниры на выбранную дату
15	Сортировка турниров	Сортировка по новизне	Показаны турниры отсортированные по новизне
16	Запуск турнира (переход в статус «В процессе»)	Менеджер стартует созданный турнир	Создаётся турнирная сетка, турнир переходит в статус «В процессе»
17	Проверка генерации турнирной сетки	Открыть турнир после запуска	Турнирная сетка отображается корректно
18	Редактирование турнирной сетки	Менеджер клуба изменяет пары игроков в активном турнире	Турнирная сетка корректно обновляется и сохраняет изменения
19	Завершение турнира	Завершить турнир через интерфейс менеджера	Статус турнира обновляется на «Завершён»

20	Отмена турнира	Менеджер отменяет созданный турнир	Турнир переходит в статус «Отменён»
21	Удаление турнира	Удалить турнир менеджером	Турнир исчезает из списков и результатов поиска
22	Верификация менеджера клуба администратором	Администратор подтверждает менеджера клуба	Менеджер клуба получает доступ ко всем функциям платформы
23	Ограничение попыток OTP	Превысить лимит запросов OTP-кода	Блокировка повторных запросов на время
24	Редактирование профиля пользователя	Изменить персональные данные (имя, телефон и т.п.)	Изменения профиля сохраняются и отображаются корректно
25	Загрузка аватара пользователя	Пользователь загружает изображение для профиля	Аватар успешно загружен и отображается

Приложение В

Листинг кода генерации сетки турнира

```
public generateSingleEliminationMatches(playerCount: number): MatchData[] {
    const totalRounds = Math.log2(playerCount);
    let matchNumber = 1;
    const matches: MatchData[] = [];

    let currentRoundMatches = playerCount / 2;
    for (let round = 1; round <= totalRounds; round++) {
      for (let i = 0; i < currentRoundMatches; i++) {
        matches.push({
          id: nanoid(),
          round,
          matchNumber: matchNumber++,
          bracketPosition: 'WINNERS',
        });
      }
      currentRoundMatches /= 2;
    }

    for (let i = 0; i < matches.length; i++) {
      const match = matches[i];

      if (match.round === totalRounds) {
        continue;
      }

      const nextRound = match.round + 1;
      const matchesInCurrentRound = playerCount / Math.pow(2, match.round);
```

```
    const matchesInNextRound = playerCount / Math.pow(2, nextRound);
    const matchIndexInRound = (match.matchNumber - 1) %
matchesInCurrentRound;
    const nextMatchIndexInRound = Math.floor(matchIndexInRound / 2);

    const nextMatch = matches.find(
      (m) =>
        m.round === nextRound &&
        (m.matchNumber - 1) % matchesInNextRound === nextMatchIndexInRound,
    );

    if (nextMatch) {
      match.nextMatchId = nextMatch.id;
    }
  }

  return matches;
}
```

Приложение Г

Модель effector страницы турнирной сетки

```
import { MatchDto } from "@/shared/api/types.gen";
import { createEvent, sample } from "effector";
import * as $$playerSetup from "./player-setup";
import * as $$tournament from "./tournament";
import { or, spread } from "patronum";

import * as $$matches from "./matches";
import * as $$participants from "./participants";

export const inited = createEvent<number>();
export const reset = createEvent<void>();
export const refreshed = createEvent<void>();
export const matchPicked = createEvent<MatchDto>();
export const $isLoading = or(
  $$tournament.$isTurnamentLoading,
  $$matches.$isMatchesLoading,
  $$participants.$isParticipantsLoading
);

sample({
  clock: inited,
  target: [
    $$tournament.fetchTournamentFx,
    $$matches.fetchMatchesFx,
    $$participants.fetchParticipantsFx,
```

```
  ],
});

sample({
  clock: refreshed,
  source: $$tournament.$tournament,
  filter: Boolean,
  fn: (tournament) => tournament!.id,
  target: [
    $$matches.fetchMatchesFx,
    $$tournament.fetchTournamentFx,
    $$participants.fetchParticipantsFx,
  ],
});

sample({
  clock: matchPicked,
  fn: (match) => {
    return {
      openModal: true,
      editingMatch: match,
      selectedPlayer1Id: match.player1Id ?? null,
      selectedPlayer2Id: match.player2Id ?? null,
      winnerId: match.winnerId ?? null,
      currentRound: match.round,
    };
  },
  target: spread({
```

```
    openModal: $$playerSetup.$$modal.changed,
    editingMatch: $$playerSetup.$editingMatch,
    selectedPlayer1Id: $$playerSetup.$selectedPlayer1Id,
    selectedPlayer2Id: $$playerSetup.$selectedPlayer2Id,
    winnerId: $$playerSetup.$winnerId,
    currentRound: $$playerSetup.$currentRound,
  }),
});

sample({
  clock: reset,
  target: [
    $$playerSetup.reset,
    $$matches.$matches.reinit,
    $$participants.$participants.reinit,
    $$tournament.$tournament.reinit,
  ],
});

export { $$playerSetup, $$matches, $$participants, $$tournament };
```

Приложение Д

Workflow в Github Actions

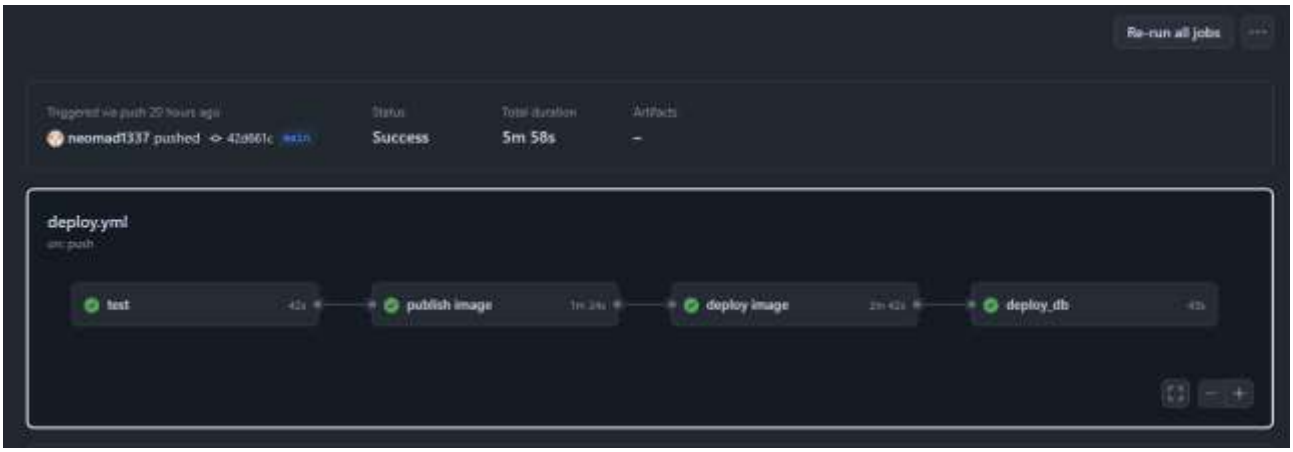

Рисунок Д.1 – Workflow в Github Actions

Приложение Е

Листинг docker-compose.yml

```
services:
  backend:
    container_name: gotennis_backend
    image: ghcr.io/neomad1337/gotennis:latest
    environment:
      DOCKER: "true"
      DATABASE_URL: /run/secrets/database_url
      EMAIL_HOST: /run/secrets/email_host
      EMAIL_USER: /run/secrets/email_user
      EMAIL_PASSWORD: /run/secrets/email_password
      JWT_ACCESS_TOKEN_EXPIRATION_TIME: "1h"
      JWT_REFRESH_TOKEN_EXPIRATION_TIME: "7d"
      JWT_ACCESS_TOKEN_SECRET: /run/secrets/jwt_access_token_secret
      JWT_REFRESH_TOKEN_SECRET:
/run/secrets/jwt_refresh_token_secret
      REDIS_HOST: /run/secrets/redis_host
      REDIS_PASSWORD: /run/secrets/redis_password
      S3_ENDPOINT: /run/secrets/s3_endpoint
      S3_ACCESS_KEY: /run/secrets/s3_access_key
      S3_SECRET_KEY: /run/secrets/s3_secret_key
      S3_BUCKET_NAME: /run/secrets/s3_bucket_name
      S3_REGION: /run/secrets/s3_region
      CDN_BASE_URL: /run/secrets/cdn_base_url
    secrets:
      - database_url
      - email_host
      - email_user
      - email_password
      - jwt_access_token_secret
      - jwt_refresh_token_secret
      - redis_host
      - redis_password
      - s3_endpoint
      - s3_access_key
      - s3_secret_key
      - s3_bucket_name
      - s3_region
      - cdn_base_url
    networks:
      - app
```

```
  depends_on:
    - db
    - redis
db:
  image: postgres
  restart: always
  shm_size: 128mb
  environment:
    POSTGRES_PASSWORD_FILE: /run/secrets/database_password
    POSTGRES_USER_FILE: /run/secrets/database_user
    POSTGRES_DB_FILE: /run/secrets/database_name
  secrets:
     - database_password
     - database_user
     - database_name
  ports:
    - 5432:5432
  networks:
    - app
  volumes:
    - ~/app/db:/var/lib/postgresql/data
nginx:
  container_name: 'nginx-service'
  image: nginx:latest
  ports:
    - 80:80
    - 443:443
  volumes:
    - ./config/html:/usr/share/nginx/html:ro
    - /etc/letsencrypt:/etc/letsencrypt:ro
    - /tmp/acme_challenge:/tmp/acme_challenge
    - ./config/nginx.conf:/etc/nginx/nginx.conf:ro
    - ./secrets/basic_auth_creds:/etc/nginx/.htpasswd:ro
  networks:
    - app
  restart: always
redis:
  image: redis:latest
  restart: always
  ports:
    - 6379:6379
  volumes:
    - ./redis/data:/root/redis
    - ./redis/redis.conf:/usr/local/etc/redis/redis.conf
```

```
    networks:
      - app
secrets:
  database_password:
    file: secrets/database_password
  database_user:
    file: secrets/database_user
  database_name:
    file: secrets/database_name
  database_url:
    file: secrets/database_url
  email_host:
    file: secrets/email_host
  email_user:
    file: secrets/email_user
  email_password:
    file: secrets/email_password
  jwt_access_token_secret:
    file: secrets/jwt_access_token_secret
  jwt_refresh_token_secret:
    file: secrets/jwt_refresh_token_secret
  basic_auth_creds:
    file: secrets/basic_auth_creds
  redis_host:
    file: secrets/redis_host
  redis_password:
    file: secrets/redis_password
  s3_endpoint:
    file: secrets/s3_endpoint
  s3_access_key:
    file: secrets/s3_access_key
  s3_secret_key:
    file: secrets/s3_secret_key
  s3_bucket_name:
    file: secrets/s3_bucket_name
  s3_region:
    file: secrets/s3_region
  cdn_base_url:
    file: secrets/cdn_base_url

networks:
  app:
```

Приложение Ж

Скриншоты пользовательского интерфейса

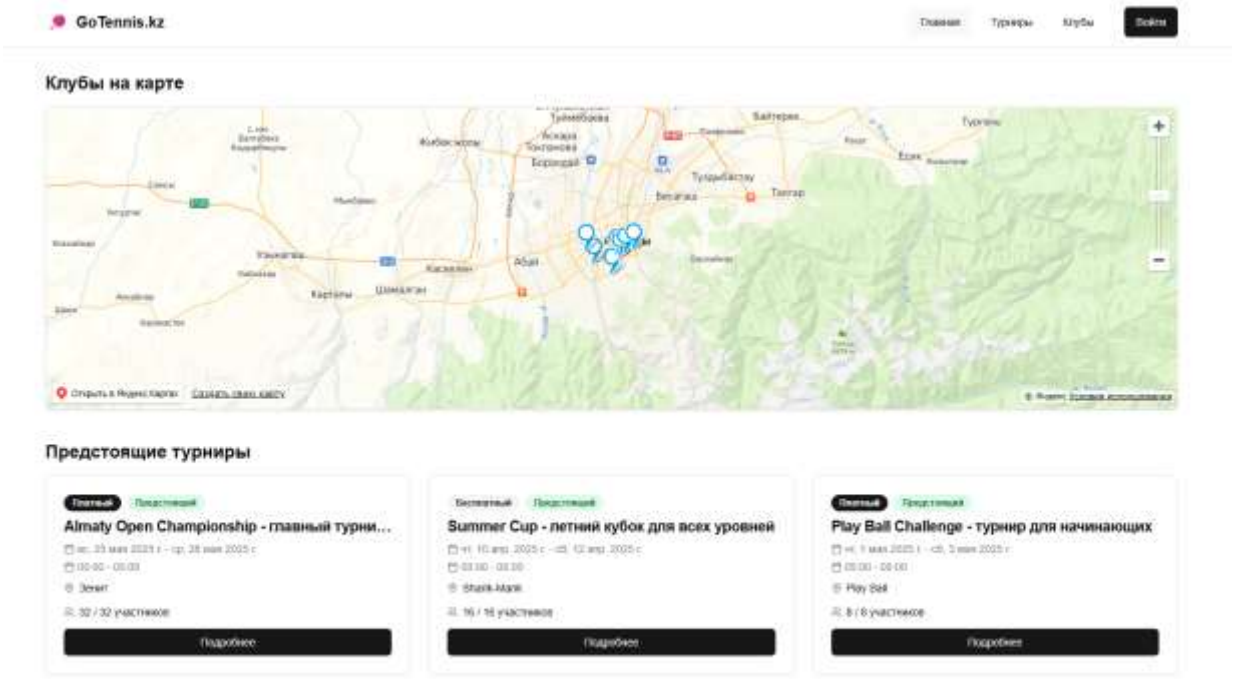

Рисунок Ж.1 – Главная страница

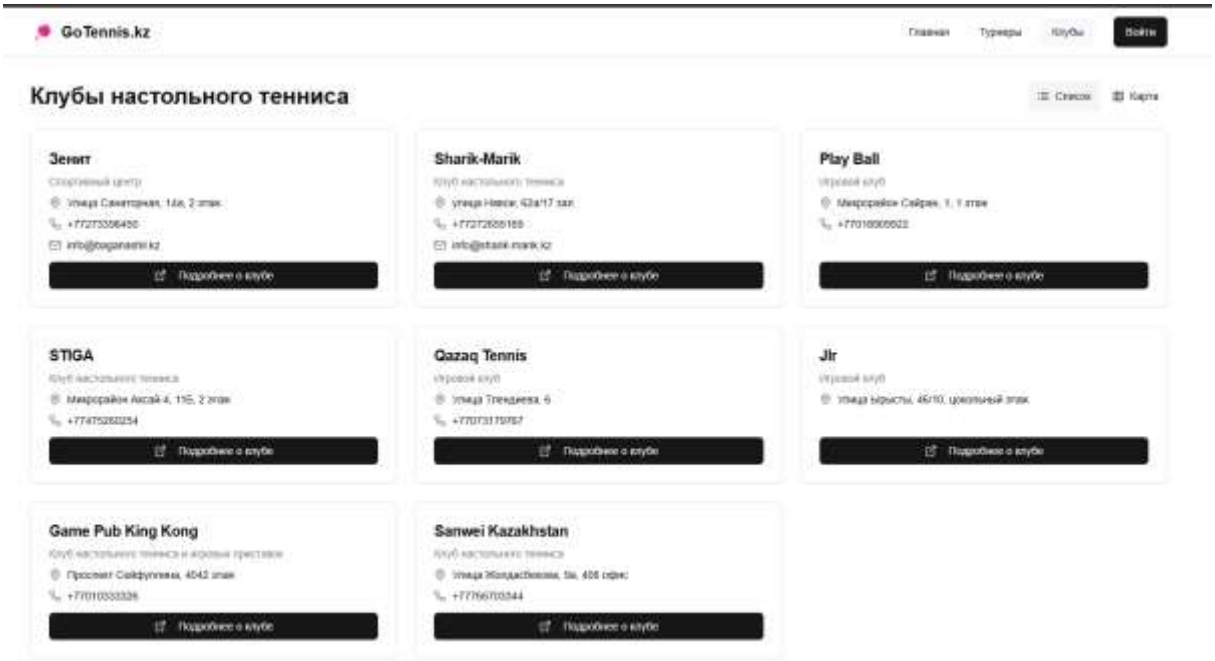

Рисунок Ж.2 – Страница списка клубов

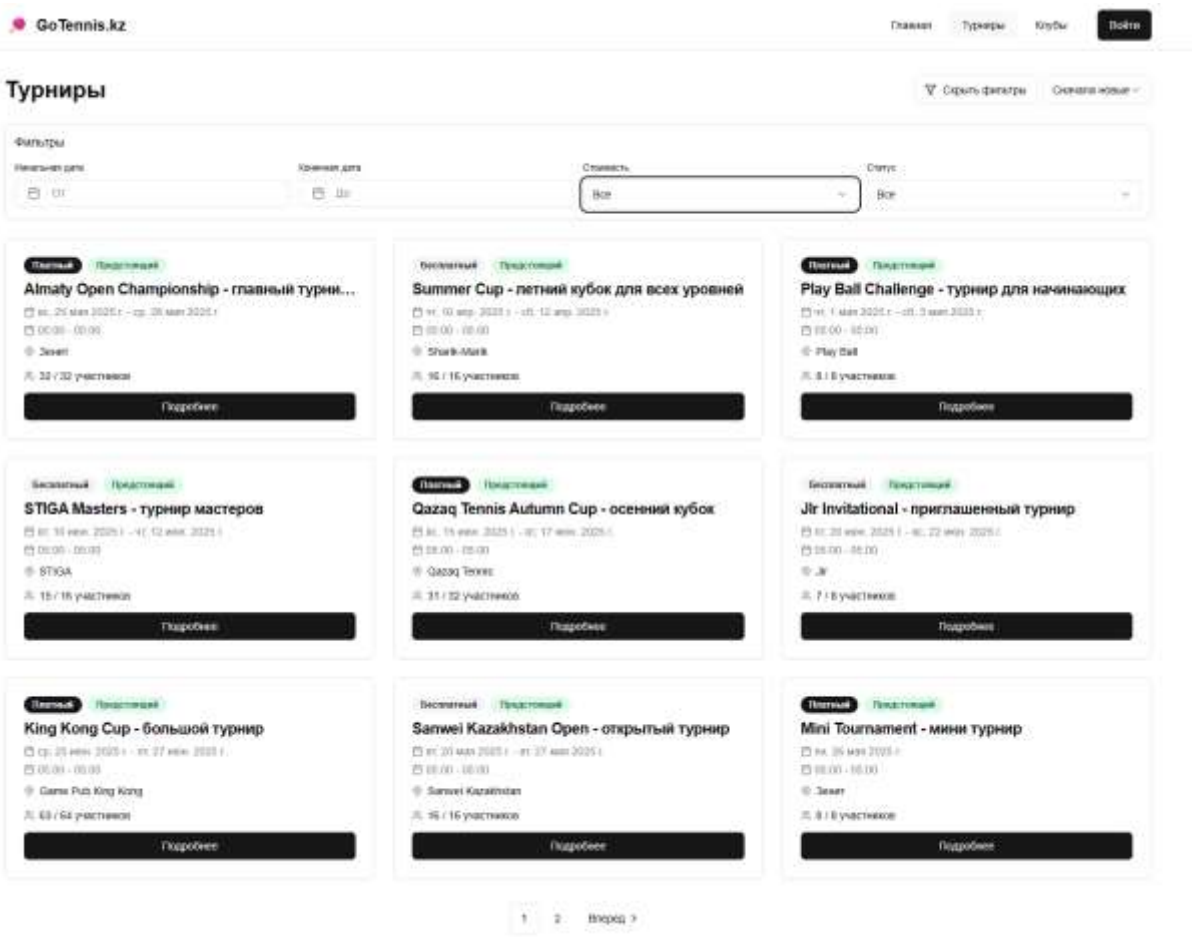

Рисунок Ж.3 – Страница списка турниров

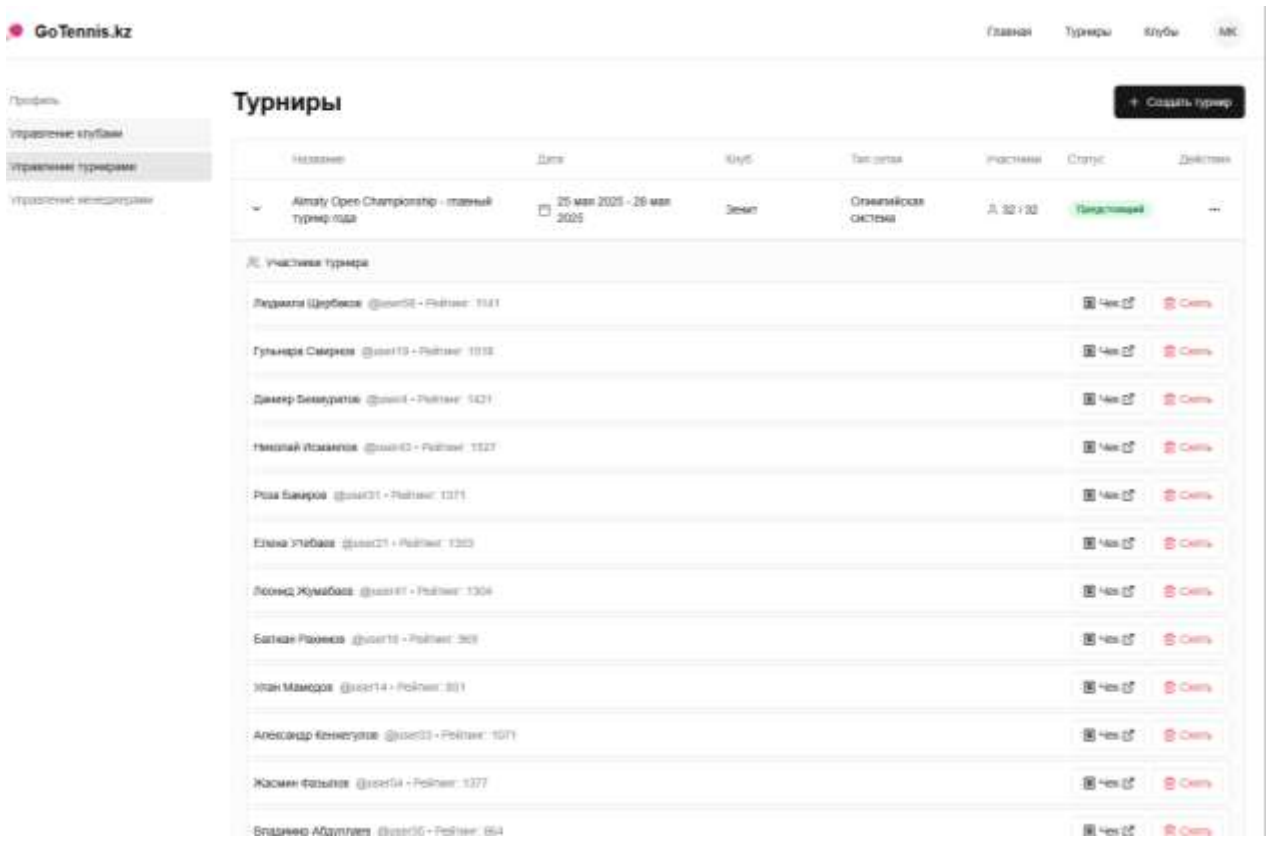

Рисунок Ж.4 – Страница управления турнирами

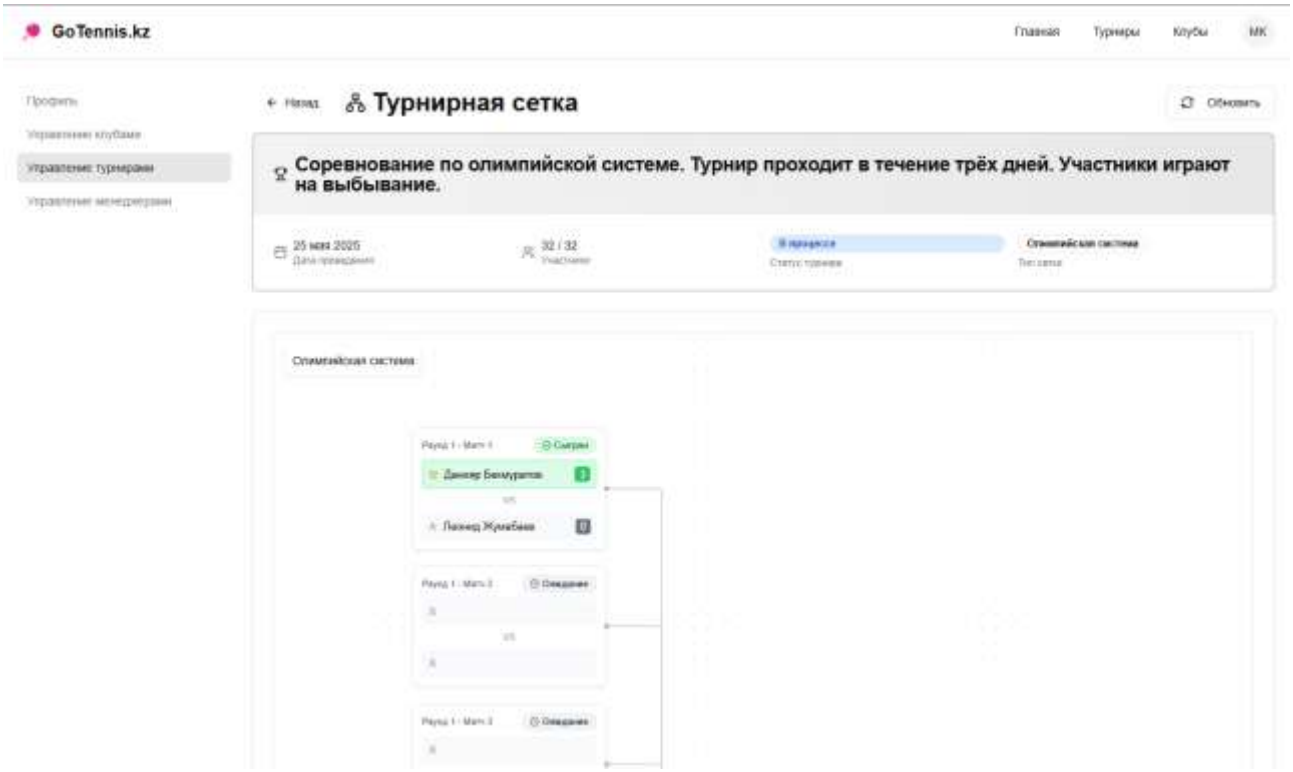

Рисунок Ж.5 – Страница редактирования турнирной сетки

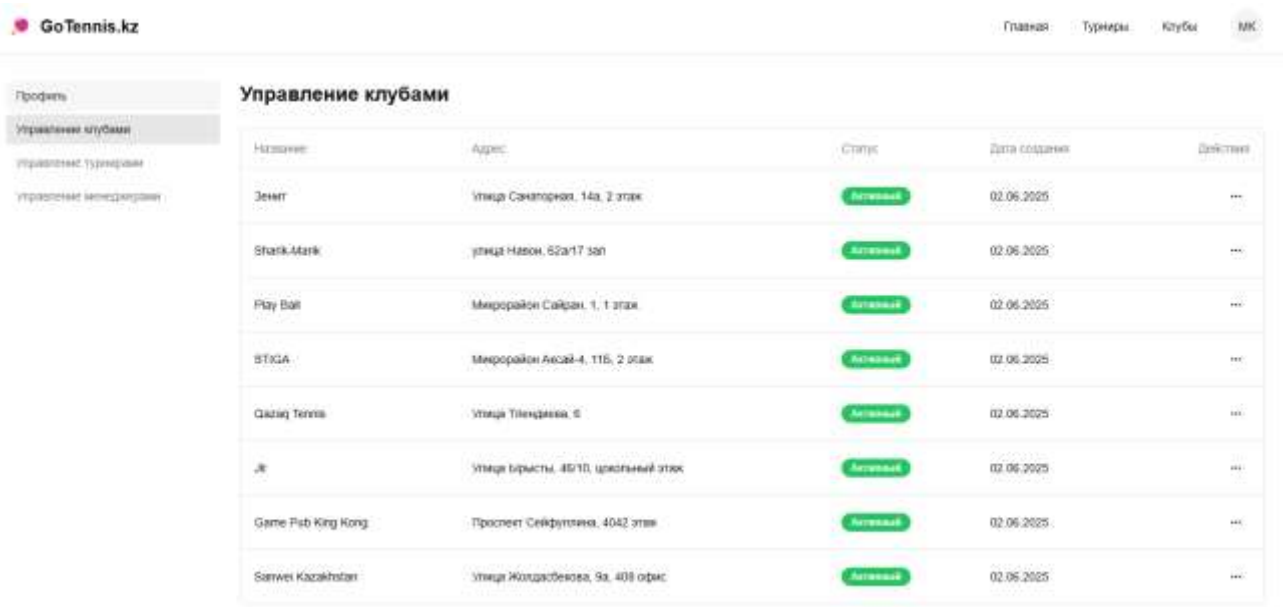

Рисунок Ж.6 – Страница управления клубами

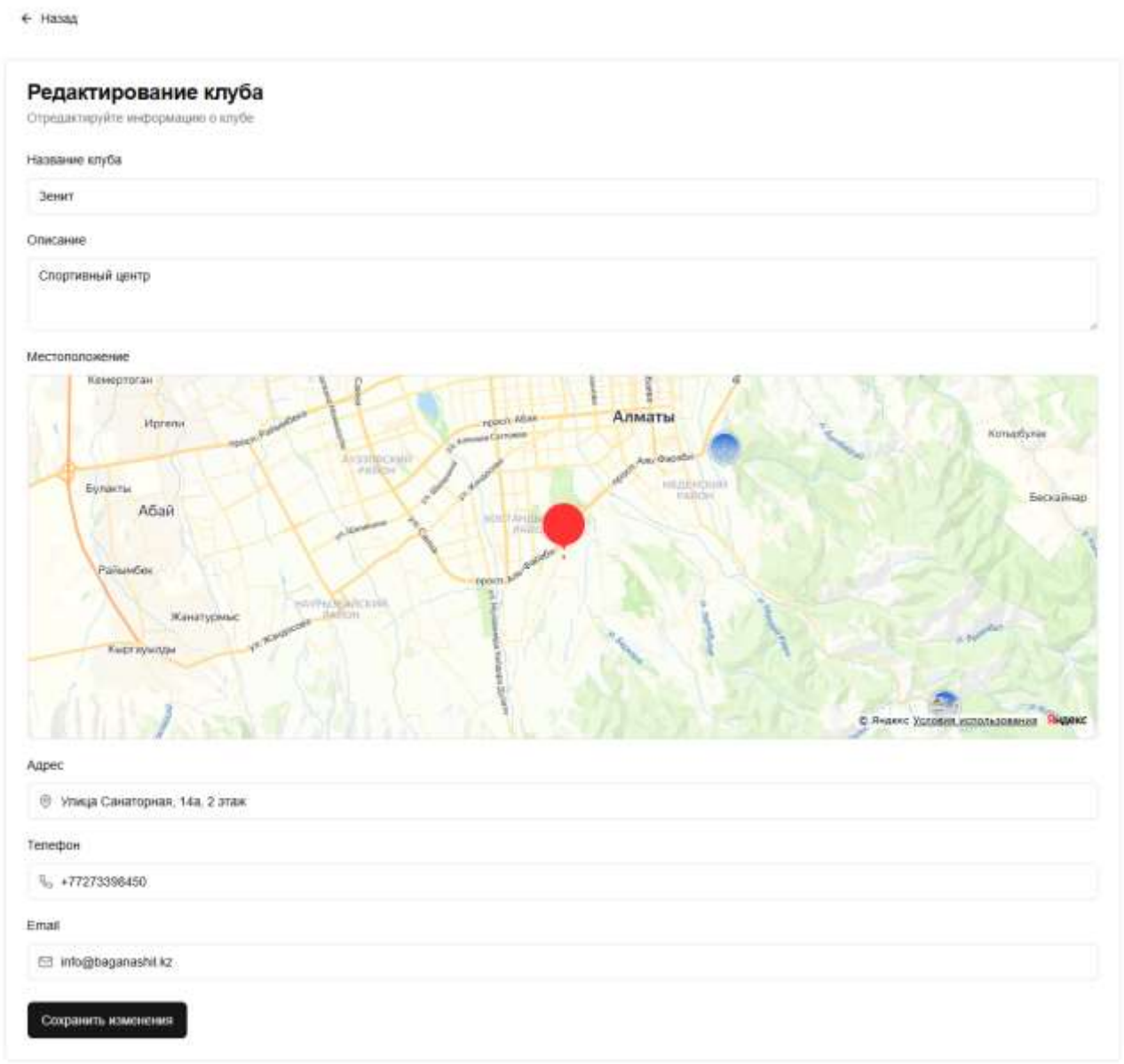

Рисунок Ж.7 – Страница редактирования клуба

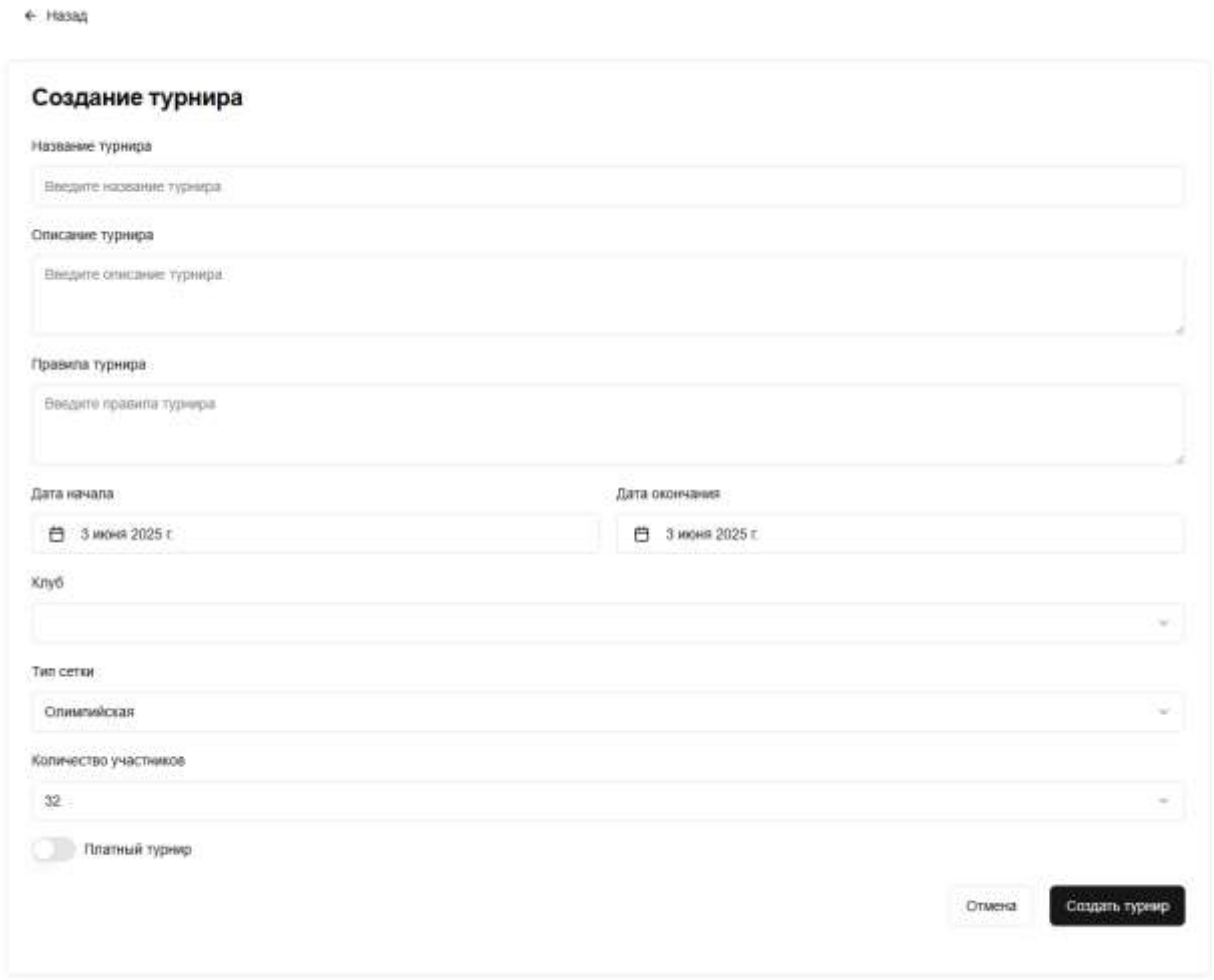

Рисунок Ж.8 – Страница создания турнира

Рисунок Ж.9 – Страница профиля

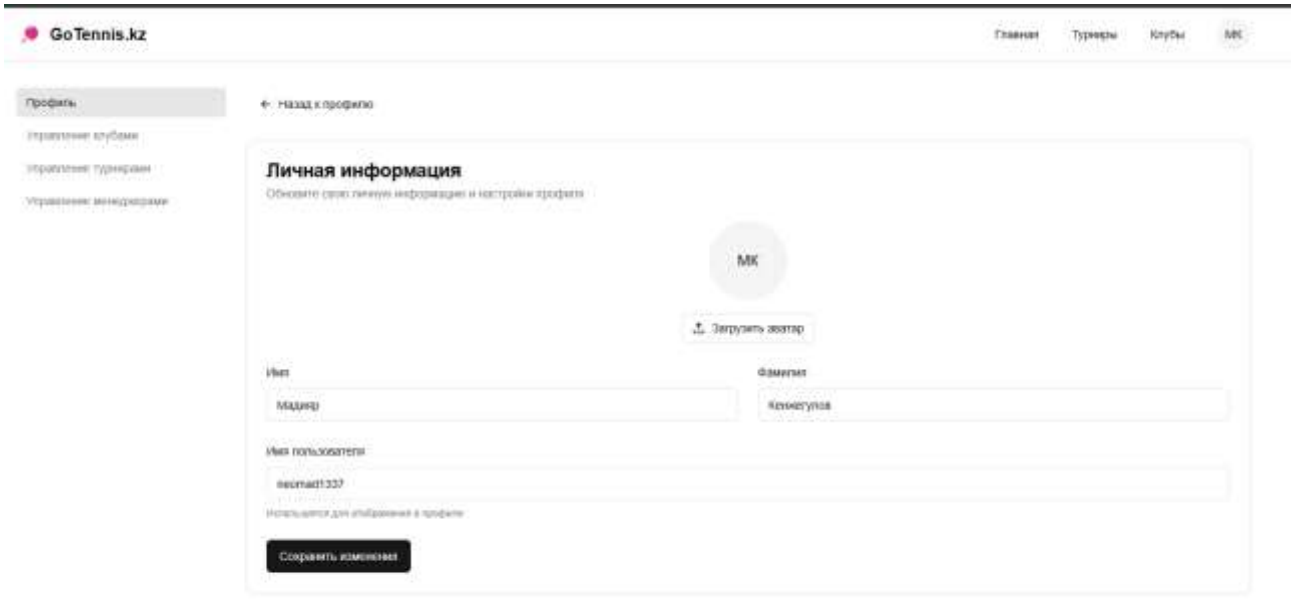

Рисунок Ж.10 – Страница редактирования профиля

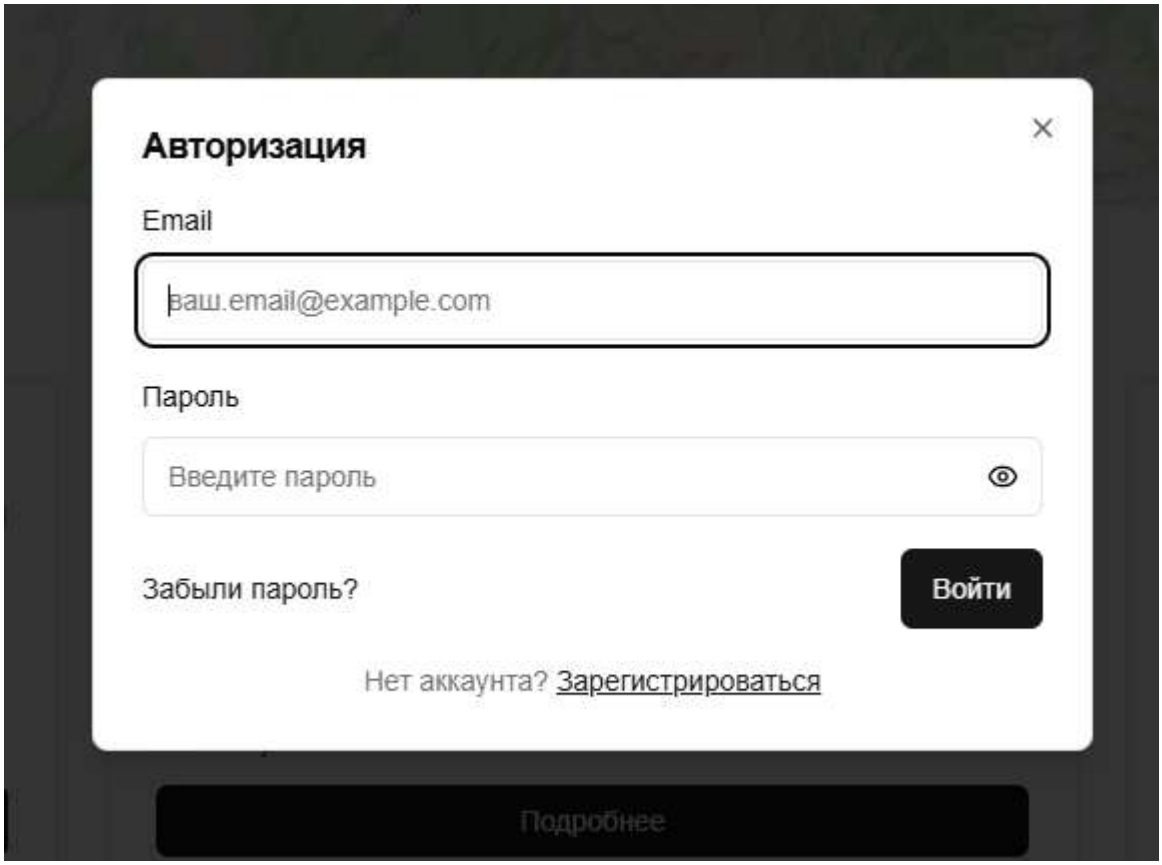

Рисунок Ж.11 – Модальное окно авторизации