**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 2](#_Toc200015785)

[**Задачи работы** 2](#_Toc200015786)

[**Объект исследования** 3](#_Toc200015787)

[**Предмет исследования** 3](#_Toc200015788)

[**Методы исследования** 3](#_Toc200015789)

[**Краткая структура** 3](#_Toc200015790)

[**1.** **ТЕОРЕТИЧЕСКАЯ ЧАСТЬ** 5](#_Toc200015791)

[**1.1 Анализ предметной области** 5](#_Toc200015792)

[**1.2 Анализ аналогов разрабатываемого решения** 7](#_Toc200015793)

[**1.3 Описание требований к разрабатываемому решению** 10](#_Toc200015794)

[**2.** **ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ** 14](#_Toc200015795)

[**2.1 Выбор инструментов и средств для реализации проекта** 14](#_Toc200015796)

[**2.2 Разработка проекта решения** 17](#_Toc200015797)

[**3. ПРАКТИЧЕСКАЯ ЧАСТЬ** 32](#_Toc200015798)

[**3.1 Разработка составных элементов** 32](#_Toc200015799)

[**3.2 Установка и настройка** 35](#_Toc200015800)

[**3.3 Тестирование** 36](#_Toc200015801)

[**3.4 Ввод в эксплуатацию** 39](#_Toc200015802)

[**3.5 Разработка сопроводительной документации** 39](#_Toc200015803)

[**ЗАКЛЮЧЕНИЕ** 41](#_Toc200015804)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 42](#_Toc200015805)

[**ПРИЛОЖЕНИЕ** 43](#_Toc200015806)

# **ВВЕДЕНИЕ**

**Актуальность работы**

Современный футбол – это не только про сам спорт, но и про соревнования, которым нужно эффективное управление данными о всем: игроках, матчах, командах и результатах. Во время модернизации спортивной индустрии, повышения требований к эффективности проведения соревнований, создание полной системы для организаторов футбольных турниров является важным фактором успеха для всех видов футбольных первенств.

Данная курсовая работа посвящена разработке программной системы для организаторов соревнований по футболу в рамках первенства страны. Во условиях растущей монетизации спорта, повышения значимости аналитических данных и обеспечения удобства для болельщиков, тема автоматизации процессов организации футбольного первенства будет особенно актуальна. Так же создание такой системы будет позволять сформировать новый подход к задачам информационной поддержки спортивных мероприятий с учетом новых требований к функциональности и пользовательскому опыту.

**Цель работы**

Целью данной курсовой работы является разработка программной системы для организаторов соревнований по футболу в рамках первенства страны. В системе будет разработана и интегрирована база данных, с помощью которой будет обеспечено управление и хранение информации о игроках, командах, стадионах, расписании матчей, их результатах и ценообразовании билетов на игры.

## **Задачи работы**

1. Провести анализ предметной области футбольных соревнований и организации спортивных мероприятий
2. Проанализировать аналоги систем управления футбольными соревнованиями.
3. Сформировать требования к разрабатываемой системе и базе данных
4. Разработать архитектуру системы с учетом потребностей организаторов соревнований.
5. Спроектировать и реализовать базу данных.
6. Разработать пакет процедур и функций для работы с данными.
7. Создать триггеры, обеспечивабщие каскадные изменения в связанных таблицах.
8. Разработать пользовательский интерфейс для организаторов соревнований
9. Составить отчет по работе.

## **Объект исследования**

Объектом исследования является система организации футбольных соревнований, которая включает в себя управление и учет информацией о игроках, стадионах, матчей и их результатах, а также о ценах на билеты.

## **Предмет исследования**

Предметом исследования является автоматизация процессов хранения и обработки информации о командах, игроках, стадионах, расписании матчей, их результатах, а также управление ценообразованием билетов на игры в рамках футбольного первенства страны.

## **Методы исследования**

В качестве основных методов исследования применены анализ, сравнение, моделирование и синтез. Практическая работа реализация поставленной задачи соответствует базовым подходам к разработке информационных систем и баз данных.

## **Краткая структура**

В этом отчете будет представлен процесс разработки для организаторов футбольных соревнований, в том числе теоретический обзор предметной области, анализ аналогов существующих решений, формирование требований к разрабатываемому решению, проектирование системы и базы данных, разработка пользовательского интерфейса, а так же тестирование работы системы и ведение документации.

1. **ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

## **1.1 Анализ предметной области**

**Футбольная индустрия и организация соревнований**

Футбольная индустрия представляет собой комплексную систему, включающую спортивные клубы, национальные федерации, лиги, игроков, тренеров, судей, стадионы и болельщиков. Организация футбольных соревнований – один из ключевых процессов в этой индустрии, требующий тщательного планирования, администрирования и информационного обеспечения [1].

Футбольные соревнования национального уровня значительно модернизировались за последние десятилетия. Современные футбольные первенства стран – это не только спортивные мероприятия, но и сложные бизнес-процессы, требующие эффективной информационной поддержки [2].

В данном контексте организаторам соревнований нужны комплексные решения, которые позволят эффективно управлять всеми аспектами турнира: от формирования календаря матчей до анализа статистики и формирования отчетности. Такие системы должны обеспечивать точность данных, прозрачность информации и удобство для всех участников футбольной экосистемы.

**Ключевые сущности в системе футбольных соревнований**

В контексте разрабатываемой системы можно выделить следующие ключевые сущности:

1. **Команда** — основная единица соревнований, представляющая определенный город, район или регион. Информация о команде включает в себя её название, город, где она базируется, имя главного тренера, место в турнирной таблице за прошый сезон и расписание встреч в текущем сезоне.
2. **Игрок** — участник команды, непосредственно принимающий участие в матчах. Для каждого игрока записываются фамилия и имя, возраст, номер на форме и амплуа в команде (вратарь, защитник, полузащитник, нападающий).
3. **Стадион** — спортивное сооружение, на котором проводятся матчи. Характеризуется местоположением (город), названием и вместимостью, что влияет на ценообразование билетов.
4. **Матч** — спортивное мероприятие, в котором принимают участие две команды. Информация о матче включает дату проведения, участвующие команды, стадион, где проходит сам матч и итоговый счет встречи.

**Процессы в организации футбольных соревнований**

Основными процессами, которые должны быть отражены в разрабатываемой системе, являются:

1. **Формирование расписания** — составление календаря игр с учетом требований регламента соревнований, возможностей стадионов и других ограничений (например, команда может участвовать только в одной встрече в день).
2. **Управление составами команд** — регистрация игроков, отслеживание изменений в составах, учет основных характеристик игроков.
3. **Учет результатов матчей** — фиксация счета встреч, формирование турнирной таблицы, расчет статистики.
4. **Управление стадионами** — учет информации о спортивных сооружениях, их вместимости и различных особенностях.
5. **Ценообразование билетов** — определение стоимости билета на матч в зависимости от вместимости стадиона и рейтинга встречающихся команд (с учетом их позиций в турнирной таблице прошлого сезона).
6. **Внесение изменений** — возможность модификации данных о командах, перенос встреч и другие корректировки в процессе проведения турнира.

Взаимосвязи между этими сущностями и процессами формируют основу для функционирования всей системы организации футбольных соревнований и позволяют реализовать необходимую функциональность для организаторов [3].

## **1.2 Анализ аналогов разрабатываемого решения**

В настоящее время на рынке представлено несколько типов систем для организации и управления футбольными соревнованиями. Рассмотрим наиболее известные аналоги разрабатываемой системы, чтобы выявить их сильные и слабые стороны, а также определить, что можно улучшить в моем решении.

**FIFA Connect**

FIFA Connect — платформа, разработанная Международной федерацией футбольных ассоциаций (FIFA) для централизованного управления регистрацией игроков и клубов на международном уровне.

**Сильные стороны:**

* Глобальная система идентификации игроков и клубов
* Строгие стандарты безопасности и защиты данных
* Интеграция с международными базами данных трансферов
* Поддержка многоязычности и адаптация к особенностям национальных федераций

**Слабые стороны:**

* Ориентация преимущественно на административные процессы, а не на организацию соревнований
* Сложность и избыточность системы для локальных турниров
* Ограниченная гибкость настройки под специфические требования отдельных лиг
* Высокая стоимость внедрения и поддержки [4]

**Sports TG (SportsTG)**

SportsTG — платформа для управления спортивными лигами, клубами и соревнованиями, широко используемая в различных видах спорта, включая футбол.

**Сильные стороны:**

* Комплексное решение, охватывающее регистрацию участников, управление соревнованиями и коммуникацию
* Удобный интерфейс для администраторов лиг и клубов
* Интегрированные инструменты для работы с болельщиками и спонсорами
* Гибкая система отчетности и аналитики

**Слабые стороны:**

* Универсальность решения может ограничивать реализацию специфических футбольных функций
* Недостаточная адаптация к национальным особенностям проведения соревнований
* Ограниченные возможности для глубокого анализа спортивных показателей команд и игроков [5]

**League Republic**

League Republic — специализированная платформа для организации футбольных лиг и турниров, ориентированная на пользовательскую простоту и эффективность.

**Сильные стороны:**

* Интуитивно понятный интерфейс для организаторов соревнований
* Автоматизированное формирование расписания матчей
* Интегрированная система управления судейским корпусом
* Публичный веб-сайт для болельщиков с актуальной информацией о турнире

**Слабые стороны:**

* Ограниченные возможности для крупных профессиональных лиг
* Недостаточно развитые инструменты финансового учета и ценообразования
* Отсутствие глубокой интеграции с телевизионными трансляциями и другими медиа-платформами [6]

**Альтернативный подход к решению**

На основе анализа существующих аналогов можно выделить несколько ключевых идей для совершенствования разрабатываемой системы организации футбольных соревнований:

1. **Специализация на национальном первенстве** — фокус на специфических требованиях и особенностях проведения национальных футбольных чемпионатов с учетом местных регламентов.
2. **Акцент на ценообразование билетов** — разработка гибкой системы формирования стоимости посещения матчей с учетом вместимости стадионов и статуса встречающихся команд.
3. **Улучшенная система управления изменениями** — создание эффективных механизмов для внесения корректировок в составы команд и расписание матчей в процессе проведения турнира.
4. **Интеграция статистических показателей** — разработка инструментов для детального анализа результатов матчей, формирования различных рейтингов команд и игроков.
5. **Удобство для организаторов** — создание интуитивно понятного интерфейса, ориентированного на потребности администраторов соревнований, с минимизацией рутинных операций.

Такой подход позволит создать систему, которая будет сочетать в себе лучшие характеристики существующих решений, при этом учитывая специфические потребности организаторов футбольных соревнований национального уровня.

## **1.3 Описание требований к разрабатываемому решению**

На основе анализа предметной области и изучения аналогов, можно сформулировать следующие требования к разрабатываемой системе для организаторов футбольных соревнований:

**Функциональные требования**

Требования к базе данных:

1. База данных должна обеспечивать хранение информации о следующих сущностях:
   * + Команды (название, город, главный тренер, место в таблице прошлого сезона)
     + Игроки (фамилия и имя, возраст, номер, амплуа)
     + Стадионы (город, название, вместимость)
     + Матчи (дата проведения, участвующие команды, стадион, счет)
2. База данных должна поддерживать выполнение следующих операций:
   * + Определение дат встреч указанной команды, ее противников и счета матчей
     + Вывод номеров и фамилий игроков команд, участвовавших во встрече на указанном стадионе в определенный день
     + Расчет цены билета на матч между указанными командами
     + Определение команд с наилучшей и наихудшей разницей забитых и пропущенных мячей
     + Формирование списка команд, занявших призовые места
     + Составление расписания игр по стадионам
3. База данных должна обеспечивать целостность данных при выполнении каскадных изменений в связанных таблицах с помощью соответствующих триггеров.

**Требования к функциональности системы:**

1. Управление командами:

* Добавление новых команд
* Редактирование информации о существующих командах
* Просмотр состава команды

1. Управление игроками:

* Добавление новых игроков в команду
* Редактирование информации об игроках
* Удаление игроков из команды

1. Управление стадионами:

* Добавление новых стадионов
* Редактирование информации о стадионах
* Просмотр расписания матчей на конкретном стадионе

1. Управление матчами:

* Формирование расписания встреч
* Внесение результатов проведенных матчей
* Перенос встреч при необходимости

1. Управление ценообразованием:

* Автоматический расчет цены билетов на основе вместимости стадиона и рейтинга команд
* Возможность ручной корректировки стоимости билетов

1. Формирование отчетов:

* Турнирная таблица
* Статистика команд и игроков
* Расписание предстоящих матчей
* Результаты прошедших встреч

**Нефункциональные требования**

1. Производительность:

* Система должна обеспечивать быстрый отклик на запросы пользователей (не более 2 секунд для стандартных операций)
* База данных должна эффективно обрабатывать запросы даже при большом объеме данных (до 20 команд, 500 игроков, 100 матчей)

1. Безопасность:

* Защита данных от несанкционированного доступа
* Разграничение прав доступа в соответствии с ролями пользователей
* Протоколирование критически важных операций

1. Масштабируемость:

* Возможность расширения функциональности системы без значительной перестройки архитектуры
* Поддержка увеличения объема данных и количества пользователей

1. Удобство использования:

* Интуитивно понятный интерфейс для организаторов соревнований
* Минимизация количества действий пользователя для выполнения типовых операций
* Наличие справочной информации и подсказок

1. Надежность:

* Обеспечение целостности данных при выполнении транзакций
* Регулярное резервное копирование данных
* Обработка исключительных ситуаций без потери данных

*Таблица 1.1 – Требования к разрабатываемой системе*

|  |  |  |
| --- | --- | --- |
| **№** | **Требование** | **Значение** |
| 1 | Структура базы данных | База данных должна содержать сущности "Команда", "Игрок", "Стадион", "Матч" с соответствующими атрибутами и связями между ними |
| 2 | Корректность работы | Система должна запускаться и поддерживать стабильный цикл работы от момента старта до завершения |
| 3 | Функции управления командами | Система должна обеспечивать возможность добавления новых команд, редактирования информации о существующих командах и просмотра состава команды |
| 4 | Функции управления игроками | Система должна обеспечивать возможность добавления новых игроков в команду, редактирования информации об игроках и удаления игроков из команды |
| 5 | Функции управления стадионами | Система должна обеспечивать возможность добавления новых стадионов, редактирования информации о стадионах и просмотра расписания матчей на конкретном стадионе |
| 6 | Функции управления матчами | Система должна обеспечивать возможность формирования расписания встреч, внесения результатов проведенных матчей и переноса встреч при необходимости |
| 7 | Ценообразование билетов | Система должна обеспечивать автоматический расчет цены билетов на основе вместимости стадиона и рейтинга команд, а также возможность ручной корректировки стоимости |
| 8 | Отчеты | Система должна обеспечивать формирование турнирной таблицы, статистики команд и игроков, расписания предстоящих матчей и результатов прошедших встреч |

# **ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ**

## **2.1 Выбор инструментов и средств для реализации проекта**

При разработке программы для организаторов соревнований по футболу в рамках первенства страны, необходимо выбрать оптимальные технологии, которые будут обеспечивать эффективную реализацию всевозможные требования проекта. Ниже представлен сравнительный анализ возможных технологических решений для каждого компонента системы.

**2.1.1 Система управления базами данных (СУБД)**

Для хранения данных необходима надежная СУБД. Рассмотрим варианты:

*Таблица 2.1 — Сравнение СУБД для реализации проекта*

| **Критерий** | **PostgreSQL** | **MySQL** | **Oracle Database** | **MS SQL Server** |
| --- | --- | --- | --- | --- |
| Производительность при сложных запросах | Высокая | Средняя | Высокая | Высокая |
| Поддержка триггеров и процедур | Полная | Частичная | Полная | Полная |
| Открытый исходный код | Да | Да | Нет | Нет |
| Совместимость с различными ОС | Высокая | Высокая | Средняя | Низкая |
| Сложность администрирования | Средняя | Низкая | Высокая | Средняя |
| Безопасность | Высокая | Средняя | Высокая | Высокая |
| Маштабируемость | Высокая | Средняя | Высокая | Высокая |

**Вывод:** Для реализации проекта будет использоваться СУБД PostgreSQL, так как она обеспечивает:

* Подходит под все требования разработки
* Высокую производительность при обработке сложных запросов
* Открытый исходный код, что уменьшает стоимость разработки
* Хорошую масштабируемость, с помощью чего в будущем будет возможность расширять систему
* Отличную совместимость с выбранными технологиями разработки

**2.1.2 Технологии backend-разработки**

Для реализации серверной части приложения рассмотрены следующие технологии:

*Таблица 2.2 — Сравнение технологий backend-разработки*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Критерий** | **FastAPI (Python)** | **Django (Python)** | **Spring Boot (Java)** | **ASP.NET Core (C#)** |
| Производительность | Высокая (асинхронная обработка) | Средняя | Высокая | Высокая |
| Поддержка ORM | SQLAlchemy, Tortoise ORM | Django ORM | Hibernate | Entity Framework |
| Безопасность | Встроенные механизмы | Высокая | Высокая | Высокая |
| Документация | Активно растущая | Обширная | Обширная | Обширная |
| Скорость разработки | Очень высокая | Высокая | Средняя | Средняя |
| Асинхронность | Нативная поддержка | Ограниченная | Через WebFlux | Через Task API |

**Вывод:** Для реализации backend-части системы выбран фреймворк FastAPI по следующим причинам:

* Очень высокая производительность благодаря асинхронной обработке запросов
* Автоматическая генерация Swagger документации
* Встроенная валидация данных через Pydantic
* Легкая интеграция с PostgreSQL с помощью библиотеки SQLAlchemy
* Современная архитектура, соответствующая принципам REST

**2.1.3 Технологии frontend-разработки**

Для разработки пользовательского интерфейса системы рассмотрены следующие фреймворки:

*Таблица 2.3 — Сравнение технологий frontend-разработки*

| **Критерий** | **React** | **Angular** | **Vue.js** | **Svelte** |
| --- | --- | --- | --- | --- |
| Производительность | Высокая | Средняя | Высокая | Очень высокая |
| Количество Библиотек | Обширное (около 100 тысяч) | Хорошее (около 10 тысяч) | Хорошее (около 10000) | Растущее (около 1000) |
| Компонентный подход | Да | Да | Да | Да |
| Типизация | Java Script + TypeScript (опционально) | TypeScript | Java Script + TypeScript (опционально) | Java Script + TypeScript (опционально) |
| Поддержка мобильного интерфейса | React Native | Ionic, NativeScript | Vue Native | Limited |
| Размер бандла | Средний | Большой | Маленький | Очень маленький |

**Вывод:** Для разработки frontend-части выбран фреймворк React с использованием TypeScript по следующим причинам:

* Подходит под все требования разработки
* Высокая производительность
* Компонентный подход
* Высокая надежность кода благодаря Typyscript
* Большой выбор библиотек

**2.1.4 Средства проектирования и моделирования**

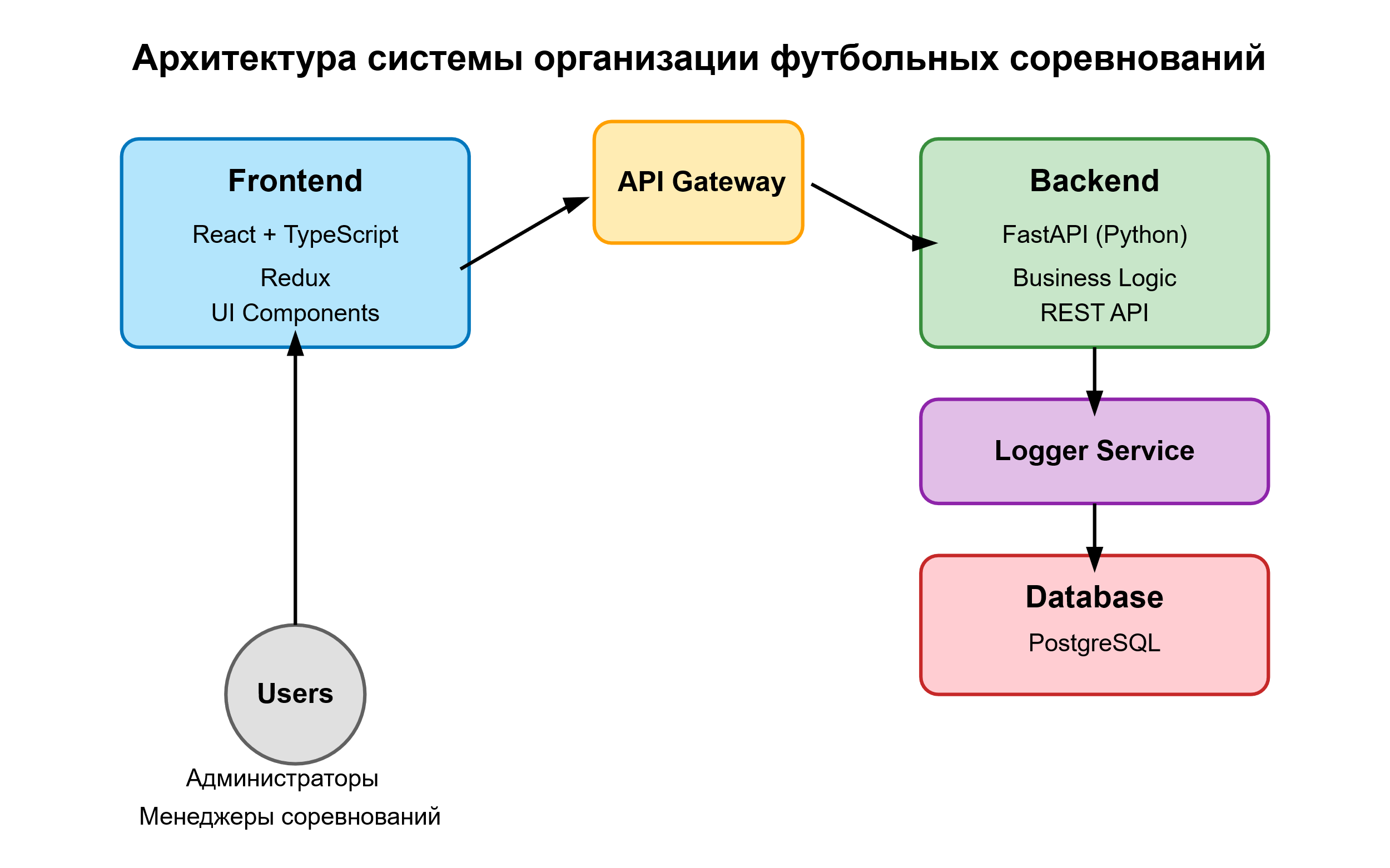
Для проектирования архитектуры системы и создания диаграмм выбраны следующие инструменты:

* **Enterprise Architect** – для создания UML-диаграмм (диаграммы классов, последовательностей, состояний и т.д.)
* **DbDesigner** – для проектирования структуры базы данных и создания ER-диаграмм
* **Swagger** — для автоматического документирования REST API
* **Mermaid** — для создания и поддержки процессных диаграмм

## **2.2 Разработка проекта решения**

**2.2.1 Архитектура системы**

Разрабатываемая система построена на основе архитектуры распределенного монолита с разделением на frontend и backend части. Схема архитектуры системы представлена на рисунке 2.1.



*Рисунок 2.1 — Общая архитектура системы*

Ключевые части архитектуры:

1. **Клиентская часть (Frontend)** – React-приложение, обеспечивающее взаимодействие пользователя с системой через графический интерфейс.
2. **Серверная часть (Backend)** — FastAPI-приложение, реализующее логику и предоставляющее REST API для взаимодействия с клиентской частью.
3. **База данных** – PostgreSQL, хранящая все данные.
4. **API Gateway** – компонент, обеспечивающий маршрутизацию запросов и базовую авторизацию.
5. **Logger Service** — компонент для ведения логирования действий пользователей.

Взаимодействие между клиентской и серверной частями осуществляется с помощью REST API с использованием JSON для обмена данными.

**2.2.2 Проектирование базы данных**

На основе анализа предметной области разработана схема базы данных, представленная на рисунке 2.2.



*Рисунок 2.2 — ER-диаграмма базы данных (нотация Crow's Foot)*

База данных включает следующие основные таблицы:

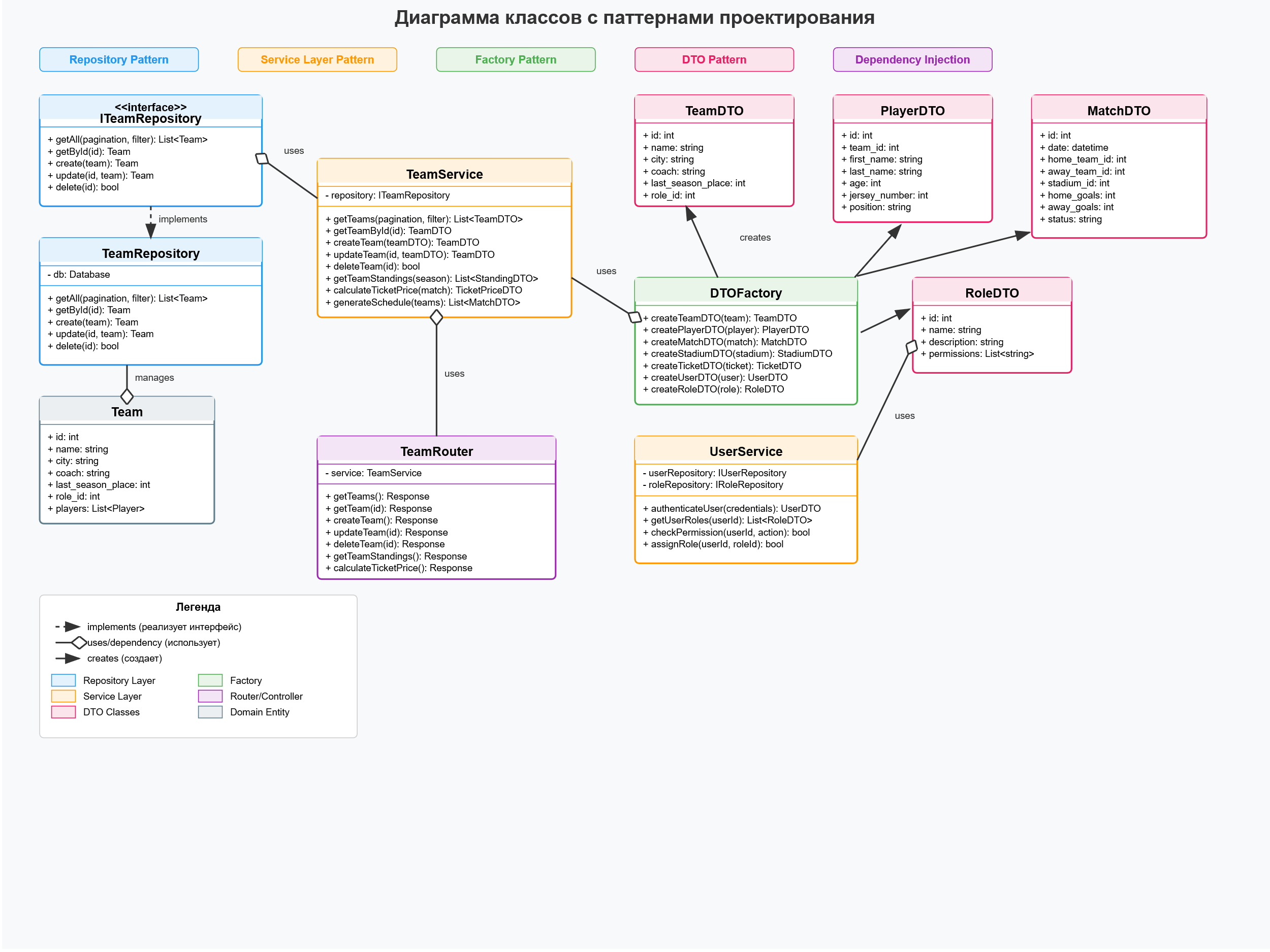
1. **Teams (Команды)**
   * team\_id (PK) – уникальный идентификатор команды
   * name – название команды
   * city – город, который представляет команда
   * coach – имя главного тренера
   * last\_season\_place – место команды в турнирной таблице прошлого сезона
2. **Players (Игроки)**
   * player\_id (PK) – уникальный идентификатор игрока
   * team\_id (FK) – ссылка на команду
   * first\_name – имя игрока
   * last\_name – фамилия игрока
   * age – возраст игрока
   * jersey\_number – номер на форме
   * position – амплуа (вратарь, защитник, полузащитник, нападающий)
3. **Stadiums (Стадионы)**
   * stadium\_id (PK) – уникальный идентификатор стадиона
   * name – название стадиона
   * city – город, в котором расположен стадион
   * capacity – вместимость стадиона
4. **Matches (Матчи)**
   * match\_id (PK) – уникальный идентификатор матча
   * date – дата проведения матча
   * home\_team\_id (FK) – ссылка на команду хозяев
   * away\_team\_id (FK) – ссылка на команду гостей
   * stadium\_id (FK) – ссылка на стадион
   * home\_goals – количество голов команды хозяев
   * away\_goals – количество голов команды гостей
   * status – статус матча (запланирован, завершен, отменен, перенесен)
5. **Tickets (Билеты)**
   * ticket\_id (PK) – уникальный идентификатор типа билета
   * match\_id (FK) – ссылка на матч
   * category – категория билета (VIP, стандарт, эконом)
   * price – стоимость билета
6. **Users (Пользователи системы)**
   * user\_id (PK) – уникальный идентификатор пользователя
   * username – имя пользователя
   * password\_hash – хеш пароля
   * role – роль в системе (администратор, менеджер)
   * email – электронная почта
7. **Audit\_Log (Журнал аудита)**
   * log\_id (PK) – уникальный идентификатор записи
   * user\_id (FK) – ссылка на пользователя
   * action – выполненное действие
   * entity\_type – тип сущности, с которой выполнено действие
   * entity\_id – идентификатор сущности
   * timestamp – дата и время действия
   * details – дополнительные сведения
8. **Roles (Роли в программе)**
   * role\_id (PK) – индетификатор роли
   * role\_name (FK) – наименование роли

**2.2.3 Паттерны проектирования**

В разрабатываемой системе реализованы следующие паттерны проектирования для взаимодействия с FastAPI и Python:

1. **Repository Pattern** — для скрытия доступа к данным и отделения бизнес-логики от деталей работы с базой данных.
2. **Dependency Injection** — для управления зависимостями и облегчения тестирования (стандартный механизм FastAPI).
3. **Service Layer** — для инкапсуляции бизнес-логики и создания четкого интерфейса.
4. **Factory** — для создания объектов без указания конкретных классов.
5. **Data Transfer Object (Pydantic models)** — для валидации входных данных.

Диаграмма классов, иллюстрирующая применение паттернов проектирования, представлена на рисунке 2.3.



*Рисунок 2.3 — Диаграмма классов с применением паттернов проектирования*

**2.2.4 Проектирование серверной части (Backend)**

Серверная часть разработана с использованием FastAPI организована в соответствии с многослойной архитектурой:

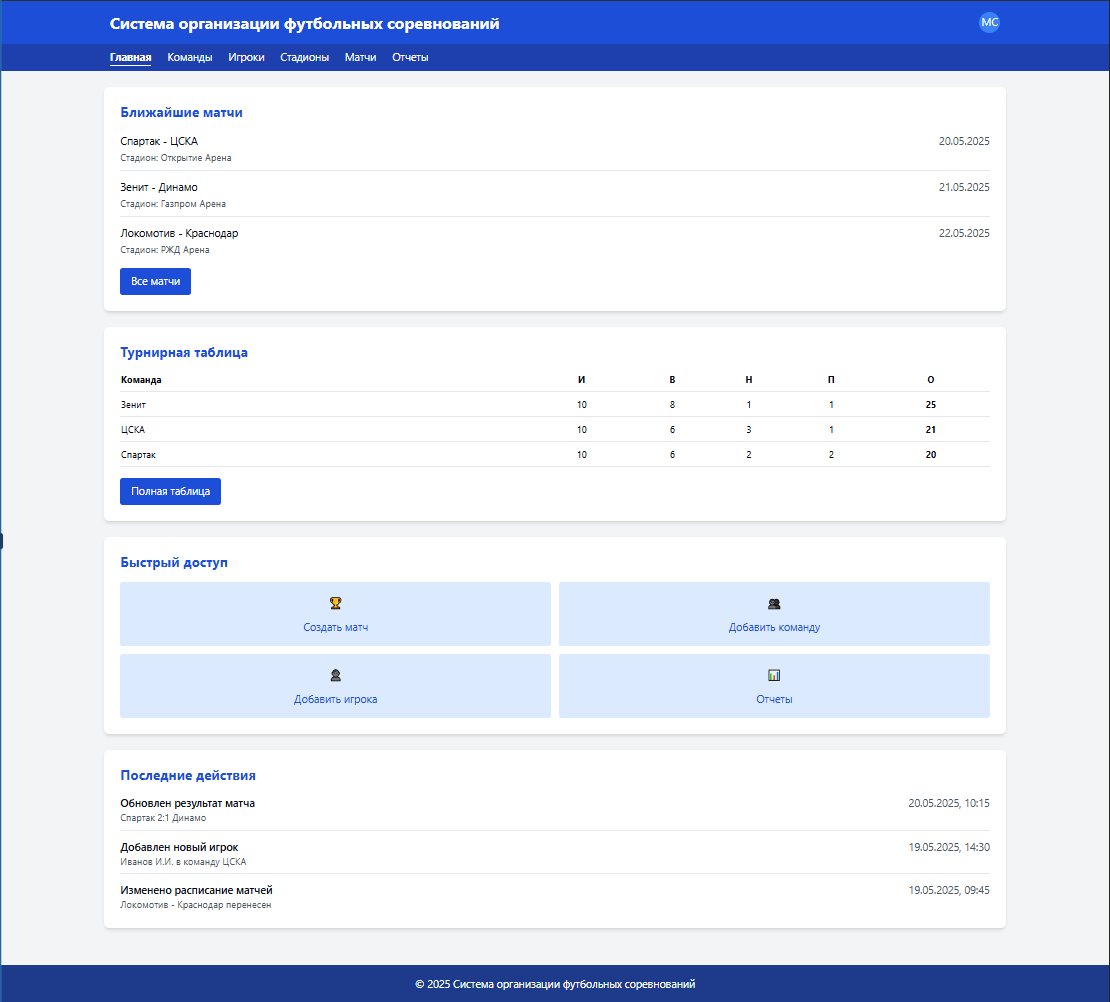
1. **API Layer (Routers)** — обрабатывает HTTP-запросы, преобразует данные и возвращает HTTP-ответы согласно REST-принципам.
2. **Service Layer** — содержит бизнес-логику приложения и координирует работу с данными.
3. **Repository Layer** — обеспечивает взаимодействие с базой данных через SQLAlchemy.
4. **Domain Model** — представляет основные сущности предметной области через Pydantic модели.

**2.2.5 Проектирование клиентской части (Frontend)**

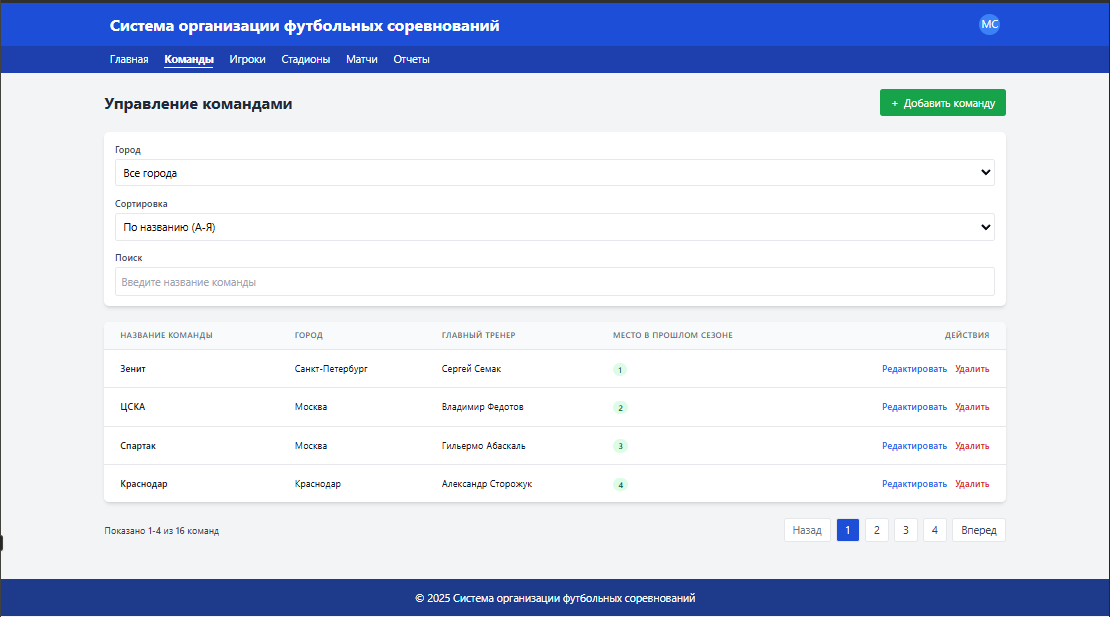
Клиентская часть разработана с использованием React и TypeScript и организована в соответствии с компонентным подходом.

**Макет пользовательского интерфейса**

На основе требований разработаны прототипы пользовательского интерфейса. Примеры макетов основных экранов представлены на рисунках 2.4-2.5.



*Рисунок 2.4 — Макет главной страницы системы*



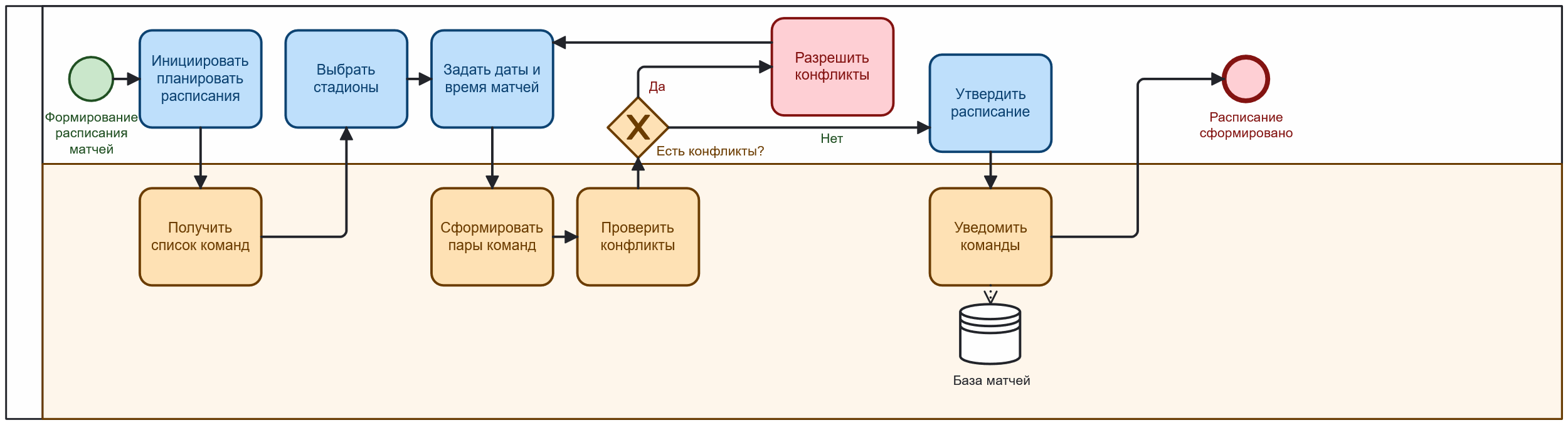
*Рисунок 2.5 — Макет страницы управления командами*

При разработке пользовательского интерфейса учтены следующие принципы:

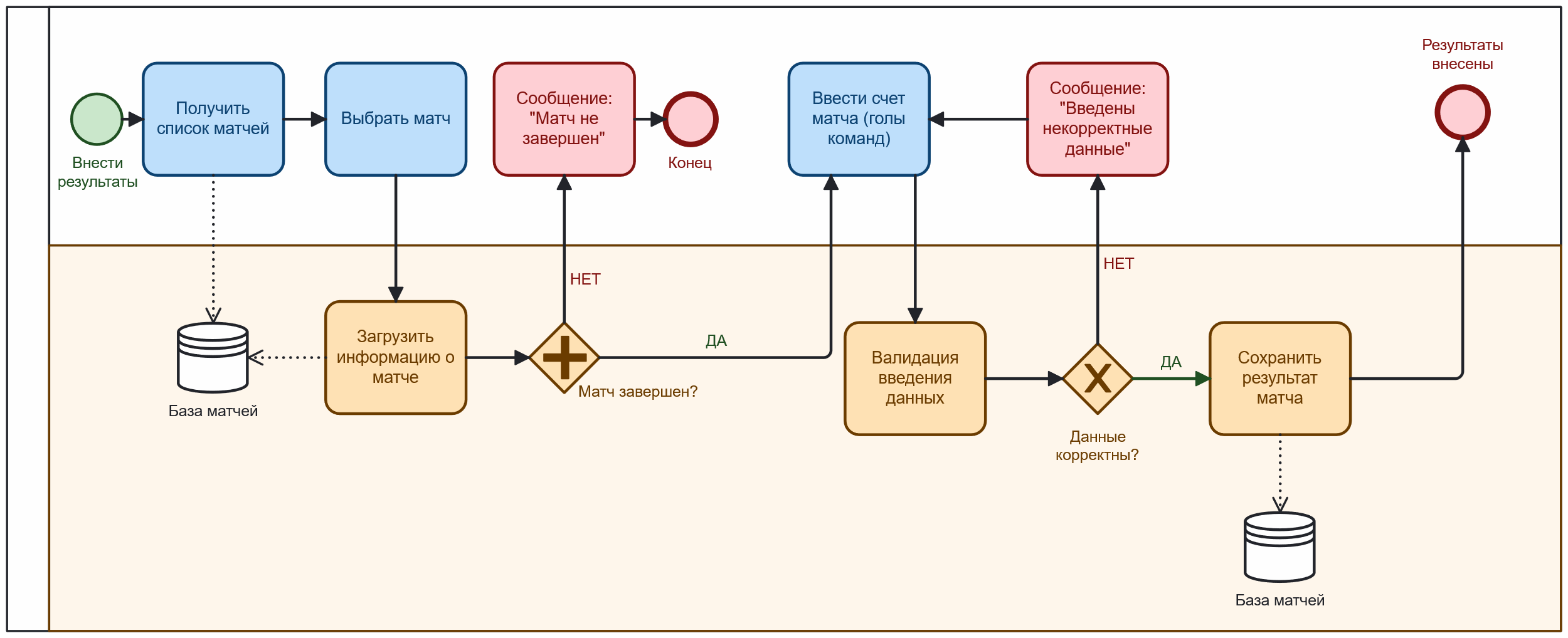
* **Единообразие** — все элементы управления имеют схожий стиль и расположены в соответствии с общей логикой интерфейса.
* **Информативность** — пользователю предоставляется вся необходимая информации, с возможностью получения дополнительных сведений при необходимости.
* **Предотвращение ошибок** — форматы ввода данных заявлены однозначно, также предусмотрена валидация вводимых значений.
* **Обратная связь** — система информирует пользователя о результатах выполнения операций и возможных проблемах.
* **Адаптивность** — интерфейс корректно отображается на различных устройствах и при различных разрешениях экрана.

**2.2.6 Процессы и алгоритмы**

Для визуализации основных бизнес-процессов системы разработаны диаграммы процессов, демонстрирующие последовательность действий и взаимодействие участников.



*Рисунок 2.6 —диаграмма процесса формирования расписания матчей*



*Рисунок 2.7 —диаграмма процесса внесения результатов матча*

**Алгоритм расчета стоимости билетов**

Одним из ключевых алгоритмов системы является расчет стоимости билетов на матч. Алгоритм учитывает следующие факторы:

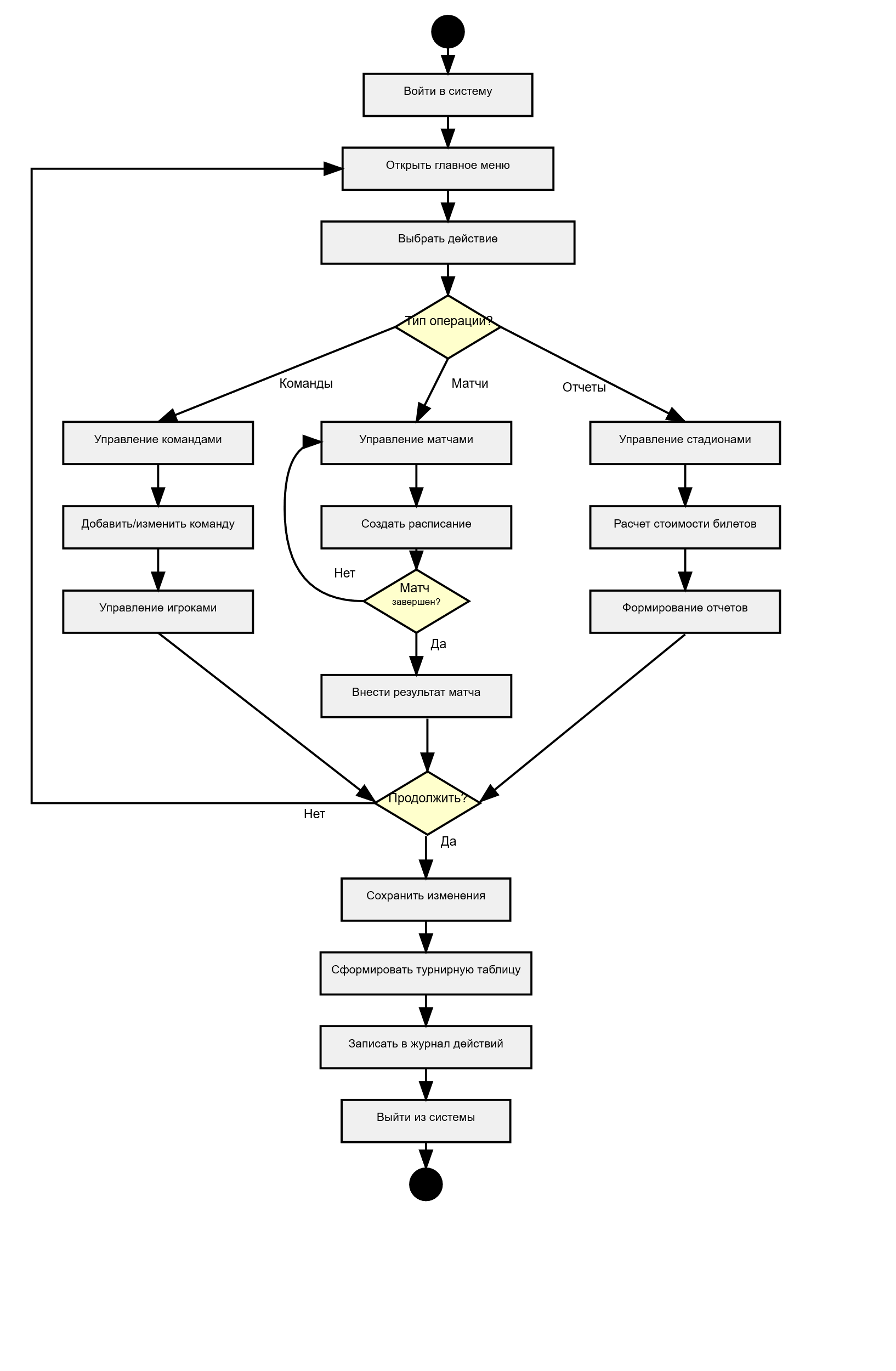
* Вместимость стадиона
* Положение команд в турнирной таблице прошлого сезона
* Категорию билета (VIP, стандарт, эконом)

Формула расчета стоимости билета:

1. Базовая стоимость = 500 - (Вместимость стадиона / 10000) \* 50
2. Коэффициент престижа = (21 - (Место команды 1 + Место команды 2) / 2) / 10
3. Стоимость билета = Базовая стоимость \* Коэффициент престижа \* Множитель категории
4. Где множитель категории равен:

* VIP: 2.5
* Стандарт: 1.0
* Эконом: 0.7

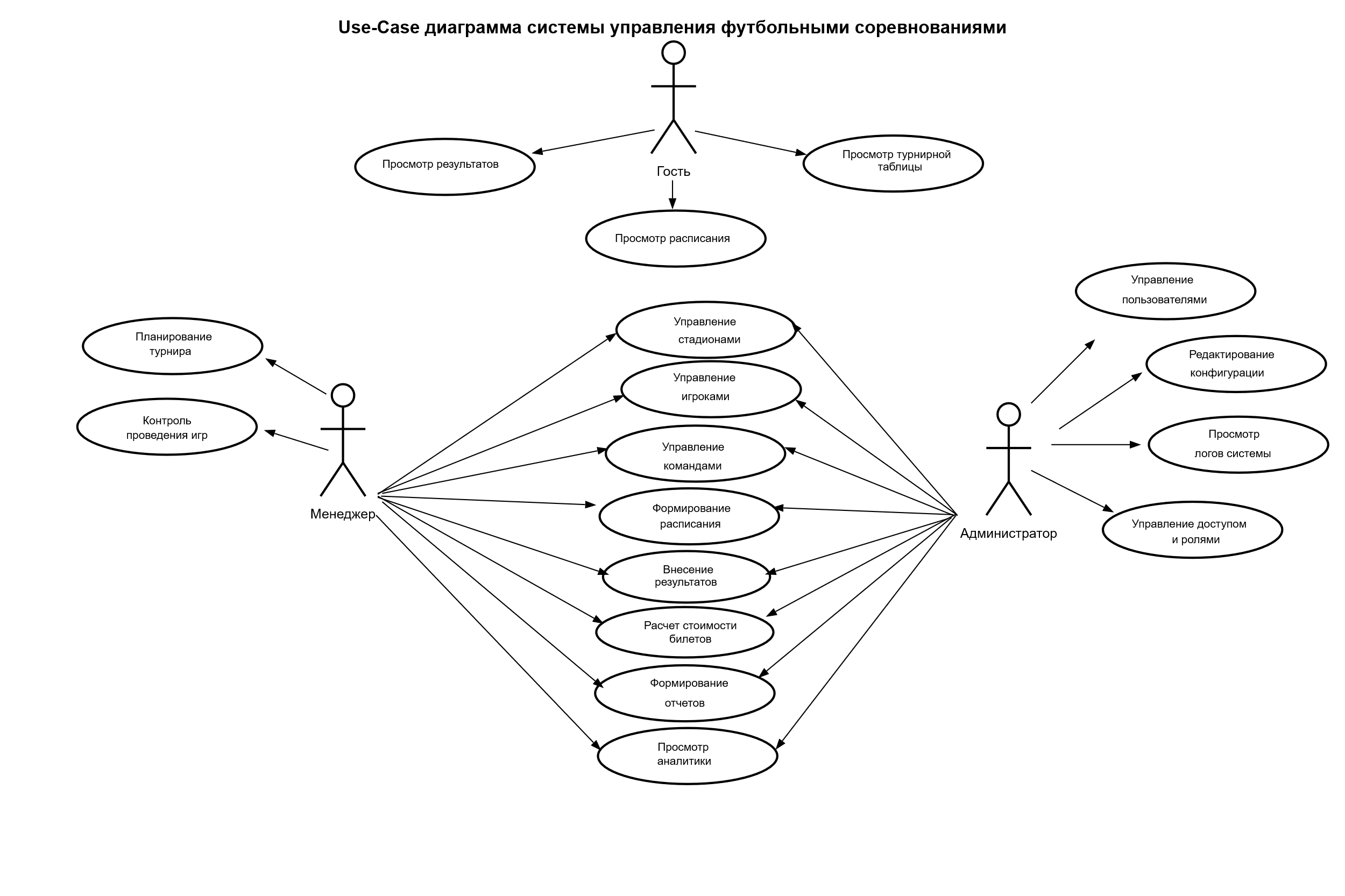
Диаграмма активности (рисунок 2.8) демонстрирует последовательность действий менеджера соревнований при работе с системой. После авторизации пользователь может выбрать один из трех основных типов операций: управление командами и игроками, управление матчами или работу со стадионами и отчетами. Каждый поток включает соответствующие действия, после выполнения которых система предлагает продолжить работу или завершить сессию с сохранением всех изменений и записью в журнал аудита.



*Рисунок 2.8 — Диаграмма активности менеджера соревнований*

**2.2.7 Use-Case диаграмма**

Для наглядного представления функциональности системы и взаимодействия пользователей с ней разработана Use-Case диаграмма, представленная на рисунке 2.13.



*Рисунок 2.9 — Use-Case диаграмма системы*

На диаграмме выделены следующие действующие лица:

1. **Администратор системы** – имеет полный доступ ко всем функциям системы, включая управление пользователями.
2. **Менеджер соревнований** – осуществляет управление командами, игроками, стадионами, матчами и формирует отчеты.

Основные прецеденты использования:

* Управление командами
* Управление игроками
* Управление стадионами
* Формирование расписания матчей
* Внесение результатов матчей
* Расчет стоимости билетов
* Формирование отчетов
* Управление пользователями системы

**2.2.8 Безопасность системы**

Для обеспечения безопасности данных и предотвращения несанкционированного доступа в системе реализованы следующие механизмы:

1. **Аутентификация и авторизация** – использование JWT-токенов для проверки подлинности пользователей и определения их прав доступа.
2. **Логирование действий пользователей** – все критические операции фиксируются в логах с указанием пользователя, времени и типа действия.
3. **Хеширование паролей** – хранение паролей в виде хешей с использованием алгоритма BCrypt.
4. **Валидация входных данных** – проверка всех входных данных на стороне сервера для предотвращения атак типа SQL-инъекций.
5. **HTTPS** – использование защищенного протокола передачи данных для шифрования трафика между клиентом и сервером.

**2.2.9 Тестирование системы**

Для обеспечения качества разрабатываемой системы предусмотрены различные виды тестирования:

1. **Модульное тестирование (Insomnia)** – проверка корректности работы отдельных компонентов системы.
2. **Интеграционное тестирование** – проверка взаимодействия между компонентами системы.
3. **Системное тестирование** – проверка соответствия системы функциональным и нефункциональным требованиям.
4. **Нагрузочное тестирование** – оценка производительности системы при различных уровнях нагрузки.
5. **Тестирование безопасности** – выявление потенциальных уязвимостей в системе защиты данных.

Примеры запланированных тестовых сценариев представлены в таблице 2.5.

*Таблица 2.4 — Примеры тестовых сценариев*

| **ID** | **Название** | **Описание** | **Ожидаемый результат** |
| --- | --- | --- | --- |
| TS-01 | Добавление новой команды | Создание новой команды с заполнением всех обязательных полей | Команда успешно добавлена в систему |
| TS-02 | Редактирование информации о команде | Изменение названия и главного тренера существующей команды | Информация о команде успешно обновлена |
| TS-03 | Добавление игрока в команду | Добавление нового игрока в существующую команду | Игрок успешно добавлен в команду |
| TS-04 | Формирование расписания матчей | Создание новых матчей с указанием дат, команд и стадионов | Матчи успешно добавлены в расписание |
| TS-05 | Внесение результата матча | Ввод счета завершенного матча | Результат матча успешно сохранен, обновлена статистика команд |
| TS-06 | Расчет стоимости билетов | Расчет стоимости билетов разных категорий для конкретного матча | Правильно рассчитанные стоимости билетов |
| TS-07 | Формирование турнирной таблицы | Запрос на формирование текущей турнирной таблицы | Корректно сформированная турнирная таблица |

В результате проектирования разработана полная архитектура системы для организаторов футбольных соревнований, включающая структуру базы данных, спецификацию API, макеты пользовательского интерфейса и алгоритмы обработки данных. Данное решение учитывает все функциональные и нефункциональные требования, выявленные на этапе анализа, и использует современные технологии разработки.

# **3. ПРАКТИЧЕСКАЯ ЧАСТЬ**

**3.1 Разработка составных элементов**

**3.1.1 Разработка серверной части (Backend)**

Разработка бекенда производилась с использованием FastAPI и принципов многослойной архитектуры.

**Создание структуры проекта**

Первым этапом стало создание структуры проекта, включающей следующие директории:

* app/ - основная директория приложения
* app/models/ - модели данных SQLAlchemy
* app/schemas/ - Pydantic схемы для валидации
* app/routers/ - маршруты API
* app/services/ - бизнес-логика приложения
* app/repositories/ - слой доступа к данным
* app/core/ - конфигурация и базовые компоненты

**Реализация слоя доступа к данным (Repository Layer)**

Для каждой основной сущности (команды, игроки, матчи, стадионы) были созданы репозитории, реализующие паттерн Repository. Это обеспечило скрытие доступа к данным от деталей работы с базой данных.

**Разработка бизнес-логики (Service Layer)**

Слой сервисов содержит основную бизнес-логику приложения, включая:

* Алгоритм расчета стоимости билетов с учетом престижности матча
* Логику формирования турнирной таблицы
* Валидацию бизнес-правил при создании матчей
* Обработку результатов матчей и обновление статистики

**REST API**

Для взаимодействия с клиентской частью разработан REST API, основные эндпоинты которого представлены в таблице 3.1. API полностью документирован с использованием OpenAPI (Swagger), который автоматически генерируется FastAPI.

*Таблица 3.1 — Основные эндпоинты REST API*

| **Метод** | **URL** | **Описание** | **Параметры** | **Ответ** |
| --- | --- | --- | --- | --- |
| GET | /api/teams | Получение списка команд | page, size, sort | Список команд |
| GET | /api/teams/{id} | Получение информации о команде | Id | Информация о команде |
| POST | /api/teams | Создание новой команды | Данные команды | Созданная команда |
| PUT | /api/teams/{id} | Обновление информации о команде | id, данные команды | Обновленная команда |
| DELETE | /api/teams/{id} | Удаление команды | Id | Статус операции |
| GET | /api/players | Получение списка игроков | page, size, sort, teamId | Список игроков |
| GET | /api/stadiums | Получение списка стадионов | page, size, sort, city | Список стадионов |
| GET | /api/matches | Получение списка матчей | page, size, sort, status, teamId, date | Список матчей |
| POST | /api/matches | Создание нового матча | Данные матча | Созданный матч |
| PUT | /api/matches/{id}/result | Внесение результата матча | id, счет | Обновленный матч |
| GET | /api/tickets/calculate | Расчет стоимости билета | matchId, category | Стоимость билета |
| GET | /api/reports/standings | Получение турнирной таблицы | Season | Турнирная таблица |
| POST | /api/auth/login | Аутентификация пользователя | Логин, пароль | |  | | --- | | JWT токен | |

**3.1.2 Разработка клиентской части (Frontend)**

Клиентская часть разрабатывалась на React с использованием TypeScript для обеспечения типобезопасности.

**Создание компонентной архитектуры**

Разработаны следующие основные компоненты:

* Компоненты представления данных (таблицы команд, списки игроков)
* Формы для ввода и редактирования данных
* Компоненты навигации и общего интерфейса
* Модальные окна для подтверждения операций

**Реализация состояния приложения**

Для управления состоянием использованы React Hooks (useState, useEffect, useContext). Реализована централизованная обработка ошибок.

**Интеграция с API**

Создан отдельный слой для взаимодействия с backend API, включающий обработку HTTP-запросов, преобразование данных и обработку ошибок.

**3.1.3 Разработка базы данных**

**Проектирование схемы данных**

На основе ER-диаграммы была спроектирована схема базы данных PostgreSQL, включающая таблицы для хранения информации о командах, игроках, стадионах, матчах, билетах и пользователях системы.

**Реализация аудита**

Создана система логирования действий пользователей через таблицу audit\_log, обеспечивающую отслеживание всех изменений в системе.

**3.1.4 Реализация системы безопасности**

**Аутентификация и авторизация**

Реализована система аутентификации на основе JWT-токенов. Созданы механизмы для регистрации пользователей, входа в систему и проверки прав доступа.

async def get\_current\_user(

*credentials*: HTTPAuthorizationCredentials = Depends(bearer\_scheme),

*db*: Session = Depends(get\_db)

):

    token = *credentials*.credentials

    credentials\_exception = HTTPException(

*status\_code*=status.HTTP\_401\_UNAUTHORIZED,

*detail*="Недействительные учетные данные",

*headers*={"WWW-Authenticate": "Bearer"},

    )

    try:

        payload = jwt.decode(token, SECRET\_KEY, *algorithms*=[ALGORITHM])

        user\_id: str = payload.get("sub")

        if user\_id is None:

            raise credentials\_exception

    except JWTError:

        raise credentials\_exception

    user = *db*.query(models.User).filter(models.User.id == user\_id).first()

    if user is None:

        raise credentials\_exception

    return user

def create\_access\_token(*data*: dict, *expires\_delta*: Optional[timedelta] = None):

    to\_encode = *data*.copy()

    if *expires\_delta*:

        expire = datetime.now() + *expires\_delta*

    else:

        expire = datetime.now() + timedelta(*minutes*=ACCESS\_TOKEN\_EXPIRE\_MINUTES)

    to\_encode.update({"exp": expire})

    encoded\_jwt = jwt.encode(to\_encode, SECRET\_KEY, *algorithm*=ALGORITHM)

    return encoded\_jwt

**Защита данных**

Пароли пользователей хранятся в виде хешей с использованием библиотеки bСrypt. Реализована валидация всех входных данных для предотвращения атак типа SQL-инъекций.

**3.2 Установка и настройка**

**3.2.1 Настройка среды разработки**

**Подготовка серверной части**

1. Установка Python 3.9+ и создание виртуального окружения
2. Установка зависимостей из файла requirements.txt
3. Настройка переменных окружения для подключения к базе данных
4. Установка и настройка PostgreSQL
5. Создание структуры БД

**Подготовка клиентской части**

1. Установка Node.js версии 16+
2. Установка зависимостей через npm
3. Настройка переменных окружения для подключения к API
4. Конфигурация сборщика (Webpack/Vite)

**3.2.2 Настройка производственной среды**

**Конфигурация сервера**

Для развертывания в производственной среде были выполнены следующие настройки:

* Настройка веб-сервера Nginx для обслуживания статических файлов
* Настройка SSL-сертификатов для обеспечения HTTPS

**Настройка базы данных**

* Создание производственной базы данных PostgreSQL

**3.2.3 Развертывание системы**

Процесс развертывания включал:

1. Клонирование репозитория с исходным кодом
2. Сборка production-версии frontend приложения
3. Установка и настройка всех зависимостей
4. Запуск сервисов и проверка их работоспособности

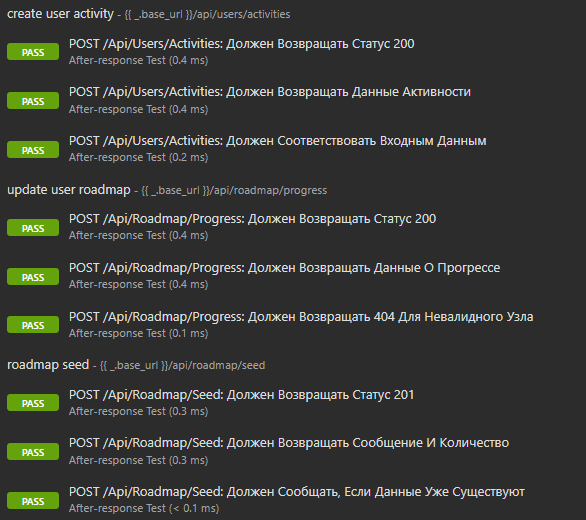
**3.3 Тестирование**

**3.3.1 Модульное тестирование**

Разработаны модульные тесты для всех основных компонентов системы:

**Тестирование backend**

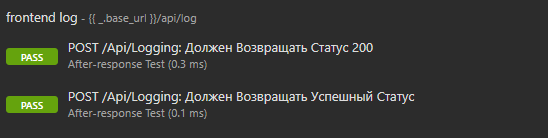
* Тесты сервисного слоя для проверки бизнес-логики
* Тесты репозиториев для проверки корректности работы с данными
* Тесты API эндпоинтов для проверки HTTP-интерфейса
* Покрытие тестами составило более 80% кода



*Рисунок 3.1 – скриншоты тестирования API*

**Тестирование frontend**

* Тесты пользовательских сценариев
* Тесты интеграции с API

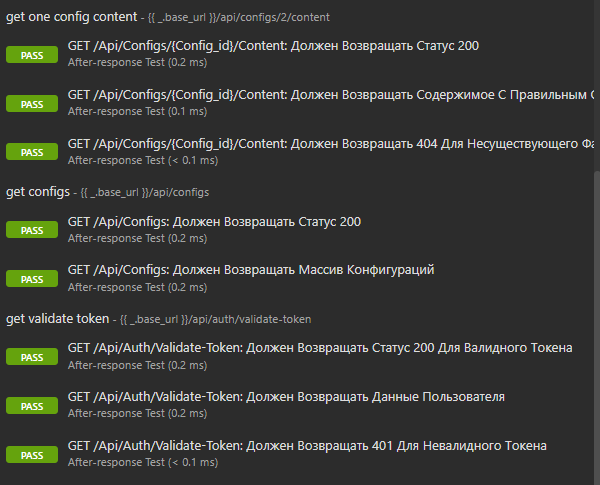


*Рисунок 3.2 – скриншоты тестирования frontend*

**3.3.2 Интеграционное тестирование**

Проведено тестирование взаимодействия между компонентами:

* Тестирование API с использованием Insomnia
* Проверка корректности обмена данными между frontend и backend
* Тестирование работы с базой данных



*Рисунок 3.3 – скриншоты тестирования backend*

**3.3.3 Системное тестирование**

Выполнена проверка системы в целом:

* Тестирование всех пользовательских сценариев
* Проверка соответствия функциональным требованиям
* Тестирование производительности при различных нагрузках
* Проверка безопасности системы

**3.3.4 Результаты тестирования**

По результатам тестирования были выявлены и устранены следующие проблемы:

* Ошибки валидации данных в формах
* Проблемы с обработкой граничных случаев
* Оптимизация производительности запросов к базе данных
* Улучшение пользовательского интерфейса

**3.4 Ввод в эксплуатацию**

**3.4.1 Подготовка к вводу в эксплуатацию**

**Подготовка данных**

* Создание начальных данных для тестирования системы
* Импорт справочной информации о командах и стадионах
* Создание учетных записей администраторов

**Обучение пользователей**

* Разработка инструкций по работе с системой
* Проведение обучающих сессий для администраторов
* Создание FAQ по наиболее часто возникающим вопросам

**3.4.2 Мониторинг и поддержка**

**Система мониторинга**

* Настройка логирования всех операций системы
* Мониторинг производительности и доступности
* Отслеживание ошибок

**Техническая поддержка**

* Организация процедур резервного копирования
* Планирование регулярного обслуживания системы
* Создание процедур восстановления после сбоев

**3.5 Разработка сопроводительной документации**

**3.5.1 Техническая документация**

Разработана следующая техническая документация:

* Руководство по архитектуре системы
* Документация API с примерами использования
* Инструкции по развертыванию и настройке

**3.5.2 Пользовательская документация**

Создана документация для пользователей:

* Руководство пользователя с описанием всех функций
* Инструкции по выполнению основных операций
* Справочник по устранению типовых проблем
* Видеоинструкции по работе с ключевыми функциями

**3.5.3 Документация для разработчиков**

Подготовлена документация для дальнейшего развития системы:

* Описание архитектуры и принципов проектирования
* Руководство по стилю кодирования
* Инструкции по добавлению новых функций
* Документация по процессам сборки и развертывания

**ЗАКЛЮЧЕНИЕ**

В результате выполнения данной курсовой работы была разработана информационная система для организаторов соревнований по футболу в рамках первенства страны. В процессе анализа были рассмотрены существующие аналоги, после чего были сформированы требования к системе.

Разработанный программный продукт полностью соответствует поставленным требованиям и включает в себя функционал для управления командами, игроками, стадионами, планирования матчей, ведения статистики и расчета стоимости билетов. Система построена на современной архитектуре с разделением на фронтенд и бекенд части, что успешно обеспечивает масштабируемость компонентов.

В разработке были использованы следующие технологии: FastAPI для бекенда, React с TypeScript для фронтенда части, и PostgreSQL в качестве системы управления базами данных. Использование таких паттернов проектирования как Repository, Service Layer и Dependency Injection обеспечило создание гибкой архитектуры.

Система прошла полный цикл тестирования, включая модульное, интеграционное и системное тестирование. Разработанная документация обеспечивает возможность дальнейшего сопровождения и развития системы.

Перспективы развития данного продукта могут состоять в добавлении мобильного приложения для болельщиков с возможностью покупки билетов, интеграции с системами онлайн-трансляций.

Дополнительно можно рассмотреть внедрение системы управления травмами игроков, интеграцию с социальными сетями для автоматической публикации результатов.

Практическая и исследовательская значимость данной работы заключается не только в получении непосредственного опыта в разработке полнофункциональной информационной системы, но и в изучении и освоении общего контекста процесса разработки программного продукта, включая анализ требований, проектирование архитектуры, реализацию.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Карякин В.Л. Организация и проведение соревнований по футболу: теория и практика. — М.: Спорт, 2023. — 284 с. (дата обращения 15.04.2025).
2. Петров А.Н. Информационные системы в спорте: архитектура и применение. — СПб.: Питер, 2024. — 320 с. (дата обращения 15.04.2025).
3. Сидоров К.В. Проектирование баз данных для спортивных информационных систем. — М.: МЦНМО, 2024. — 346 с. (дата обращения: 15.04.2025).
4. FIFA Connect: Система управления регистрацией игроков и клубов / [Электронный ресурс] // FIFA Digital Hub : [сайт]. — URL: https://digitalhub.fifa.com/fifa-connect/ (дата обращения: 15.04.2025).
5. SportsTG: Платформа для управления спортивными соревнованиями / [Электронный ресурс] // SportsTG Blog : [сайт]. — URL: https://blog.sportstg.com/about-sportstg/ (дата обращения: 15.04.2025).
6. League Republic: Система организации футбольных турниров / [Электронный ресурс] // League Republic : [сайт]. — URL: https://www.leaguerepublic.com/about (дата обращения: 23.04.2025).

# **ПРИЛОЖЕНИЕ**

Приложение 1А – Документация API (Swagger)



