1. **Выбор и обоснование средств проектирования и реализации**

Для успешной реализации информационной системы для любительского уровня волейбольных соревнований, важно правильно выбрать технологический стек, который обеспечит высокую производительность, безопасность и устойчивость системы. В рамках проекта мы используем FastAPI для серверной части, PostgreSQL для управления данными, а также NGINX в качестве веб-сервера, что позволит создать эффективную и масштабируемую платформу.

Backend реализован на базе FastAPI — легковесного Python-фреймворка для создания API. FastAPI поддерживает асинхронные запросы, что обеспечивает высокую скорость обработки данных и позволяет обрабатывать большое количество одновременных подключений. Этот фреймворк также предлагает встроенные механизмы для валидации данных и генерации документации API, что упрощает разработку и тестирование.

Для хранения данных мы выбрали реляционную базу данных PostgreSQL. Этот СУБД является одним из самых надёжных решений для хранения больших объёмов данных, обеспечивая при этом высокую производительность и безопасность. PostgreSQL отлично поддерживает транзакционную целостность и сложные запросы, что позволит обрабатывать большое количество данных и сохранять целостность системы.

В качестве веб-сервера мы используем NGINX, который зарекомендовал себя как надёжное и производительное решение для обработки HTTP-запросов. NGINX эффективно управляет трафиком, распределяя нагрузку между серверами и увеличивая устойчивость системы. Благодаря встроенным функциям кэширования и поддержки обратного прокси, NGINX позволяет ускорить загрузку страниц и обрабатывать большое количество запросов, что особенно важно при росте числа пользователей. Это обеспечит бесперебойную работу системы.

1. **Проектирование архитектуры приложения**

На рисунке (?) представлена схема архитектуры веб-приложения для информационной системы для любительского уровня волейбольных соревнований:

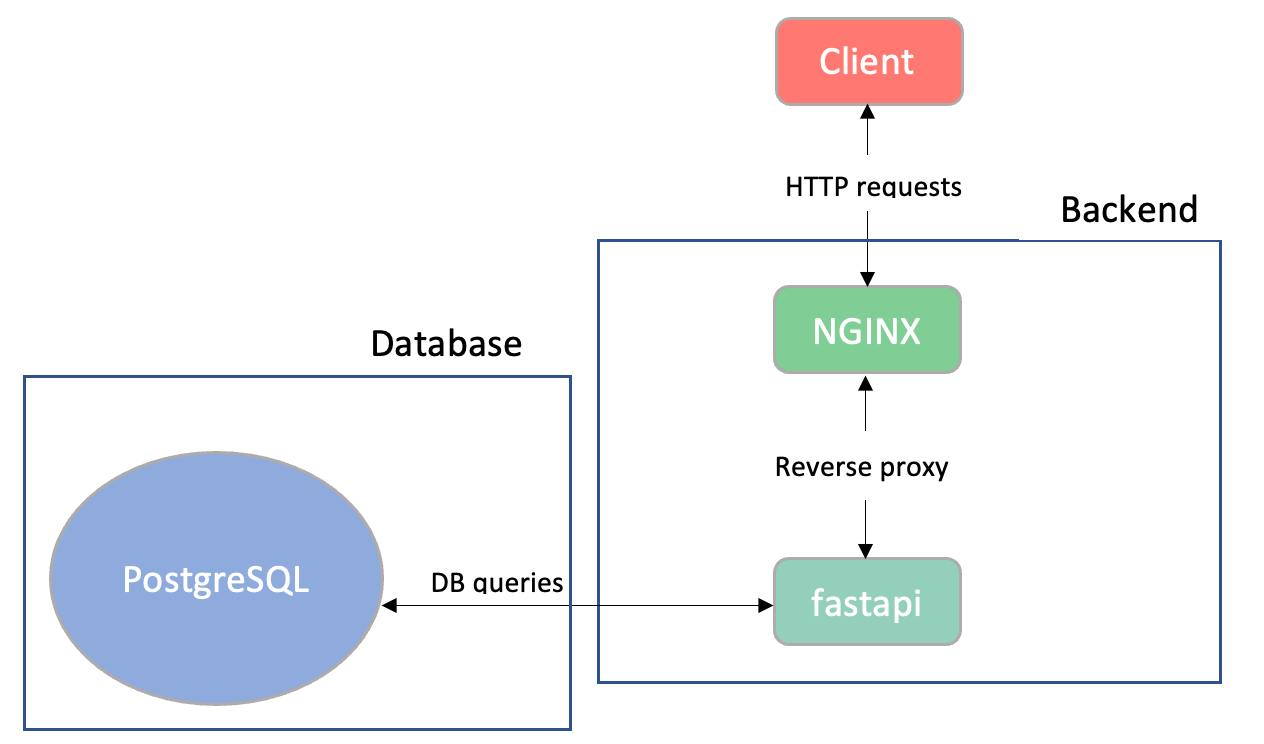


Рисунок 1 - схема архитектуры веб-приложения

Схема включает в себя следующие компоненты:

* клиентская сторона представлена фреймворком React.js, который позволяет пользователям взаимодействовать с порталом в реальном времени. React.js отправляет HTTP-запросы на серверную часть, чтобы получить данные и выполнить различные операции;
* серверная часть представлена веб-сервером nginx и fastapi, которые отвечают за всю бизнес-логику портала: прием и обработка HTTP-запросов от клиентской части и взаимодействие с базой данных;
* база данных используется для хранения и управления данными приложения. Серверная часть обращается к базе данных через SQL-запросы, чтобы совершить действия над информацией

1. **Проектирование хранилища данных**

**Основные сущности и атрибуты**

В таблице 1 приведены основные сущности с описанием их атрибутов.

Таблица 1 – сущности и их атрибуты

|  |  |  |
| --- | --- | --- |
| **Сущность** | **Атрибуты** | **Описание** |
| teams | team\_id, team\_name, team\_captain\_name, team\_captain\_phone | Команды |
| players | player\_id, player\_name, player\_team\_id | Игроки |
| matches | match\_id, match\_date, match\_completed, match\_team1\_id, match\_team2\_id | Матчи |
| match\_sets | match\_sets\_id, match\_sets\_match\_id, match\_sets\_team1\_score, match\_sets\_team2\_score, match\_sets\_set\_number | Партии |
| team\_stats | team\_stats\_id, team\_stats\_team\_id, team\_stats\_matches\_played, team\_stats\_wins, team\_stats\_losses, team\_stats\_score | Результаты |
| user | user\_id, is\_adm, is\_ref | Пользователи системы |
| user\_metadata | user\_metadata\_id, user\_id, email, hashed\_password, user\_name, user\_surname | Метаданные пользователей |

**Логическая модель данных**

Логическая модель данных представлена ключевыми связями между сущностями. Сущности связаны через внешние ключи, которые определяют связь между таблицами.

1. **Связи между игроками и командой:**

* Таблица players связана с teams через атрибут team\_id.

1. **Связи для матчей с командами:**

* Таблица matches связана с teams через атрибуты team1\_id и team2\_id.

1. **Партии матчей:**

* Таблица match\_sets связана с matches через атрибут match\_id.

1. **Результаты соревнований:**

* С помощью team\_id получаем team\_name из teams и match\_id из matches.
* match\_id позволяет получить данные из match\_sets, на основе которых подсчитываются matches\_played, wins, losses и score.

1. **Связи для пользователей с их метаданными:**

* Таблица user связана с user\_metadata через атрибут user\_metadata\_id.

**Инфологическая модель данных**

В инфологической модели описываются основные сущности и их взаимосвязи:

* Пользователи связаны с метаданными пользователей и могут быть судьями или администраторами.
* На матч назначаются две команды.
* Счет каждой партии в матче сохраняется, на основе партий в уже завершенных матчах вычисляется победитель и ведется подсчет итоговых очков.
* Каждый игрок состоит в команде.

**Дополнительные ограничения целостности**

1. **Уникальные ограничения:**

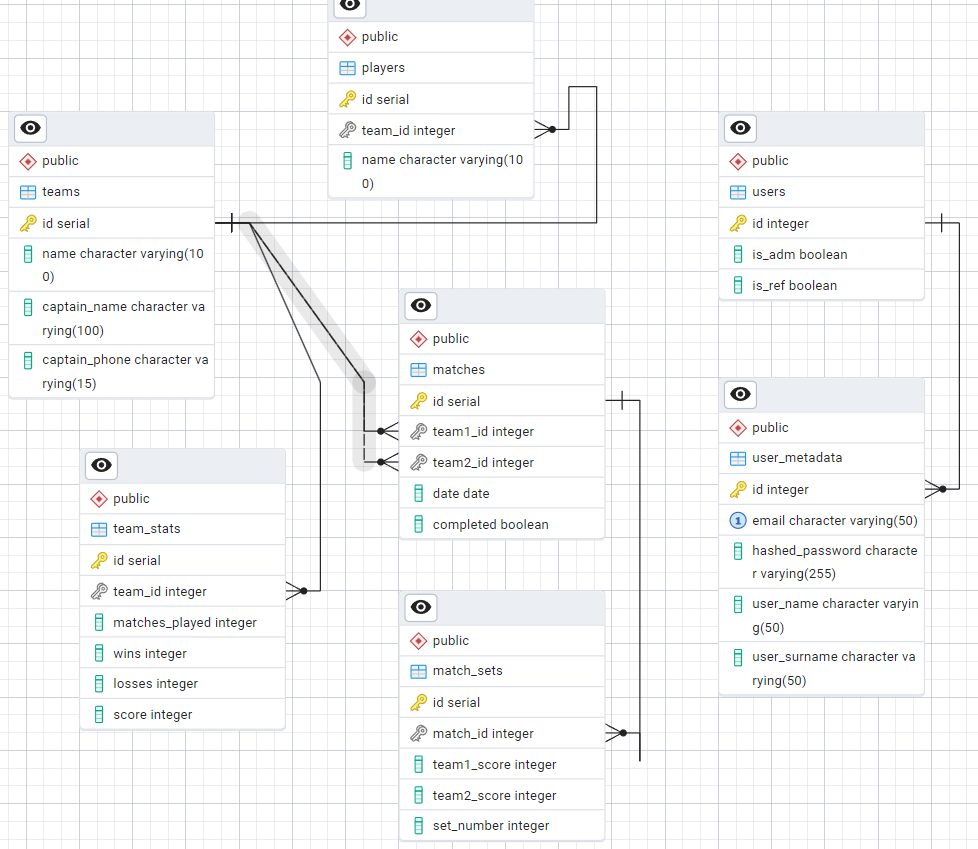
* Поле email в таблице user\_metadata должно быть уникальным.

1. **Каскадные операции:**

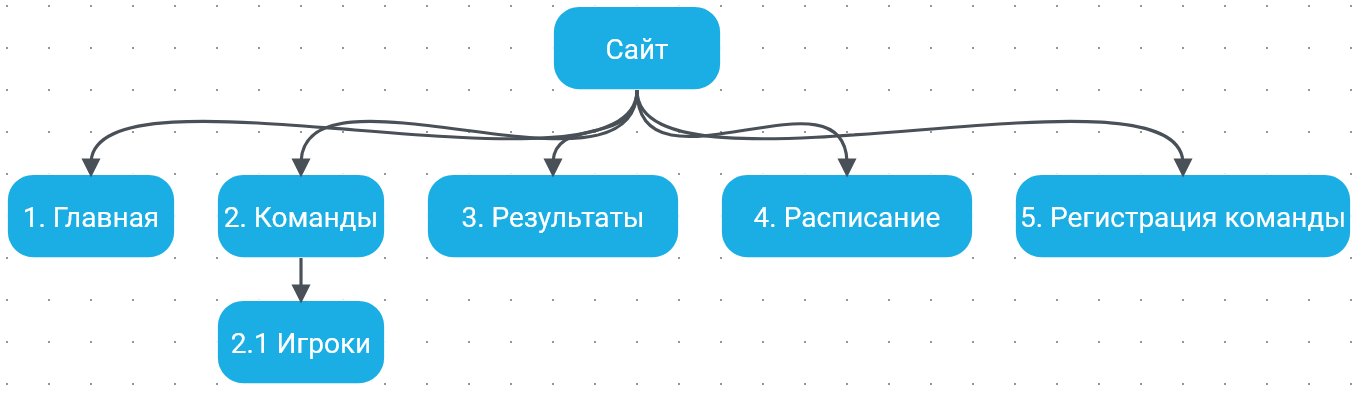
* При удалении записи в таблице matches записи в таблице match\_sets удаляются каскадно.

1. **Проверки на пустые значения и обязательные поля:**

* Поля, которые участвуют в связях или являются ключевыми, должны быть не NULL (например, team\_id в таблице teams должен быть заполнен для указания страны).

  
Рисунок 2 – Схема БД

1. **Проектирование пользовательского интерфейса**
   1. **Верхнеуровневое определение экранов группируем по разделам**



* 1. **Назначение экранов**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № экрана | |  | | --- | | Краткое название |  |  | | --- | |  | | |  | | --- | | Поля ввода для валидации |  |  | | --- | |  | | Описание экрана и его поведения |
| 1 | |  | | --- | | Главная | | |  | | --- | | - |  |  | | --- | |  | | Страница содержит приветственную информацию |
| 2 | |  | | --- | | Команды |  |  | | --- | |  | | Название команды(<50, любые символы) | Страница содержит таблицу зарегистрированных команд, а также поля для поиска по названию |
| 2.1 | Игроки | ФИО(<100, только буквы) | Отображает данные об игроках выбранной команды, а также содержит форму добавления нового игрока |
| 3 | |  | | --- | | Результаты |  |  | | --- | |  | | Название команды(<50, любые символы) | |  | | --- | |  |  |  | | --- | | Содержит |   фильтры по датам, при выборе временного промежутка выводит таблицу с результатами команд в этот период времени, а так же содержит поиск по названию |
| 4 | Расписание | Название команды(<50, любые символы) | Содержит информацию о матчах, есть поиск по названию команды и фильтры по датам |
| 5 | Регистрация команды | Название команды(<50, любые символы), ФИО капитана (<100, только буквы), Номер телефона капитана (11, цифры) | |  | | --- | | Форма для регистрации команды, требует заполнить все поля, при пустом поле выводится сообщение с указанием, при успешной регистрации выводит соответствующее сообщение | |

* 1. **Отрисовка и описание макетов ваших экранов**

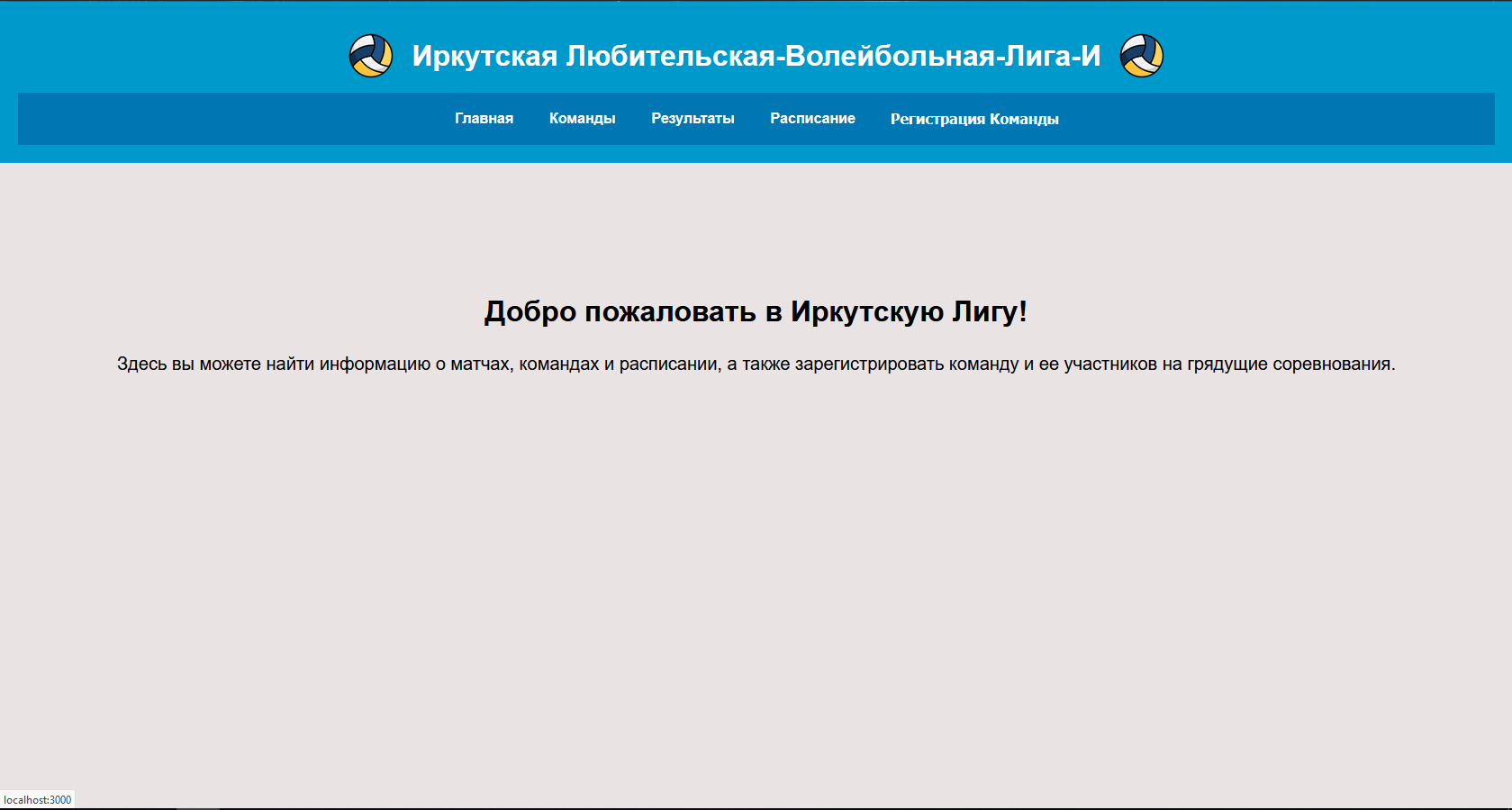
****

Рисунок 3 – Экран №1

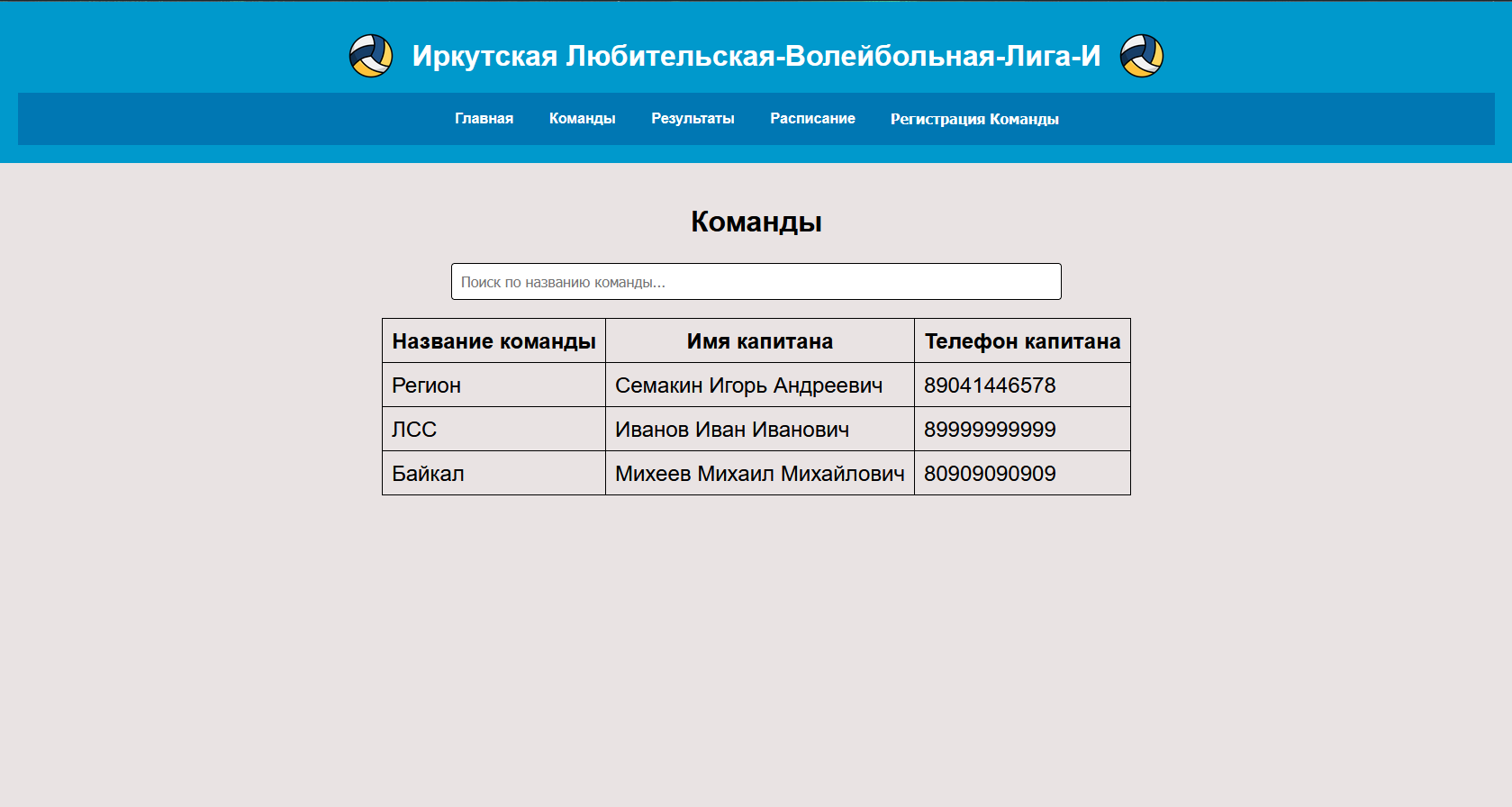
****

Рисунок 4 – Экран №2

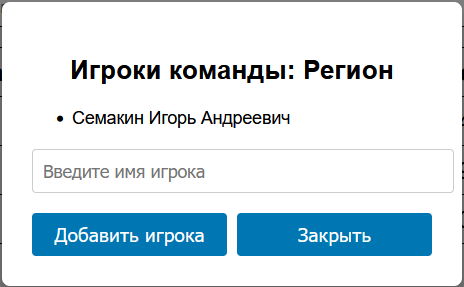
****

Рисунок 5 – Экран №2.1

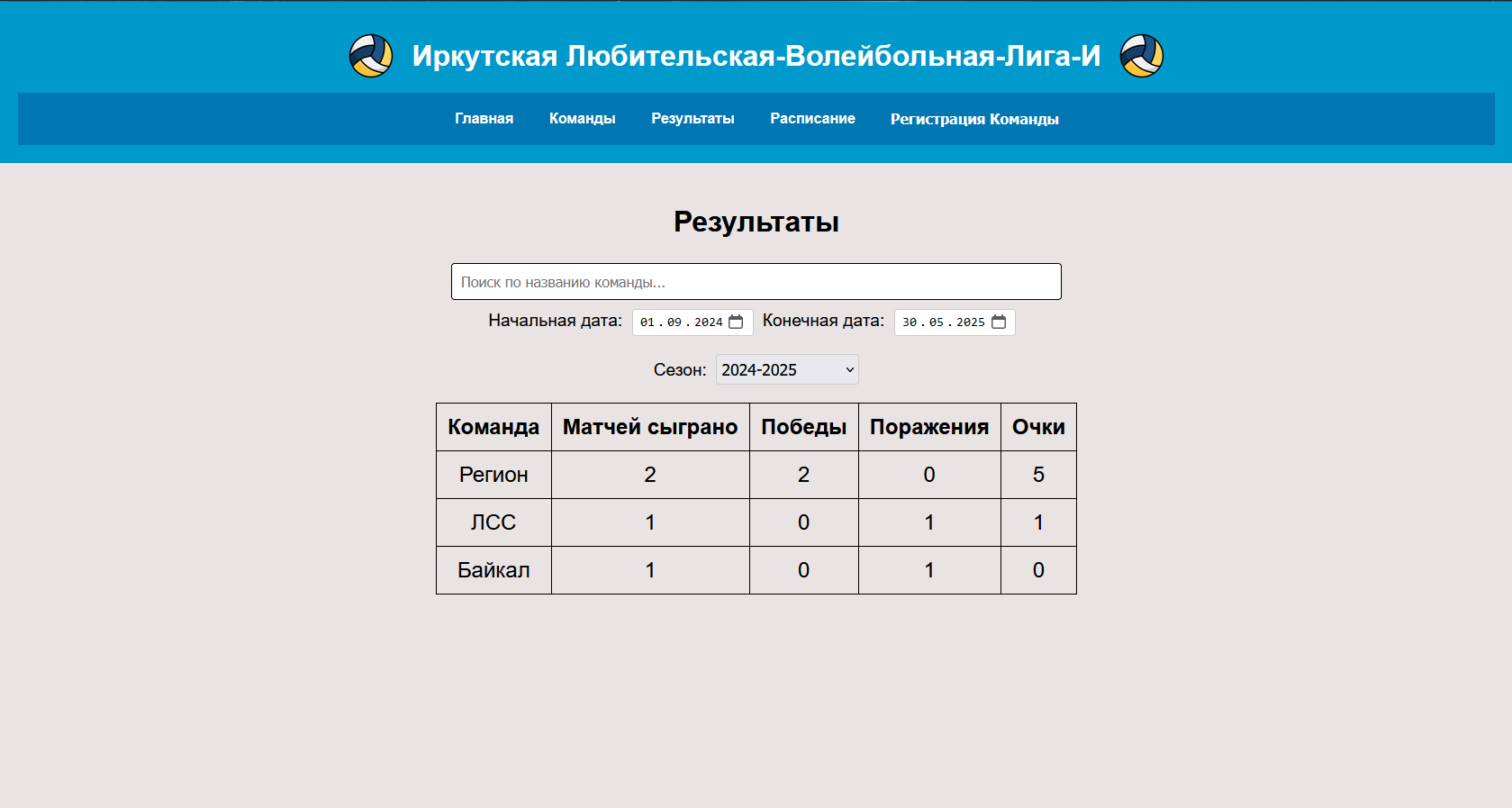
****

Рисунок 6 – Экран №3

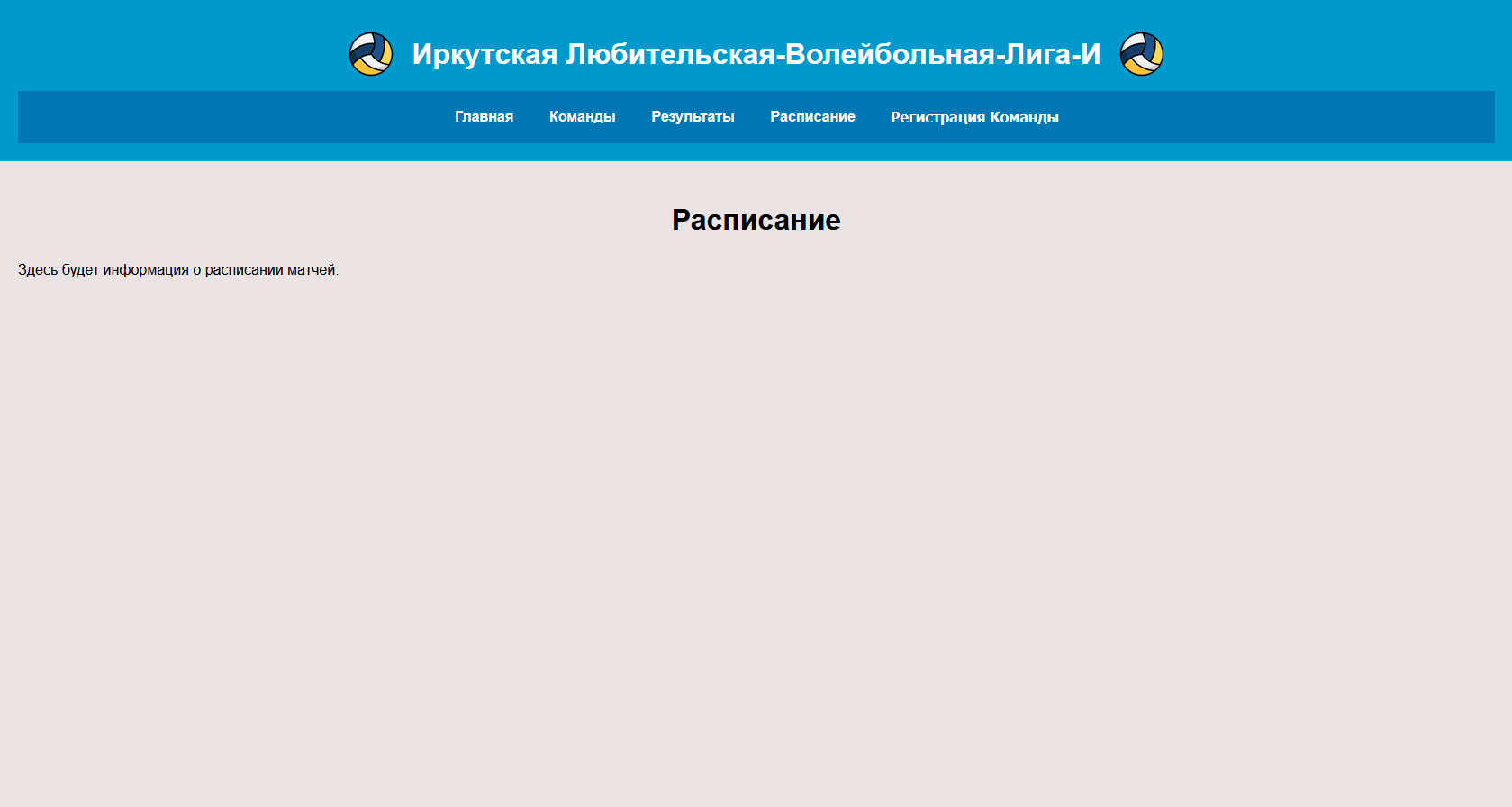
****

Рисунок 7 – Экран №4

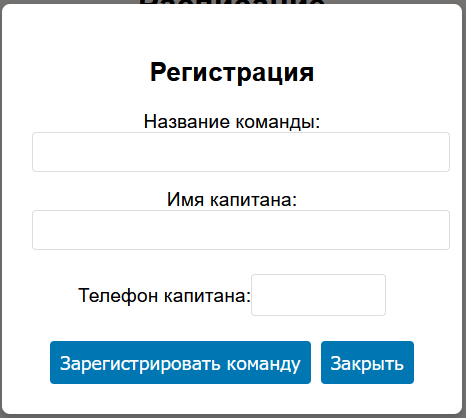
****

Рисунок 8 – Экран №5

* 1. **Карта экранов**

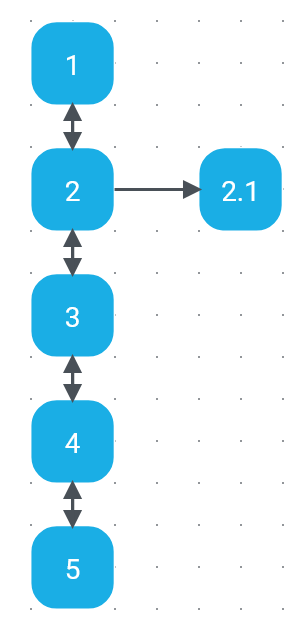
****

Рисунок 9 – карта экранов