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

**Средства проектирования**

1. **PlantUML**:
   * **Описание**: PlantUML — это инструмент для создания UML-диаграмм с помощью простого текстового языка. Он удобен для быстрого прототипирования и документирования архитектуры программного обеспечения.
   * **Обоснование выбора**: Мы выбрали PlantUML, потому что он прост в использовании, имеет открытый исходный код и хорошо интегрируется с различными инструментами разработки. Кроме того, он позволяет создавать читаемые и легко редактируемые диаграммы, что важно для командного сотрудничества.
2. **Mermaid**:
   * **Описание**: Mermaid — это ещё один инструмент для создания диаграмм с использованием текстовых инструкций. Он поддерживает широкий спектр типов диаграмм, таких как диаграммы классов, последовательности, состояния и многие другие.
   * **Обоснование выбора**: Mermaid был выбран благодаря своей гибкости и поддержке множества типов диаграмм. Это позволило нам быстро создавать сложные схемы, которые помогают лучше понимать архитектуру нашего приложения.
3. **Figma**:
   * **Описание**: Figma — это облачный инструмент для дизайна пользовательских интерфейсов. Он предоставляет возможности для совместной работы над проектами, создания интерактивных прототипов и экспорта ресурсов для разработчиков.
   * **Обоснование выбора**: Figma был выбран из-за его удобства для совместной работы, широкого набора инструментов для дизайна и возможности быстрой итерации над макетами. Это помогло нам ускорить процесс проектирования пользовательского интерфейса.

**2. Проектирование архитектуры приложения**

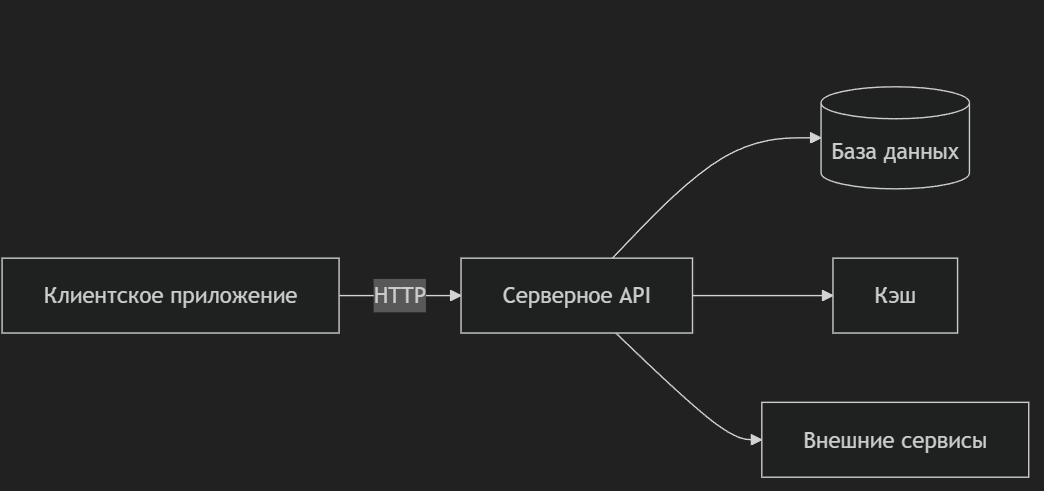
**Архитектура приложения**

Рисунок 1 – Архитектура приложение

**2.1 Описание архитектуры**

Архитектура представляет собой классический клиент-серверный подход с использованием RESTful API. Клиентское приложение взаимодействует с серверным API посредством HTTP-запросов. Серверное API обрабатывает запросы, обращается к базе данных для получения и сохранения данных, а также использует кэш для ускорения доступа к часто запрашиваемым данным. При необходимости серверное API может обращаться к внешним сервисам для выполнения специфических задач.

**2.2 Почему была выбрана эта архитектура?**

Мы выбрали эту архитектуру, потому что она обеспечивает высокую масштабируемость, гибкость и удобство в обслуживании. Разделение на клиентскую и серверную части позволяет независимо развивать и тестировать разные компоненты системы. Использование RESTful API делает систему совместимой с различными клиентами, будь то веб-приложения, мобильные устройства или настольные клиенты. Кэширование ускоряет выполнение запросов, а возможность интеграции с внешними сервисами расширяет функциональность системы.

**3. Проектирование хранилища данных**

**3.1 Основные сущности**

|  |  |
| --- | --- |
| **Название сущности** | **Атрибуты** |
| Пользователь (User) | id, имя, email, пароль, роль |
| Турнир (Tournament) | id, название, дата начала, дата окончания, статус |
| Команда (Team) | id, название, капитан\_id (внешний ключ на User.id) |
| Матч (Match) | id, турнир\_id (внешний ключ на Tournament.id), команда1\_id (внешний ключ на Team.id), команда2\_id (внешний ключ на Team.id), результат |
| Участник матча (MatchParticipant) | id, матч\_id (внешний ключ на Match.id), пользователь\_id (внешний ключ на User.id), команда\_id (внешний ключ на Team.id) |

Таблица 1 – Основные сущности

**3.2 Логическая модель данных**

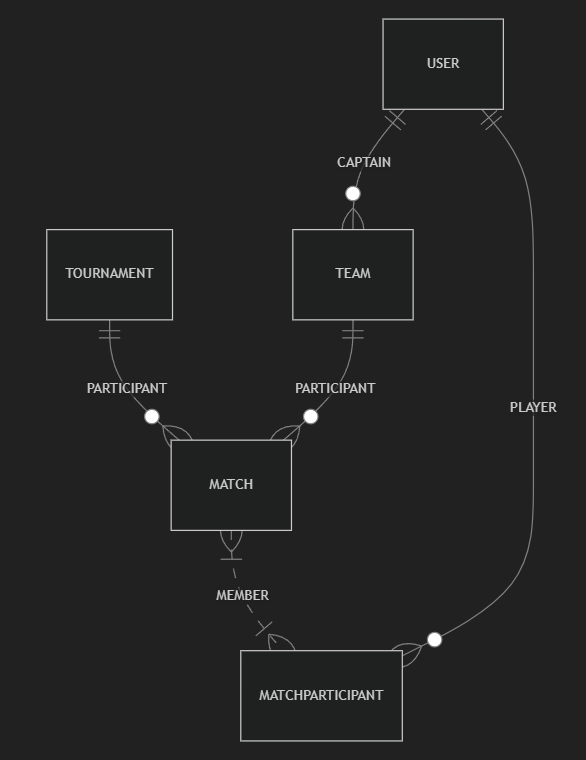


Рисунок 2 **–** Логическая модель данных

**Дополнительные ограничения целостности**

1. **Уникальность имени пользователя**: Каждый пользователь должен иметь уникальное имя.
2. **Уникальность названия команды**: Названия команд должны быть уникальными в пределах одного турнира.
3. **Наличие капитана у каждой команды**:У каждой команды должен быть назначен капитан.
4. **Роль участника в матче**: Один и тот же пользователь не может участвовать в одном матче одновременно за разные команды.

**4. Проектирование пользовательского интерфейса**

**4.1 Разделы**

**Раздел "Account"**

* Вход
* Регистрация
* Профиль
* Редактирование профиля

**Раздел "Tournaments"**

* Список турниров
* Подробности турнира
* Регистрация на турнир
* Управление турниром (для организаторов)

**Раздел "Matches"**

* Календарь матчей
* Результаты матчей
* Прямая трансляция матча

**Раздел "Social"**

* Обсуждения
* Комментарии
* Лента новостей

**Таблица назначения экранов**

|  |  |  |  |
| --- | --- | --- | --- |
| **№ экрана** | **Краткое название (Name)** | **Поля ввода для валидации (Validation)** | **Описание экрана и его поведения (Behavior)** |
| 1 | Вход | Email, Пароль | Экран входа в систему. Пользователи вводят свои учетные данные для аутентификации. |
| 2 | Регистрация | Имя, Email, Пароль, Подтверждение пароля | Экран регистрации новых пользователей. Необходимо ввести имя, email и дважды подтвердить пароль. |
| 3 | Профиль | Имя, Email, Роль | Экран профиля пользователя. Здесь отображается основная информация о пользователе, такая как имя, email и роль. |
| 4 | Редактирование профиля | Имя, Email, Пароль, Новый пароль, Подтверждение нового пароля | Экран для изменения личных данных пользователя. Можно изменить имя, email и пароль. |
| 5 | Список турниров | Фильтры: Дата, Статус | Экран со списком доступных турниров. Пользователи могут фильтровать список по дате и статусу турнира. |
| 6 | Подробности турнира | Название, Дата начала, Дата окончания, Статус, Участники | Экран с подробной информацией о конкретном турнире. Включает название, даты начала и окончания, текущий статус и список участников. |
| 7 | Регистрация на турнир | Команда, Капитан | Экран для регистрации на турнир. Пользователю предлагается выбрать команду и назначить капитана. |
| 8 | Управление турниром | Действия: Добавить матч, Изменить статус, Удалить турнир | Экран для организаторов турнира. Позволяет добавлять новые матчи, изменять статус турнира и удалять турнир. |
| 9 | Календарь матчей | Фильтры: Дата, Турнир | Экран с календарем матчей. Пользователи могут просматривать предстоящие и прошедшие матчи, фильтруя их по дате и турниру. |
| 10 | Результаты матчей | Турнир, Команда 1, Команда 2, Результат | Экран с результатами прошедших матчей. Показывает, кто выиграл или проиграл в каждом матче. |
| 11 | Прямая трансляция матча | Видео, Чат | Экран для просмотра прямой трансляции матча. Включает видео поток и чат для обсуждения происходящего. |
| 12 | Обсуждения | Тема, Сообщение | Экран для создания и ведения обсуждений. Пользователи могут создавать темы и оставлять сообщения. |
| 13 | Комментарии | Сообщение | Экран для добавления комментариев к различным событиям и материалам. Пользователи оставляют комментарии. |
| 14 | Лента новостей | Новости, Объявления | Экран с лентой новостей и объявлений. Содержит актуальные новости и важные объявления для пользователей. |

Таблица 2 - Данные

**4.2 Отрисовка и описание макетов экранов**

****

Рисунок 3 – макет №1

Эта карта экранов демонстрирует структуру навигации между различными разделами и экранами приложения. Пользователи могут перемещаться между разделами и экранами в зависимости от своих потребностей и ролей в системе.

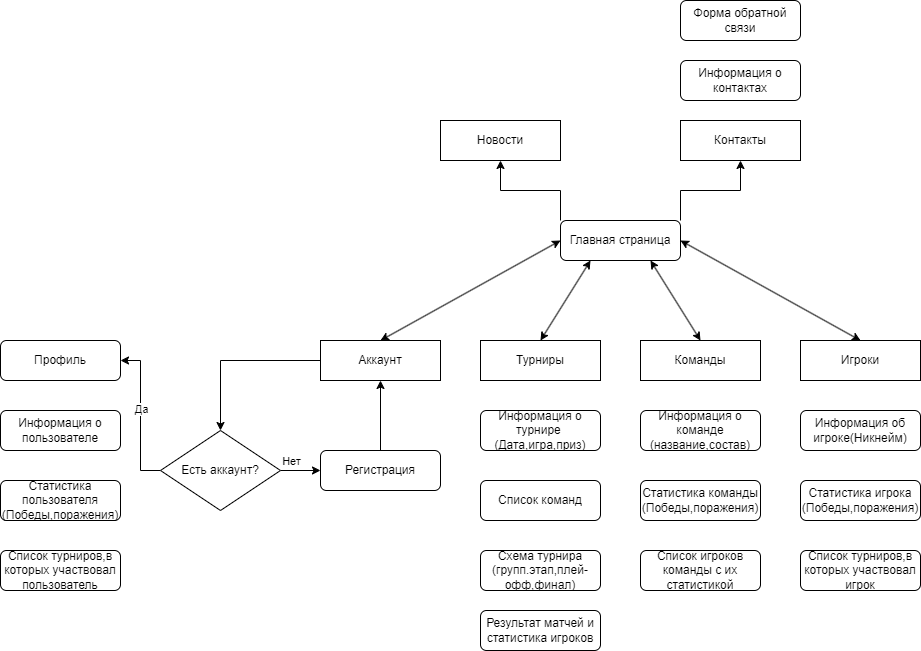


Рисунок 4 – Блок схема сайта

**4.3 Хранение и репликация данных**

Система должна иметь надежное решение для хранения данных, способное обрабатывать большие объемы данных, включая информацию об игроках, результаты турниров и системные журналы.

Должны быть реализованы механизмы репликации данных, обеспечивающие доступность и актуальность данных.

Система должна быть способна обрабатывать различные форматы хранения данных, такие как реляционные базы данных PostgreSQL.

Примеры протоколов и систем обмена данными включают:

API (интерфейсы прикладного программирования);

Протоколы обмена сообщениями;

EDI (электронный обмен данными)

Глобальная система обмена информацией (GIES);

Система должна уметь обрабатывать различные форматы данных и преобразовывать их по мере необходимости для обеспечения совместимости с другими системами. Это включает в себя:

Форматы изображений: Система должна уметь обрабатывать такие форматы изображений, как JPEG, PNG и GIF.

Текстовые форматы: Система должна уметь обрабатывать текстовые форматы, такие как TXT, CSV и JSON.

Преобразование данных: Система должна иметь возможность преобразовывать данные из одного формата в другой по мере необходимости для обеспечения совместимости с другими системами.

**4.4** **Обслуживание и обновления системы**

Система должна регулярно проходить техническое обслуживание, чтобы гарантировать, что все компоненты обновлены и функционируют правильно.

Система должна быть способна обрабатывать обновления и патчи, не нарушая работу турнира и не ставя под угрозу безопасность.

В системе должен быть предусмотрен процесс резервного копирования и восстановления, гарантирующий возможность восстановления данных в случае сбоя или потери данных.

**4.5** **Безопасность и контроль доступа**

В системе должны быть приняты надежные меры безопасности для предотвращения несанкционированного доступа или вмешательства в данные турниров или системные компоненты.

Должны быть внедрены механизмы контроля доступа, гарантирующие, что только уполномоченный персонал имеет доступ к конфиденциальным данным или функциям системы.

Система должна быть способна обнаруживать угрозы безопасности и реагировать на них, такие как попытки взлома или заражение вредоносными программами.

**4.6 Реляционные базы данных (PostgreSQL)**

**4.6.1 Преимущества**:

* **ACID** (Atomicity, Consistency, Isolation, Durability): реляционные базы данных обеспечивают целостность данных и транзакции.
* **SQL**: использование стандартного языка SQL для работы с данными.
* **Журналы транзакций**: записи всех операций сохраняются в журнале транзакций, что позволяет отслеживать изменения и восстанавливать данные.
* **Поддержка сложных запросов**: реляционные базы данных поддерживают сложные запросы, позволяющие выполнять агрегирование данных и анализ.

**4.6.2 Недостатки**:

* **Сложность в работе с большими объемами данных**: при большом количестве записей могут возникать сложности с производительностью.
* **Гибкость**: реляционные базы данных требуют строгого определения схем данных, что ограничивает гибкость.

**4.6.3 Использование**:

* **Финансовые транзакции**: транзакции должны быть атомарными и непротиворечивыми.
* **Работа с большими наборами данных**: обработка больших объемов данных требует использования агрегации и обработки данных.
* **Бизнес-аналитика**: поддержка сложных запросов для анализа данных.

**4.6.4 Структура**:

1. **Пользователи (Users)**:
   * ID (primary key)
   * Имя (name)
   * Электронная почта (email)
   * Пароль (password)
   * Роль (role) (например, участник, организатор, судья)
2. **Турниры (Tournaments)**:
   * ID (primary key)
   * Название (title)
   * Описание (description)
   * Дата начала (start\_date)
   * Дата окончания (end\_date)
   * Статус (status) (например, запланирован, идет, завершен)
3. **Команды (Teams)**:
   * ID (primary key)
   * Название (name)
   * Капитан (captain\_id) (внешний ключ на Users.ID)
4. **Матчи (Matches)**:
   * ID (primary key)
   * Турнир (tournament\_id) (внешний ключ на Tournaments.ID)
   * Команда 1 (team1\_id) (внешний ключ на Teams.ID)
   * Команда 2 (team2\_id) (внешний ключ на Teams.ID)
   * Результат (result)
5. **Участники матчей (MatchParticipants)**:
   * ID (primary key)
   * Матч (match\_id) (внешний ключ на Matches.ID)
   * Пользователь (user\_id) (внешний ключ на Users.ID)
   * Команда (team\_id) (внешний ключ на Teams.ID)
6. **Результаты (Results)**:
   * ID (primary key)
   * Матч (match\_id) (внешний ключ на Matches.ID)
   * Победитель (winner\_team\_id) (внешний ключ на Teams.ID)
   * Итоговый счет (score)

**4.6.5 Структура базы данных в виде кода**

CREATE TABLE Users (

user\_id SERIAL PRIMARY KEY,

name VARCHAR(255),

email VARCHAR(255) UNIQUE NOT NULL,

password VARCHAR(255),

role VARCHAR(50)

);

CREATE TABLE Tournaments (

tournament\_id SERIAL PRIMARY KEY,

title VARCHAR(255),

description TEXT,

start\_date TIMESTAMP,

end\_date TIMESTAMP,

status VARCHAR(20)

);

CREATE TABLE Teams (

team\_id SERIAL PRIMARY KEY,

name VARCHAR(255),

captain\_id INT REFERENCES Users(user\_id),

UNIQUE (name)

);

CREATE TABLE Matches (

match\_id SERIAL PRIMARY KEY,

tournament\_id INT REFERENCES Tournaments(tournament\_id),

team1\_id INT REFERENCES Teams(team\_id),

team2\_id INT REFERENCES Teams(team\_id),

result VARCHAR(100)

);

CREATE TABLE MatchParticipants (

participant\_id SERIAL PRIMARY KEY,

match\_id INT REFERENCES Matches(match\_id),

user\_id INT REFERENCES Users(user\_id),

team\_id INT REFERENCES Teams(team\_id)

);

CREATE TABLE Results (

result\_id SERIAL PRIMARY KEY,

match\_id INT REFERENCES Matches(match\_id),

winner\_team\_id INT REFERENCES Teams(team\_id),

score VARCHAR(255)

);

**4.6.6 Объяснение структуры**

1. **Таблица Users** хранит информацию о пользователях, участвующих в турнирах. У каждого пользователя есть уникальный идентификатор, имя, электронная почта, пароль и роль (например, участник, организатор, судья).
2. **Таблица Tournaments** содержит информацию о самих турнирах, включая название, описание, даты начала и окончания, а также статус турнира.
3. **Таблица Teams** хранит информацию о командах, участвующих в турнирах. Каждая команда имеет уникальный идентификатор, название и ссылку на капитана (через внешний ключ на таблицу Users).
4. **Таблица Matches** хранит информацию о матчах, включающую идентификатор матча, ссылки на участвующие команды и результат.
5. **Таблица MatchParticipants** связывает пользователей с матчами и командами, показывая, кто участвует в каком матче и за какую команду.
6. **Таблица Results** хранит результаты матчей, включая победителя и общий счет.

**4.7** **Турнирные операции и управление игроками**

1) Система должна быть способна управлять турнирными операциями, включая регистрацию игроков, планирование турниров и подсчет очков.

2) Система должна быть способна выполнять различные роли игроков и команд, такие как игроки, тренеры и администраторы.

3) Система должна быть способна предоставлять профили игроков и команд, включая статистику и показатели производительности.

4) Система должна быть способна обрабатывать различные форматы турниров, такие как турниры на выбывание или по круговой системе.

5) Система должна быть способна предоставлять обновления и уведомления в режиме реального времени игрокам и администраторам во время проведения турниров.

**4.8 Лучшие практики кибербезопасности**

Система должна соответствовать лучшим практикам кибербезопасности, таким как шифрование, безопасная аутентификация и контроль доступа.

Шифрование: Использование шифрования SSL / TLS для защиты данных при передаче и шифрование конфиденциальных данных в режиме ожидания, таких как пароли и личная информация.

Безопасная аутентификация: Внедрение многофакторной аутентификации (MFA) для обеспечения доступа к системе только авторизованным пользователям, а также использование безопасного хранения паролей и алгоритмов хэширования.

Контроль доступа: Внедрение ролевого контроля доступа (RBAC) для обеспечения доступа пользователей только к ресурсам и данным, необходимым им для выполнения своих задач, и использование доступа с наименьшими привилегиями для минимизации поверхности атаки.

Система должна быть способна обнаруживать угрозы безопасности и реагировать на них, такие как попытки взлома или заражение вредоносными программами.

В системе должен быть разработан план реагирования на инциденты для обработки инцидентов безопасности или утечек данных.

**4.9 Требования**

**4.9.1 Функциональные требования**

Сайт должен быть сделан на Bitrix24 или Wordpress.

На одном из предложенных языков: PHP,CSS,HTML,JAVASCRIPT

У сайта должна быть кроссбраузерность Opera, Safari, Chrome.

Адаптивность к мониторам 1920x1080

Функциональные требования к системе автоматизации кибертурниров могут включать:

1) Регистрация и аутентификация пользователей. Личный кабинет с историей игрока;

2) Управление турнирами;

3) Управление командой и игроками;

4) Учет и ранжирование турниров;

5) Инструменты коммуникации для игроков и администраторов;

6) Интеграция с внешними системами для обмена данными.

Эти требования должны быть конкретными, измеримыми, достижимыми, актуальными и привязанными ко времени (SMART), чтобы обеспечить ясность и осуществимость.

**4.9.2 Технические требования**

Требования к оборудованию:

Процессор: не менее 1 ГГц

Оперативная память: не менее 256 МБ

Хранилище: достаточно места для хранения файлов веб-сайта и базы данных, до 1 ГБ.

Требования к программному обеспечению:

- Операционная система: серверно-ориентированная ОС, такая как Ubuntu Server, Windows Server или дистрибутив Linux.

Другие соображения:

- подключение к Интернету: стабильное подключение к Интернету со статическим IP-адресом или динамической службой DNS

- безопасность: Обеспечьте безопасность вашего сервера, настроив брандмауэры, регулярно обновляя программное обеспечение и используя надежные пароли

- резервное копирование и восстановление: регулярно создавайте резервные копии данных вашего веб-сайта и разработайте план восстановления на случай сбоев

- сайт должен загружаться быстро любая страница сайта должна иметь больше 80 баллов в Google PageSpeed Insights.

- большие нагрузки: 50 тысяч посетителей одновременно.

**5. Вид страниц**

На главной странице выводится список последних 6 опубликованных турниров.

**5.1.Примерная структура страниц**

В шапке сайта должно быть расположена переходы на другие страницы, блоки с турнирами, командами, играми. В основе страницы расположить информацию, соответствующие названию страницы, то есть название турниров, счет, команды, игру, если пользователь нажал на блок с турнирами. В подвале расположить информацию о контактах.

**5.2 Дизайн**

Дизайн системы должен соответствовать следующим требованиям:

1) Минималистичный и простой дизайн, сочетающийся с удобством использования и быстрым доступом к основным функциям;

2) Использование адаптивного дизайна, позволяющего корректно;

3) Применение интуитивно понятного и современного пользовательского интерфейса, с возможностью персонализации интерфейса для удобства пользователей;

4) Использование качественных и выразительных графических элементов, таких как иконки, изображения, фоны и т.д.;

5) Применение цветовой гаммы, соответствующей бренду компании и общим трендам дизайна, опираясь на Dribbble;

6) Применение анимации и интерактивных элементов для улучшения пользовательского опыта и привлекательности дизайна.

3.Возможности администрирования

Возможность, создания/удаления/редактирования информации о турнирах, команд, игроков, новостей. Возможность редактирования контактов, добавления/удаления дополнительной информации.

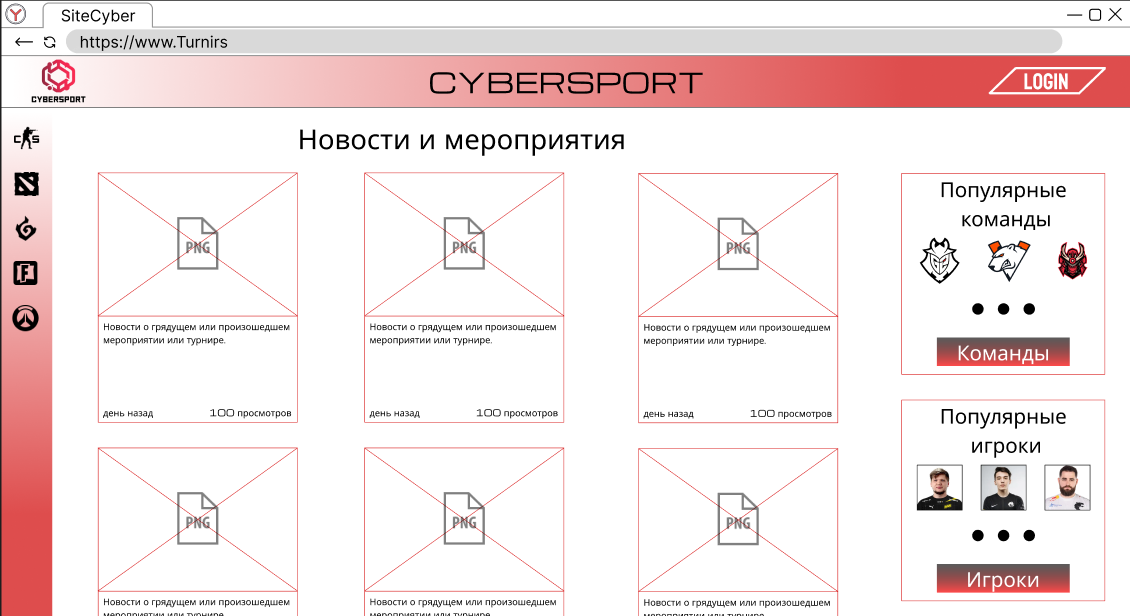


Рисунок 4 – макет главной страница сайта