**СОДЕРЖАНИЕ**

[ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ 5](#_Toc106126944)

[ВВЕДЕНИЕ 6](#_Toc106126945)

[1 Современное информационное обеспечение мониторинга спортивных событий 9](#_Toc106126946)

[1.1 Обзор аналогов агрегаторов спортивных событий 9](#_Toc106126947)

[1.1.1 FlashScore 9](#_Toc106126948)

[1.1.2 Championat 12](#_Toc106126949)

[1.2 Постановка задачи 16](#_Toc106126950)

[2 Выбор технологий и инструментов для разработки приложения 18](#_Toc106126951)

[2.1 Выбор технологий для разработки приложения 18](#_Toc106126952)

[2.1.1 Серверная часть 18](#_Toc106126953)

[2.1.2 Клиентская часть 19](#_Toc106126954)

[2.2 Программное обеспечение для разработки 22](#_Toc106126955)

[2.2.1 Среда разработки 22](#_Toc106126956)

[2.2.2 Сборка проекта 23](#_Toc106126957)

[2.2.3 Система контроля версий 23](#_Toc106126958)

[2.3 Инструменты для упрощения разработки 25](#_Toc106126959)

[3 Реализация клиентской части информационной системы для мониторинга спортивных событий 28](#_Toc106126960)

[3.1 Прототипирование интерфейса 28](#_Toc106126961)

[3.2 Базовая конфигурация приложения 30](#_Toc106126962)

[3.2.1 Инициализация JavaScript-проекта 30](#_Toc106126963)

[3.2.2 Стартовая конфигурация файлов 31](#_Toc106126964)

[3.2.3 Добавление TypeScript в проект 33](#_Toc106126965)

[3.2.4 Установка библиотек 33](#_Toc106126966)

[3.3 Разработка приложения 33](#_Toc106126967)

[3.3.1 Разработка шапки 34](#_Toc106126968)

[3.3.2 Разработка маршрутизации 35](#_Toc106126969)

[3.3.3 Разработка модуля «События» 36](#_Toc106126970)

[3.3.4 Разработка модуля «Избранное» 45](#_Toc106126971)

[3.3.5 Разработка модуля «Новости» 46](#_Toc106126972)

[ЗАКЛЮЧЕНИЕ 48](#_Toc106126973)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 49](#_Toc106126974)

[ПРИЛОЖЕНИЕ А 51](#_Toc106126975)

[ПРИЛОЖЕНИЕ Б 57](#_Toc106126976)

[ПРИЛОЖЕНИЕ В 58](#_Toc106126977)

[ПРИЛОЖЕНИЕ Г 102](#_Toc106126978)

[ПРИЛОЖЕНИЕ Д 112](#_Toc106126979)

[ПРИЛОЖЕНИЕ Е 114](#_Toc106126980)

[ПРИЛОЖЕНИЕ Ж 122](#_Toc106126981)

[ПРИЛОЖЕНИЕ З 127](#_Toc106126982)

[ПРИЛОЖЕНИЕ И 132](#_Toc106126983)

[ПРИЛОЖЕНИЕ К 135](#_Toc106126984)

[ПРИЛОЖЕНИЕ Л 136](#_Toc106126985)

# ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ПО – Программное обеспечение

JS – JavaScript

TS – TypeScript

HTTP – HyperText Transfer Protocol – протокол передачи гипертекста

URL – Uniform Resource Locator – унифицированный указатель ресурса

JSON – JavaScript Object Notation – текстовый формат обмена данными,

основанный на JavaScript

SPA – Single Page Application – одностраничное приложение

HTML – HyperText Markup Language – язык гипертекстовой разметки

CSS – Cascading Style Sheets – каскадные таблицы стилей

AJAX – Asynchronous JavaScript and XML – асинхронный JavaScript и XML

DOM – Document Object Model – объектная модель документа

JSX – JavaScript XML

TSX – TypeScript XML

API – Application Programming Interface – программный интерфейс

приложения

REST – Representational State Transfer – репрезентативная передача состояния

SVG – Scalable Vector Graphics – масштабируемая векторная графика

UI – User Interface – пользовательский интерфейс

SPA – Single Page Application – одностраничное приложение

GUI – Graphical User Interface – графический интерфейс пользователя

Лайв – идущее в данный момент событие

Npm – node package manager – менеджер пакетов Node.js

Компонента – функция, возвращающая JSX(TSX)-разметку

Хук – функция, имеющая возможность «зацепиться» к состоянию компоненты извне

H2H – Head To Head – матчи друг против друга

# ВВЕДЕНИЕ

В наше время проблематично встретить человека, который никогда в жизни не интересовался спортом и событиями, которые происходят в этой сфере, ведь действительно он является неотъемлемой частью жизни каждого из нас. Одни посвящают жизнь спорту ради карьеры, другие ведут активный образ жизни для себя, для своего тела и здоровья, болельщики посещают спортивные мероприятия ради получения эмоций. Несмотря весомое количество дисциплин и видов спортивной деятельности, всех поклонников спорта объединяет одно - желание позитивных эмоций и воля к победе.

В век цифровых технологий, во всех сферах жизни общества, свойственно увеличение информатизации, за счет непрерывного развития цифровых технологий.

Доступ к информации важен и нужен, ведь это путь к цивилизации и развитию, доступ к современному миру в целом. Благодаря интернету у людей есть неограниченные возможности в получении актуальной информации по многим видам спорта, событиям, новостям, результатам матчей.

Информационные системы используют повсеместно, в любых сферах, в том числе и в спорте. В связи с этим возник спрос, на системы, дающие людям доступ к информации о спортивных мероприятиях, новостях, статистиках.

В настоящее время спортивные мобильные и веб-приложения создаются не только для занятий спортом, но и несут такую важную функцию, как предоставление возможности просмотра значимых спортивных событий с любого устройства, что, в свою очередь, необходимо для такой части интернет-аудитории, как болельщики. Для того, чтобы облегчить поиск нужной информации пользователям-болельщикам, были разработаны приложения – агрегаторы спортивных мероприятий.

Приложения-агрегаторы спортивных мероприятий собирают всю необходимую для болельщика информацию в одном месте. Именно благодаря работе данных систем люди с легкостью получают доступ к результатам матчей, статистике и составам команд, таблицам чемпионатов, актуальным новостям из мира спорта[[1]](#footnote-1). Но все же трудности возникают достаточно часто, ведь не все сервисы, всё оборудование работают всегда стабильно.

В Российском сегменте таких приложений насчитывается не так много, однако в общем мировом таких приложений масса. В основном все они работают по примерно одинаковой схеме – пользователь может выбрать интересующий его вид спорта, чемпионат, лигу, турнир, и получает список прошедших, идущих, предстоящих матчей в удобной форме. Для людей, которые активно следят за спортом, фанатов, отслеживающих результаты

Разрабатываемая информационная система поможет в анализе спорта, благодаря возможности ее использования через веб-браузер и простому, понятному интерфейсу.

Исходя из вышеизложенного, данная работа, а именно разработка клиентской части информационной системы для мониторинга спортивных событий, является актуальной.

Просмотр анализа прошедших игр команд, ее состав позволит сформировать человеку некоторые ожидания перед грядущим матчем, а если пользователь увлекается ставками на спорт, то и к возможной прибыли, поскольку грамотный анализ позволит сделать грамотные предположения по предстоящим играм.

Цель дипломного проекта – разработать клиентскую часть информационной системы для мониторинга спортивных событий, не содержащую рекламных сообщений.

Задачи, необходимые для решения поставленной цели:

* выполнить анализ современного информационного обеспечения мониторинга спортивных событий;
* выбрать программное обеспечение и инструменты для разработки;
* реализовать клиентскую часть информационной системы для мониторинга спортивных событий.

# 1 Современное информационное обеспечение мониторинга спортивных событий

Спортивное событие – это мероприятие, в котором принимают участие две команды, соревнующиеся в определенном виде спорта.

Под мониторингом спортивных событий подразумевается получение и отображение данных по законченным, текущим и будущим матчам в разных видах спорта, а также просмотр статистики команд, таблиц чемпионатов, новостей в сфере спорта.

Таким образом, агрегатор спортивных мероприятий – это комплексная платформа мультиспортивного контента, которая обеспечивает анализ внутриигровых данных, исследования в области спорта, результаты матчей в реальном времени [8].

Web-приложение для мониторинга спортивных событий должно предоставлять возможность получения данных из мира спортивных игр, матчей в удобном виде для пользователей с различными устройствами.

Подобная система очень важна для удобного получения информации из мира спорта. Ей пользуются как приближенные к спортивной сфере (профессиональные спортсмены, тренеры, аналитики), так и обычные люди, которым интересно узнать результат игры их любимой команды. Особенно актуальными агрегаторы становятся в период проведения крупных чемпионатов, таких как чемпионат мира, Европы, Лиги Чемпионов.

## Обзор аналогов агрегаторов спортивных событий

Естественно, в этой сфере уже существуют аналогичные системы, прежде чем начать разработку, рассмотрим уже реализованные приложения.

### 1.1.1 FlashScore

FlashScore — один из самых популярных сервисов для отслеживания спортивных событий. Основой проекта был веб-сайт, однако позже были реализованы и мобильные приложения.

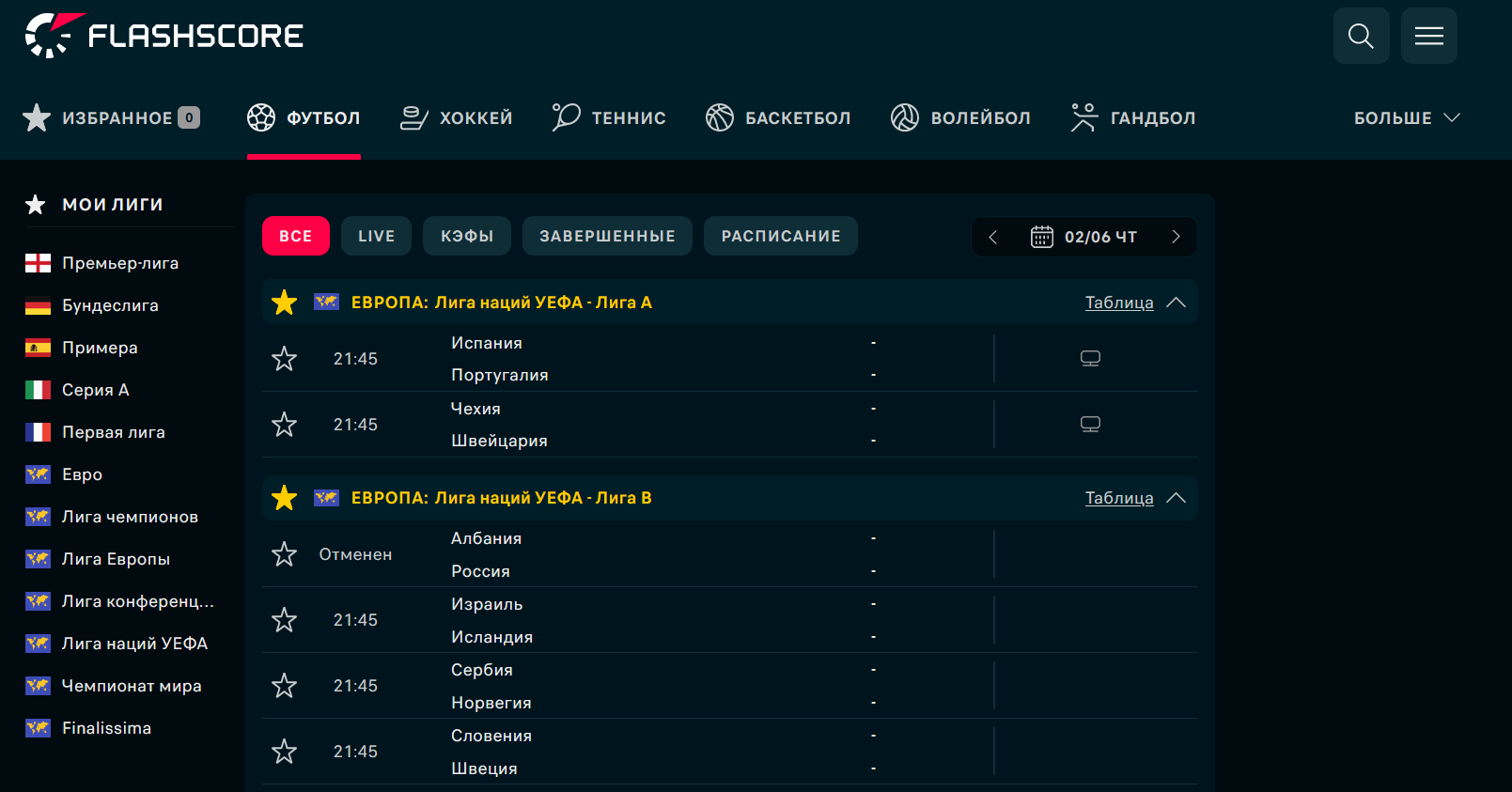


Рисунок 1 ­— Интерфейс веб-приложения FlashScore[[2]](#footnote-2)

Основной критерий выбора информационной системы мониторинга спортивных событий для обычного пользователя — количество поддерживаемых сервисом спортивных дисциплин. В случае с FlashScore — жаловаться не на что: в системе пристуствует тридцать два вида спорта, несколько тысяч лиг, включая самые низшие. Ежедневно обрабатываются тысячи событий.

Интерфейс приложения выполнен в одной стилистике как для веб, так и для мобильного приложения. Домашний экран представляет из себя каталог лиг и чемпионатов.

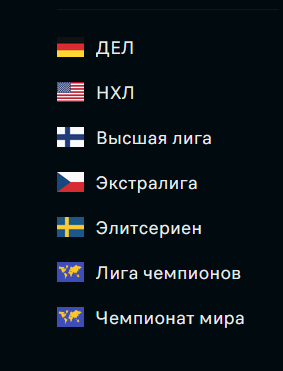


Рисунок 2 — Список чемпионатов для выбранного вида спорта - хоккей

Внутри каждого чемпионата доступен список спортивных событий на текущий день. Идущие в данный момент игры выделяются цветом. Любое событие можно добавить в избранное для того, чтобы была возможность получать push-уведомления при каждом изменении счета, нарушении, окончании таймов. Помимо этого, в этом разделе можно ознакомиться с актуальной турнирной таблицей.

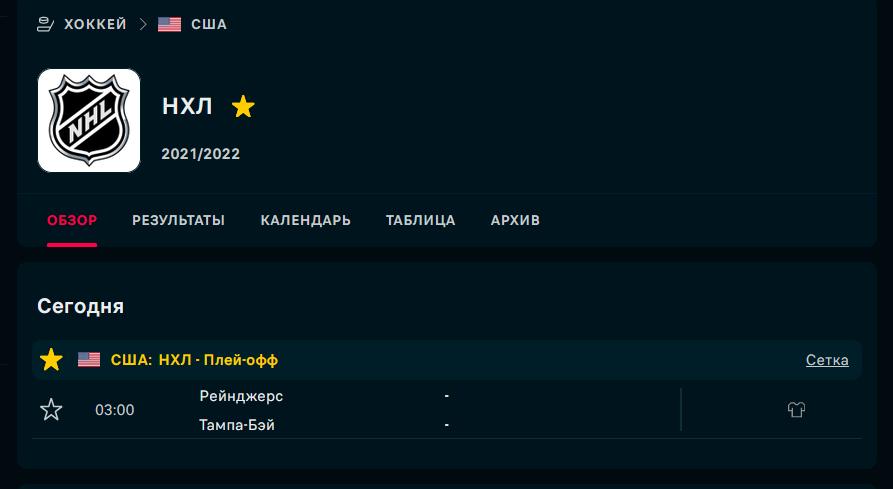


Рисунок 3 — Список событий для выбранного чемпионата

По конкретному событию можно посмотреть краткую статистику, результаты прошлых игр, а также инофрмацию о чемпионате, к которому относится матч. Заполненность данными зависит от вида спорта и важности соыбтия. Информация постоянно обновляется в режиме реального времени. Также к просмотру доступны средние коэффициенты от букмекерских контор. На сайте, как и в мобильном приложении присутствует всплывающая реклама, отключить которую невозможно.

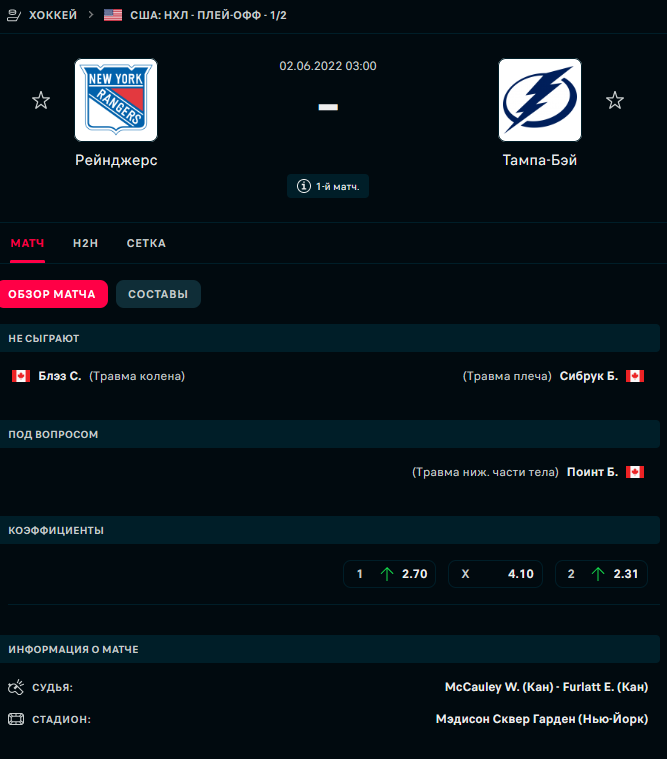


Рисунок 4 — Интерфейс выбранного события

### 1.1.2 Championat

Чемпионат — приложение-клиент от разработчиков Rambler.

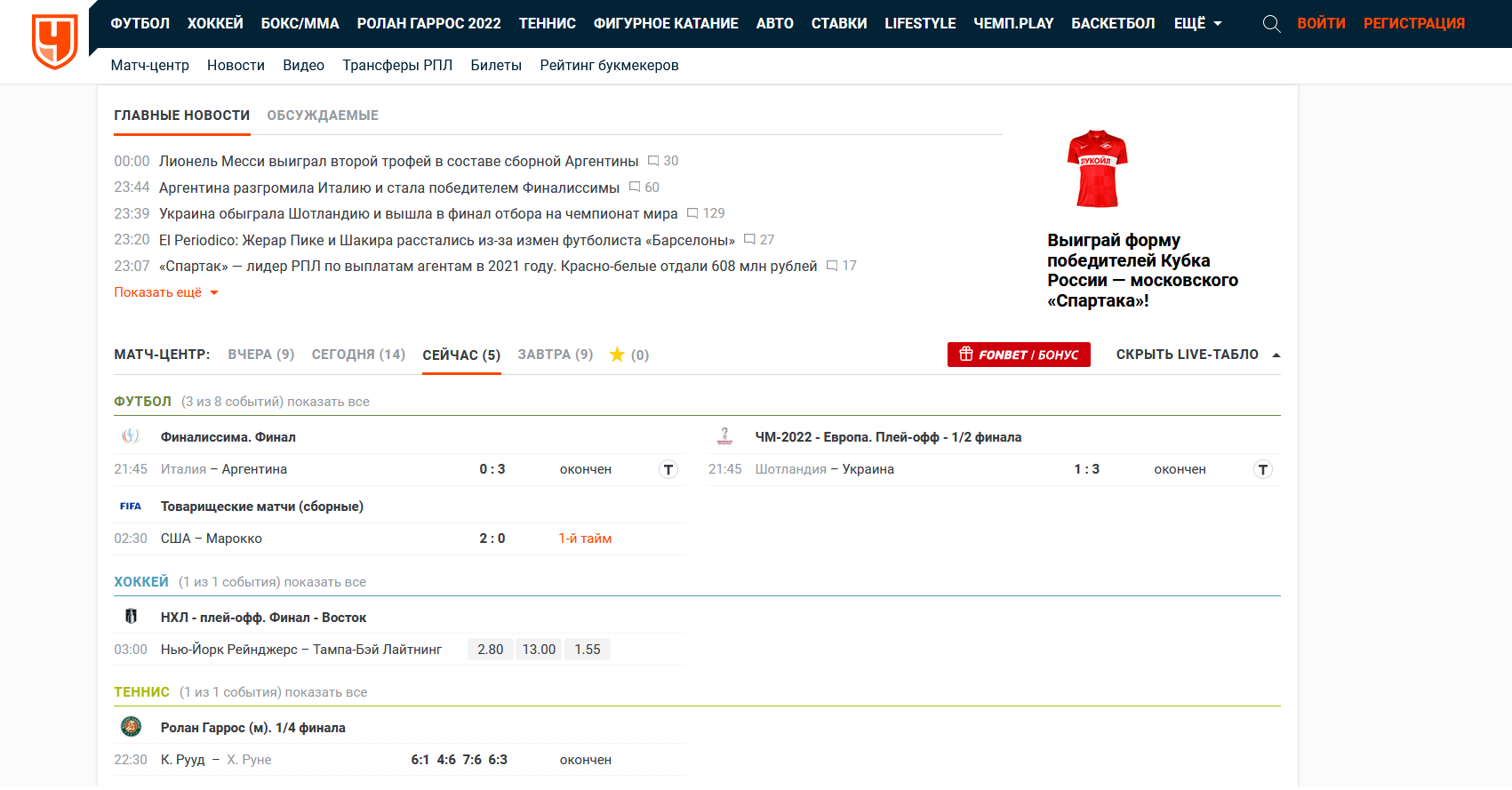


Рисунок 5 — Интерфейс веб-приложения Чемпионат[[3]](#footnote-3)

В отличии от FlashScore, Чемпионат не поддерживает огромное количество дисциплин. Доступны всего восемь видов спорта. Система расчитана на предоставление комфортного минимума основных и популярных чемпионатов для интересующихся людей.

Все события находятся в разделе «Матч-центр». Виды спорта для выбора отображены в удобном месте над контентной частью списка событий. Все матчи сгруппированы по чемпионатам, которые, в свою очередь, отсортированы по важности. В сервисе доступны к просмотру как прошедшие, так и будущие события.

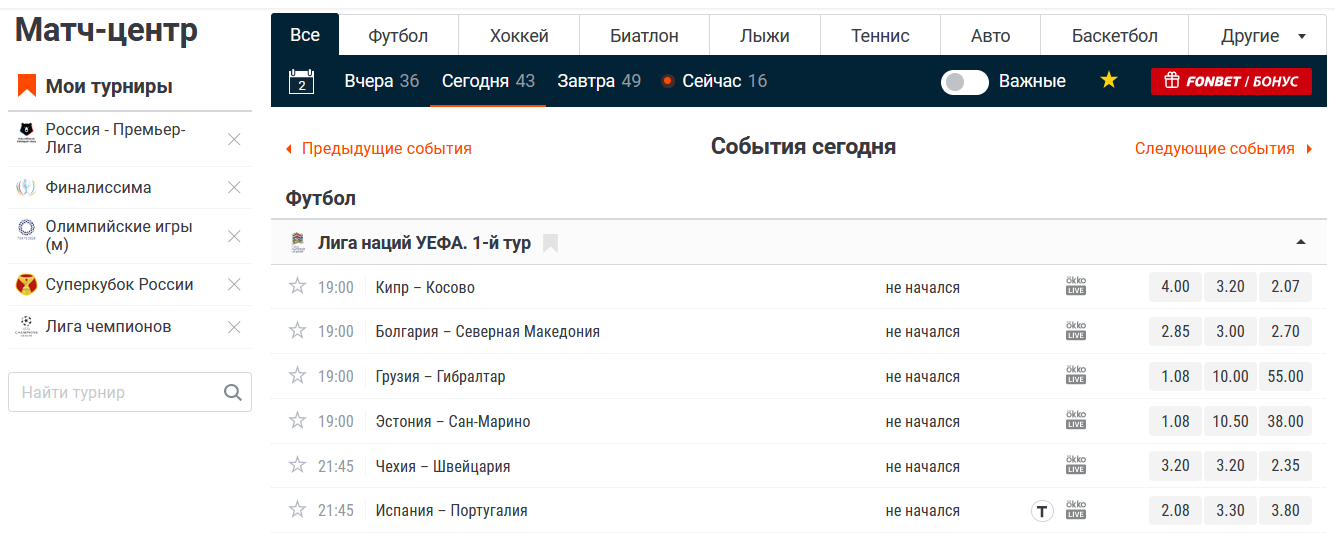


Рисунок 6 — Интерфейс «Матч-центра»

На страничке мероприятия используется удобный и привлекательный интерфейс. Также как и у FlashScore, количество охватываемых статистикой данных зависит от важности события, и сами события можно добавить в «избранное». Интересной функцией Чемпионата является возможность комментирования матчей на страничке мероприятия.

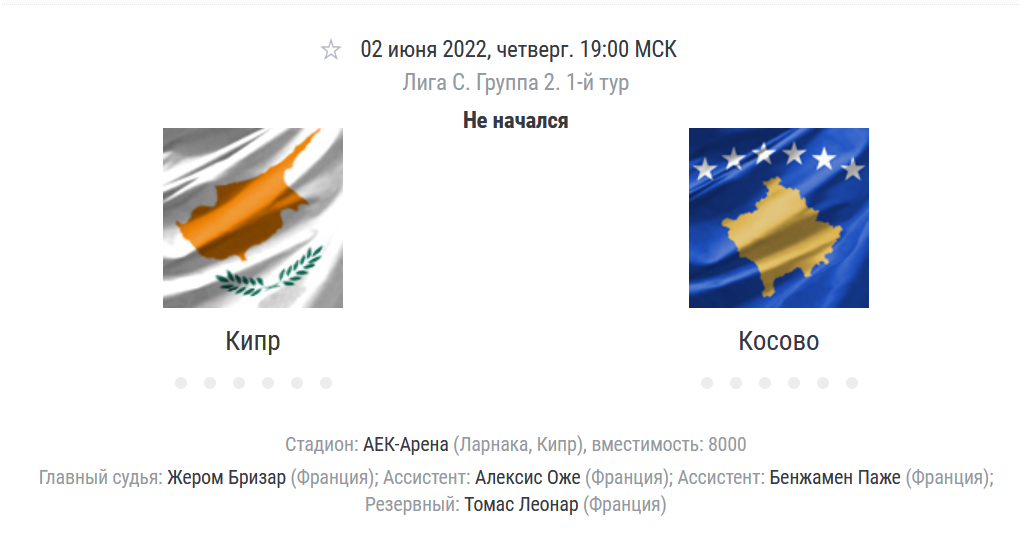


Рисунок 7 — Интерфейс мероприятия

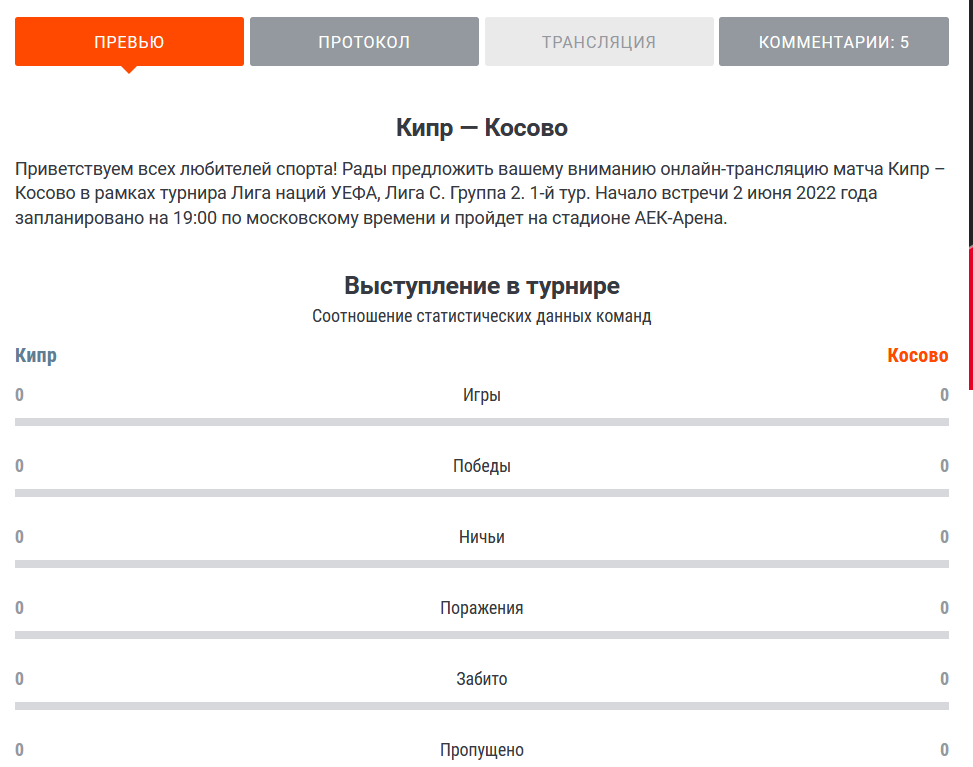


Рисунок 8 — Интерфейс мероприятия

Интересная функция мобильного приложения — «Фан-зона», в ней можно выбрать любимые команды и спортсменов для того, чтобы отслеживать все новости и матчи.

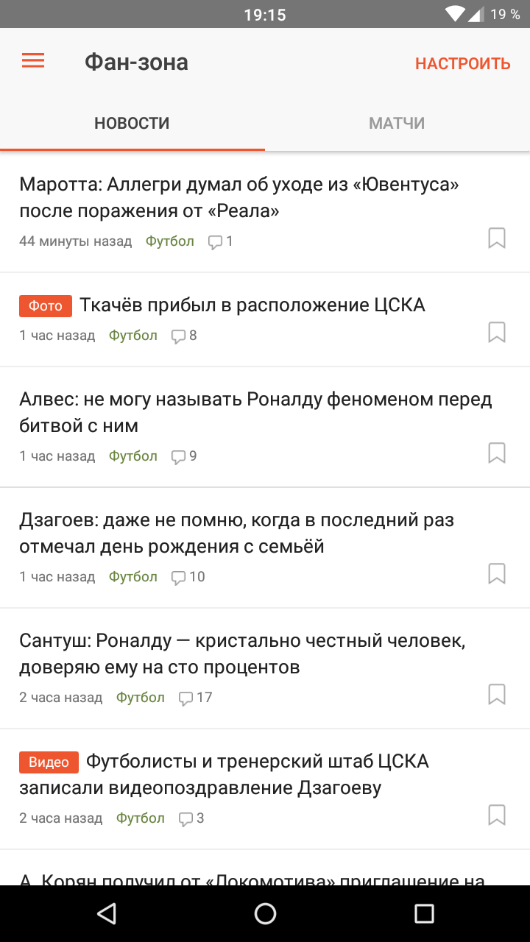


Рисунок 9 — Интерфейс «Фан-Зоны»

В обоих рассматриваемых системах мониторинга спортивных событий присутствует значительное количество всплывающих баннеров и рекламы букмекерских контор.

В основном все подобные агрегаторы работают одинаково, чего-то конкретно уникального в них нет, в основном уникальность заключается в мелких нюансах и дизайне, исходя из этого не вижу смысла разбирать каждую систему, но хотелось бы отметить еще LiveScore, 365Score, SofaScore.

## 1.2 Постановка задачи

Во всех рассмотренных приложениях реализация пользовательского интерфейса не самая удобная: избыточное количество рекламы, которую невозможно отключить, всплывающие баннеры, просмотр событий реализован на отдельной вкладке, что нарушает принципы SPA.

На основе рассмотренных приложений можно сделать следующие выводы: у разрабатываемого web-приложения должен быть интересный, современный и приятный дизайн, понятное меню, удобные механизмы работы с системой, а также не должно быть присутствия рекламы.

Одна из важных стадий разработки программного продукта – разработка требований. Она описывает функционал, дизайн и ограничения системы.

Было принято решение реализовать следующий функционал на проекте:

- поддержка шести видов спорта,

- просмотр данных по завершенным матчам,

- просмотр данных по текущим матчам,

- просмотр данных по будущим матчам,

- возможность выбора чемпионата для просмотра данных по событиям,

- возможность выбора страны для просмотра данных по событиям,

- возможность просмотра новостей спорта,

- возможность добавить чемпионат в «Избранное».

Также приложение должно быть производительным и оптимизированным и не содержать никакой рекламы. Для достижения этой цели будут выбраны библиотеки с минимальным размером и высоким быстродействием.

# 2 Выбор технологий и инструментов для разработки приложения

Для успешной реализации приложения произведем выбор необходимых технологий, программного обеспечения и инструментов для разработки:

## 2.1 Выбор технологий для разработки приложения

Классическое веб-приложение является клиент-серверным, то есть в нем осуществляется взаимодействие пользователя с сервером благодаря браузеру [14].

### 2.1.1 Серверная часть

В качестве серверной части был использован REST API от Tipsscore.

С его помощью можно получить доступ к расширенной спортивной статистике по пяти видам спорта: футбол, баскетбол, теннис, гандбол и волейбол, подробные данные о командах, турнирных таблицах, игроков, тренера, стартовые составы, стадионы команд, места проведения матча, видео голов и опасных моментов, статистику анализа xG. Благодаря разработчикам команды Tipsscore мне не нужно беспокоиться о хранении и заполнении данных. Информация о спортивных событиях поступает и обрабатывается в автоматическом режиме. Частота обновления информации – один раз в 15 секунд.

Данный API покрывает шесть видов спорта, более восьми тысяч лиг и чемпионатов, более шестидесяти тысяч команд и почти двести тысяч игроков.

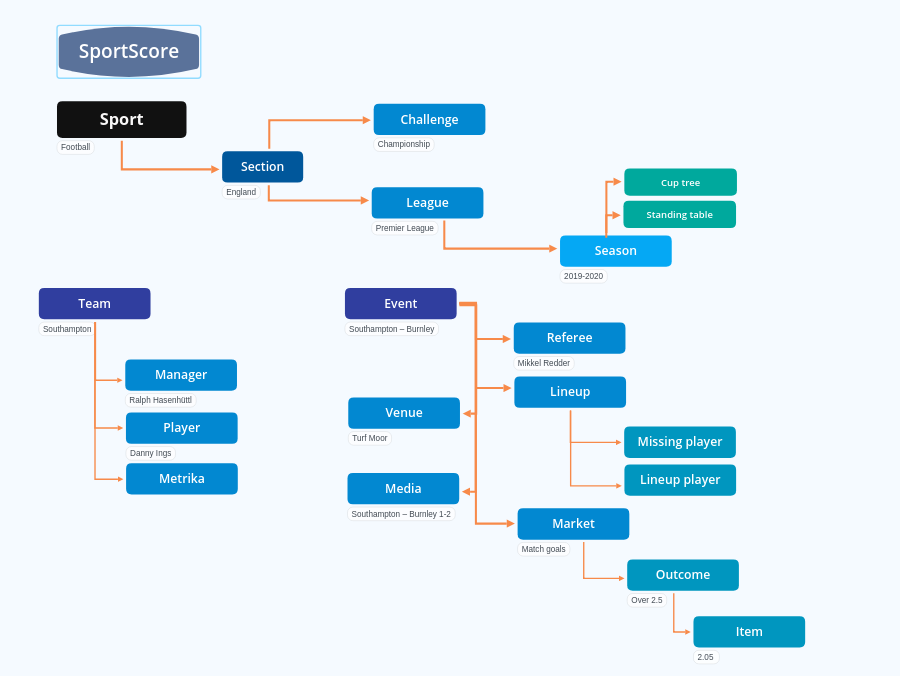


Рисунок 10 — Архитектура REST API Tipsscore[[4]](#footnote-4)

### 2.1.2 Клиентская часть

Для реализации клиентской части потребовались следующие технологии:

**Node JS**[[5]](#footnote-5) **—** кроссплатформенная среда выполнения, основанная на исполняемой JavaScript-библиотеке Chrome. Она упрощает создание масштабируемых приложений. Среда использует эффективную и оптимизированную модель ввода-вывода, поэтому она идеально подходит для разработки приложений [3]. В моем случае, Node.js необходим для того, чтобы получить доступ к npm - стандартный менеджер пакетов.

**JavaScript** — это язык программирования, позволяющий работать с документом как с объектной моделью, манипулировать контентом и всеми элементами, находящимся на странице путем написания функциональных модулей[[6]](#footnote-6).

**HTML**[[7]](#footnote-7) — это язык разметки, используемый для того, чтобы структурировать всю контентную часть web-страницы как строительные кирпичи, создать иерархию расположения элементов [16].

**CSS**[[8]](#footnote-8) - это язык стилей, с помощью которого мы стилизируем разметку страницы, например устанавливаем размеры элемента, цвет текста, цвет фона, способ отображения [16].

**TypeScript**[[9]](#footnote-9) – расширенная версия JavaScript, добавляющая типизацию, он позволяет разработчикам избежать ошибок, вызванных несоблюдением типов данных на этапе компиляции, а не на этапе непосредственного выполнения кода, что существенно ускоряет процесс разработки [10].

Разрабатываемое приложение было реализовано по принципу работы SPA – приложения, то есть был испоьзован всего один HTML-файл, являющийся основой всего приложения, а контент наполняется динамически при подгрузке необходимых данных посредством AJAX[[10]](#footnote-10). Основными фреймворками/библиотеками для реализации SPA-приложений являются React, Vue, Angular. Сравним их.

Таблица 1 — Сравнение библиотек для построения UI

|  |  |  |
| --- | --- | --- |
| Библиотека | Плюсы | Минусы |
| Angular | * Предоставляет жесткую структуру и архитектуру * Отлично подходит для создания крупных проектов * Большая кодовая база | * Необходимо изучать множество различных структур, которые присущи только Angular * Относительно слабая производительность * Наименьшая популярность * Не подходит для создания небольших проектов |
| Vue | * Отличная документация и поддержка сообщества * Высокая производительность * Легкость в изучении | * Много вспомогательных ресурсов на китайском языке * Не стабилен для создания больших проектов * Нет достаточного количества пакетов, для реализации различных задач |
| React | • Хорошая документация  • Отличная поддержка сообщества  • Много различных библиотек, для реализации задач  • Высокая производительность  • Не навязывает структуру, можно использовать что угодно и как угодно. | • Для полноценной работы необходимо подключение и знание дополнительных библиотек  • Новому разработчику будет тяжело стартовать в проекте |

Для реализации приложения я выбрал React JS[[11]](#footnote-11) – JavaScript библиотека для построения пользовательских интерфейсов, созданная разработчиками FaceBook и пропагандирующая философию декомпозиции кода проекта на функциональные модули.

Для хранения данных всего приложения в одном месте используется Redux **-** менеджер состояний. Чаще всего его используют с React, но его возможности не ограничиваются одной этой библиотекой. Хотя в React есть собственный метод управления состояниями, он плохо масштабируется.

Библиотека Redux[[12]](#footnote-12) — это способ управления состоянием приложения. Она основана на нескольких концепциях, изучив которые, можно с лёгкостью решать проблемы с состоянием.

Для упрощения работы с менеджером состояний была использована библиотека Redux Toolkit[[13]](#footnote-13), она позволяет значительно оптимизировать работу с состоянием.

Для упрощения работы со стилизацией использована styled-components[[14]](#footnote-14) – библиотека, которая позволяет прописывать CSS - стили методами JavaScript, а также избежать необходимости передавать наименования классов для блоков HTML[[15]](#footnote-15).

Для обеспечения маршрутизации в приложении была использована библиотека react-router-dom[[16]](#footnote-16).

Таким образом, для разработки клиентской части информационной системы мониторинга спортивных событий выбраны следующие технологии:

• REST API (Tippsscore)

• JavaScript

• TypeScript

• React

• Redux

• Экосистема React и Redux

Это лишь основные и самые важные зависимости для проекта, полный список подключенных библиотек указан в файле package.json (Приложение А) – документ, содержащий общую информацию о проекте.

## 2.2 Программное обеспечение для разработки

### 2.2.1 Среда разработки

Для того, чтобы комфортно разрабатывать приложение нужна производительная, удобная, умная среда разработки. Основываясь на своем опыте я выбрал WebStorm[[17]](#footnote-17) – интегрированная среда для разработки на JS и связанных с ним технологиях от JetBrains. Позволяет автоматизировать большую часть работы и посредством гибких опций настроить среду под свои предпочтения.

### 2.2.2 Сборка проекта

Поскольку проект разбит на функциональные модули, потребовалось ПО, которое собирает приложение воедино и запускает на локальном сервере – WebPack[[18]](#footnote-18). По своей сути — это сборщик статических модулей для современных приложений JavaScript.

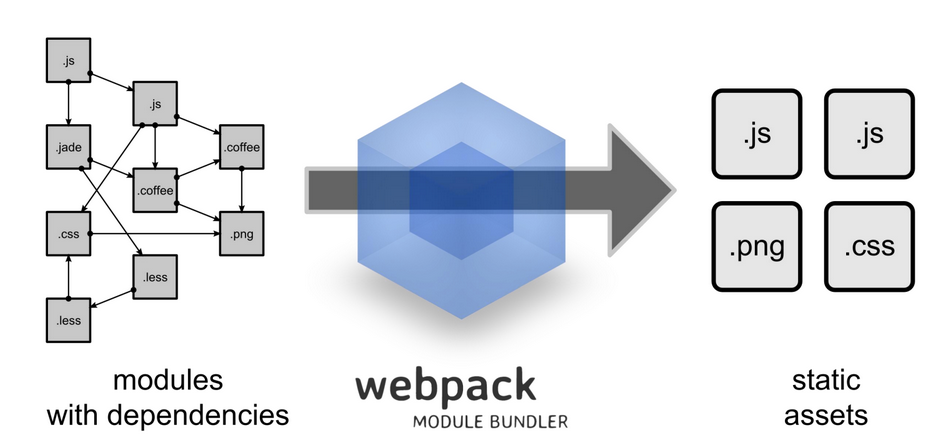


Рисунок 11 — Схема работы WebPack.

### 2.2.3 Система контроля версий

Для того, чтобы хранить информацию буквально об изменении каждой строчки кода в проекте используется система контроля версий. Это нужно для того, чтобы в случае каких-либо проблем не было непоправимых изменений и была возможность откатиться до предыдущей версии приложения.

Git[[19]](#footnote-19) — абсолютный лидер по популярности среди современных систем управления версиями. Это развитый проект с активной поддержкой и открытым исходным кодом. Для того, чтобы хранить информацию о проекте в системе требуется репозиторий – место, где хранится код и вся информация о его изменениях.

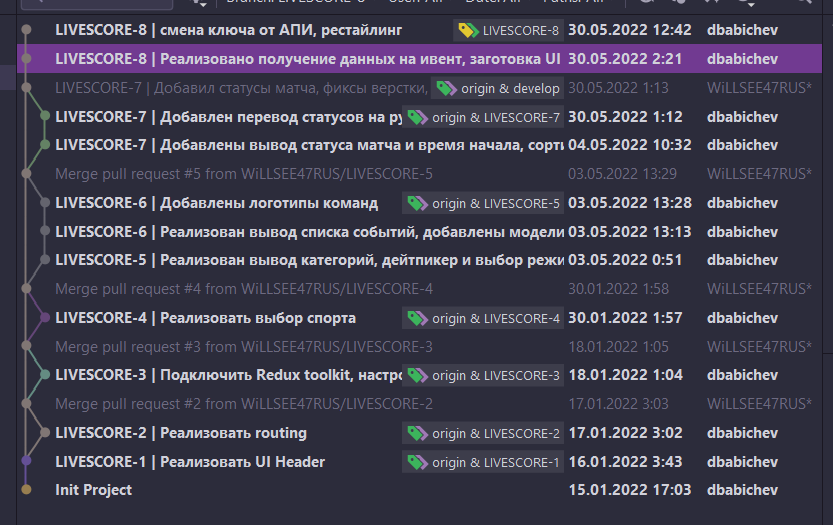


Рисунок 12 — Дерево изменения версий проекта, ветка «develop»

У файлов в репозитории есть так называемый жизненный цикл состояний. Для того, чтобы файл попал в нужную ветку, его сначала нужно добавить в список измененных файлов, модифицировать его, а затем зафиксировать изменения (на практике файлы группируются автоматически, в WebStorm есть GUI для работы с системой контроля версий) [12].

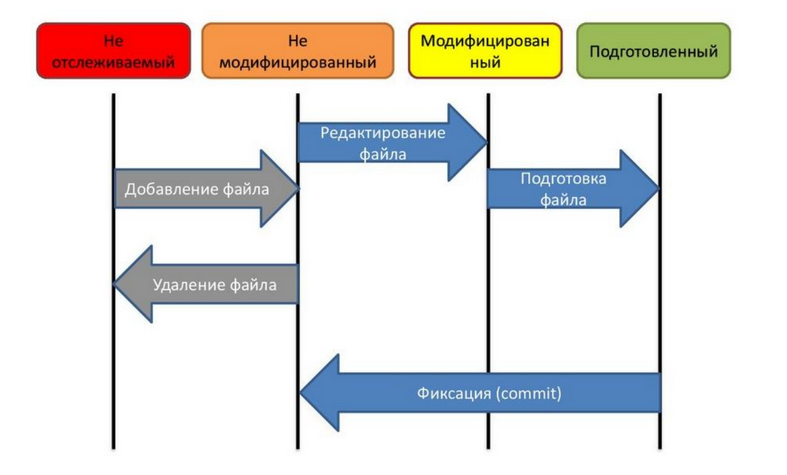


Рисунок 13 — Жизненный цикл файлов в системе контроля версий[[20]](#footnote-20)

## 2.3 Инструменты для упрощения разработки

Существует ряд инструментов, которые значительно помогают в процессе разработки.

**Lint** - это название, которое носила программа, целью которой был поиск подозрительных и неперносимых конструкций, ошибок. Однако сейчас этот термин применим к инструментам, упрощающим поиск ошибок на этапе разработки для всех языков программирования. В случае с JavaScript, ошибки могут быть обнаружены только при непосредственном выполнении кода, поскольку отсутствует этап компиляции [1].

Соответственно, линтер – инструмент, позволяющий найти ошибку в коде еще на этапе написания, значительно сокращает вероятность допущения ошибки и ускоряет процесс разработки. В моем проекте используется ESLint[[21]](#footnote-21) (конфигурация приведена в приложении А) – гибко настраиваемый инструмент по линтингу, который позволяет:

1. Найти существующие ошибки в коде;

2. Избежать глупых ошибок;

3. Избежать бесконечные циклы в условиях цикла for;

4. Убедится, что все методы getter возвращают что-то;

5. Не разрешить выражения debugger, console.log (и аналогичные);

6. Проверить наличие дубликатов cases в switch;

7. Проверить недоступный код.

На большом проекте, как правило, работает несколько программистов. У каждого из них может быть свой стиль написания кода, начиная от размера отступов, заканчивая стилем написания названия переменных. EsLint может помочь в поддержании определенной стилистике кода проекта при помощи утилиты Prettier[[22]](#footnote-22). Большая часть правил оформления уже задана по умолчанию, но есть возможность гибко настроить правила форматирования под определенный проект в формате .json/.yml. Это значит, что мы можем написать файл с едиными настройками для всей команды (приложение А).

Также значительно упрощают процесс разработки, а особенно поиска багов, расширения для браузера – React DevTools[[23]](#footnote-23) и Redux DevTools[[24]](#footnote-24), они позволяют просмотреть предыдущее и текущее состояние как компонентов, так и глобальное по приложению, отследить изменения [7].

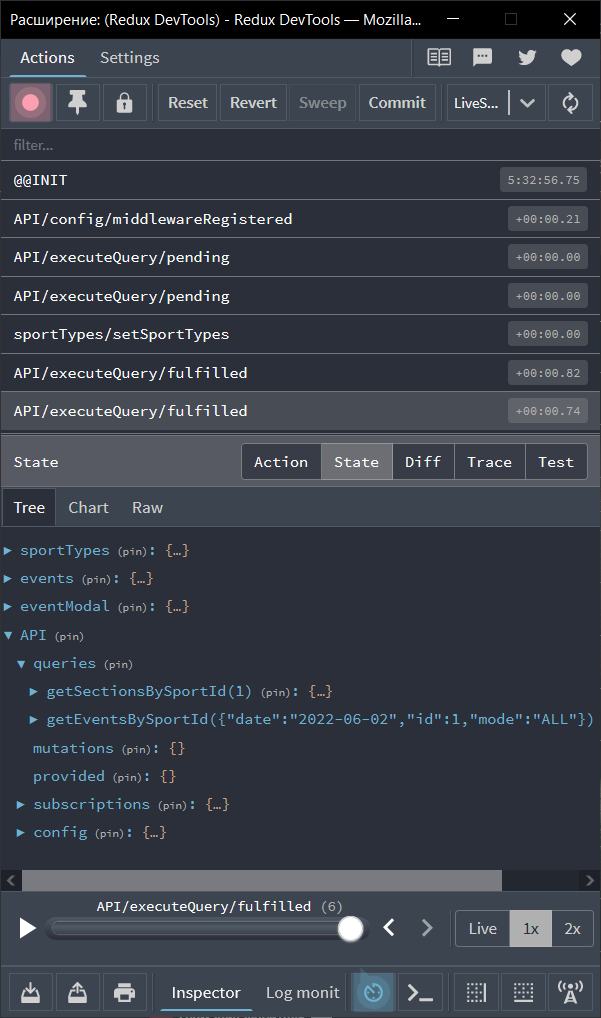


Рисунок 14 — Redux DevTools

# 3 Реализация клиентской части информационной системы для мониторинга спортивных событий

Всего было реализовано 3 вкладки – «События», «Избранное», «Новости».

По структуре приложение работает следующим образом:

Пользователь заходит на веб-сайт и сразу попадает на страницу с событиями по выбранному по умолчанию виду спорта, откуда у него есть возможность выбрать категорию или лигу для сортировки, а также для просмотра информации о них. Также со страницы событий есть возможность выбрать режим просмотра – «Все» или «Лайв» и установить фильтр по дате и названию команд, где «Все» - выводятся все события на текущую дату по виду спорта, где «Лайв» - текущие. Фильтр по дате доступен только в режиме «Все». Также у пользователя есть возможность выбрать вид спорта, устанавливаемый по умолчанию, добавить в избранное матч или чемпионат для более удобного просмотра информации.

В разделе «Избранное» отображаются события, которые были добавлены пользователем, либо события чемпионатов, добавленных пользователем с группировкой по виду спорта.

В разделе «Новости» отображаются последние новости спорта в формате сетки новостей с возможностью перейти к источнику.

Помимо этого, присутствует возможность открыть модальное окно с информацией о турнире, событии, команде для просмотра статистик, составов, событий, места проведения, предыдущих матчей между этими командами. А также ни в одном разделе не присутствует никаких рекламных баннеров.

## 3.1 Прототипирование интерфейса

Для создания прототипа интерфейса был использован бесплатный сервис NinjaMock. С помощью него можно создать примерный внешний вид интерфейса приложения, основанный на примерной структуре и требованиях к приложению.

На вкладке «События» пользователю доступен просмотр категорий по заданному виду спорта, а также список событий, полученных с сервера, основываясь на дате в двух режимах – «Все» и «Лайв».

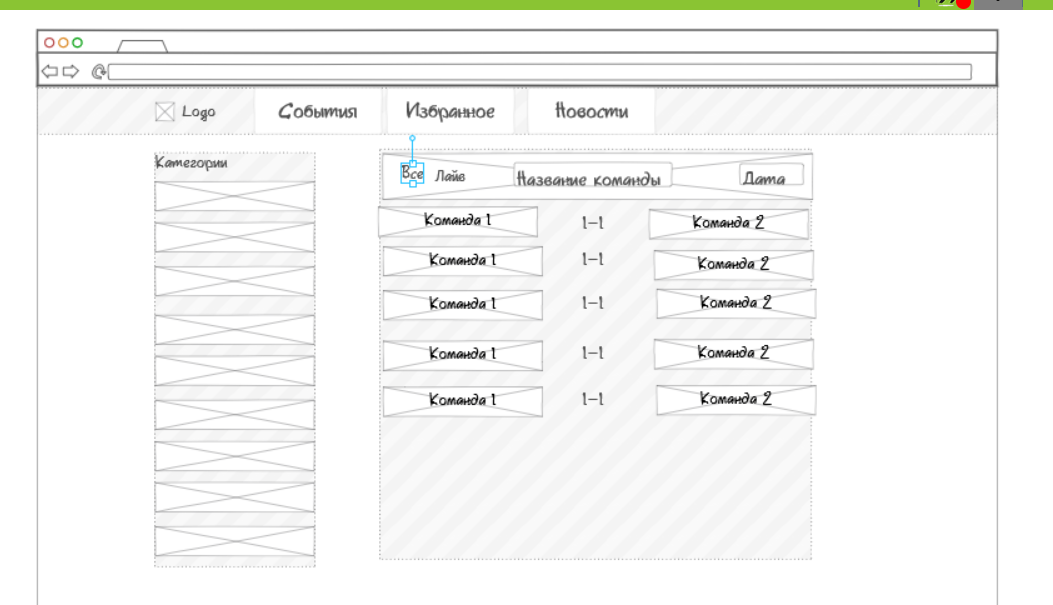


Рисунок 15 — Прототип интерфейса вкладки "События"

На вкладке «Избранное» пользователю доступен просмотр добавленных им в избранное чемпионатов и матчей, сгруппированных по виду спорта. Также доступно два режима для просмотра – «Все» и «Лайв».



Рисунок 16 — Прототип интерфейса вкладки "Избранное"

На вкладке «Новости» пользователю доступен просмотр актуальных новостей в виде слайдера, содержащий заголовок, текст и изображение новости.



Рисунок 17 — Прототип интерфейса вкладки "Новости"

## 3.2 Базовая конфигурация приложения

### 3.2.1 Инициализация JavaScript-проекта

Для обеспечения модульности приложения и получения возможности запускать скрипты сборки проекта, необходимо инициализировать приложение как npm-пакет. Для этого необходимо создать директорию проекта – место, где будут располагаться все файлы с кодом, эта директория называется корнем проекта, открыть ее в терминале – командной строке, и запустить команду «npm init».

В процессе инициализации, npm задаст несколько вопросов на тему того как должен называться npm-пакет, какая у него должна быть версия, описание и тому подобное.

В конце npm спросит все ли правильно и покажет содержимое файла package.json (приложение А), который и станет основой нового npm-пакета. После ответа yes этот файл запишется на диск и проект будет считаться готовым к работе.

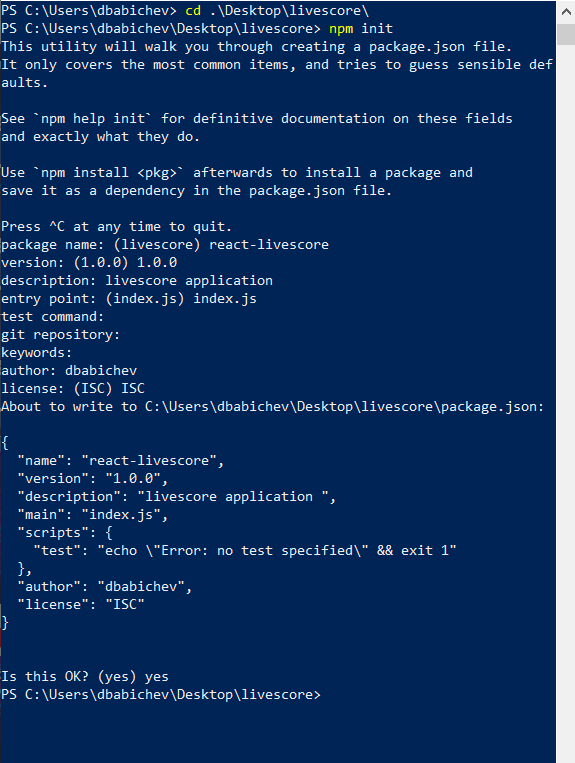


Рисунок 18 — Инициализация JavaScript-проекта как npm-пакета

### 3.2.2 Стартовая конфигурация файлов

Для того, чтобы продолжать удобную работу JavaScript-проектом была сформирована файловая структура следующего вида (обозначение «.» (точка) указывает на корневую директорию):

**./src** – директория, хранящая все исходные файлы приложения, от компонент до используемых изображений при реализации приложения;

**./src/assets –** директория, хранящая все файлы изображений, используемых при реализации;

**./src/components –** директория, хранящая все исходные файлы компонентов приложения;

**./src/constants –** директория, хранящая все постоянные величины, необходимые при реализации приложения;

**./src/hooks –** директория, хранящая вспомогательные самописные хуки (обособленные функции, имеющие возможность манипулировать с состоянием компонентов приложения);

**./src/model –** директория, хранящая все модели данных для типизации

**./src/modules –** директория, хранящая модули приложения (в моем случае – «События», «Избранное», «Новости»)

**./src/services –** директория, хранящая файлы, позволяющие работать с REST API;

**./src/store –** директория, хранящая файлы конфигурации и функциями управления глобальным состоянием приложения (Redux);

**./src/styles –** директория, хранящая файлы, используемые при стилизации веб-страниц;

**./src/utils –** директория, хранящая вспомогательные функции.

Также был создан файл index.html (приложение Б) – входной и единственный HTML-файл приложения, все отрисовки происходят именно в нем.



Рисунок 19 — Файловая структура приложения

### 3.2.3 Добавление TypeScript в проект

Для добавления типизации данных на уровне компилятора необходим Typescript. Для его установки требуется добавить typescript компилятор и ts-loader в проект с помощью команды «npm install --save-dev typescript ts-loader». Флаг –save-dev говорит о том, что эта зависимость для проекта необходима только на этапе разработки. После того, как скачались все пакеты, был создан конфигурирующий файл для TypeScript – tsconfig.json (приложение А). В нем указаны настройки компилятора и файлы деклараций типов.

### 3.2.4 Установка библиотек

Для того, чтобы пользоваться всеми благами используемых библиотек, было необходимо скачать все требуемые пакеты для приложения, для этого использована команда «npm install \*наименование пакета\*».

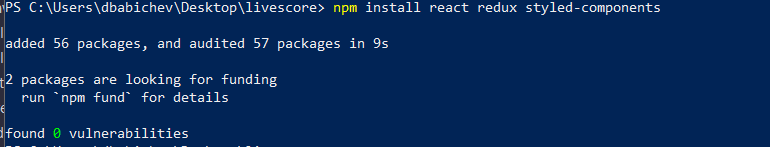


Рисунок 20 — Установка пакетов библиотек

После установки пакетов в корне проекта появилась директория node\_modules, хранящая все установленные пакеты.

Полный список используемых библиотек приведен в зависимостях (dependencies, devDependencies) в файле package.json (приложение А).

## 3.**3** Разработка приложения

Итак, после того, как все библиотеки были установлены, можно приступить к процессу разработки. В начале был создан входной скриптовый файл – index.tsx (приложение Л). Именно с него начинается сборка проекта, в нем осуществляется внедрение React-компонентов в исходный HTML файл путем исполнения метода ReactDOM.render, который принимает на вход два параметра – родительский компонент всего приложения app.tsx и ссылку на узел, в который его помещать. Глобальные стили для страниц описаны в файле GlobalStyle.ts (приложение И, с.132), в нем определены базовые переменные цветов и параметры для всего приложения. Большинство компонентов реализовано с помощью библиотеки styled-components, позволяющей использовать CSS внутри JavaScript. Как правило – на каждую компоненту выделяется своя директория, в которой должны присутствовать следующие файлы:

* исходный файл компоненты – index.tsx;
* файл, определяющий тип принимаемых параметров для компоненты - \*имя\*.types.ts;
* файл, содержащий стилизированные благодаря styled-components компоненты - \*имя\*.styled.ts.

### 3.**3.1** Разработка шапки

Шапка веб-страницы состоит из двух составных элементов – обертки (область, которую занимает вся шапка) и контентной части. Контент состоит из логотипа и навигационной панели, содержащий три ссылки на разделы приложения.

При реализации компонента необходимо идти от обратного, то есть сначала разработать компонент навигационной панели:

Компонент «NavBar» принимает на вход массив объектов, хранящих ссылки и наименования для разделов и возвращает стилизованный список компонентов «NavLink», помещенных внутрь так называемого flex-контейнера – элемента, обеспечивающего вывод элементов в ряд друг за другом. NavLink - ссылочный компонент из библиотеки react-router-dom, экосистемы Реакта, принимающий в параметры путь URL, на который перейдет пользователь при нажатии на ссылку.

Компонент шапки (приложение В, с.78) страницы носит наименование «Header», и в себе содержит стилизованные компоненты «HeaderWrapper» (приложение В, с.79) — компонент-обертка, содержащий базовые свойства стиля для всей области, используемой шапкой сайта, а именно цвет фона, цвет и размер разделителя блоков, «HeaderContent» (приложение В, с.79) — компонент, содержащий всю контентную часть шапки, содержащий определение стилей для блока с контентом, а именно размер (длина и ширина), отступы, форму отображения flex для расположения элементов в ряд, «Logo» (приложение В, с.78) — компонент, определяющий свойства логотипа — длину и ширину. А также компонент уже реализованной навигационной панели – «NavBar», принимающей на вход массив объектов, хранящих ссылки и наименования для разделов.

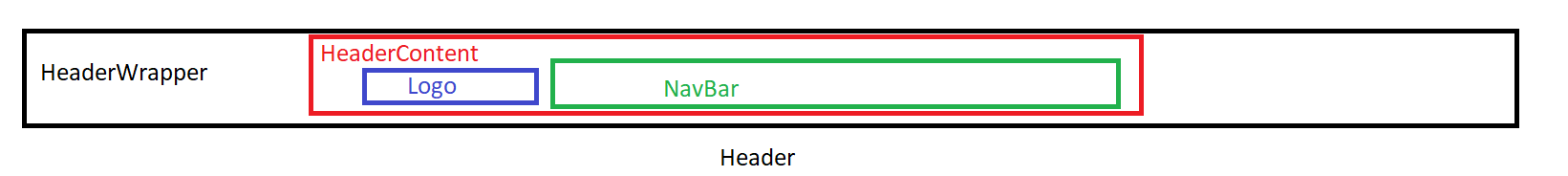


Рисунок 21 — Структура компонента Header

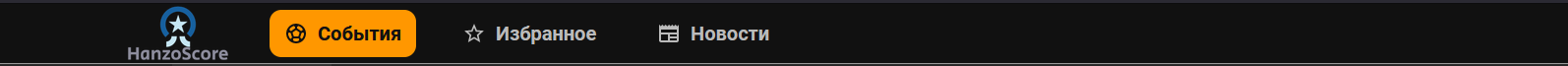


Рисунок 22 — Реализованный компонент Header

### 3.**3.2** Разработка маршрутизации

Как уже было сказано, маршрутизация осуществляется благодаря использованию библиотеки react-router-dom.

Всего в приложении доступно 3 маршрута – «/events», «/favourites», «/news» — разделы «События», «Избранное» и «Новости» соответственно. Роут «/events» может быть расширен идентификатором вида спорта, при наличии которого открывается контентная часть спортивных событий.

Все доступные маршруты описаны и реализованы в файле routeSwitcher.tsx (приложение В, с. 102). Этот компонент состоит из оберточной части Routes, содержащей набор элементов Route, получающих на вход в качестве параметров путь и используемый компонент для этого пути, поставляемых библиотекой react-router-dom. Сам по себе RouteSwitcher не является отображаемым компонентом, а лишь предоставляет доступ к определенным путям для дочерних элементов.

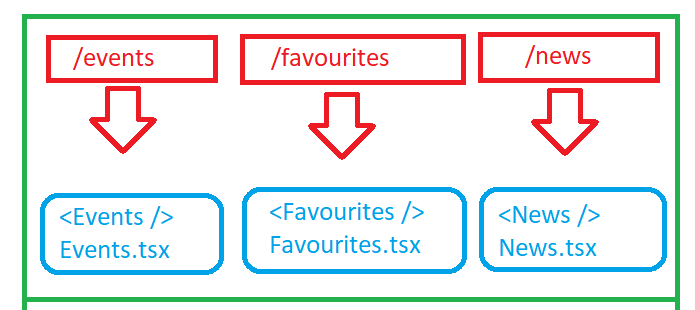


Рисунок 23 — Схема работы компонента RouteSwitcher

### 3.**3.3** Разработка модуля «События»

Модуль «События» доступен по маршруту «/events», и состоит из следующих составляющих частей.

**Доступные виды спорта** – виды спорта запрашиваются с сервера, обрабатываются хуком useSportTypesFetch (приложение Д, с.112), который проверяет, есть ли данный по видам спорта в локальном хранилище браузера, и, при условии их отсутствия запрашивает их с сервера, после чего данные помещаются в глобальное хранилище данных Redux.

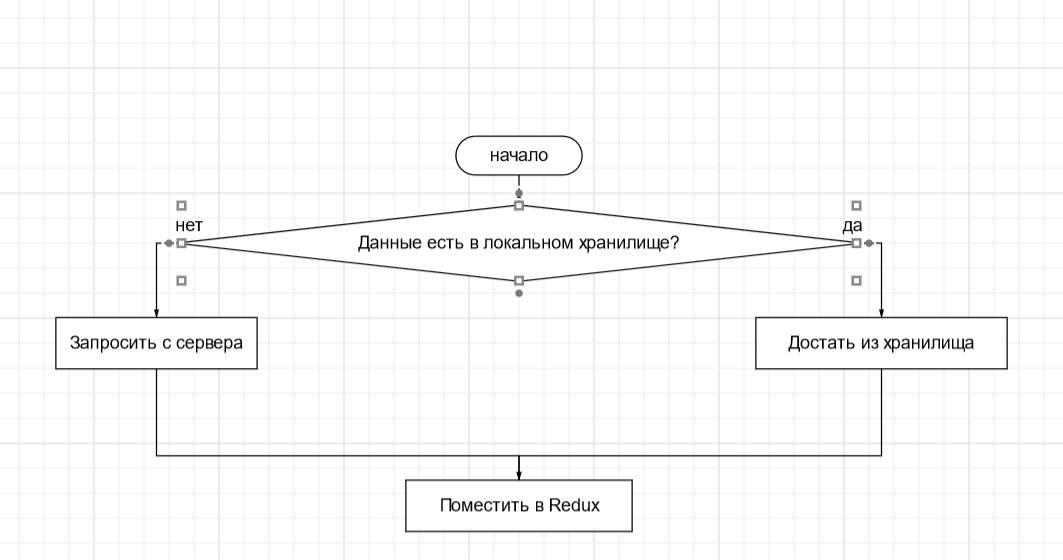


Рисунок 24 — Схема работы хука useSportTypesFetch

Далее в компоненте «SportTypes» (приложение В, с.94) эти данные достаются из глобального хранилища данных приложения (Redux) и отображаются с доступностью выбора аналогично компоненте NavBar . При выборе вида спорта происходит переход к маршруту, ему соответствующему. Компонент состоит из контейнерной обертки «SportTypesContainer» (приложение В, с.94), в которой определены базовые стили – размеры, способ отображения flex, цвет текста, внутри которого находится список элементов «NavLink», обеспечивающий роутинг.

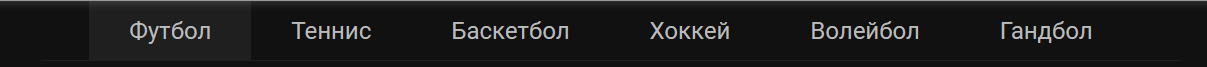


Рисунок 25 — Реализованный компонент SportTypes

**Список категорий** – категории запрашиваются с сервера, обрабатываются и записываются в глобальное состояние с помощью Redux Toolkit Query – встроенной функции в redux-toolkit, позволяющая упростить работу с API, запросами данных, путем формирования так называемых эндпоинтов и назначения для них обработчиков (приложение Ж) аналогично предыдущему самописному хуку useSportsTypesFetch, только без проверки хранения данных в localStorage, а сам принцип работы ровно такой же. Из созданной API экспортируются хуки, позволяющие с легкостью обработать процесс получения данных (хуки от redux toolkit query возвращают полученные данные, а также состояние выполнения запроса, индикатор загрузки), в данном случае нужный хук - useGetSectionsBySportIdQuery. Компонент «SectionsList» (приложение В, с. 92) состоит из контейнерной части — «SectionsListContainer» (приложение В, с. 91), в котором определены стили фона, способа отображения, отступов, размеров, содержащей полученный список компонентов «SectionItem» (приложение В, с. 91), в котором определены стили фона, отображения для конкретного элемента — доступных категорий, который обновляется при каждой смене выбранного вида спорта и отображаемый в виде flex-контейнера, содержащем логотип или флаг категории и название. При нажатии на категорию в списке событий происходит фильтрация по выбранной категории.

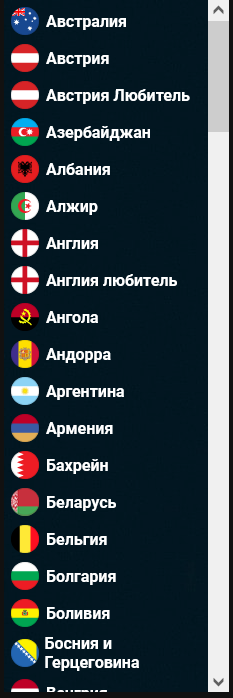


Рисунок 26 — Реализованный компонент SectionsList

**Cписок событий** – события запрашиваются с сервера на основе идентификатора спорта и выбранной даты, обрабатываются и записываются в глобальное состояние с помощью Redux Toolkit Query и хука useGetEventsBySportIdQuery аналогично предыдущим запросам, принимающим на вход следующие параметры: id спорта, режим отображения, дата для сортировки. Данные обновляются каждую минуту, а также при изменении вида спорта. В списке событий реализована фильтрация по наименованию команды и группировка по чемпионату (или лиге, если информация о чемпионате отсутствует). Компонент «EventsList» (приложение В, с. 73) разделен на две составляющие части – шапка «EventsListHeader» (реализовано аналогично «Header») с переключателем режима («Все» или «Лайв»), текстовым полем для фильтрации и компонентом выбора даты — по сути, обычный flex-контейнер, о котором было сказано ранее, содержащий стилизованные элементы кнопок EventsModeButton (приложение В, с. 73), от активного значения которых зависит выполняемый запрос на сервер, а также поле ввода и поле выбора даты, предоставленных библиотекой Material UI, и сам список событий, хранящийся внутри компонента «EventsListContainer» (приложение В, с. 72), в котором определены стили для списка событий — размер, фоновое изображение, способ отображения, отфильтрованный по наименованию и выбранной категории, содержащий группировку по чемпионату.



Рисунок 27 — Реализованный компонент EventsList

**Модальное окно события**, компонент Event (приложение В, с. 67) – при нажатии на какое-либо событие, открывается модальное окно, содержащее всю информацию о нем. Данные запрашиваются с сервера на основе идентификатора события, обрабатываются и записываются в глобальное состояние с помощью Redux Toolkit Query и хука useGetEventDataByIdQuery (приложение Ж) аналогично предыдущим запросам. Компонент состоит из шапки «EventHeaderWrapper» (приложение В, с. 58) (реализовано аналогично «Header») с полным наименованием события и его текущим статусом. В левом углу находится кнопка для добавления события в «Избранное», при ее нажатии происходит заполнение данными о событии в localStorage – локальное хранилище браузера по ключу «favouritesEvents». Далее идет блок информации об участниках события и его статусе — «EventInfoWrapper» (приложение В, с. 66), если событие находится в лайве, показывается счет и текущий период, если оно еще не началось, формируется обратный отсчет до старта с точностью до секунды с помощью компоненты «CountDownTimer» (приложение В, с.64). Ниже идет блок информации о матче — «MatchInfo», содержащий в себе несколько разделов, реализованных в виде кнопок-переключателей аналогично «EventsModeButton»: статистика по периодам, составы, события матча, H2H, место проведения (приложение В, компоненты MatchStatistics (с. 89), MatchInfoLineUp (с. 85), MatchIncidents (с. 81), H2HEvents (с. 77), StadiumInfo (с. 94) соответственно), вся информация запрашивается соответственно при выборе раздела благодаря хукам Redux Toolkit Query (приложение Ж) аналогично предыдущим запросам. Данные обновляются каждую минуту при условии, что событие в лайве. Если по какому-то из разделов данные отсутствуют, раздел не отображается к выбору.

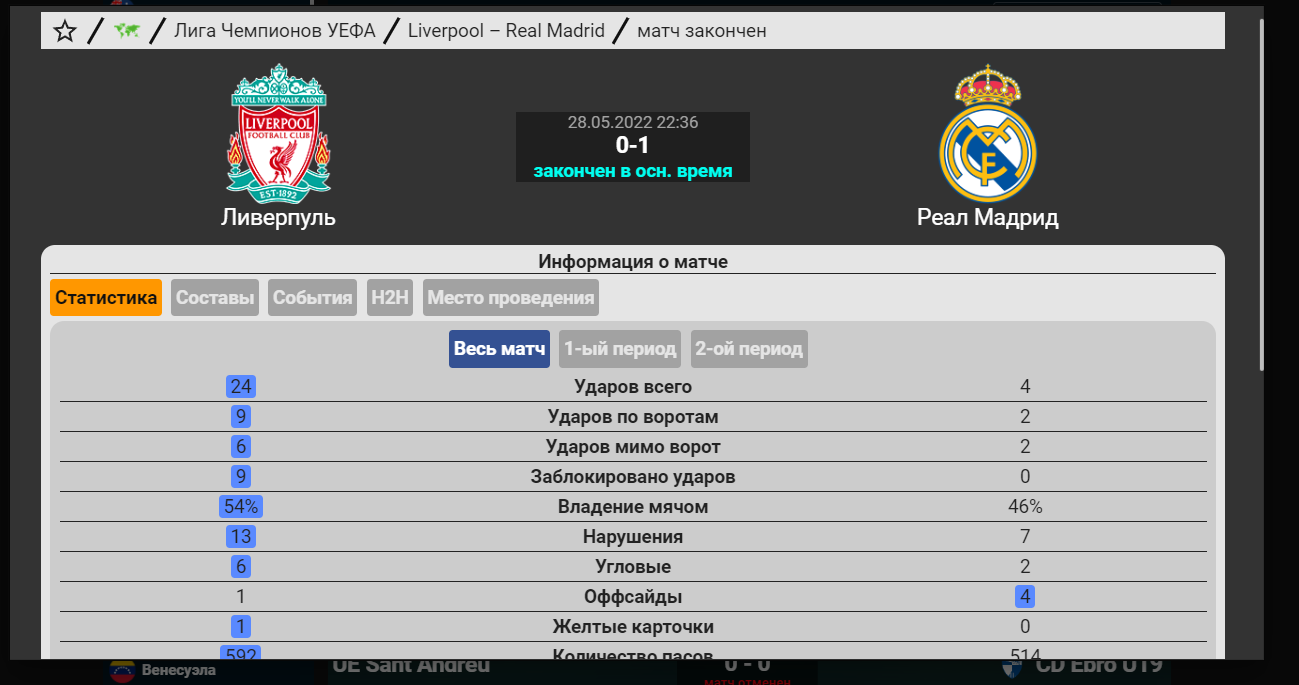


Рисунок 28 — Реализованный компонент EventModal – статистика



Рисунок 29 — Реализованный компонент EventModal – составы

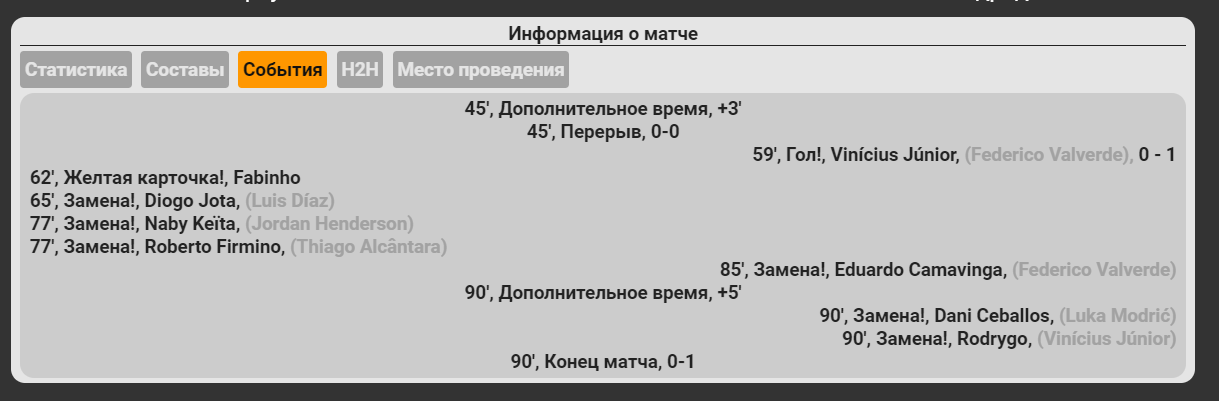


Рисунок 30 — Реализованный компонент EventModal – события



Рисунок 31 — Реализованный компонент EventModal – H2H

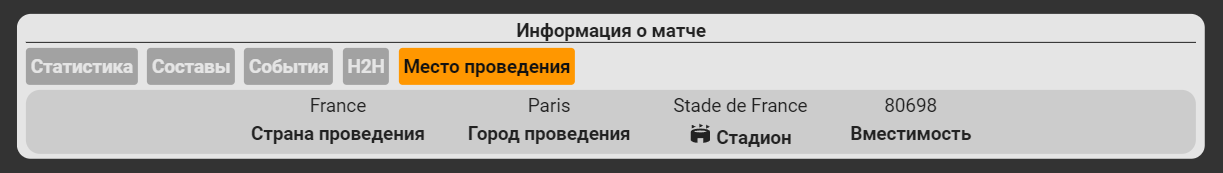


Рисунок 32 — Реализованный компонент EventModal – место проведения

**Модальное окно лиги / чемпионата**, компонент ChallengeModal (приложение В, с. 61) – реализовано аналогично другим модальным окнам, при нажатии на какое-либо наименование лиги, будь то в списке или карточке события, открывается модальное окно, содержащее всю информацию о турнире. Данные запрашиваются с сервера на основе идентификатора турнира, обрабатываются и записываются в глобальное состояние с помощью Redux Toolkit Query и хука useGetChallengeDataByIdQuery (приложение Ж) аналогично предыдущим запросам. Компонент состоит из шапки с полным наименованием события и его текущим статусом. В левом углу находится кнопка для добавления события в «Избранное», при ее нажатии происходит заполнение данными о событии в localStorage – локальное хранилище браузера по ключу «favouritesLeagues». Далее идет блок информации о турнире, разделенный на сезоны по годам, список которых запрашивается с сервера благодаря хуку из Redux Toolkit Query useGetSeasonsByLeagueIdQuery. В зависимости от сезона с сервера получаются разные данные по таблицам, данным турнира, событиям турнира, статистикам, деревьям кубков. Вся информация запрашивается соответственно при выборе сезона благодаря хукам Redux Toolkit Query (приложение Ж).

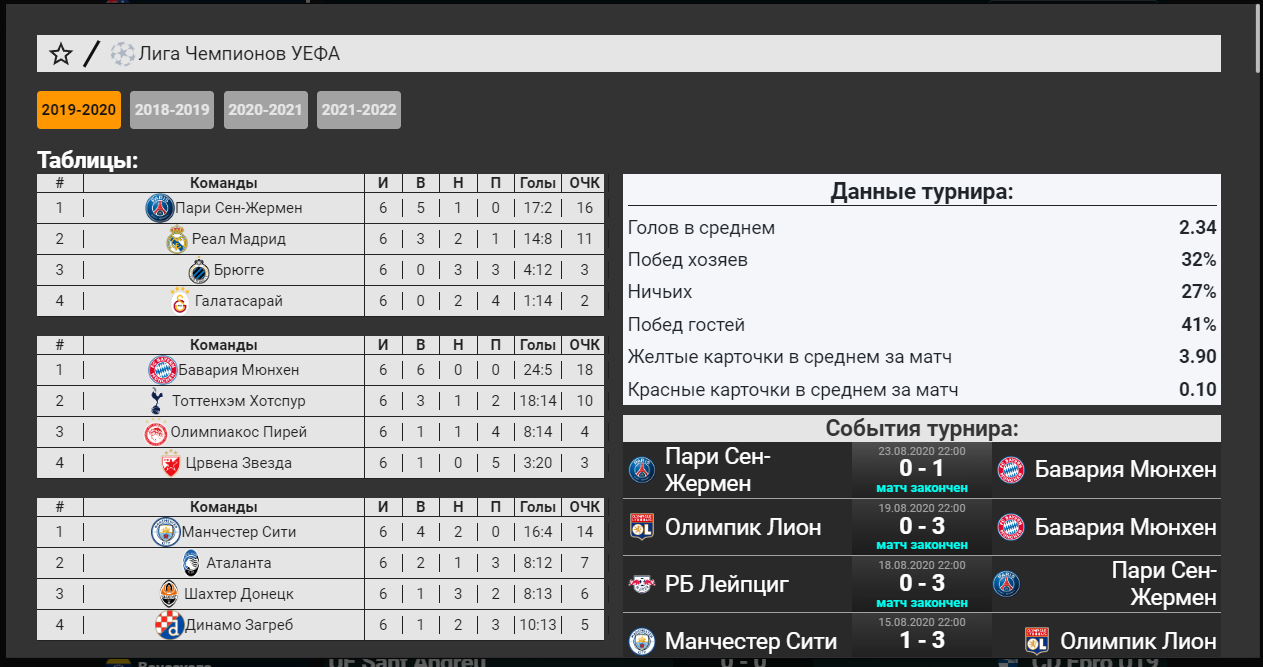


Рисунок 33 — Реализованный компонент ChallengeModal

**Модальное окно команды**, реализованный компонент TeamModal (приложение В, с. 97) – реализован аналогично другим модальным окнам, при нажатии на какое-либо наименование или логотип команды, будь то в списке или карточке события, открывается модальное окно, содержащее всю информацию о команде. Данные запрашиваются с сервера на основе идентификатора, обрабатываются и записываются в глобальное состояние с помощью Redux Toolkit Query и хука useGetTeamDataByIdQuery (приложение Ж) аналогично другим запросам. Компонент состоит из шапки с полным наименованием события и его текущим статусом. Далее идет блок информации о команде, тренере, составам, таблицам, турнирам, событиям, статистикам. Вся информация запрашивается благодаря хукам Redux Toolkit Query (приложение Ж).

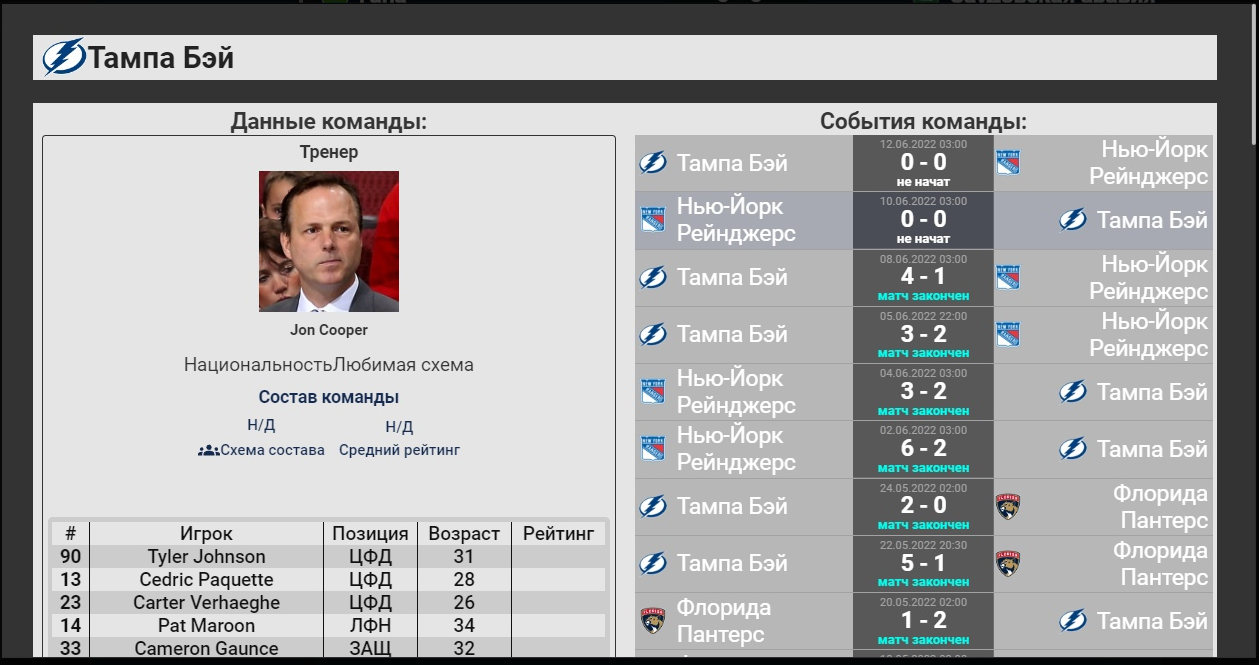


Рисунок 34 — Реализованный компонент TeamModal

Таким образом, модуль «Events» (приложение Е, с.116) состоит из обертки, контейнерной части, содержащей два отображаемых по умолчанию элемента – блок с категориями и блок с списком событий, с возможностью выбора категории для фильтрации и поиска событий по наименованию команд, а также возможностью выбора даты, и три скрытых по умолчанию модальных окна, вызываемых при открытии события, команды или чемпионата.

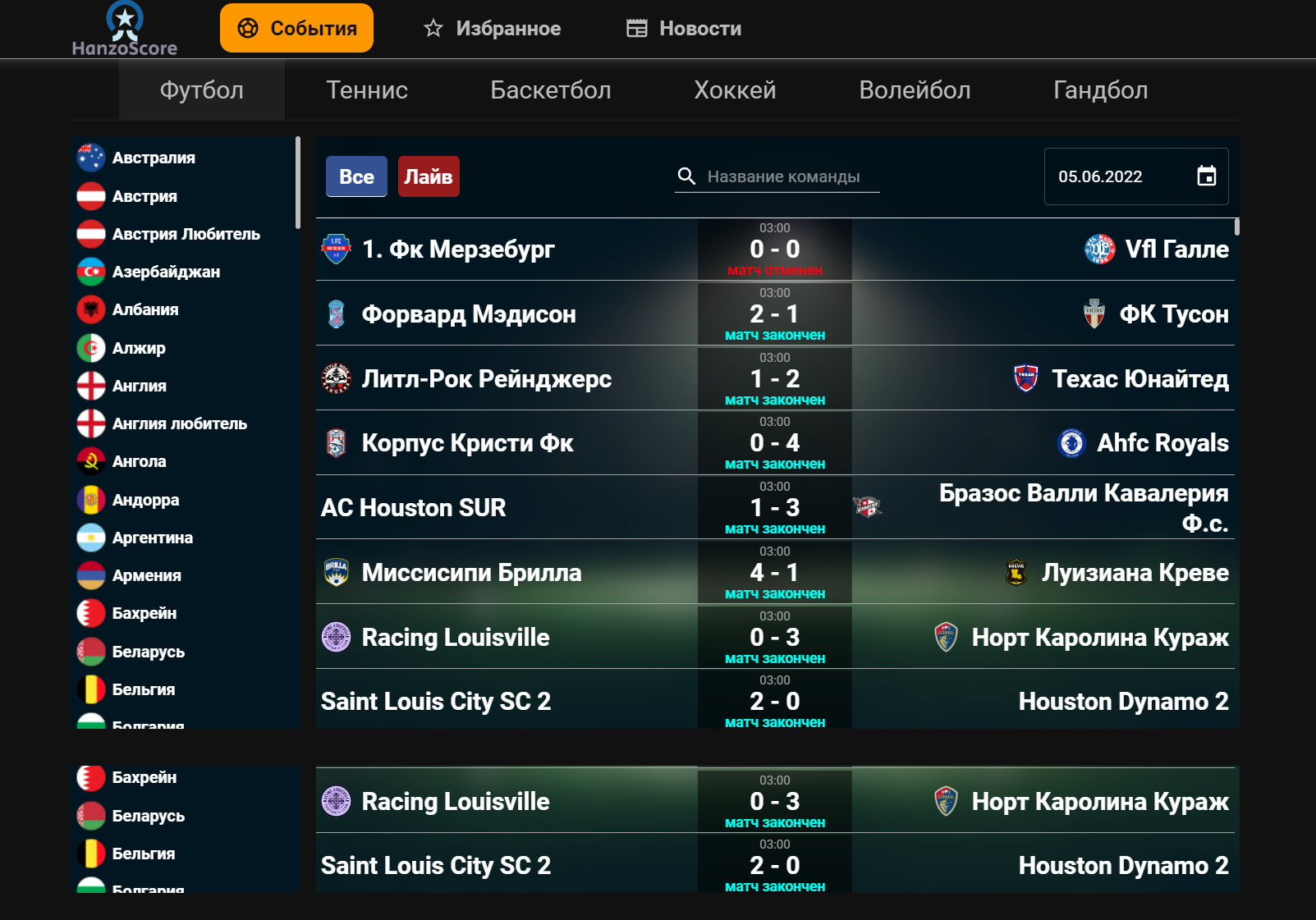


Рисунок 35 — Реализованный модуль Events

### 3.**3.4** Разработка модуля «Избранное»

Модуль «Избранное» (приложение Е, с. 117) реализован абсолютно идентчино модулю «События», доступен по маршруту «/favourites», и состоит из следующих составляющих частей:

**Список избранных чемпионатов** – чемпионаты попадают в этот список при их добавлении из карточки чемпионата в localStorage. Происходит парсинг информации о чемпионатах из JSON в массив данных, и обрабатывается и отображается с помощью переиспользования компонентов «SectionsListContainer», «SectionList» и «SectionItem», с добавлением шапки, указывающей на принадлежность списка к чемпионатам. При нажатии на какой-либо из перечисленных элементов открывается модальное окно чемпионата, о котором сказано ранее

**Список избранных событий** - события попадают в этот список при их добавлении из карточки матча в localStorage. Происходит парсинг информации о событиях из JSON в массив данных, и обрабатывается в компоненте «Favourites» (Приложение Е, с. 117). Происходит группировка событий по виду спорта с помощью функции groupBy (приложение К) и отображение с помощью переиспользования компонентов «EventsList», «EventItem, с добавлением шапки, указывающей на принадлежность списка к виду спорта. При нажатии на какой-либо из перечисленных элементов открывается модальное окно события, о котором сказано ранее.



Рисунок 36 — Реализованный компонент Favourites

### 3.**3.5** Разработка модуля «Новости»

Модуль «Новости» (приложение Е, с. 121) доступен по маршруту «/news», и состоит из следующих составляющих частей:

**Текстовая строка** для поиска новостей, при внесении в нее каких-либо символов отправляется запрос на сервер для получения новостей.

**Блок с новостями**, которые запрашиваются с сервера при помощи хука от Redux Toolkit Query – useGetNewsQuery (приложение Ж) аналогично предыдущим запросам, основываясь на содержании текстовой строки. Если же она пустая – запрашиваются последние «горячие» и обсуждаемые новости. По умолчанию отображается только превью-изображение и заголовок новости, однако при наведении курсора мыши отображается и содержание, при клике на новость осуществляется переход к ее источнику в новой вкладке браузера. Компонент состоит из обертки «NewsWrapper» (приложение Е, с 118.), в которой определены стили блока новостей, а именно размеры и отступ, внутри нее находится текстовое поле ввода «TextField», импортированное из библиотеки Material UI аналогично полю фильтрации в разделе событий, а также компонент «NewsContent» (приложение Е, с 121.), отвечающий за контентную часть полученных новостей, и содержащий определение стилей для данного блока, а именно отступы, способ отображения, размеры.

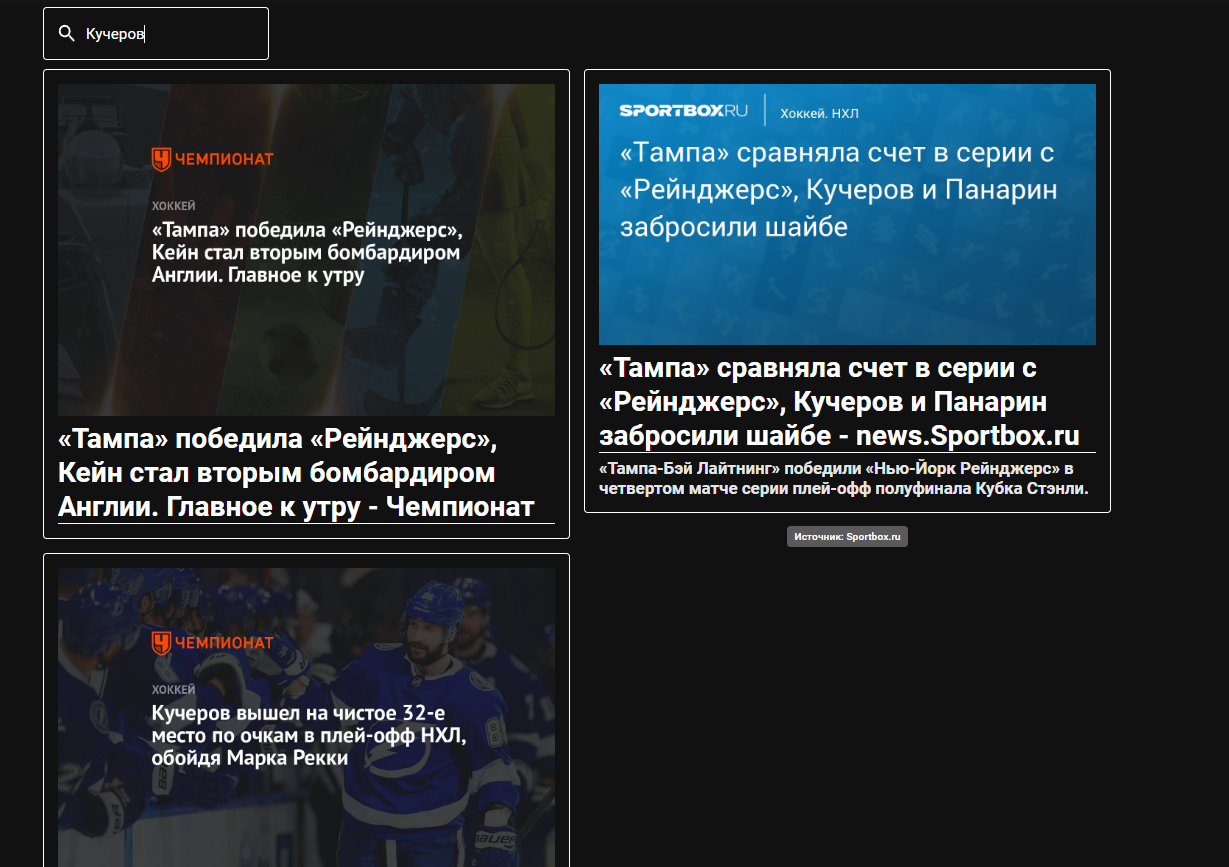


Рисунок 37 — Реализованный компонент News

# ЗАКЛЮЧЕНИЕ

Проанализировано современное информационное обеспечение мониторинга спортивных событий. Проведен обзор аналогов приложений-агрегаторов, выделены недостатки таких систем. На основе существующих приложений для мониторинга спортивных событий были выявлены недостатки подобных систем, самый главный из которых — избыточное количество рекламной информации. Также были составлены функциональные требования к разрабатываемому приложению.

Были выбраны необходимые технологии, инструменты и программное обеспечение для разработки клиентской части информационной системы для мониторинга спортивных событий. В качестве серверной части использован открытый REST API от разработчиков Tipsscore, охватывающий 6 видов спорта. Для реализации клиентской части отобран список необходимых библиотек и вспомогательных инструментов разработчика: React JS и его экосистема, Redux Toolkit, Styled-components, ESLint, React DevTools, Redux DevTools, работающих с языком программирования JavaScript.

Была реализована клиентская часть информационной системы для мониторинга спортивных событий. Данная система не содержит рекламных баннеров и сообщений, что позволяет обеспечить более удобный процесс получения и просмотра информации о спортивных мероприятиях.

Тестирование web-приложения проведено успешно, была проверена основная функциональность приложения.

Разработанное web-приложение мониторинга спортивных мероприятий полностью готово к эксплуатации пользователями, заинтересованными в получении данных о спортивных мероприятиях в удобной форме, без раздражающей рекламы. Однако, есть огромные перспективы в доработках, а именно в дизайне, в функциональности, оптимизации.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Браун И. Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript. / И. Браун, 2-е изд., Питер, 2021. 336 c.

2. Дакетт Д. Javascript и jQuery. Интерактивная веб-разработка / Д. Дакетт, Эксмо, 2020. 640 c.

3. Кантелон М. [и др.]. Node.js в действии / М. Кантелон, М. Хартер, Т. Головайчук, Н. Райлих, Питер, 2014. 548 c.

4. Мартин Р. Быстрая разработка программ. Принципы, примеры, практика / Р. Мартин, Вильямс, 2004. 752 c.

5. Мартин Ф. Рефакторинг кода на JavaScript: улучшение проекта существующего кода / Ф. Мартин, Вильямс, 2019. 464 c.

6. Мехди М. Непрерывное развитие API. Правильные решения в изменчивом технологическом ландшафте / М. Мехди, Питер, 2021. 272 c.

7. Миковски М., Пауэлл Д. Разработка одностраничных веб-приложений / М. Миковски, Д. Пауэлл, ЛитРес, 2022. 512 c.

8. Мишакин Т. Наука и спорт: современные тенденции // Информационное обеспечение крупных спортивных событий. 2017. 10 с.

9. Порселло Е. React: современные шаблоны для разработки приложений / Е. Порселло, O’Reilly, 2022. 320 c.

10. Черный Б. Профессиональный TypeScript. Разработка масштабируемых JavaScript-приложений / Б. Черный, 2021. 352 c.

11. Abbot D. JavaScript и jQuery: исчерпывающее руководство / D. Abbot, Эксмо, 2015. 880 c.

12. Chacon S., Straub B. Pro Git / S. Chacon, B. Straub, Apress, 2014.

13. Fenton S. Pro TypeScript: Application-Scale JavaScript Development / S. Fenton, Apress, 2014. 248 c.

14. Maccow A. JavaScript Web Applications: jQuery Developers’ Guide to Moving State to the Client / A. Maccow, O’Reilly Media, Inc., 2011. 278 c.

15. Maharry D. TypeScript revealed / D. Maharry, Apress, 2013. 123 c.

16. Musciano C., Kennedy B. HTML & XHTML: The Definitive Guide: The Definitive Guide / C. Musciano, B. Kennedy, O’Reilly Media, Inc., 2006. 681 c.

# ПРИЛОЖЕНИЕ А

**Конфигурационные файлы JavaScript-проекта**

**Файл package.json**

{

"name": "react-livescore",

"version": "1.0.0",

"description": "liveScore app",

"main": "index.js",

"scripts": {

"start": "webpack serve",

"test": "echo \"Error: no test specified\" && exit 1"

},

"repository": {

"type": "git",

"url": "git+https://github.com/WiLLSEE47RUS/react-livescore.git"

},

"author": "HanzoWillSee",

"license": "ISC",

"bugs": {

"url": "https://github.com/WiLLSEE47RUS/react-livescore/issues"

},

"homepage":"https://github.com/WiLLSEE47RUS/react-livescore#readme",

"devDependencies": {

"@types/react": "^17.0.38",

"@types/react-dom": "^17.0.11",

"@typescript-eslint/eslint-plugin": "^5.9.1",

"@typescript-eslint/parser": "^5.9.1",

"eslint": "^8.6.0",

"eslint-config-prettier": "^8.3.0",

"eslint-plugin-prettier": "^4.0.0",

"eslint-plugin-react": "^7.28.0",

"eslint-plugin-react-hooks": "^4.3.0",

"prettier": "2.5.1",

"react-error-overlay": "^6.0.9",

"styled-components": "^5.3.3",

"ts-loader": "^9.2.6",

"typescript": "^4.5.4",

"webpack": "^5.66.0",

"webpack-cli": "^4.9.1",

"webpack-dev-server": "^4.7.3"

},

"dependencies": {

"@date-io/date-fns": "^2.13.1",

"@date-io/dayjs": "^2.13.1",

"@emotion/react": "^11.7.1",

"@emotion/styled": "^11.6.0",

"@mui/icons-material": "^5.2.5",

"@mui/material": "^5.2.8",

"@mui/x-date-pickers": "^5.0.0-alpha.2",

"@reduxjs/toolkit": "^1.7.1",

"@types/react-router": "^5.1.17",

"@types/react-router-dom": "^5.3.2",

"@types/styled-components": "^5.1.20",

"axios": "^0.24.0",

"css-loader": "^6.7.1",

"date-fns": "^2.28.0",

"eslint-webpack-plugin": "^3.1.1",

"html-webpack-plugin": "^5.5.0",

"moment": "^2.29.3",

"qs": "^6.10.3",

"react": "^17.0.2",

"react-dom": "^17.0.2",

"react-redux": "^7.2.6",

"react-reveal": "^1.2.2",

"react-router": "^6.2.1",

"react-router-dom": "^6.2.1",

"react-transition-group": "^4.4.2",

"style-loader": "^3.3.1"

}

}

**Файл .eslintrc.js**

module.exports = {

'env': {

'browser': true,

'es2021': true

},

'extends': [

'eslint:recommended',

'plugin:react/recommended',

'plugin:react-hooks/recommended',

'plugin:@typescript-eslint/eslint-recommended',

'plugin:@typescript-eslint/recommended',

'prettier'

],

'parser': '@typescript-eslint/parser',

'parserOptions': {

'ecmaFeatures': {

'jsx': true

},

'ecmaVersion': 13,

'sourceType': 'module',

project: './tsconfig.json',

tsconfigRootDir: \_\_dirname,

},

'plugins': [

'react',

'react-hooks',

'@typescript-eslint',

'prettier'

],

'rules': {

'@typescript-eslint/interface-name-prefix': 0,

'@typescript-eslint/no-unnecessary-type-assertion': 'warn',

'@typescript-eslint/no-unsafe-assignment': 'warn',

'@typescript-eslint/no-non-null-assertion': 'error',

'@typescript-eslint/no-inferrable-types': 0,

'@typescript-eslint/no-empty-interface': 0,

'@typescript-eslint/no-explicit-any': 0,

'@typescript-eslint/unbound-method': 0,

'@typescript-eslint/member-ordering': 'warn',

'@typescript-eslint/no-magic-numbers': [

'error',

{

ignoreNumericLiteralTypes: true,

ignoreEnums: true,

enforceConst: true,

ignoreReadonlyClassProperties: true,

ignore: [0, 1, 2, 3, 4, 5, 12, 24, 60, 100]

},

],

'@typescript-eslint/naming-convention': [

'error',

{

selector: 'default',

format: ['camelCase', 'snake\_case'],

},

{

selector: 'function',

format: ['camelCase', 'PascalCase'],

},

{

selector: 'variable',

format: ['camelCase', 'UPPER\_CASE']

},

{

selector: 'parameter',

format: ['camelCase'],

leadingUnderscore: 'allow',

},

{

selector: 'memberLike',

modifiers: ['private'],

format: ['camelCase'],

leadingUnderscore: 'require',

},

{

selector: 'method',

format: ['camelCase'],

},

{

selector: 'enumMember',

format: ['PascalCase', 'UPPER\_CASE'],

},

{

selector: 'typeLike',

format: ['PascalCase'],

},

],

"react/react-in-jsx-scope": "off",

'indent': [

'error',

2

],

'linebreak-style': [

'error',

'windows'

],

'quotes': [

'error',

'single'

],

'semi': [

'error',

'always'

]

}

};

**Файл .prettierrc.json**

{

"singleQuote": true,

"trailingComma": "es5",

"printWidth": 120,

"tabWidth": 2

}

**Файл tsconfig.json**

{

"compilerOptions": {

"jsx": "react-jsx",

"target": "es6",

"lib": ["dom", "dom.iterable", "esnext"],

"allowJs": true,

"skipLibCheck": true,

"esModuleInterop": true,

"allowSyntheticDefaultImports": true,

"strict": true,

"forceConsistentCasingInFileNames": true,

"module": "esnext",

"moduleResolution": "node",

"resolveJsonModule": true,

"isolatedModules": true,

"noImplicitAny": false,

"noFallthroughCasesInSwitch": true

},

"include": ["src", "index.d.ts"]

}

# ПРИЛОЖЕНИЕ Б

**Содержимое файла index.html**

<!DOCTYPE html>

<html lang='en'>

<head>

<meta charset='UTF-8'>

<meta http-equiv='X-UA-Compatible' content='IE=edge'>

<meta name='viewport' content='width=device-width, initial-scale=1.0'>

<link

rel="stylesheet"

href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700&display=swap"

/>

<title>LiveScore</title>

</head>

<body>

<div id='root'></div>

</body>

</html>

# ПРИЛОЖЕНИЕ В

**Исходный код компонентов приложения**

**ChallengeModal.styled.ts**

import styled from 'styled-components';

import {

EventHeaderWrapper,

EventsHeaderDivider,

EventWrapper

} from '../Event/Event.styled';

import { MatchInfoViewModesWrapper } from '../MatchInfo/MatchInfo.styled';

import { FlexContainer } from '../common/common.styled';

export const ChallengeWrapper = styled(EventWrapper)``;

export const ChallengeHeaderWrapper = styled(EventHeaderWrapper)`

img{

height: 32px;

width: 32px;

}

`;

export const ChallengeHeaderDivider = styled(EventsHeaderDivider)``;

export const ChallengeSeasonsWrapper = styled(MatchInfoViewModesWrapper)``;

export const ChallengeSeasonButton = styled.div<{ active?: boolean }>`

display: flex;

min-height: 40px;

padding: 0 5px;

align-items: center;

justify-content: center;

color: ${props => !props.active ? 'var(--lightGray)' : 'var(--darkGray)'};

font-weight: 700;

background-color: ${props => !props.active ? 'var(--gray)' : 'var(--activeLinkBackground)'};

cursor: pointer;

margin-right: 10px;

transition: all 0.5s ease;

border-radius: 4px;

`;

export const ChallengeGrid = styled.div`

display: grid;

grid-template-columns: 605px 1fr;

grid-gap: 20px;

align-items: start;

justify-items: end;

`;

export const ChallengeSeasonTableGrid = styled.div`

display: grid;

grid-template-columns: 605px;

grid-gap: 20px;

`;

export const ChallengeSeasonTableGridTemplate = styled.div`

display: grid;

grid-template-columns: 50px 300px 40px 40px 40px 40px 50px 50px;

align-items: center;

color: var(--midGray);

img {

height: 32px;

}

border-bottom: 1px solid var(--midGray);

span {

display: flex;

align-items: center;

justify-content: center;

border-right: 1px solid var(--midGray);

}

`;

export const InfoDataFlex = styled(FlexContainer)`

width: 100%;

color: #212529;

font-size: 20px;

span{

&:last-child {

font-weight: 700;

}

}

`;

**ChallengeModal.tsx**

import React, { FC, useMemo, useState } from 'react';

import { Box, Modal, Tooltip } from '@mui/material';

import { useAppDispatch, useAppSelector } from '../../store';

import {

useGetChallengeDataByIdQuery,

useGetSeasonsByLeagueIdQuery, useGetSeasonsEventsByIdQuery,

useGetSeasonsTableByIdQuery,

} from '../../services/api';

import { Spinner } from '../Spinner';

import { getTranslations } from '../../utils/translations';

import { challengeModalActions, challengeModalSelector } from '../../store/challengeModal/challengeModal.slice';

import { ILeague } from '../../model/league.model';

import {

ChallengeGrid,

ChallengeHeaderDivider,

ChallengeHeaderWrapper, ChallengeSeasonButton,

ChallengeSeasonsWrapper, ChallengeSeasonTableGrid, ChallengeSeasonTableGridTemplate,

ChallengeWrapper, InfoDataFlex,

} from './ChallengeModal.styled';

import StarBorderIcon from '@mui/icons-material/StarBorder';

import {

EventTableNameTranslations,

EventTypesBackgroundColors,

EventTypesTranslations,

} from '../../constants/events.constants';

import { FlexContainer } from '../common/common.styled';

import EventItem from '../EventItem';

const style = {

position: 'absolute',

top: '50%',

left: '50%',

transform: 'translate(-50%, -50%)',

width: 1340,

height: 700,

bgcolor: 'background.paper',

border: '2px solid #000',

boxShadow: 24,

p: 4,

color: 'text.primary',

zIndex: 15555,

overflow: 'auto',

};

const ChallengeModal: FC = () => {

const { challengeId, isOpen } = useAppSelector(challengeModalSelector);

const dispatch = useAppDispatch();

const [selectedSeason, setSelectedSeason] = useState<number | null>(null);

const handleClose = () => {

dispatch(challengeModalActions.closeChallengeModal());

};

const { data, isLoading, isFetching } = useGetChallengeDataByIdQuery(challengeId, { skip: !challengeId });

const { data: seasonsResponse } = useGetSeasonsByLeagueIdQuery(challengeId, { skip: !challengeId });

const { data: tableResponse } = useGetSeasonsTableByIdQuery(selectedSeason, { skip: !selectedSeason });

const { data: eventsResponse } = useGetSeasonsEventsByIdQuery(selectedSeason, { skip: !selectedSeason });

const challengeData = useMemo(() => data?.data, [data]);

const seasonsData = useMemo(() => seasonsResponse?.data || [], [seasonsResponse]);

const tableData = useMemo(() => tableResponse?.data || [], [tableResponse]);

const eventsData = useMemo(() => eventsResponse?.data || [], [eventsResponse]);

// eslint-disable-next-line @typescript-eslint/no-unsafe-member-access,@typescript-eslint/no-unsafe-call

let isEventInFavourites = JSON.parse(localStorage.getItem('favouritesLeagues') || '[]').some(el => el.id === challengeId);

const handleAddToFavourites = () => {

if (!challengeData) return;

const currentFavourites: ILeague[] = JSON.parse(localStorage.getItem('favouritesLeagues') || '[]');

currentFavourites.push(challengeData);

localStorage.setItem('favouritesLeagues', JSON.stringify(currentFavourites));

// eslint-disable-next-line @typescript-eslint/no-unsafe-member-access,@typescript-eslint/no-unsafe-call

isEventInFavourites = JSON.parse(localStorage.getItem('favouritesLeagues') || '[]').some(el => el.id === challengeId);

};

const handleDeleteFromFavourites = () => {

if (!challengeData) return;

const currentFavourites: ILeague[] = JSON.parse(localStorage.getItem('favouritesLeagues') || '[]');

const newFavourites = currentFavourites.filter(el => el.id !== challengeId);

localStorage.setItem('favouritesLeagues', JSON.stringify(newFavourites));

// eslint-disable-next-line @typescript-eslint/no-unsafe-member-access,@typescript-eslint/no-unsafe-call

isEventInFavourites = JSON.parse(localStorage.getItem('favouritesLeagues') || '[]').some(el => el.id === challengeId);

};

return (

<Modal

open={isOpen}

onClose={handleClose}

>

<Box sx={style}>

{!isLoading && !isFetching && challengeData ? (

<ChallengeWrapper>

<ChallengeHeaderWrapper>

<Tooltip

title={isEventInFavourites ? 'Удалить лигу из избранного' : 'Добавить лигу в избранное'}

placement='top-end'

style={{ cursor: 'pointer', color: isEventInFavourites ? 'gold' : 'inherit', fontSize: '32px' }}

>

<StarBorderIcon onClick={isEventInFavourites ? handleDeleteFromFavourites : handleAddToFavourites} />

</Tooltip>

<ChallengeHeaderDivider />

<img src={challengeData.logo} alt='logo' />

<span>{getTranslations(challengeData)}</span>

</ChallengeHeaderWrapper>

<ChallengeSeasonsWrapper>

{seasonsData.map(el => (

<ChallengeSeasonButton

key={el.slug}

active={selectedSeason === el.id}

onClick={() => setSelectedSeason(el.id)}

>

{el.slug}

</ChallengeSeasonButton>

))}

</ChallengeSeasonsWrapper>

<ChallengeGrid>

<div>

<h2>Таблицы:</h2>

<ChallengeSeasonTableGrid>

{tableData.map(table => (

<div style={{ background: 'var(--lightGray)' }}>

<ChallengeSeasonTableGridTemplate style={{ fontWeight: 700 }}>

<span>#</span>

<span>Команды</span>

{Object.values(table.total\_keys).map(el => <span>{EventTableNameTranslations[el]}</span>)}

</ChallengeSeasonTableGridTemplate>

{table.standings\_rows.map(row => (

<ChallengeSeasonTableGridTemplate>

<span>{row.position}</span>

<span>

{row.team.has\_logo && <img src={row.team.logo} alt='logo' />}

{getTranslations(row.team)}

</span>

<span>{row.fields.matches\_total}</span>

<span>{row.fields.wins\_total}</span>

<span>{row.fields.draws\_total}</span>

<span>{row.fields.losses\_total}</span>

<span>{row.fields.goals\_total}</span>

<span>{row.fields.points\_total}</span>

</ChallengeSeasonTableGridTemplate>

))}

</div>

))}

</ChallengeSeasonTableGrid>

</div>

<FlexContainer fd="column" ai="flex-start" jc="center" rg="10px" style={{ width: '100%', marginTop: '28px'}}>

<FlexContainer fd="column" ai="flex-start" jc="center" rg="10px" style={{ width: '100%', padding: '5px', background: '#f4f6fa', color: '#212529'}}>

<h2 style={{ borderBottom: '1px solid var(--midGray)', width: '100%', textAlign: 'center'}}>Данные турнира:</h2>

<InfoDataFlex jc="space-between" ai="center">

<span>Голов в среднем</span>

<span>2.34</span>

</InfoDataFlex>

<InfoDataFlex jc="space-between" ai="center">

<span>Побед хозяев</span>

<span>32%</span>

</InfoDataFlex>

<InfoDataFlex jc="space-between" ai="center">

<span>Ничьих</span>

<span>27%</span>

</InfoDataFlex>

<InfoDataFlex jc="space-between" ai="center">

<span>Побед гостей</span>

<span>41%</span>

</InfoDataFlex>

<InfoDataFlex jc="space-between" ai="center">

<span>Желтые карточки в среднем за матч</span>

<span>3.90</span>

</InfoDataFlex>

<InfoDataFlex jc="space-between" ai="center">

<span>Красные карточки в среднем за матч</span>

<span>0.10</span>

</InfoDataFlex>

</FlexContainer>

<div style={{ backgroundColor: 'rgba(0,0,0,0.2)', backdropFilter: 'blur(15px)', height: '100%'}}>

<h2 style={{ background: 'var(--lightGray)', textAlign: 'center', color: '#353535'}}>

События турнира:

</h2>

{eventsData.map(el => <EventItem event={el} key={el.id}/>)}

</div>

</FlexContainer>

</ChallengeGrid>

</ChallengeWrapper>

) : (

<Spinner width='50' />

)}

</Box>

</Modal>

);

};

export default ChallengeModal;

**Common.styled.ts**

import styled from 'styled-components';

export const FlexContainer = styled.div<{ jc?: string; ai?: string, cg?: string; fd?: string; rg?: string; }>`

display: flex;

justify-content: ${props => props.jc};

align-items: ${props => props.ai};

column-gap: ${props => props.cg};

row-gap: ${props => props.rg};

flex-direction: ${props => props.fd || 'row'};

`;

export const Divider = styled.div`

width: 2px;

height: 100%;

background-color: var(--darkBlue);

`;

**CountDownTimer.tsx**

import { FC, useCallback, useEffect, useRef, useState } from 'react';

import moment from 'moment';

const calculateDuration = eventTime => moment.duration(Math.max(eventTime - (moment().valueOf()), 0));

const CountDownTimer: FC<{ eventTime: number, interval: number }> = ({ eventTime, interval }) => {

const [duration, setDuration] = useState(calculateDuration(eventTime));

const timerRef = useRef(0);

const timerCallback = useCallback(() => {

setDuration(calculateDuration(eventTime));

}, [eventTime]);

useEffect(() => {

// eslint-disable-next-line @typescript-eslint/ban-ts-comment

// @ts-ignore

timerRef.current = setInterval(timerCallback, interval);

return () => {

clearInterval(timerRef.current);

};

}, [eventTime, interval, timerCallback]);

return (

<div>

{!!duration.days() && `${duration.days()} дн.`} {!!duration.hours() && `${duration.hours()} ч. `}

{!!duration.minutes() && `${duration.minutes()} мин.`} {!!duration.seconds() && `${duration.seconds()} сек.`}

</div>

);

};

export default CountDownTimer;

**ProgressBar.tsx**

import styled from 'styled-components';

import { FC } from 'react';

const Container = styled.div`

height: 20px;

width: 100px;

position: relative;

`;

const BaseBox = styled.div`

height: 100%;

position: absolute;

left: 0;

top: 0;

border-radius: 3px;

transition: width 10s ease-in-out;

`;

const Background = styled(BaseBox)`

background: var(--gray);

width: 100%;

`;

const Progress = styled(BaseBox)<{ percent: number }>`

padding: 0 5px;

background: var(--darkBlue);

width: ${({ percent }) => percent}%;

color: var(--white);

font-size: 16px;

font-weight: 700;

`;

const ProgressBar: FC<{ percent: number }> = ({ percent }) => {

return (

<Container>

<Background />

<Progress percent={percent}>

{percent}%

</Progress>

</Container>

);

};

export default ProgressBar;

**Event.styled.ts**

import styled from 'styled-components';

import { EventsStatuses, EventsStatusesColors } from '../../constants/events.constants';

export const EventWrapper = styled.div`

width: 100%;

display: flex;

flex-direction: column;

row-gap: 15px;

overflow-y: scroll;

`;

export const EventHeaderWrapper = styled.div`

width: 100%;

height: 40px;

display: flex;

padding: 0 10px;

align-items: center;

font-size: 20px;

background-color: var(--lightGray);

color: var(--midGray);

`;

export const EventsHeaderDivider = styled.div`

height: 32px;

margin: 0 15px;

border: none;

border-left: 4px var(--darkGray) solid;

transform: rotate(30deg);

`;

export const EventInfoWrapper = styled.div`

width: 100%;

display: grid;

grid-template-columns: 1fr 250px 1fr;

justify-content: center;

align-items: center;

font-size: 24px;

`;

export const EventTeamInfo = styled.div`

display: flex;

flex-direction: column;

align-items: center;

img {

height: 150px;

width: 150px;

}

`;

export const EventScoreInfo = styled.div<{ status: EventsStatuses }>`

display: flex;

flex-direction: column;

text-align: center;

font-weight: 700;

color: var(--white);

background-color: rgba(0, 0, 0, 0.4);

backdrop-filter: blur(50px);

span {

font-size: 18px;

font-weight: 400;

color: var(--gray)

}

.status {

font-size: 20px;

font-weight: 600;

color: ${props => EventsStatusesColors[props.status]}

}

`;

**Event.tsx**

import React, { FC, useMemo } from 'react';

import { Box, Modal, Tooltip } from '@mui/material';

import { useAppDispatch, useAppSelector } from '../../store';

import { eventModalActions, eventModalSelector } from '../../store/eventModal/eventModal.slice';

import { useGetEventDataByIdQuery } from '../../services/api';

import { Spinner } from '../Spinner';

import { getTranslations } from '../../utils/translations';

import {

EventsStatuses,

EventsStatusesMoreTranslations,

EventsStatusesTranslations,

} from '../../constants/events.constants';

import { DEFAULT\_EVENT\_TIME\_FORMAT, DEFAULT\_TIME\_ZONE } from '../../constants/common.constants';

import { formatEventStartDate } from '../../utils/common';

import {

EventTeamInfo,

EventScoreInfo,

EventInfoWrapper,

EventHeaderWrapper,

EventsHeaderDivider,

EventWrapper,

} from './Event.styled';

import CountDownTimer from '../common/CountDownTimer';

import MatchInfo from '../MatchInfo';

import StarBorderIcon from '@mui/icons-material/StarBorder';

import { IEvent } from '../../model/events.model';

import { teamModalActions } from '../../store/teamModal/teamModal.slice';

const style = {

position: 'absolute',

top: '50%',

left: '50%',

transform: 'translate(-50%, -50%)',

width: 1340,

height: 700,

bgcolor: 'background.paper',

border: '2px solid #000',

boxShadow: 24,

p: 4,

color: 'text.primary',

overflow: 'auto',

};

const Event: FC = () => {

const { eventId, isOpen } = useAppSelector(eventModalSelector);

const dispatch = useAppDispatch();

const handleClose = () => {

dispatch(eventModalActions.closeEventModal());

};

const handleOpenTeamModal = (id: number) => {

dispatch(teamModalActions.openChallengeModal(id));

};

const { data, isLoading, isFetching } = useGetEventDataByIdQuery(eventId, { skip: !eventId });

const eventData = useMemo(() => data?.data, [data]);

// eslint-disable-next-line @typescript-eslint/no-unsafe-member-access,@typescript-eslint/no-unsafe-call

let isEventInFavourites = JSON.parse(localStorage.getItem('favouritesEvents') || '[]').some(el => el.id === eventId);

const handleAddToFavourites = () => {

if(!eventData) return;

const currentFavourites: IEvent[] = JSON.parse(localStorage.getItem('favouritesEvents') || '[]');

currentFavourites.push(eventData);

localStorage.setItem('favouritesEvents', JSON.stringify(currentFavourites));

// eslint-disable-next-line @typescript-eslint/no-unsafe-member-access,@typescript-eslint/no-unsafe-call

isEventInFavourites = JSON.parse(localStorage.getItem('favouritesEvents') || '[]').some(el => el.id === eventId);

};

const handleDeleteFromFavourites = () => {

if(!eventData) return;

const currentFavourites: IEvent[] = JSON.parse(localStorage.getItem('favouritesEvents') || '[]');

const newFavourites = currentFavourites.filter(el => el.id !== eventId);

localStorage.setItem('favouritesEvents', JSON.stringify(newFavourites));

// eslint-disable-next-line @typescript-eslint/no-unsafe-member-access,@typescript-eslint/no-unsafe-call

isEventInFavourites = JSON.parse(localStorage.getItem('favouritesEvents') || '[]').some(el => el.id === eventId);

};

return (

<Modal

open={isOpen}

onClose={handleClose}

>

<Box sx={style}>

{!isLoading && !isFetching && eventData ? (

<EventWrapper>

<EventHeaderWrapper>

<Tooltip

title={isEventInFavourites ? 'Удалить матч из избранного' : 'Добавить матч в избранное'}

placement="top-end"

style={{ cursor: 'pointer', color: isEventInFavourites ? 'gold' : 'inherit', fontSize: '32px' }}

>

<StarBorderIcon onClick={isEventInFavourites ? handleDeleteFromFavourites : handleAddToFavourites}/>

</Tooltip>

<EventsHeaderDivider />

<span className={`flags flags--category flags--md flags--${eventData.section.flag}`} />

<EventsHeaderDivider />

{eventData.league

? (<span>{getTranslations(eventData.league)}</span>)

: (<span>{getTranslations(eventData.challenge)}</span>)

}

<EventsHeaderDivider />

<span>{eventData.name}</span>

<EventsHeaderDivider />

<span>{EventsStatusesTranslations[eventData.status]}</span>

{eventData.result\_only && (

<>

<EventsHeaderDivider />

<span>только результат</span>

</>

)}

</EventHeaderWrapper>

<EventInfoWrapper>

<EventTeamInfo onClick={() => handleOpenTeamModal(eventData.home\_team.id)}>

{eventData.home\_team.has\_logo && (<img src={eventData.home\_team.logo} alt='homeTeamLogo' />)}

{getTranslations(eventData.home\_team)}

</EventTeamInfo>

<EventScoreInfo status={eventData.status}>

<>

<span>

{formatEventStartDate(eventData.start\_at).tz(DEFAULT\_TIME\_ZONE).format(DEFAULT\_EVENT\_TIME\_FORMAT)}

</span>

{eventData.status === EventsStatuses.NOT\_STARTED ? (

<CountDownTimer

eventTime={formatEventStartDate(eventData.start\_at).tz(DEFAULT\_TIME\_ZONE).valueOf()}

interval={1000}

/>

) : (

eventData.status !== EventsStatuses.POSTPONED && eventData.status !== EventsStatuses.CANCELED && (

<>

{eventData.home\_score ? eventData.home\_score.current : 0}

-

{eventData.away\_score ? eventData.away\_score.current : 0}

</>

)

)}

<span className='status'>

{!eventData.status\_more

? EventsStatusesTranslations[eventData.status]

: eventData.status\_more in EventsStatusesMoreTranslations

? EventsStatusesMoreTranslations[eventData.status\_more]

: !isNaN(parseInt(eventData.status\_more, 10)) ? eventData.status\_more + '\'' : ''

}

</span>

</>

</EventScoreInfo>

<EventTeamInfo onClick={() => handleOpenTeamModal(eventData.away\_team.id)}>

{eventData.away\_team.has\_logo && (<img src={eventData.away\_team.logo} alt='awayTeamLogo' />)}

{getTranslations(eventData.away\_team)}

</EventTeamInfo>

</EventInfoWrapper>

{!eventData.result\_only

&& eventData.status !== EventsStatuses.POSTPONED

&& eventData.status !== EventsStatuses.CANCELED

&& <MatchInfo eventData={eventData} />

}

</EventWrapper>

) : (

<Spinner width='50' />

)}

</Box>

</Modal>

);

};

export default Event;

**EventItem.tsx**

import React, { FC } from 'react';

import { IEvent } from '../../model/events.model';

import { EventItemWrapper, TeamInfo } from '../EventsList/EventsList.styled';

import { getTranslations } from '../../utils/translations';

import { formatEventStartDate } from '../../utils/common';

import { DEFAULT\_EVENT\_TIME\_FORMAT, DEFAULT\_TIME\_FORMAT, DEFAULT\_TIME\_ZONE } from '../../constants/common.constants';

import { EventsStatusesTranslations } from '../../constants/events.constants';

import { eventModalActions } from '../../store/eventModal/eventModal.slice';

import { useAppDispatch } from '../../store';

const EventItem: FC<{ event: IEvent }> = ({ event }) => {

const dispatch = useAppDispatch();

const handleOpenEventModal = (id: number) => {

dispatch(eventModalActions.openEventModal(id));

};

return (

<EventItemWrapper

key={event.id}

status={event.status}

onClick={() => handleOpenEventModal(event.id)}

>

<TeamInfo>

{event.home\_team.has\_logo && (<img src={event.home\_team.logo} alt='homeTeamLogo' />)}

{getTranslations(event.home\_team)}

</TeamInfo>

<div className='score'>

<span>

{formatEventStartDate(event.start\_at).tz(DEFAULT\_TIME\_ZONE).format(DEFAULT\_EVENT\_TIME\_FORMAT)}

</span>

{event.home\_score ? event.home\_score.current : 0} - {event.away\_score ? event.away\_score.current : 0}

<span className='status'>

{EventsStatusesTranslations[event.status]}

</span>

</div>

<TeamInfo className='awayTeam'>

{event.away\_team.has\_logo && (<img src={event.away\_team.logo} alt='awayTeamLogo' />)}

{getTranslations(event.away\_team)}

</TeamInfo>

</EventItemWrapper>

);

};

export default EventItem;

**EventsList.styled.ts**

import styled from 'styled-components';

import { FlexContainer } from '../common/common.styled';

import { EventsStatuses, EventsStatusesColors, EventsViewModes } from '../../constants/events.constants';

export const EventsHeader = styled(FlexContainer)`

display: grid;

grid-template-columns: 1fr 200px 1fr;

justify-content: center;

align-items: center;

padding: 10px;

border-bottom: 1px solid var(--lightGray);

position: sticky;

top: 0;

left: 0;

background: rgba(0, 0, 0, 0.4);

backdrop-filter: blur(50px);

height: 80px;

div:last-child {

justify-self: end;

}

`;

export const EventsListContainer = styled.div`

height: calc(100% - 80px);

overflow-y: scroll;

background-color: rgba(0, 0, 0, 0.2);

backdrop-filter: blur(15px);

display: flex;

flex-direction: column;

row-gap: 2px;

h2{

cursor: pointer;

}

`;

export const EventsModeButton = styled.div<{ mode: EventsViewModes, active?: boolean }>`

display: flex;

width: 60px;

height: 40px;

border-radius: 5px;

padding: 5px;

align-items: center;

justify-content: center;

color: var(--white);

font-weight: 700;

background-color: ${props => props.mode === EventsViewModes.ALL ? 'var(--darkBlue)' : 'var(--darkOrange)'};

cursor: pointer;

border-bottom: ${props => props.active ? '1px solid var(--white)' : 'none'};

margin-right: 10px;

transition: all 0.5s ease;

`;

export const EventItemWrapper = styled.div<{ status: EventsStatuses }>`

display: grid;

width: 100%;

grid-template-columns: 1fr 150px 1fr;

justify-content: center;

align-items: center;

padding: 0 5px;

font-size: 24px;

.score {

display: flex;

flex-direction: column;

text-align: center;

font-weight: 700;

color: var(--white);

line-height: 20px;

background-color: rgba(0,0,0,0.4);

backdrop-filter: blur(50px);

span {

font-size: 12px;

font-weight: 400;

color: var(--gray)

}

.status {

font-size: 14px;

font-weight: 600;

color: ${props => EventsStatusesColors[props.status]}

}

}

.awayTeam {

justify-self: end;

text-align: end;

}

&:not(:last-child) {

border-bottom: 1px solid var(--gray);

}

transition: all 0.3s ease;

&:hover {

background-color: var(--darkBlueTP);

cursor: pointer;

}

`;

export const TeamInfo = styled.div`

display: flex;

align-items: center;

img {

width: 30px;

height: 30px;

margin-right: 10px;

}

color: var(--white);

`;

**EventsList.tsx**

import React, { ChangeEvent, FC, useEffect, useMemo, useState } from 'react';

import { EventsHeader, EventsListContainer, EventsModeButton } from './EventsList.styled';

import { FlexContainer } from '../common/common.styled';

import {

EventsViewModes,

EventsViewModesTitle,

} from '../../constants/events.constants';

import { InputAdornment, TextField } from '@mui/material';

import { Search } from '@mui/icons-material';

import { LocalizationProvider } from '@mui/x-date-pickers/LocalizationProvider';

import { AdapterMoment } from '@mui/x-date-pickers/AdapterMoment';

import { DatePicker } from '@mui/x-date-pickers';

import { getTranslations } from '../../utils/translations';

import { groupBy } from '../../utils/common';

import { Spinner } from '../Spinner';

import moment from 'moment-timezone';

import { useDebounce } from '../../hooks/useDebounce';

import { DEFAULT\_DEBOUNCE } from '../../constants/api.constants';

import { useAppDispatch, useAppSelector } from '../../store';

import { useGetEventsBySportIdQuery } from '../../services/api';

import { setSearchValue } from '../../store/events/events.slice';

import EventItem from '../EventItem';

import { challengeModalActions } from '../../store/challengeModal/challengeModal.slice';

const EventsList:FC<{ sportId: number }> = ({ sportId }) => {

const [date, setDate] = useState(moment());

const [isInit, setIsInit] = useState(false);

const [eventsViewMode, setEventsViewMode] = useState<EventsViewModes>(EventsViewModes.ALL);

const debouncedDate = useDebounce(date, DEFAULT\_DEBOUNCE);

const { searchValue, selectedSectionId } = useAppSelector(state => state.events);

const dispatch = useAppDispatch();

useEffect(() => {

setIsInit(true);

}, []);

const { data: eventsData, isLoading: isEventsLoading, isFetching: isEventsFetching } = useGetEventsBySportIdQuery({

id: sportId,

mode: eventsViewMode,

date: moment(isInit ? debouncedDate : date).format('YYYY-MM-DD'),

}, {

// скрыто дабы не тратить кол-во запросов доступных

// ...(eventsViewMode === EventsViewModes.LIVE && { pollingInterval: DEFAULT\_LIVE\_POLLING\_INTERVAL })

});

const events = useMemo(() => eventsData && [...eventsData.data]

.filter(event => event.home\_team.name.toLowerCase().includes(searchValue.toLowerCase())

|| getTranslations(event.home\_team).toLowerCase().includes(searchValue.toLowerCase())

|| getTranslations(event.away\_team).toLowerCase().includes(searchValue.toLowerCase())

|| getTranslations(event.away\_team).toLowerCase().includes(searchValue.toLowerCase()))

.sort((a, b) => moment(a.start\_at, 'YYYY-MM-DD HH:mm:ss')

// eslint-disable-next-line @typescript-eslint/no-magic-numbers

.isBefore(moment(b.start\_at, 'YYYY-MM-DD HH:mm:ss')) ? -1 : 1)

.filter(event => event.section.id === selectedSectionId || !selectedSectionId) || [],

[eventsData, searchValue, selectedSectionId]);

const handleChangeSearchValue = (event: ChangeEvent<HTMLTextAreaElement | HTMLInputElement>) => {

dispatch(setSearchValue(event.target.value));

};

// eslint-disable-next-line @typescript-eslint/ban-ts-comment

// @ts-ignore

const groupedByChallenge = Object.values(groupBy(

events,

event => event?.league?.slug || event.challenge.slug || event.section.slug)

).sort((a, b) => b[0].section.priority - a[0].section.priority);

const handleOpenChallengeModal = (id: number) => {

dispatch(challengeModalActions.openChallengeModal(id));

};

return (

<>

<EventsHeader>

<FlexContainer>

<EventsModeButton

active={eventsViewMode === EventsViewModes.ALL}

onClick={() => setEventsViewMode(EventsViewModes.ALL)}

mode={EventsViewModes.ALL}

>

{EventsViewModesTitle[EventsViewModes.ALL]}

</EventsModeButton>

<EventsModeButton

active={eventsViewMode === EventsViewModes.LIVE}

onClick={() => setEventsViewMode(EventsViewModes.LIVE)}

mode={EventsViewModes.LIVE}

>

{EventsViewModesTitle[EventsViewModes.LIVE]}

</EventsModeButton>

</FlexContainer>

<TextField

value={searchValue}

onChange={handleChangeSearchValue}

variant='standard'

placeholder='Название команды'

InputProps={{

startAdornment: (

<InputAdornment position='start'>

<Search />

</InputAdornment>

),

}}

/>

{eventsViewMode === EventsViewModes.ALL && (

<LocalizationProvider dateAdapter={AdapterMoment} locale={'ru'}>

<DatePicker

value={date}

onChange={(date) => {

date && setDate(date);

}}

renderInput={(params) =>

<TextField

{...params}

classes={{ root: 'datePickerInput' }}

/>}

/>

</LocalizationProvider>

)}

</EventsHeader>

<EventsListContainer>

{!isEventsLoading && !isEventsFetching ? (groupedByChallenge.map(group => (

<>

<h2

onClick={() => handleOpenChallengeModal(group[0].league\_id)}

style={{ width: '100%', backgroundColor: 'var(--midGray)', color: 'var(--lightGray'}}

>

{getTranslations(group[0]?.league || group[0]?.challenge)}

</h2>

{group.map (el => <EventItem event={el} key={el.id}/>)}

</>

))) : (

<Spinner width='50' />

)}

</EventsListContainer>

</>

);

};

export default EventsList;

**H2HEvents.styled.ts**

import styled from 'styled-components';

import { EventItemWrapper, TeamInfo } from '../EventsList/EventsList.styled';

export const H2HEventsWrapper = styled.div`

width: 100%;

padding: 5px 10px;

border-radius: 15px;

font-size: 20px;

background-color: var(--midGrayTP);

color: var(--midGray);

display: flex;

flex-direction: column;

align-items: center;

h2 {

text-align: center;

font-size: inherit;

border-bottom: 1px solid var(--midGray);

margin: 5px 0;

}

`;

export const H2HEventItem = styled(EventItemWrapper)`

&:not(:last-child) {

border-bottom: 1px solid var(--midGray);

}

`;

export const H2HTeamInfo = styled(TeamInfo)`

color: var(--midGray);

`;

**H2HEvents.tsx**

import React, { FC } from 'react';

import { H2HEventItem, H2HEventsWrapper, H2HTeamInfo } from './H2HEvents.styled';

import { getTranslations } from '../../utils/translations';

import { formatEventStartDate } from '../../utils/common';

import { DEFAULT\_EVENT\_TIME\_FORMAT, DEFAULT\_TIME\_ZONE } from '../../constants/common.constants';

import { EventsStatusesTranslations, EventTypes } from '../../constants/events.constants';

import { eventModalActions } from '../../store/eventModal/eventModal.slice';

import { useAppDispatch } from '../../store';

import { IEvent } from '../../model/events.model';

const H2HEvents: FC<{ events: IEvent[], eventData: IEvent }> = ({ events, eventData }) => {

const dispatch = useAppDispatch();

const handleOpenEventModal = (id: number) => {

dispatch(eventModalActions.openEventModal(id));

};

return (

<H2HEventsWrapper>

<h2>Другие встречи {eventData.sport.slug === EventTypes.TENNIS ? 'игроков' : 'команд'}</h2>

{events.map(el => (

<H2HEventItem

key={el.id}

status={el.status}

onClick={() => handleOpenEventModal(el.id)}

>

<H2HTeamInfo>

{el.home\_team.has\_logo && (<img src={el.home\_team.logo} alt='homeTeamLogo' />)}

{getTranslations(el.home\_team)}

</H2HTeamInfo>

<div className='score'>

<span>

{formatEventStartDate(el.start\_at).tz(DEFAULT\_TIME\_ZONE).format(DEFAULT\_EVENT\_TIME\_FORMAT)}

</span>

{el.home\_score ? el.home\_score.current : 0} - {el.away\_score ? el.away\_score.current : 0}

<span className='status'>

{EventsStatusesTranslations[el.status]}

</span>

</div>

<H2HTeamInfo className='awayTeam'>

{el.away\_team.has\_logo && (<img src={el.away\_team.logo} alt='awayTeamLogo' />)}

{getTranslations(el.away\_team)}

</H2HTeamInfo>

</H2HEventItem>

))}

</H2HEventsWrapper>

);

};

export default H2HEvents;

**Header.styled.ts**

import styled from 'styled-components';

export const HeaderWrapper = styled.div`

border-bottom: 1px #999999 solid;

box-shadow: 0 1px 10px #444;

`;

export const HeaderContent = styled.div`

margin: 0 auto;

max-width: var(--containerWidth);

height: 60px;

display: flex;

`;

export const Logo = styled.img`

max-height: 100%;

`;

**Header.tsx**

import React from 'react';

import {HeaderContent, HeaderWrapper, Logo} from './Header.styled';

import logo from '../../assets/header/logo.svg';

import NavBar from '../NavBar';

import SportsSoccerIcon from '@mui/icons-material/SportsSoccer';

import StarBorderIcon from '@mui/icons-material/StarBorder';

import NewspaperIcon from '@mui/icons-material/Newspaper';

const Header = (): JSX.Element => {

return (

<HeaderWrapper>

<HeaderContent>

<Logo src={logo} alt="logo"/>

<NavBar linkObjects={[

{

link: '/events',

label: 'События',

icon: <SportsSoccerIcon/>,

},

{

link: '/favourites',

label: 'Избранное',

icon: <StarBorderIcon/>,

},

{

link: '/news',

label: 'Новости',

icon: <NewspaperIcon/>,

},

]}/>

</HeaderContent>

</HeaderWrapper>

);

};

export default Header;

**MatchIncidents.tsx**

import React, { FC, useMemo } from 'react';

import { MatchIncident, MatchIncidentsWrapper } from './MatchIncidents.styled';

import { IEvent } from '../../model/events.model';

import { useGetEventIncidentsByIdQuery } from '../../services/api';

import { Spinner } from '../Spinner';

import { IIncidentsModel } from '../../model/incidents.model';

import { getTranslations } from '../../utils/translations';

const MatchIncidents: FC<{ eventData: IEvent }> = ({ eventData }) => {

const { data, isLoading } = useGetEventIncidentsByIdQuery(eventData.id);

const incidentsData = useMemo(() => data && [...data.data].sort((a, b) => a.order - b.order) || [], [data]);

const getMatchIncident = (incident: IIncidentsModel) => {

switch (incident.incident\_type) {

case 'period': {

const type = incident.text.split(' ')[0];

const text = type === 'HT' ? 'Перерыв' : 'Конец матча';

// eslint-disable-next-line react/no-unescaped-entities

return <span>{incident.time}', {text}, {incident.text.split(' ').slice(1).join('')}</span>;

}

case 'injuryTime': {

const text = 'Дополнительное время';

// eslint-disable-next-line react/no-unescaped-entities

return <span>{incident.time}', {text}, +{incident.length}'</span>;

}

case 'goal': {

const text = 'Гол!';

// eslint-disable-next-line react/no-unescaped-entities

return <div style={{ display: 'flex', columnGap: '5px' }}>

<span>{incident.time}',</span>

<span>{text},</span>

<span>{incident.player && getTranslations(incident.player)},</span>

<span style={{ color: 'var(--gray)' }}>{incident.player\_two\_in && ` (${getTranslations(incident.player\_two\_in)})`},</span>

<span> {incident.home\_score} - {incident.away\_score}</span>

</div>;

}

case 'card': {

const text = incident.card\_type === 'Yellow' ? 'Желтая карточка!' : 'Красная карточка!';

// eslint-disable-next-line react/no-unescaped-entities

return <div style={{ display: 'flex', columnGap: '5px' }}>

<span>{incident.time}',</span>

<span>{text},</span>

<span>{incident.player && getTranslations(incident.player)}</span>

</div>;

}

case 'substitution': {

const text = 'Замена!';

// eslint-disable-next-line react/no-unescaped-entities

return <div style={{ display: 'flex', columnGap: '5px' }}>

<span>{incident.time}',</span>

<span>{text},</span>

<span>{incident.player && getTranslations(incident.player)},</span>

<span style={{ color: 'var(--gray)' }}>{incident.player\_two\_in && ` (${getTranslations(incident.player\_two\_in)})`}</span>

</div>;

}

default: {

return <span>{incident.text || incident.reason || incident.incident\_type}</span>;

}

}

};

if (isLoading) return <Spinner width='50px' />;

return (

<MatchIncidentsWrapper>

{incidentsData.map((el, index) => (

<MatchIncident type={el.incident\_type} playerTeam={el.player\_team} key={index}>

{getMatchIncident(el)}

</MatchIncident>

))}

</MatchIncidentsWrapper>

);

};

export default MatchIncidents;

**MatchIncidents.styled.ts**

import styled from 'styled-components';

import { H2HEventsWrapper } from '../H2HEvents/H2HEvents.styled';

export const MatchIncidentsWrapper = styled(H2HEventsWrapper)`

display: flex;

flex-direction: column;

`;

export const MatchIncident = styled.div<{ type: string, playerTeam: number }>`

align-self: ${props => props.type === 'period' || props.type === 'injuryTime'

? 'center'

: props.playerTeam === 1

? 'flex-start'

: 'flex-end'

};

span{

font-weight: 700;

}

`;

**MatchInfo.styled.ts**

import styled from 'styled-components';

import { MatchInfoViewModes } from '../../constants/events.constants';

export const MatchInfoWrapper = styled.div`

width: 100%;

padding: 5px 10px;

border-radius: 15px;

font-size: 20px;

background-color: var(--lightGray);

color: var(--midGray);

h2{

text-align: center;

font-size: inherit;

border-bottom: 1px solid var(--midGray);

}

`;

export const MatchInfoViewModesWrapper = styled.div`

display: flex;

align-items: flex-end;

padding: 5px 0;

`;

export const MatchInfoViewModeButton = styled.div<{ mode?: MatchInfoViewModes, active?: boolean }>`

display: flex;

min-height: 40px;

padding: 0 5px;

align-items: center;

justify-content: center;

color: ${props => !props.active ? 'var(--lightGray)' : 'var(--darkGray)'};

font-weight: 700;

background-color: ${props => !props.active ? 'var(--gray)' : 'var(--activeLinkBackground)'};

cursor: pointer;

margin-right: 10px;

transition: all 0.5s ease;

border-radius: 4px;

`;

export const MatchInfoContent = styled.div`

width: 100%;

color: var(--darkBlue);

`;

**MatchInfo.tsx**

import React, { FC, useMemo, useState } from 'react';

import {

MatchInfoContent,

MatchInfoViewModeButton,

MatchInfoViewModesWrapper,

MatchInfoWrapper,

} from './MatchInfo.styled';

import { MatchInfoViewModes, MatchInfoViewModesTitle } from '../../constants/events.constants';

import { IEvent } from '../../model/events.model';

import { useGetEventLineupsByIdQuery, useGetH2HEventsQuery } from '../../services/api';

import MatchInfoLineup from '../MatchInfoLineup';

import H2HEvents from '../H2HEvents';

import MatchStatistics from '../MatchStatistics';

import MatchIncidents from '../MatchIncidents';

import StadiumInfo from '../StadiumInfo';

const MatchInfo: FC<{ eventData: IEvent }> = ({ eventData }) => {

const [selectedMatchInfoViewMode, setSelectedMatchInfoViewMode] = useState<MatchInfoViewModes | null>(null);

const { data } = useGetEventLineupsByIdQuery(eventData.id);

const { homeTeamLineup, awayTeamLineup } = useMemo(() =>{

const homeTeamLineup = data?.data.some(el => el.is\_confirmed)

? data?.data.find(lineup => lineup.team\_id === eventData.home\_team\_id && lineup.is\_confirmed)

: data?.data.find(lineup => lineup.team\_id === eventData.home\_team\_id);

const awayTeamLineup = data?.data.some(el => el.is\_confirmed)

? data?.data.find(lineup => lineup.team\_id === eventData.away\_team\_id && lineup.is\_confirmed)

: data?.data.find(lineup => lineup.team\_id === eventData.away\_team\_id);

return { homeTeamLineup, awayTeamLineup };

}, [data, eventData]);

const { data: h2hdata } = useGetH2HEventsQuery(

{ homeTeam: eventData.home\_team\_id, awayTeam: eventData.away\_team\_id },

{ skip: !eventData.home\_team\_id || !eventData.away\_team\_id},

);

const h2hevents = useMemo(() => h2hdata && [...h2hdata.data].filter(event => event.id !== eventData.id) || [], [h2hdata, eventData]);

return (

<MatchInfoWrapper>

<h2>Информация о матче</h2>

<MatchInfoViewModesWrapper>

<MatchInfoViewModeButton

active={selectedMatchInfoViewMode === MatchInfoViewModes.STATISTICS}

mode={MatchInfoViewModes.STATISTICS}

onClick={() => setSelectedMatchInfoViewMode(MatchInfoViewModes.STATISTICS)}

>

{MatchInfoViewModesTitle[MatchInfoViewModes.STATISTICS]}

</MatchInfoViewModeButton>

{(homeTeamLineup || awayTeamLineup) && (

<MatchInfoViewModeButton

active={selectedMatchInfoViewMode === MatchInfoViewModes.LINEUPS}

mode={MatchInfoViewModes.LINEUPS}

onClick={() => setSelectedMatchInfoViewMode(MatchInfoViewModes.LINEUPS)}

>

{MatchInfoViewModesTitle[MatchInfoViewModes.LINEUPS]}

</MatchInfoViewModeButton>

)}

<MatchInfoViewModeButton

active={selectedMatchInfoViewMode === MatchInfoViewModes.INCIDENTS}

mode={MatchInfoViewModes.INCIDENTS}

onClick={() => setSelectedMatchInfoViewMode(MatchInfoViewModes.INCIDENTS)}

>

{MatchInfoViewModesTitle[MatchInfoViewModes.INCIDENTS]}

</MatchInfoViewModeButton>

{!!h2hevents.length && (

<MatchInfoViewModeButton

active={selectedMatchInfoViewMode === MatchInfoViewModes.H2H}

mode={MatchInfoViewModes.H2H}

onClick={() => setSelectedMatchInfoViewMode(MatchInfoViewModes.H2H)}

>

{MatchInfoViewModesTitle[MatchInfoViewModes.H2H]}

</MatchInfoViewModeButton>

)}

{!!eventData?.venue?.stadium && (

<MatchInfoViewModeButton

active={selectedMatchInfoViewMode === MatchInfoViewModes.STADIUM}

mode={MatchInfoViewModes.STADIUM}

onClick={() => setSelectedMatchInfoViewMode(MatchInfoViewModes.STADIUM)}

>

{MatchInfoViewModesTitle[MatchInfoViewModes.STADIUM]}

</MatchInfoViewModeButton>

)}

</MatchInfoViewModesWrapper>

<MatchInfoContent>

{selectedMatchInfoViewMode === MatchInfoViewModes.LINEUPS && (homeTeamLineup && awayTeamLineup) && (

<MatchInfoLineup homeTeamLineup={homeTeamLineup} awayTeamLineup={awayTeamLineup} eventData={eventData}/>

)}

{selectedMatchInfoViewMode === MatchInfoViewModes.H2H && !!h2hevents.length && (

<H2HEvents eventData={eventData} events={h2hevents}/>

)}

{selectedMatchInfoViewMode === MatchInfoViewModes.STATISTICS && (

<MatchStatistics eventId={eventData.id}/>

)}

{selectedMatchInfoViewMode === MatchInfoViewModes.INCIDENTS && (

<MatchIncidents eventData={eventData}/>

)}

{selectedMatchInfoViewMode === MatchInfoViewModes.STADIUM && (

<StadiumInfo venue={eventData.venue}/>

)}

</MatchInfoContent>

</MatchInfoWrapper>

);

};

export default MatchInfo;

**MatchInfoLineup.styled.ts**

import styled from 'styled-components';

export const TeamLineupInfo = styled.div`

display: flex;

flex-direction: column;

row-gap: 10px;

align-items: center;

font-weight: 500;

color: #142a50;

`;

export const TeamLineupConfirmationStatus = styled.h2<{ isConfirmed: boolean }>`

color: ${props => props.isConfirmed ? 'var(--green)' : 'var(--lightRed)'};

border-bottom: none;

`;

export const MatchInfoLineupWrapper = styled.div`

display: grid;

grid-template-columns: 1fr auto 1fr;

justify-content: center;

column-gap: 15px;

`;

**MatchInfoLineup.tsx**

import React, { FC, useMemo } from 'react';

import { MatchInfoLineupWrapper, TeamLineupConfirmationStatus, TeamLineupInfo } from './MatchInfoLineup.styled';

import { Divider, FlexContainer } from '../common/common.styled';

import GroupsIcon from '@mui/icons-material/Groups';

import ProgressBar from '../common/ProgressBar';

import TeamLineupTable from '../TeamLineUpTable';

import { lineupsPositionsShortNaming } from '../../constants/lineups.constants';

import { ILineupModel } from '../../model/lineup.model';

import { IEvent } from '../../model/events.model';

const MatchInfoLineup: FC<{ homeTeamLineup: ILineupModel, awayTeamLineup: ILineupModel, eventData: IEvent }>

= ({ homeTeamLineup, awayTeamLineup, eventData }) => {

const homeTeamLineupRows = useMemo(() => {

return homeTeamLineup?.lineup\_players.filter(player => !player.substitute).map(lineupPlayer => ({

id: lineupPlayer.player.id,

shirt\_number: lineupPlayer.player.shirt\_number,

name: lineupPlayer.player.name,

// eslint-disable-next-line @typescript-eslint/no-unsafe-member-access,@typescript-eslint/no-unsafe-assignment

position: lineupsPositionsShortNaming[eventData.sport.slug][lineupPlayer.player.position],

age: lineupPlayer.player.age,

rating: lineupPlayer.player.rating,

})) || [];

}, [homeTeamLineup, eventData.sport.slug]);

const awayTeamLineupRows = useMemo(() => {

return awayTeamLineup?.lineup\_players.filter(player => !player.substitute).map(lineupPlayer => ({

id: lineupPlayer.player.id,

shirt\_number: lineupPlayer.player.shirt\_number,

name: lineupPlayer.player.name,

// eslint-disable-next-line @typescript-eslint/no-unsafe-member-access,@typescript-eslint/no-unsafe-assignment

position: lineupsPositionsShortNaming[eventData.sport.slug][lineupPlayer.player.position],

age: lineupPlayer.player.age,

rating: lineupPlayer.player.rating,

})) || [];

}, [awayTeamLineup, eventData.sport.slug]);

const homeTeamLineupSubstitutionRows = useMemo(() => {

return homeTeamLineup?.lineup\_players.filter(player => player.substitute).map(lineupPlayer => ({

id: lineupPlayer.player.id,

shirt\_number: lineupPlayer.player.shirt\_number,

name: lineupPlayer.player.name,

// eslint-disable-next-line @typescript-eslint/no-unsafe-member-access,@typescript-eslint/no-unsafe-assignment

position: lineupsPositionsShortNaming[eventData.sport.slug][lineupPlayer.player.position],

age: lineupPlayer.player.age,

rating: lineupPlayer.player.rating,

})) || [];

}, [homeTeamLineup, eventData.sport.slug]);

const awayTeamLineupSubstitutionRows = useMemo(() => {

return awayTeamLineup?.lineup\_players.filter(player => player.substitute).map(lineupPlayer => ({

id: lineupPlayer.player.id,

shirt\_number: lineupPlayer.player.shirt\_number,

name: lineupPlayer.player.name,

// eslint-disable-next-line @typescript-eslint/no-unsafe-member-access,@typescript-eslint/no-unsafe-assignment

position: lineupsPositionsShortNaming[eventData.sport.slug][lineupPlayer.player.position],

age: lineupPlayer.player.age,

rating: lineupPlayer.player.rating,

})) || [];

}, [awayTeamLineup, eventData.sport.slug]);

return (

<MatchInfoLineupWrapper>

<TeamLineupInfo>

<FlexContainer ai='center' cg='15px'>

<FlexContainer ai='center' rg='5px' fd='column'>

{homeTeamLineup?.formation || 'Н/Д'}

<FlexContainer ai='center'><GroupsIcon /> Схема состава</FlexContainer>

</FlexContainer>

<FlexContainer ai='center' rg='5px' fd='column'>

{homeTeamLineup?.avg\_rating || 'Н/Д'}

<FlexContainer ai='center'>Средний рейтинг</FlexContainer>

</FlexContainer>

</FlexContainer>

<FlexContainer ai='center' cg='10px' style={{ height: '40px', width: '100%' }}>

{homeTeamLineup?.attacking && (

<FlexContainer ai='center' rg='5px' fd='column'>

<ProgressBar percent={homeTeamLineup.attacking} />

<FlexContainer ai='center'>Атака</FlexContainer>

</FlexContainer>

)}

{homeTeamLineup?.technical && (

<FlexContainer ai='center' rg='5px' fd='column'>

<ProgressBar percent={homeTeamLineup.technical} />

<FlexContainer ai='center'>Техника</FlexContainer>

</FlexContainer>

)}

{homeTeamLineup?.defending && (

<FlexContainer ai='center' rg='5px' fd='column'>

<ProgressBar percent={homeTeamLineup.defending} />

<FlexContainer ai='center'>Защита</FlexContainer>

</FlexContainer>

)}

{homeTeamLineup?.tactical && (

<FlexContainer ai='center' rg='5px' fd='column'>

<ProgressBar percent={homeTeamLineup.tactical} />

<FlexContainer ai='center'>Тактика</FlexContainer>

</FlexContainer>

)}

{homeTeamLineup?.creativity && (

<FlexContainer ai='center' rg='5px' fd='column'>

<ProgressBar percent={homeTeamLineup.creativity} />

<FlexContainer ai='center'>Креативность</FlexContainer>

</FlexContainer>

)}

</FlexContainer>

<TeamLineupConfirmationStatus isConfirmed={!!homeTeamLineup?.is\_confirmed}>

Стартовый состав {homeTeamLineup?.is\_confirmed ? '' : 'не '}подтвержден

</TeamLineupConfirmationStatus>

<TeamLineupTable rows={homeTeamLineupRows} />

<h2>Замены:</h2>

<TeamLineupTable rows={homeTeamLineupSubstitutionRows} />

</TeamLineupInfo>

<Divider />

<TeamLineupInfo>

<FlexContainer ai='center' cg='10px'>

<FlexContainer ai='center' rg='5px' fd='column' jc='center'>

{awayTeamLineup?.formation || 'Н/Д'}

<FlexContainer ai='center'><GroupsIcon /> Схема состава</FlexContainer>

</FlexContainer>

<FlexContainer ai='center' rg='5px' fd='column'>

{awayTeamLineup?.avg\_rating || 'Н/Д'}

<FlexContainer ai='center'>Средний рейтинг</FlexContainer>

</FlexContainer>

</FlexContainer>

<FlexContainer ai='center' cg='10px' style={{ height: '40px', width: '100%' }}>

{awayTeamLineup?.attacking && (

<FlexContainer ai='center' rg='5px' fd='column'>

<ProgressBar percent={awayTeamLineup.attacking} />

<FlexContainer ai='center'>Атака</FlexContainer>

</FlexContainer>

)}

{awayTeamLineup?.technical && (

<FlexContainer ai='center' rg='5px' fd='column'>

<ProgressBar percent={awayTeamLineup.technical} />

<FlexContainer ai='center'>Техника</FlexContainer>

</FlexContainer>

)}

{awayTeamLineup?.defending && (

<FlexContainer ai='center' rg='5px' fd='column'>

<ProgressBar percent={awayTeamLineup.defending} />

<FlexContainer ai='center'>Защита</FlexContainer>

</FlexContainer>

)}

{awayTeamLineup?.tactical && (

<FlexContainer ai='center' rg='5px' fd='column'>

<ProgressBar percent={awayTeamLineup.tactical} />

<FlexContainer ai='center'>Тактика</FlexContainer>

</FlexContainer>

)}

{awayTeamLineup?.creativity && (

<FlexContainer ai='center' rg='5px' fd='column'>

<ProgressBar percent={awayTeamLineup.creativity} />

<FlexContainer ai='center'>Креативность</FlexContainer>

</FlexContainer>

)}

</FlexContainer>

<TeamLineupConfirmationStatus isConfirmed={!!awayTeamLineup?.is\_confirmed}>

Стартовый состав {homeTeamLineup?.is\_confirmed ? '' : 'не '}подтвержден

</TeamLineupConfirmationStatus>

<TeamLineupTable rows={awayTeamLineupRows} />

<h2>Замены:</h2>

<TeamLineupTable rows={awayTeamLineupSubstitutionRows} />

</TeamLineupInfo>

</MatchInfoLineupWrapper>

);

};

export default MatchInfoLineup;

**MatchStatistics.styled.ts**

import styled from 'styled-components';

import { H2HEventsWrapper } from '../H2HEvents/H2HEvents.styled';

import { MatchInfoViewModeButton, MatchInfoViewModesWrapper } from '../MatchInfo/MatchInfo.styled';

export const MatchStatisticsWrapper = styled(H2HEventsWrapper)`

`;

export const MatchStatisticsViewModeWrapper = styled(MatchInfoViewModesWrapper)``;

export const MatchStatisticsViewModeButton = styled(MatchInfoViewModeButton)`

color: ${props => !props.active ? 'var(--lightGray)' : 'var(--white)'};

background-color: ${props => !props.active ? 'var(--gray)' : 'var(--darkBlue)'};

`;

export const MatchStatisticsInfoItem = styled.div`

width: 100%;

height: 32px;

display: grid;

grid-template-columns: 1fr 450px 1fr;

justify-items: center;

align-items: center;

color: var(--midGray);

font-weight: 400;

.name {

font-weight: 700;

}

&:not(:last-child){

border-bottom: 1px solid var(--midGray);

}

`;

**MatchStatistics.tsx**

import React, { FC, useEffect, useMemo, useState } from 'react';

import {

MatchStatisticsInfoItem,

MatchStatisticsViewModeButton,

MatchStatisticsViewModeWrapper,

MatchStatisticsWrapper,

} from './MatchStatistics.styled';

import { useGetEventStatisticsByIdQuery } from '../../services/api';

import { EventPeriodsTranslations, EventStatisticsNameTranslations } from '../../constants/events.constants';

import { groupBy } from '../../utils/common';

const MatchStatistics: FC<{ eventId: number }> = ({ eventId }) => {

const { data } = useGetEventStatisticsByIdQuery(eventId);

const statistics = useMemo(() => data && [...data.data] || [], [data]);

const statisticsViewModes = useMemo(() => [...new Set(statistics.map(el => el.period))], [statistics]);

const [selectedMatchInfoViewMode, setSelectedMatchInfoViewMode] = useState<string | null>(null);

useEffect(() => {

setSelectedMatchInfoViewMode(statisticsViewModes[0]);

}, [statisticsViewModes]);

const statisticsToView = useMemo(() => {

const filteredStats = statistics.filter(stat => stat.period === selectedMatchInfoViewMode);

return Object.values(groupBy(filteredStats, el => el.group)).flat();

},

[statisticsViewModes, selectedMatchInfoViewMode, statistics]

);

return (

<MatchStatisticsWrapper>

<MatchStatisticsViewModeWrapper>

{statisticsViewModes.map(el => (

<MatchStatisticsViewModeButton

active={selectedMatchInfoViewMode === el}

onClick={() => setSelectedMatchInfoViewMode(el)}

key={el}

>

{EventPeriodsTranslations[el]}

</MatchStatisticsViewModeButton>

))}

</MatchStatisticsViewModeWrapper>

{statisticsToView.map(el => (

<MatchStatisticsInfoItem key={el.name}>

<div style={{ backgroundColor: el.compare\_code === 1 ? 'var(--blue)' : 'none', padding: '0 5px', borderRadius: '4px' }}>

{el.home}

</div>

<div className='name'>{EventStatisticsNameTranslations[el.name]}</div>

<div style={{ backgroundColor: el.compare\_code === 2 ? 'var(--blue)' : 'none', padding: '0 5px', borderRadius: '4px' }}>

{el.away}

</div>

</MatchStatisticsInfoItem>

))}

</MatchStatisticsWrapper>

);

};

export default MatchStatistics;

**NavBar.styled.ts**

import styled from 'styled-components';

export const NavBarWrapper = styled.div`

max-height: 100%;

display: flex;

align-items: center;

justify-content: center;

.active, .active:hover {

background: var(--activeLinkBackground);

span {

color: var(--darkGray)

}

}

`;

**NavBar.tsx**

import { FC } from 'react';

import { INavBarProps } from './NavBar.types';

import { NavBarWrapper } from './NavBar.styled';

import NavLink from '../NavLink';

const NavBar: FC<INavBarProps> = (props) => {

const { linkObjects } = props;

const links = linkObjects.map((el, index) => {

return (

<NavLink key={index} link={el.link} label={el.label} icon={el.icon} />

);

});

return (

<NavBarWrapper>

{links}

</NavBarWrapper>

);

};

export default NavBar;

**SectionsList.styled.ts**

import styled from 'styled-components';

import { FlexContainer } from '../common/common.styled';

export const SectionsListContainer = styled.div`

background-color: rgba(0, 0, 0, 0.2);

backdrop-filter: blur(15px);

display: flex;

flex-direction: column;

padding: 5px;

row-gap: 5px;

overflow-y: scroll;

height: 100% ;

`;

export const SectionItem = styled(FlexContainer)<{isActive: boolean}>`

column-gap: 5px;

align-items: center;

justify-content: flex-start;

font-size: 16px;

font-weight: 700;

color: var(--white);

background: ${props => props.isActive ? 'rgba(255,255,255,0.4)' : 'none'};

transition: all 0.3s ease;

&:hover {

background: ${props => !props.isActive ? 'rgba(0,0,0,0.4)' : 'rgba(255,255,255,0.4)'};

cursor: pointer;

}

img {

height: 32px;

width: 32px;

}

`;

**SectionsList.tsx**

import React, { FC, useMemo } from 'react';

import { SectionItem, SectionsListContainer } from './SectionsList.styled';

import { Tooltip } from '@mui/material';

import { getTranslations } from '../../utils/translations';

import { Spinner } from '../Spinner';

import { setSelectedSection } from '../../store/events/events.slice';

import { useGetSectionsBySportIdQuery } from '../../services/api';

import { useAppDispatch, useAppSelector } from '../../store';

const SectionsList: FC<{ sportId: number }> = ({ sportId }) => {

const { data, isLoading: isSectionsLoading, isFetching: isSectionsFetching } = useGetSectionsBySportIdQuery(sportId);

const { selectedSectionId } = useAppSelector(state => state.events);

const dispatch = useAppDispatch();

const sections = useMemo(() =>

data && [...data.data]

.sort((a, b) => a.name\_translations.ru && b.name\_translations.ru

? a.name\_translations.ru.localeCompare(b?.name\_translations?.ru) : a.name.localeCompare(b.name))

|| [], [data]);

const handleSelectSection = (id: number) => {

dispatch(setSelectedSection(id));

};

return (

<SectionsListContainer>

{!isSectionsLoading && !isSectionsFetching ? sections.map(section => (

<Tooltip

key={section.slug}

title={'Выбрать категорию: ' + getTranslations(section)}

placement="top-end"

>

<SectionItem

onClick={() => handleSelectSection(section.id)}

isActive={section.id === selectedSectionId}

>

<span className={`flags flags--category flags--md flags--${section.flag}`} />

{getTranslations(section)}

</SectionItem>

</Tooltip>

)) : <Spinner width='50' />}

</SectionsListContainer>

);

};

export default SectionsList;

**Spinner.ts**

import styled from 'styled-components';

const borderPath = 0.1;

export const Spinner = styled.div<{ width: string }>`

height: ${(props): string => props.width + 'px'};

width: ${(props): string => props.width + 'px'};

margin: 0 auto;

border-radius: 50%;

border: var(--midGray) ${(props): string => `${+props.width \* borderPath}` + 'px'} solid;

border-top-color: var(--activeLinkBackground);

border-bottom-color: #eee;

animation: spin 2s infinite ease-in-out;

@keyframes spin {

50% {

transform: rotate(360deg);

border-bottom-color: var(--activeLinkBackground);

border-top-color: #eee;

}

100%{

transform: rotate(0deg);

border-top-color: var(--activeLinkBackground);

border-bottom-color: #eee;

}

}

`;

**SportTypes.styled.ts**

import styled from 'styled-components';

export const SportTypesContainer = styled.div`

height: 60px;

display: flex;

font-size: 1.5em;

justify-content: center;

border-bottom: 1px #eeeeee10 solid;

a{

display: flex;

height:100%;

padding: 0 40px;

align-items: center;

justify-content: center;

text-decoration: none;

color: var(--navLinkColor);

transition: all 0.5s ease;

:hover{

background: #eeeeee20;

}

}

.active{

background: #eeeeee10;

}

`;

**SportTypes.tsx**

import React, { FC, useEffect } from 'react';

import { NavLink } from 'react-router-dom';

import {SportTypesContainer} from './SportTypes.styled';

import {useSportTypesFetch} from '../../hooks/useSportTypesFetch';

import { useAppDispatch, useAppSelector } from '../../store';

import {Spinner} from '../Spinner';

import {Outlet} from 'react-router-dom';

import { setSelectedSection } from '../../store/events/events.slice';

import { useLocation } from 'react-router';

const SportTypes: FC = () => {

const {loading, error} = useSportTypesFetch();

const state = useAppSelector(state => state.sportTypes);

const dispatch = useAppDispatch();

const location = useLocation();

useEffect(() => {

dispatch(setSelectedSection(null));

}, [location, dispatch]);

return (

<>

<SportTypesContainer>

{!loading && !error ?

(

<>

{state.sportTypes.map(type => (

<NavLink key={type.slug} to={'/events/' + type.slug}>{type?.name\_translations?.ru}</NavLink>))}

</>

) :

<Spinner width="50"/>

}

</SportTypesContainer>

<Outlet/>

</>

);

};

export default SportTypes;

**StadiumInfo.styled.ts**

import { MatchIncidentsWrapper } from '../MatchIncidents/MatchIncidents.styled';

import styled from 'styled-components';

export const StadiumInfoWrapper = styled(MatchIncidentsWrapper)`

align-items: center;

row-gap: 5px;

`;

**StadiumInfo.tsx**

import React, { FC } from 'react';

import { IVenueModel } from '../../model/venue.model';

import { StadiumInfoWrapper } from './StadiumInfo.styled';

import { FlexContainer } from '../common/common.styled';

import StadiumIcon from '@mui/icons-material/Stadium';

const StadiumInfo: FC<{ venue: IVenueModel }> = ({ venue }) => {

return (

<StadiumInfoWrapper>

<FlexContainer jc="space-evenly" style={{ width: '70%' }}>

<FlexContainer fd="column" rg="5px" ai="center">

<span>{venue?.country\_name || 'H/Д'}</span>

<span style={{ fontWeight: 700 }}>Страна проведения</span>

</FlexContainer>

<FlexContainer fd="column" rg="5px" ai="center">

<span>{venue?.city.ru ? venue?.city.ru : venue?.city.en || 'H/Д'}</span>

<span style={{ fontWeight: 700 }}>Город проведения</span>

</FlexContainer>

<FlexContainer fd="column" rg="5px" ai="center">

<span>{venue?.stadium.ru ? venue?.stadium.ru : venue?.stadium.en || 'H/Д'}</span>

<span style={{ fontWeight: 700 }}><StadiumIcon /> Стадион</span>

</FlexContainer>

<FlexContainer fd="column" rg="5px" ai="center">

<span>{venue?.stadium\_capacity || 'H/Д'}</span>

<span style={{ fontWeight: 700 }}>Вместимость</span>

</FlexContainer>

</FlexContainer>

</StadiumInfoWrapper>

);

};

export default StadiumInfo;

**TeamLineUpTable.styled.ts**

import styled from 'styled-components';

export const TeamLineUpTableWrapper = styled.div`

width: 100%;

background: var(--midGrayTP);

border-radius: 5px;

padding: 5px;

color: var(--darkGray);

`;

export const TeamLineUpGridTemplate = styled.div`

width: 100%;

display: grid;

grid-template-columns: 40px 250px 100px 100px 100px;

justify-items: center;

span:not(:last-child){

display: block;

width: 100%;

text-align: center;

border-right: 1px solid var(--darkGray);

}

:nth-child(odd) {

background-color: var(--lightGray);

}

`;

export const TeamLineUpHeader = styled(TeamLineUpGridTemplate)`

background-color: var(--gray);

`;

**TeamLineUpTable.tsx**

import React, { FC } from 'react';

import { ILineupRow } from '../../model/lineup.model';

import { TeamLineUpGridTemplate, TeamLineUpHeader, TeamLineUpTableWrapper } from './TeamLineupTable.styled';

const TeamLineupTable: FC<{ rows: ILineupRow[] }> = ({ rows }) => {

return (

<TeamLineUpTableWrapper>

<TeamLineUpHeader>

<span>

#

</span>

<span>

Игрок

</span>

<span>

Позиция

</span>

<span>

Возраст

</span>

<span>

Рейтинг

</span>

</TeamLineUpHeader>

{rows.map(el => (

<TeamLineUpGridTemplate key={el?.id}>

<span>

{el?.shirt\_number}

</span>

<span>

{el?.name}

</span>

<span>

{el?.position}

</span>

<span>

{el?.age}

</span>

<span>

{el?.rating}

</span>

</TeamLineUpGridTemplate>

))}

</TeamLineUpTableWrapper>

);

};

export default TeamLineupTable;

**TeamModal.styled.ts**

import styled from 'styled-components';

import { EventHeaderWrapper, EventsHeaderDivider, EventWrapper } from '../Event/Event.styled';

export const TeamWrapper = styled(EventWrapper)``;

export const TeamHeaderWrapper = styled(EventHeaderWrapper)`

font-size: 32px;

font-weight: 700;

img{

height: 48px;

width: 48px;

}

height: 48px;

`;

export const TeamGrid = styled.div`

margin-top: 10px;

display: grid;

grid-template-columns: 1fr 1fr;

grid-gap: 20px;

align-items: start;

justify-items: end;

background: var(--lightGray);

padding: 5px;

`;

export const TeamDataInfo = styled.div`

padding: 5px;

display: flex;

flex-direction: column;

align-items: center;

border-radius: 4px;

row-gap: 10px;

border: 1px solid #333;

color: #333;

span{

font-size: 20px;

}

span:first-child {

font-weight: 700;

}

`;

**TeamModal.styled.tsx**

import React, { useMemo, useState } from 'react';

import { TeamDataInfo, TeamGrid, TeamHeaderWrapper, TeamWrapper } from './TeamModal.styled';

import { useAppDispatch, useAppSelector } from '../../store';

import { useGetTeamEventsByIdQuery, useGetTeamDataByIdQuery, useGetTeamLineupsByIdQuery } from '../../services/api';

import { teamModalSelector, teamModalActions } from '../../store/teamModal/teamModal.slice';

import { getTranslations } from '../../utils/translations';

import { Box, Modal } from '@mui/material';

import { Spinner } from '../Spinner';

import EventItem from '../EventItem';

import { FlexContainer } from '../common/common.styled';

import GroupsIcon from '@mui/icons-material/Groups';

import ProgressBar from '../common/ProgressBar';

import { TeamLineupConfirmationStatus, TeamLineupInfo } from '../MatchInfoLineup/MatchInfoLineup.styled';

import TeamLineupTable from '../TeamLineUpTable';

import { lineupsPositionsShortNaming } from '../../constants/lineups.constants';

const style = {

position: 'absolute',

top: '50%',

left: '50%',

transform: 'translate(-50%, -50%)',

width: 1340,

height: 700,

bgcolor: 'background.paper',

border: '2px solid #000',

boxShadow: 24,

p: 4,

color: 'text.primary',

zIndex: 15555,

overflow: 'auto',

};

const TeamModal = () => {

const { teamId, isOpen } = useAppSelector(teamModalSelector);

const dispatch = useAppDispatch();

const handleClose = () => {

dispatch(teamModalActions.closeChallengeModal());

};

const { data } = useGetTeamDataByIdQuery(teamId, { skip: !teamId });

const { data: eventsResponse } = useGetTeamEventsByIdQuery(teamId, { skip: !teamId });

const eventsData = useMemo(() => eventsResponse?.data || [], [eventsResponse]);

const { data: lineupsResponse } = useGetTeamLineupsByIdQuery(teamId, { skip: !teamId });

const homeTeamLineup = useMemo(() => lineupsResponse?.data.find(lineup => lineup.is\_confirmed), [lineupsResponse]);

const teamData = useMemo(() => data?.data, [data]);

const homeTeamLineupRows = useMemo(() => {

return homeTeamLineup?.lineup\_players.map(lineupPlayer => ({

id: lineupPlayer.player.id,

shirt\_number: lineupPlayer.player.shirt\_number,

name: lineupPlayer.player.name,

// eslint-disable-next-line @typescript-eslint/no-unsafe-member-access,@typescript-eslint/no-unsafe-assignment

position: lineupsPositionsShortNaming[eventsData[0]?.sport?.slug]

&& lineupsPositionsShortNaming[eventsData[0]?.sport?.slug][lineupPlayer.player.position] || 'Н/Д',

age: lineupPlayer.player.age,

rating: lineupPlayer.player.rating,

})) || [];

}, [homeTeamLineup, eventsData[0]?.sport.slug]);

if (!teamData) return <Spinner width='50px' />;

return (

<Modal

open={isOpen}

onClose={handleClose}

>

<Box sx={style}>

<TeamWrapper>

<TeamHeaderWrapper>

<img src={teamData.logo} alt='logo' />

<span>{getTranslations(teamData)}</span>

</TeamHeaderWrapper>

<TeamGrid>

<div>

<h2 style={{ background: 'var(--lightGray)', textAlign: 'center', color: '#353535'}}>Данные команды:</h2>

<TeamDataInfo>

<h3>Тренер</h3>

<img src={teamData?.manager?.photo} alt='photo' />

<h4>{getTranslations(teamData.manager)}</h4>

<FlexContainer>

<FlexContainer fd="column" rg="5px" ai="center">

<span>{teamData?.manager?.nationality\_code}</span>

<span>Национальность</span>

</FlexContainer>

{teamData?.manager?.date\_birth && (

<FlexContainer fd='column' rg='5px' ai='center'>

<span>{teamData?.manager?.date\_birth}</span>

<span>Дата рождения</span>

</FlexContainer>

)}

<FlexContainer fd="column" rg="5px" ai="center">

<span>{teamData?.manager?.preferred\_formation}</span>

<span>Любимая схема</span>

</FlexContainer>

</FlexContainer>

<TeamLineupInfo>

<h3>Состав команды</h3>

<FlexContainer ai='center' cg='15px'>

<FlexContainer ai='center' rg='5px' fd='column'>

{homeTeamLineup?.formation || 'Н/Д'}

<FlexContainer ai='center'><GroupsIcon /> Схема состава</FlexContainer>

</FlexContainer>

<FlexContainer ai='center' rg='5px' fd='column'>

{homeTeamLineup?.avg\_rating || 'Н/Д'}

<FlexContainer ai='center'>Средний рейтинг</FlexContainer>

</FlexContainer>

</FlexContainer>

<FlexContainer ai='center' cg='10px' style={{ height: '40px', width: '100%' }}>

{homeTeamLineup?.attacking && (

<FlexContainer ai='center' rg='5px' fd='column'>

<ProgressBar percent={homeTeamLineup.attacking} />

<FlexContainer ai='center'>Атака</FlexContainer>

</FlexContainer>

)}

{homeTeamLineup?.technical && (

<FlexContainer ai='center' rg='5px' fd='column'>

<ProgressBar percent={homeTeamLineup.technical} />

<FlexContainer ai='center'>Техника</FlexContainer>

</FlexContainer>

)}

{homeTeamLineup?.defending && (

<FlexContainer ai='center' rg='5px' fd='column'>

<ProgressBar percent={homeTeamLineup.defending} />

<FlexContainer ai='center'>Защита</FlexContainer>

</FlexContainer>

)}

{homeTeamLineup?.tactical && (

<FlexContainer ai='center' rg='5px' fd='column'>

<ProgressBar percent={homeTeamLineup.tactical} />

<FlexContainer ai='center'>Тактика</FlexContainer>

</FlexContainer>

)}

{homeTeamLineup?.creativity && (

<FlexContainer ai='center' rg='5px' fd='column'>

<ProgressBar percent={homeTeamLineup.creativity} />

<FlexContainer ai='center'>Креативность</FlexContainer>

</FlexContainer>

)}

</FlexContainer>

<TeamLineupTable rows={homeTeamLineupRows} />

</TeamLineupInfo>

</TeamDataInfo>

</div>

<div style={{ backgroundColor: 'rgba(0,0,0,0.2)', backdropFilter: 'blur(15px)', height: '100%'}}>

<h2 style={{ background: 'var(--lightGray)', textAlign: 'center', color: '#353535'}}>

События команды:

</h2>

{eventsData.map(el => <EventItem event={el} key={el.id}/>)}

</div>

</TeamGrid>

</TeamWrapper>

</Box>

</Modal>

);

};

export default TeamModal;

**App.tsx**

import { FC } from 'react';

import Header from './Header';

import { GlobalStyle } from '../styles/GlobalStyle';

import { RouteSwitcher } from './routeSwitcher';

import { Provider } from 'react-redux';

import { store } from '../store';

import '../styles/flags.css';

import { ThemeProvider } from '@mui/material';

import { theme } from '../styles/muiTheme';

const App: FC = () => {

return (

<Provider store={store}>

<ThemeProvider theme={theme}>

<Header />

<RouteSwitcher />

<GlobalStyle />

</ThemeProvider>

</Provider>

);

};

export default App;

**RouteSwitcher.tsx**

import {FC} from 'react';

import {Routes, Route, Navigate} from 'react-router';

import SportTypes from './SportTypes';

import Events from '../modules/Events';

import Favourites from '../modules/Favourites';

import News from '../modules/News';

export const RouteSwitcher: FC = () => {

return (

<div className="app-wrapper">

<div className="app-container">

<Routes>

<Route path="/events" element={<SportTypes />}>

<Route path=":id" element={<Events />} />

</Route>

<Route path="/favourites" element={<Favourites />}/>

<Route path="/news" element={<News />}/>

<Route

path="\*"

element={

<Navigate to="/events/football"/>

}

/>

</Routes>

</div>

</div>

);

};

# ПРИЛОЖЕНИЕ Г

**Константы**

**Файл api.constants.ts**

export enum ApiVersion{

Sections = '/sections',

Sports = '/sports',

}

export const DEFAULT\_SPORT\_ID = 1;

export const DEFAULT\_LIVE\_POLLING\_INTERVAL = 60000;

export const DEFAULT\_DEBOUNCE = 500;

**Файл common.constants.ts**

export const DEFAULT\_TIME\_FORMAT = 'HH:mm';

export const DEFAULT\_EVENT\_TIME\_FORMAT = 'DD.MM.YYYY HH:mm';

export const DEFAULT\_TIME\_ZONE = 'Europe/Moscow';

**Файл events.constants.ts**

import football from '../assets/backgrounds/football.jpg';

import tennis from '../assets/backgrounds/tennis.jpg';

import basketball from '../assets/backgrounds/basketball.jpg';

import hockey from '../assets/backgrounds/hockey.jpg';

import volleyball from '../assets/backgrounds/volleyball.jpg';

import handball from '../assets/backgrounds/handball.jpg';

export enum EventsViewModes {

ALL = 'ALL',

LIVE = 'LIVE',

}

export const EventsViewModesTitle = {

[EventsViewModes.ALL]: 'Все',

[EventsViewModes.LIVE]: 'Лайв',

};

export enum EventsStatuses {

NOT\_STARTED = 'notstarted',

INPROGRESS = 'inprogress',

FINISHED = 'finished',

POSTPONED = 'postponed',

CANCELED = 'canceled',

}

export const EventsStatusesTranslations = {

[EventsStatuses.NOT\_STARTED]: 'не начат',

[EventsStatuses.INPROGRESS]: 'лайв',

[EventsStatuses.FINISHED]: 'матч закончен',

[EventsStatuses.POSTPONED]: 'матч перенесен',

[EventsStatuses.CANCELED]: 'матч отменен',

};

export const EventsStatusesColors = {

[EventsStatuses.NOT\_STARTED]: 'var(--white)',

[EventsStatuses.INPROGRESS]: 'var(--lightGreen)',

[EventsStatuses.FINISHED]: 'var(--lightCyan)',

[EventsStatuses.POSTPONED]: 'var(--lightGray)',

[EventsStatuses.CANCELED]: 'var(--lightRed)',

};

export enum EventTypes {

FOOTBALL = 'football',

TENNIS = 'tennis',

BASKETBALL = 'basketball',

HOCKEY = 'hockey',

VOLLEYBALL = 'volleyball',

HANDBALL = 'handball',

}

export const EventTypesBackgrounds = {

[EventTypes.FOOTBALL]: football,

[EventTypes.TENNIS]: tennis,

[EventTypes.BASKETBALL]: basketball,

[EventTypes.HOCKEY]: hockey,

[EventTypes.VOLLEYBALL]: volleyball,

[EventTypes.HANDBALL]: handball,

};

export const EventTypesTranslations = {

[EventTypes.FOOTBALL]: 'Футбол',

[EventTypes.TENNIS]: 'Теннис',

[EventTypes.BASKETBALL]: 'Баскетбол',

[EventTypes.HOCKEY]: 'Хоккей',

[EventTypes.VOLLEYBALL]: 'Волейбол',

[EventTypes.HANDBALL]: 'Гандбол',

};

export const EventTypesBackgroundColors = {

[EventTypes.FOOTBALL]: '#008638',

[EventTypes.TENNIS]: '#3eff41',

[EventTypes.BASKETBALL]: '#ff5204',

[EventTypes.HOCKEY]: '#b7b7b7',

[EventTypes.VOLLEYBALL]: '#ffc757',

[EventTypes.HANDBALL]: '#a9a000',

};

export enum EventsStatusesMore {

FIRST\_HALF = '1st half',

FIRST\_SET = '1st set',

SECOND\_HALF = '2nd half',

SECOND\_SET = '2nd set',

THIRD\_HALF = '3rd half',

THIRD\_SET = '3rd set',

FOURTH\_HALF = '4th half',

WALKOVER = 'walkover',

CANCELED = 'Canceled',

POSTPONED = 'Postponed',

ENDED = 'Ended',

FT = 'FT',

ET = 'ET',

AET = 'AET',

}

export const EventsStatusesMoreTranslations = {

[EventsStatusesMore.FIRST\_HALF]: '1-ый тайм',

[EventsStatusesMore.FIRST\_SET]: '1-ый сет',

[EventsStatusesMore.SECOND\_HALF]: '2-ой тайм',

[EventsStatusesMore.SECOND\_SET]: '2-ой сет',

[EventsStatusesMore.THIRD\_HALF]: '3-ий тайм',

[EventsStatusesMore.THIRD\_SET]: '3-ий сет',

[EventsStatusesMore.FOURTH\_HALF]: '4-ый тайм',

[EventsStatusesMore.WALKOVER]: 'техническое поражение',

[EventsStatusesMore.CANCELED]: 'матч отменен',

[EventsStatusesMore.POSTPONED]: 'матч перенесен',

[EventsStatusesMore.ENDED]: 'матч окончен',

[EventsStatusesMore.ET]: 'овертайм',

[EventsStatusesMore.FT]: 'закончен в осн. время',

[EventsStatusesMore.AET]: 'закончен в доп. время',

};

export enum MatchInfoViewModes {

LINEUPS = 'LINEUPS',

H2H = 'H2H',

STATISTICS = 'STATISTICS',

INCIDENTS = 'INCIDENTS',

STADIUM = 'STADIUM',

}

export const MatchInfoViewModesTitle = {

[MatchInfoViewModes.LINEUPS]: 'Составы',

[MatchInfoViewModes.H2H]: 'H2H',

[MatchInfoViewModes.STATISTICS]: 'Статистика',

[MatchInfoViewModes.INCIDENTS]: 'События',

[MatchInfoViewModes.STADIUM]: 'Место проведения',

};

export enum EventPeriods {

FIRST = '1st',

SECOND = '2nd',

THIRD = '3rd',

FOURTH = '4th',

FIRSTQ = '1q',

SECONDQ = '2q',

THIRDQ = '3q',

FOURTHQ = '4q',

FIFTH = '5th',

ALL = 'all'

}

export const EventPeriodsTranslations = {

[EventPeriods.FIRST]: '1-ый период',

[EventPeriods.SECOND]: '2-ой период',

[EventPeriods.THIRD]: '3-йй период',

[EventPeriods.FOURTH]: '4-ый период',

[EventPeriods.FIFTH]: '5-ый период',

[EventPeriods.FIRSTQ]: '1-ая четверть',

[EventPeriods.SECONDQ]: '2-ая четверть',

[EventPeriods.THIRDQ]: '3-я четверть',

[EventPeriods.FOURTHQ]: '4-ая четверть',

[EventPeriods.ALL]: 'Весь матч',

};

export const EventStatisticsNameTranslations = {

ball\_possession: 'Владение мячом',

total\_shots: 'Ударов всего',

shots: 'Броски',

penalty\_minutes: 'Штрафное время',

goals: 'Голы',

goals\_in\_powerplay: 'Голы в большинстве',

shorthanded\_goals: 'Голы в меньшинстве',

faceoffs\_won: 'Вбрасываний выиграно',

blocked: 'Бросков заблокировано',

takeaways: 'Перехваты шайбы',

giveaways: 'Потери шайбы',

hits: 'Силовые приемы',

shots\_on\_target: 'Ударов по воротам',

shots\_off\_target: 'Ударов мимо ворот',

blocked\_shots: 'Заблокировано ударов',

corner\_kicks: 'Угловые',

offsides: 'Оффсайды',

hit\_woodwork: 'Попаданий в штангу',

shots\_inside\_box: 'Ударов из штрафной',

shots\_outside\_box: 'Ударов вне штрафной',

goalkeeper\_saves: 'Сейвы',

fouls: 'Нарушения',

passes: 'Количество пасов',

accurate\_passes: 'Точных пасов',

long\_balls: 'Длинных передач',

crosses: 'Кроссы',

dribbles: 'Удачных дриблингов',

possession\_lost: 'Потери мяча',

duels\_won: 'Единоборств выиграно',

aerials\_won: 'Единоборств в воздухе выиграно',

tackles: 'Отборы мяча',

interceptions: 'Перехваты',

clearances: 'Выносы мяча',

big\_chances: 'Голевые моменты',

yellow\_cards: 'Желтые карточки',

big\_chances\_missed: 'Упущенных голевых моментов',

aces: 'Эйсы',

double\_faults: 'Двойных ошибок на подаче',

first\_serve: 'Первая подача',

second\_serve: 'Вторая подача',

first\_serve\_points: 'Очков с первой подачи',

second\_serve\_points: 'Очков с второй подачи',

break\_points\_saved: 'Отыгранных брейк-поинтов',

service\_games\_played: 'Сыграно геймов с подачей',

first\_serve\_return\_points: 'Выигранных очков на приме первой подачи',

second\_serve\_return\_points: 'Выигранных очков на приме второй подачи',

break\_points\_converted: 'Реализация брейк-поинтов',

return\_games\_played: 'Выиграно геймов на приеме',

service\_points\_won: 'Выиграно очков на подаче',

receiver\_points\_won: 'Выиграно очков на приеме',

total\_points\_won: 'Всего выиграно очков',

max\_points\_in\_a\_row: 'Максимум очков подряд',

max\_games\_in\_a\_row: 'Максимум геймов подряд',

tiebreaks: 'Выигранных тай-брейков',

'2\_pointers': 'Двух-очковые',

'3\_pointers': 'Трех-очковые',

field\_goals: 'Попадания с игры',

rebounds: 'Подборы',

timeouts: 'Таймауты',

time\_spent\_in\_lead: 'Времени лидерства',

lead\_changes: 'Смены лидерства в счете',

biggest\_lead: 'Наибольшее преимущество',

service\_errors: 'Ошибок на подаче',

'7\_meters': '7-метровые',

'goal\_streak': 'Голевая серия',

'2\_min\_penalty': 'Двухминутные удаления',

};

export const EventTableNameTranslations = {

'P': 'И',

'W': 'В',

'D': 'Н',

'L': 'П',

'Goals': 'Голы',

'PTS': 'ОЧК',

}

**Файл lineups.constants.ts**

export const lineupsPositions = {

basketball: {

C:'C',

CF:'CF',

F:'F',

FC:'FC',

FG:'FG',

G:'G',

GF:'GF',

},

football: {

C:'C',

D:'D',

F:'F',

G:'G',

M:'M',

},

hockey: {

C:'C',

D:'D',

F:'F',

G:'G',

L:'L',

R:'R',

}

};

export const lineupsPositionsNaming = {

basketball: {

[lineupsPositions.basketball.C]:'Центральный',

[lineupsPositions.basketball.CF]:'Центральный форвард',

[lineupsPositions.basketball.F]:'Форвард',

[lineupsPositions.basketball.FC]:'Тяжёлый форвард',

[lineupsPositions.basketball.FG]:'Атакующий защитник',

[lineupsPositions.basketball.G]:'Разыгрывающий защитник',

[lineupsPositions.basketball.GF]:'Малый форвард',

},

football: {

[lineupsPositions.football.C]:'Центральный нападающий',

[lineupsPositions.football.D]:'Защитник',

[lineupsPositions.football.F]:'Форвард',

[lineupsPositions.football.G]:'Голкипер',

[lineupsPositions.football.M]:'Полузащитник',

},

hockey: {

[lineupsPositions.hockey.C]:'Центральный нападающий',

[lineupsPositions.hockey.D]:'Защитник',

[lineupsPositions.hockey.F]:'Форвард',

[lineupsPositions.hockey.G]:'Голкипер',

[lineupsPositions.hockey.L]:'Левый фронт, нападающий',

[lineupsPositions.hockey.R]:'Правый фронт, нападающий',

}

};

export const lineupsPositionsShortNaming = {

basketball: {

[lineupsPositions.basketball.C]:'Ц',

[lineupsPositions.basketball.CF]:'ЦФД',

[lineupsPositions.basketball.F]:'ФВД',

[lineupsPositions.basketball.FC]:'ТФД',

[lineupsPositions.basketball.FG]:'АЗ',

[lineupsPositions.basketball.G]:'РЗ',

[lineupsPositions.basketball.GF]:'МФ',

},

football: {

[lineupsPositions.football.C]:'ЦФД',

[lineupsPositions.football.D]:'ЗАЩ',

[lineupsPositions.football.F]:'ФВД',

[lineupsPositions.football.G]:'ВРТ',

[lineupsPositions.football.M]:'ПЗЩ',

},

hockey: {

[lineupsPositions.hockey.C]:'ЦФД',

[lineupsPositions.hockey.D]:'ЗАЩ',

[lineupsPositions.hockey.F]:'ФВД',

[lineupsPositions.hockey.G]:'ВРТ',

[lineupsPositions.hockey.L]:'ЛФН',

[lineupsPositions.hockey.R]:'ПФН',

}

};

# ПРИЛОЖЕНИЕ Д

**Самописные хуки**

**Файл useDebounce.ts:**

import { useState, useEffect } from 'react';

export function useDebounce<T>(value: T, delay: number): T {

const [debouncedValue, setDebouncedValue] = useState(value);

useEffect(() => {

const handler = setTimeout(() => {

setDebouncedValue(value);

}, delay);

return () => {

clearTimeout(handler);

};

}, [value, delay]);

return debouncedValue;

}

**Файл useSportTypesFetch.ts:**

import {useEffect, useState} from 'react';

import {useAppDispatch, useAppSelector} from '../store';

import {getSportTypes} from '../services/sports';

import {setSportTypes} from '../store/sportTypes/sportTypes.slice';

import {ISportPayload} from '../model/sport.model';

export interface ISportTypesFetchReturnType {

loading: boolean;

error: boolean;

}

export const useSportTypesFetch = (): ISportTypesFetchReturnType => {

const [loading, setLoading] = useState(false);

const [error, setError] = useState(false);

const dispatch = useAppDispatch();

const state = useAppSelector(state => state.sportTypes);

const fetchSports = async (): Promise<void> => {

const typesFromStorage = localStorage.getItem('sportTypes');

if (!state.sportTypes.length && !typesFromStorage) {

try {

setLoading(true);

setError(false);

const types = await getSportTypes();

localStorage.setItem('sportTypes', JSON.stringify(types));

dispatch(setSportTypes(types));

} catch (error) {

setLoading(false);

setError(true);

}

setLoading(false);

} else if (typesFromStorage) {

const typesFromStorageParsed = JSON.parse(typesFromStorage) as ISportPayload;

dispatch(setSportTypes(typesFromStorageParsed));

}

};

useEffect(() => {

void fetchSports();

// eslint-disable-next-line react-hooks/exhaustive-deps

}, []);

return {loading, error};

};

# ПРИЛОЖЕНИЕ Е

**Исходный код модульных компонентов приложения**

**Файл Events.styled.ts**

import styled from 'styled-components';

import {

EventTypes,

EventTypesBackgrounds,

} from '../../constants/events.constants';

export const Wrapper = styled.div`

height: calc(100vh - 60px - 10vh);

padding-top: 15px;

`;

export const Content = styled.main`

display: grid;

grid-template-columns: 1fr 4fr;

height: 100%;

max-height: calc(100vh - 60px - 15vh);

gap: 15px;

`;

export const GridItem = styled.div<{id: EventTypes}>`

height: calc(100vh - 60px - 15vh);

color: #aaa;

font-size: 20px;

font-weight: 700;

background: url(${props => EventTypesBackgrounds[props.id]}) no-repeat;

overflow: overlay;

&:last-child {

background-position: bottom;

background-size: cover;

}

`;

**Файл Events.tsx**

import React, { FC, useMemo } from 'react';

import {

Content,

GridItem,

Wrapper,

} from './Events.styled';

import { EventsPropsType } from './Events.types';

import Fade from 'react-reveal/Fade';

import { TransitionGroup } from 'react-transition-group';

import {

EventTypes,

} from '../../constants/events.constants';

import Event from '../../components/Event';

import SectionsList from '../../components/SectionsList';

import EventsList from '../../components/EventsList';

import { useParams } from 'react-router';

import { DEFAULT\_SPORT\_ID } from '../../constants/api.constants';

import { useAppSelector } from '../../store';

import ChallengeModal from '../../components/ChallengeModal';

import TeamModal from '../../components/TeamModal';

const Events: FC<EventsPropsType> = () => {

const { id } = useParams<{ id: EventTypes }>();

const { sportTypes } = useAppSelector(state => state.sportTypes);

const sportId = useMemo(() => {

return sportTypes?.find(el => el.slug === id)?.id || DEFAULT\_SPORT\_ID;

}, [id, sportTypes]);

return (

<Wrapper>

<Content>

<GridItem id={id || EventTypes.FOOTBALL}>

<SectionsList sportId={sportId} />

</GridItem>

<TransitionGroup appear enter exit key={id}>

<Fade>

<GridItem id={id || EventTypes.FOOTBALL}>

<EventsList sportId={sportId}/>

</GridItem>

</Fade>

</TransitionGroup>

</Content>

<Event />

<ChallengeModal/>

<TeamModal/>

</Wrapper>

);

};

export default Events;

**Файл Favourites.tsx**

import React from 'react';

import { Content, Wrapper, GridItem } from '../Events/Events.styled';

import { EventTypes, EventTypesBackgroundColors, EventTypesTranslations } from '../../constants/events.constants';

import { IEvent } from '../../model/events.model';

import EventItem from '../../components/EventItem';

import { groupBy } from '../../utils/common';

import Event from '../../components/Event';

import { Tooltip } from '@mui/material';

import { getTranslations } from '../../utils/translations';

import { SectionItem, SectionsListContainer } from '../../components/SectionsList/SectionsList.styled';

import { ILeague } from '../../model/league.model';

import { challengeModalActions } from '../../store/challengeModal/challengeModal.slice';

import { useAppDispatch } from '../../store';

import ChallengeModal from '../../components/ChallengeModal';

import TeamModal from '../../components/TeamModal';

const Favourites = () => {

const currentFavourites: IEvent[] = JSON.parse(localStorage.getItem('favouritesEvents') || '[]');

const currentFavouritesLeagues: ILeague[] = JSON.parse(localStorage.getItem('favouritesLeagues') || '[]');

const groupedBy = Object.values(groupBy(currentFavourites, el => el.sport.slug));

const dispatch = useAppDispatch();

const handleOpenChallengeModal = (id: number) => {

dispatch(challengeModalActions.openChallengeModal(id));

};

return (

<Wrapper>

<Content>

<GridItem id={EventTypes.FOOTBALL}>

<h2 style={{ background: 'var(--lightGray)', textAlign: 'center', color: '#333'}}>

Чемпионаты

</h2>

<SectionsListContainer>

{currentFavouritesLeagues.map(league => (

<Tooltip

key={league.slug}

title={'Выбрать категорию: ' + getTranslations(league)}

placement="top-end"

>

<SectionItem

onClick={() => handleOpenChallengeModal(league.id)}

isActive={false}

>

{league.has\_logo && <img src={league.logo} alt='logo' />}

{getTranslations(league)}

</SectionItem>

</Tooltip>

))}

</SectionsListContainer>

</GridItem>

<GridItem id={EventTypes.FOOTBALL}>

<div style={{ backgroundColor: 'rgba(0,0,0,0.2)', backdropFilter: 'blur(15px)', height: '100%'}}>

{groupedBy.map((group) => (

<>

<h2 style={{ background: EventTypesBackgroundColors[group[0].sport.slug], textAlign: 'center', color: '#353535'}}>

{EventTypesTranslations[group[0].sport.slug]}

</h2>

{group.map(el => <EventItem event={el} key={el.id}/>)}

</>

))}

</div>

</GridItem>

</Content>

<Event />

<ChallengeModal />

<TeamModal />

</Wrapper>

);

};

export default Favourites;

**Файл News.styled.ts**

import styled from 'styled-components';

export const NewsWrapper = styled.div`

height: calc(100vh - 10vh);

padding-top: 15px;

`;

export const NewsContent = styled.main`

margin-top: 10px;

display: grid;

grid-template-columns: 1fr 1fr;

height: 100%;

max-height: calc(100vh - 15vh);

gap: 15px;

`;

export const NewsGridItem = styled.div`

cursor: pointer;

height: fit-content;

color: #fff;

font-size: 20px;

font-weight: 700;

border-radius: 4px;

border: 1px solid white;

padding: 15px;

grid-template-areas:

"i i i i"

"i i i i"

"t t t t"

"d d d d";

img{

grid-area: i;

width: 100%;

}

h2{

grid-area: t;

border-bottom: 1px solid white;

}

&:hover{

p{

display: block;

}

}

transition: all 0.5s ease;

p{

margin-top: 5px;

grid-area: d;

display: none;

font-size: 18px;

color: var(--lightGray);

}

`;

**Файл News.tsx**

import React, { useEffect, useMemo, useState } from 'react';

import { NewsContent, NewsGridItem, NewsWrapper } from './News.styled';

import { InputAdornment, TextField, Tooltip } from '@mui/material';

import { Search } from '@mui/icons-material';

import { useDebounce } from '../../hooks/useDebounce';

import { DEFAULT\_DEBOUNCE } from '../../constants/api.constants';

import { useGetNewsQuery } from '../../services/news';

import { Spinner } from '../../components/Spinner';

const News = () => {

const [searchValue, setSearchValue] = useState('');

const [isInit, setIsInit] = useState(false);

const debouncedSearchValue = useDebounce(searchValue, DEFAULT\_DEBOUNCE);

useEffect(() => {

setIsInit(true);

}, []);

const { data, isLoading, isFetching } = useGetNewsQuery(isInit ? debouncedSearchValue : searchValue);

const newsData = useMemo(() => data && data.articles || [], [data]);

const handleNewsClick = (url: string) => {

if(url) window.open(url, '\_blank').focus();

};

return (

<NewsWrapper>

<TextField

value={searchValue}

onChange={(e) => setSearchValue(e.target.value)}

variant='outlined'

placeholder='Название новости'

InputProps={{

startAdornment: (

<InputAdornment position='start'>

<Search />

</InputAdornment>

),

}}

/>

{newsData && !isLoading && !isFetching ? (

<NewsContent>

{newsData.map(news => (

<Tooltip title={`Источник: ${ news.source.name }`} onClick={() => handleNewsClick(news.url)}>

<NewsGridItem>

<img src={news.urlToImage} alt='logo' />

<h2>{news.title}</h2>

<p>{news.description}</p>

</NewsGridItem>

</Tooltip>

))}

</NewsContent>

) : (

<Spinner width="50px" />

)}

</NewsWrapper>

);

};

# ПРИЛОЖЕНИЕ Ж

**Конфигурационные файлы API для работы с сервером**

**Файл api.ts**

import { ISectionModel } from '../model/sections.model';

import { baseURL } from './rest';

import { createApi, fetchBaseQuery } from '@reduxjs/toolkit/query/react';

import { TFetchedData } from '../model/common.model';

import { IEvent } from '../model/events.model';

import { IGetEventsBySportIdParams } from '../model/api.model';

import { EventsViewModes } from '../constants/events.constants';

import { ILineupModel } from '../model/lineup.model';

import { IStatisticsModel } from '../model/statistics.model';

import { IIncidentsModel } from '../model/incidents.model';

import { ILeague, ISeason, ISeasonTable } from '../model/league.model';

import { IPlayerModel } from '../model/player.model';

import { ITeam } from '../model/team.model';

export const API = createApi({

reducerPath: 'API',

baseQuery: fetchBaseQuery({

baseUrl: baseURL,

prepareHeaders: headers => {

headers.set('x-rapidapi-host', 'sportscore1.p.rapidapi.com');

headers.set('x-rapidapi-key', 'fcf0f94828mshe4442eea88a7d01p15a177jsn1344e9c69c0e');

return headers;

},

}),

endpoints: (builder) => ({

getSectionsBySportId: builder.query<TFetchedData<ISectionModel[]>, number>({

query: (id) => `/sports/${id}/sections`,

}),

getEventsBySportId: builder.query<TFetchedData<IEvent[]>, IGetEventsBySportIdParams>({

query: ({ id, mode, date }) => `/sports/${id}/events${mode === EventsViewModes.ALL ?( '/date/' + date) : '/live'}`,

}),

getEventDataById: builder.query<TFetchedData<IEvent>, number | null>({

query: (id) => `/events/${String(id)}`,

}),

getChallengeDataById: builder.query<TFetchedData<ILeague>, number | null>({

query: (id) => `/leagues/${String(id)}`,

}),

getH2HEvents: builder.query<TFetchedData<IEvent[]>, { homeTeam: number, awayTeam: number }>({

query: ({ homeTeam, awayTeam }) => `/teams/${String(homeTeam)}/h2h-events/${String(awayTeam)}`,

}),

getEventLineupsById: builder.query<TFetchedData<ILineupModel[]>, number | null>({

query: (id) => `/events/${String(id)}/lineups`,

}),

getEventIncidentsById: builder.query<TFetchedData<IIncidentsModel[]>, number | null>({

query: (id) => `/events/${String(id)}/incidents`,

}),

getEventStatisticsById: builder.query<TFetchedData<IStatisticsModel[]>, number | null>({

query: (id) => `/events/${String(id)}/statistics`,

}),

getSeasonsByLeagueId: builder.query<TFetchedData<ISeason[]>, number | null>({

query: (id) => `/leagues/${String(id)}/seasons`,

}),

getSeasonsTableById: builder.query<TFetchedData<ISeasonTable[]>, number | null>({

query: (id) => `/seasons/${String(id)}/standings-tables`,

}),

getSeasonsEventsById: builder.query<TFetchedData<IEvent[]>, number | null>({

query: (id) => `/seasons/${String(id)}/events`,

}),

getTeamDataById: builder.query<TFetchedData<ITeam>, number | null>({

query: (id) => `/teams/${String(id)}`,

}),

getTeamEventsById: builder.query<TFetchedData<IEvent[]>, number | null>({

query: (id) => `/teams/${String(id)}/events`,

}),

getTeamLineupsById: builder.query<TFetchedData<ILineupModel[]>, number | null>({

query: (id) => `/teams/${String(id)}/lineups`,

}),

getPlayerDataById: builder.query<TFetchedData<IPlayerModel>, number | null>({

query: (id) => `/players/${String(id)}`,

}),

}),

});

export const {

useGetSectionsBySportIdQuery,

useGetEventsBySportIdQuery,

useGetEventDataByIdQuery,

useGetEventLineupsByIdQuery,

useGetH2HEventsQuery,

useGetEventStatisticsByIdQuery,

useGetEventIncidentsByIdQuery,

useGetChallengeDataByIdQuery,

useGetSeasonsByLeagueIdQuery,

useGetSeasonsTableByIdQuery,

useGetSeasonsEventsByIdQuery,

useGetTeamDataByIdQuery,

useGetTeamEventsByIdQuery,

useGetTeamLineupsByIdQuery,

useGetPlayerDataByIdQuery,

} = API;

**Файл news.ts**

import { createApi, fetchBaseQuery } from '@reduxjs/toolkit/dist/query/react';

import { INewsModel, TFetchNews } from '../model/news.model';

export const NEWSAPI = createApi({

reducerPath: 'NEWS\_API',

baseQuery: fetchBaseQuery({

baseUrl: 'https://newsapi.org/v2/',

}),

endpoints: (builder) => ({

getNews: builder.query<TFetchNews<INewsModel>, string>({

query: (q) => ({

url:'top-headlines',

params: { q, country: 'ru', pageSize:30 , category: 'sports', apiKey: '7f2791e2360a4917b836537a869d3e7b'}

}),

}),

}),

});

export const { useGetNewsQuery, } = NEWSAPI;

**Файл rest.js**

import axios from 'axios';

import {getBaseUrl, paramsSerializer} from '../utils/url';

// eslint-disable-next-line @typescript-eslint/explicit-function-return-type

export const getHeader = () => {

return {

'x-rapidapi-host': 'sportscore1.p.rapidapi.com',

'x-rapidapi-key': 'fcf0f94828mshe4442eea88a7d01p15a177jsn1344e9c69c0e'

};

};

export const baseURL = getBaseUrl();

const instance = axios.create({

baseURL,

paramsSerializer

});

// eslint-disable-next-line @typescript-eslint/explicit-function-return-type

export const restGet = (url, config) => {

// eslint-disable-next-line @typescript-eslint/no-unsafe-argument,@typescript-eslint/no-unsafe-assignment

return instance.get(url, {params: {...config}, headers: {...getHeader()}});

};

**Файл sports.ts**

import {restGet} from './rest';

import {ApiVersion} from '../constants/api.constants';

import {ISportPayload} from '../model/sport.model';

export const getSportTypes = async (): Promise<ISportPayload> => {

const response = await restGet(`${ApiVersion.Sports}`);

return response.data as ISportPayload;

};

# ПРИЛОЖЕНИЕ З

**Конфигурационные файлы хранилища состояния приложения (Redux)**

**Файл challengeModal.slice.ts**

import { createSlice, PayloadAction } from '@reduxjs/toolkit';

import { TAppState } from '../index';

interface IChallengeModalState {

challengeId: number | null;

isOpen: boolean;

}

const initialState: IChallengeModalState = {

challengeId: null,

isOpen: false,

};

export const challengeModalSlice = createSlice({

name: 'challengeModal',

initialState,

reducers: {

openChallengeModal: (state: IChallengeModalState, action: PayloadAction<number>) => {

state.isOpen = true;

state.challengeId = action.payload;

},

closeChallengeModal: (state: IChallengeModalState) => {

state.isOpen = false;

state.challengeId = null;

}

},

});

export const challengeModalSelector = (state: TAppState) => state.challengeModal;

export const challengeModalActions = challengeModalSlice.actions;**Файл eventModal.slice.ts**

import { createSlice, PayloadAction } from '@reduxjs/toolkit';

import { TAppState } from '../index';

interface IEventModalState {

eventId: number | null;

isOpen: boolean;

}

const initialState: IEventModalState = {

eventId: null,

isOpen: false,

};

export const eventModalSlice = createSlice({

name: 'eventModal',

initialState,

reducers: {

openEventModal: (state: IEventModalState, action: PayloadAction<number>) => {

state.isOpen = true;

state.eventId = action.payload;

},

closeEventModal: (state: IEventModalState) => {

state.isOpen = false;

state.eventId = null;

}

},

});

export const eventModalSelector = (state: TAppState) => state.eventModal;

export const eventModalActions = eventModalSlice.actions;

**Файл events.slice.ts**

import { createSlice, PayloadAction } from '@reduxjs/toolkit';

interface IEventsState {

searchValue: string;

selectedSectionId: number | null;

}

const initialState: IEventsState = {

searchValue: '',

selectedSectionId: null,

};

export const eventsSlice = createSlice({

name: 'events',

initialState,

reducers: {

setSearchValue: (state: IEventsState, action: PayloadAction<string>) => {

state.searchValue = action.payload;

},

setSelectedSection: (state: IEventsState, action: PayloadAction<null | number>) => {

state.selectedSectionId = action.payload;

},

},

});

export const { setSearchValue, setSelectedSection } = eventsSlice.actions;

**Файл sportTypes.slice.ts**

import {createSlice, PayloadAction} from '@reduxjs/toolkit';

import {ISportModel, ISportPayload} from '../../model/sport.model';

interface ISportTypesState {

sportTypes: ISportModel[];

}

const initialState: ISportTypesState = {

sportTypes: [],

};

export const sportTypeSlice = createSlice({

name: 'sportTypes',

initialState,

reducers: {

setSportTypes: (state:ISportTypesState, action: PayloadAction<ISportPayload>) => {

state.sportTypes = action.payload.data;

}

}

});

export const {setSportTypes} = sportTypeSlice.actions;

**Файл teamModal.slice.ts**

import { createSlice, PayloadAction } from '@reduxjs/toolkit';

import { TAppState } from '../index';

interface ITeamModalState {

teamId: number | null;

isOpen: boolean;

}

const initialState: ITeamModalState = {

teamId: null,

isOpen: false,

};

export const teamModalSlice = createSlice({

name: 'playerModal',

initialState,

reducers: {

openChallengeModal: (state: ITeamModalState, action: PayloadAction<number>) => {

state.isOpen = true;

state.teamId = action.payload;

},

closeChallengeModal: (state: ITeamModalState) => {

state.isOpen = false;

state.teamId = null;

}

},

});

export const teamModalSelector = (state: TAppState) => state.teamModal;

export const teamModalActions = teamModalSlice.actions;

**Файл index.ts**

import { configureStore } from '@reduxjs/toolkit';

import { TypedUseSelectorHook, useDispatch, useSelector } from 'react-redux';

import { sportTypeSlice } from './sportTypes/sportTypes.slice';

import { API } from '../services/api';

import { eventsSlice } from './events/events.slice';

import { eventModalSlice } from './eventModal/eventModal.slice';

import { challengeModalSlice } from './challengeModal/challengeModal.slice';

import { playerModalSlice } from './playerModal/playerModal.slice';

import { teamModalSlice } from './teamModal/teamModal.slice';

import { NEWSAPI } from '../services/news';

export const store = configureStore({

reducer: {

sportTypes: sportTypeSlice.reducer,

events: eventsSlice.reducer,

eventModal: eventModalSlice.reducer,

challengeModal: challengeModalSlice.reducer,

playerModal: playerModalSlice.reducer,

teamModal: teamModalSlice.reducer,

[API.reducerPath]: API.reducer,

[NEWSAPI.reducerPath]: NEWSAPI.reducer,

},

middleware: (getDefaultMiddleware) =>

getDefaultMiddleware().concat(API.middleware).concat(NEWSAPI.middleware),

});

export type TAppState = ReturnType<typeof store.getState>;

export type TAppDispatch = typeof store.dispatch;

export const useAppDispatch = () => useDispatch<TAppDispatch>();

export const useAppSelector: TypedUseSelectorHook<TAppState> = useSelector;

# ПРИЛОЖЕНИЕ И

**Исходные файлы стилизации**

**Файл GlobalStyle.ts**

import { createGlobalStyle } from 'styled-components';

export const GlobalStyle = createGlobalStyle`

:root {

--midGray: #222;

--darkGray: #111;

--midGrayTP: #22222220;

--containerWidth: 1140px;

--navLinkColor: #bbb;

--activeLinkBackground: #ff9700;

--white: #fff;

--lightGray: #e5e5e5;

--gray: #a2a2a2;

--darkBlue: #345193;

--blue: #5989ff;

--darkBlueTP: #34519320;

--darkOrange: #981c1c;

--lightGreen: #49ff00;

--green: #007c06;

--lightRed: #ff000d;

--lightCyan: #00fff0;

--color-scrollbar: #bdbdbd;

--color-scrollbar-hover: #b3b3b3;

}

\* {

margin: 0;

padding: 0;

-webkit-box-sizing: border-box;

box-sizing: border-box;

font-family: Roboto, sans-serif;

&::-webkit-scrollbar-thumb {

background: var(--color-scrollbar);

}

&::-webkit-scrollbar-thumb:hover {

background: var(--color-scrollbar-hover);

}

&::-webkit-scrollbar {

width: 5px;

height: 5px;

}

&::-webkit-scrollbar-track {

border-radius: 15px;

}

&::-webkit-scrollbar-thumb {

border-radius: 3px;

}

}

html {

font-size: 16px;

}

body {

background: var(--darkGray);

}

.app-wrapper {

}

.app-container {

max-width: var(--containerWidth);

margin: 0 auto;

}

.datePickerInput {

width: 180px;

input {

pointer-events: none;

position: relative;

&:after {

pointer-events: none;

position: absolute;

left: 0;

top: 0;

width: 200px;

height: 40px;

content: ' ';

}

}

}

`;

**Файл muiTheme.ts**

import { createTheme } from '@mui/material';

const backgroundColor = '#fff';

const paperBackgroundColor = '#333';

const textColor = '#fff';

const primaryColor = '#fff';

export const theme = createTheme({

palette: {

mode: 'dark',

background: {

default: backgroundColor,

paper: paperBackgroundColor

},

text: {

primary: textColor,

},

primary: {

main: primaryColor

},

}

});

# ПРИЛОЖЕНИЕ К

**Исходный код вспомогательных файлов**

**Файл common.ts:**

import moment from 'moment-timezone';

export const groupBy = <T>(array: T[], predicate: (v: T) => string | number) =>

array.reduce((acc, value) => {

(acc[predicate(value)] ||= []).push(value);

return acc;

}, {} as { [key: string]: T[] });

export const formatEventStartDate = (date: string) => {

return moment.tz(date, 'Etc/GMT+0');

};

**Файл translations.ts**

export const getTranslations = (obj: any): string => {

// eslint-disable-next-line @typescript-eslint/no-unsafe-member-access

return (obj?.name\_translations?.ru ? obj.name\_translations.ru : obj?.name) as string;

};

**Файл url.ts**

import qs from 'qs';

export function getBaseUrl(): string | undefined {

return 'https://sportscore1.p.rapidapi.com';

}

export const paramsSerializer = (obj): string =>

qs.stringify(obj);

# ПРИЛОЖЕНИЕ Л

**Исходный код входного компонента index.tsx**

import ReactDOM from 'react-dom';

import React from 'react';

import App from './components/app';

import {BrowserRouter} from 'react-router-dom';

ReactDOM.render(

<BrowserRouter>

<App/>

</BrowserRouter>,

document.getElementById('root'),

);

1. trashbox.ru/link/top-sport-aggregators [↑](#footnote-ref-1)
2. www.flashscore.ru.com [↑](#footnote-ref-2)
3. www.championat.com [↑](#footnote-ref-3)
4. www.tipsscore.com/api/documentation [↑](#footnote-ref-4)
5. www.nodejs.org/en [↑](#footnote-ref-5)
6. www.learn.javascript.ru/intro [↑](#footnote-ref-6)
7. www.w3.org/html [↑](#footnote-ref-7)
8. www.w3.org/Style/CSS [↑](#footnote-ref-8)
9. www.typescriptlang.org [↑](#footnote-ref-9)
10. www.codenet.ru/webmast/js/spa [↑](#footnote-ref-10)
11. www.reactjs.org [↑](#footnote-ref-11)
12. www.redux.js.org [↑](#footnote-ref-12)
13. www.redux-toolkit.js.org [↑](#footnote-ref-13)
14. www.styled-components.com [↑](#footnote-ref-14)
15. www.tproger.ru/articles/styled-components-idealnaja-stilizacija-react-prilozhenija/ [↑](#footnote-ref-15)
16. www.reactrouter.com/docs/en/v6 [↑](#footnote-ref-16)
17. www.jetbrains.com/ru-ru/webstorm [↑](#footnote-ref-17)
18. www.webpack.js.org [↑](#footnote-ref-18)
19. www.atlassian.com/ru/git/tutorials/what-is-git [↑](#footnote-ref-19)
20. www.git-scm.com/book/ru/v2 [↑](#footnote-ref-20)
21. www.eslint.org/ [↑](#footnote-ref-21)
22. www.prettier.io [↑](#footnote-ref-22)
23. chrome.google.com/webstore/detail/react-developer-tools [↑](#footnote-ref-23)
24. chrome.google.com/webstore/detail/redux-devtools [↑](#footnote-ref-24)