УДК 004.584

Разработка онлайн-платформы для осуществления букмекерской деятельности

Нестеров С.А., Пономарев И.В.
Алтайский государственный университет, г. Барнаул
ice22333@gmail.com, igorpon@mail.ru

Аннотация

Работа посвящена разработке веб-приложения для прогнозирования исходов спортивных событий, которое позволит оценить составленные программой проценты на исходы определенной подборки матчей по разным видам спорта.

Ключевые слова: разработка веб-приложения, разработка веб-сервиса, букмекерская деятельность, прогнозирование исходов спортивных событий, проценты.

1. Введение

В настоящее время среди пользователей сети Интернет большой популярностью пользуются веб-приложения для ведения заметок. Они позволяют вести записи на самые разные темы – от распорядка дня до научных конспектов. Однако не каждый сервис для ведения личных записей позволяет пользователю внедрять в тексты заметок гиперссылки на другие заметки. Между тем, такая функция, на наш взгляд, может оказаться весьма полезной при ведении научных записей. Возможность быстро перейти от одной заметки к другой, кликнув по гиперссылке, позволяет видеть контекст каждой заметки и легче ориентироваться в своих записях.

WEB-разработка – процедура создания WEB-приложения или WEB-сайта. Основными этапами этого процесса являются такие мероприятия, как WEB-дизайн, вёрстка страниц сайта, WEB-программирование на стороне сервера и клиента, а также работы по конфигурированию WEB-сервера. На передовых позициях WEB-разработки остались три основные технологии. Простота PHP и допустимость использования в нем встроенных ссылок на программу базы данных MySQL обеспечили этому языку более чем двойное превосходство по количеству пользователей над другими подобными языками. A JavaScript, ставший важнейшей составной частью уравнения, используемого для динамического манипулирования каскадными таблицами стилей (Cascading Style Sheets – CSS) и HTML, в настоящее время берет на себя наиболее трудоемкие задачи осуществления асинхронного обмена данными.

Букмекерская контора – это игорное заведение, занимающееся приемом ставок. Букмекер оценивает вероятность того или иного исхода события и на каждый из возможных исходов выставляет коэффициент — числовое значение, на которое умножается ставка в случае успеха. Букмекерские конторы могут осуществлять свою деятельность в наземных пунктах приема ставок (ППС), а также в сети Интернет (онлайн-букмекеры). В пунктах приема ставок осуществляется наличный расчет, а выплата выигрышей производится через кассу. Некоторые компании называют свои ППС клубами и оформляют их так, чтобы у клиентов была возможность не только делать ставки, но и полноценно отдыхать, просматривая прямые трансляции спортивных событий.

Несмотря на комфорт, который букмекеры стараются обеспечить посетителям своих пунктов приема ставок, из года в год растет популярность онлайн-букмекеров – тех, которые принимают ставки через Интернет.

2. Программная реализация

Для реализации веб-сервиса необходимо решить следующие задачи:

1. Разработка API: Создание и настройка API со статистикой, который будет хранить информацию о матчах, командах и их статистике.
2. Разработка функциональности прогнозирования: Реализация алгоритмов и логики, позволяющих программе оценивать проценты на победу команд на основе имеющихся данных и статистики.
3. Создание пользовательского интерфейса: Разработка веб-интерфейса, который позволит пользователям выбирать вид спорта и просматривать соответствующие матчи и проценты на победу.
4. Интеграция с API: Установка соединения с API и реализация функциональности, позволяющей получать данные о матчах и командах из API со статистикой.
5. Тестирование и отладка: Проверка работоспособности приложения, обнаружение и устранение возможных ошибок и проблем.
6. Документация: Создание документации, которая содержит описание функциональности приложения, инструкции по использованию и другую необходимую информацию.

Далее, необходимо выделить основной инструментарий, используемый для разработки приложения:

1. HTML (HyperText Markup Language) – язык гипертекстовой разметки. Он нужен, чтобы размещать на веб-странице элементы: текст, картинки, таблицы и видео.
2. Cascading Style Sheets (CSS) – это язык иерархических правил (таблиц стилей), используемый для представления внешнего вида документа, написанного на HTML.
3. JavaScript – это интерпретируемый язык программирования с объектно-ориентированными возможностями.
4. React – это декларативная, эффективная и гибкая JavaScript библиотека для создания пользовательских интерфейсов.
5. Redux – это библиотека JavaScript для управления состоянием приложения. Он позволяет легко управлять состоянием приложения в едином хранилище, что делает его более простым и предсказуемым.
6. Redux Toolkit – упрощает процесс управления состояния приложением и предлагает удобный API для работы с асинхронными запросами и управления состоянием приложения.
7. Tailwind CSS – инструмент, который предоставляет набор готовых классов для быстрой и удобной разработки веб-приложений. Он позволяет создавать пользовательские интерфейсы на основе набора классов, которые можно применять к HTML-элементам для быстрого создания стилей.
8. React Router – это библиотека маршрутизации для React, которая позволяет создавать SPA (Single Page Application), где каждый компонент приложения может отобразиться на странице в зависимости от URL-адреса. Он предоставляет компоненты, которые позволяют определить маршруты и связанные с ними компоненты.

9. Средой программирования было выбрано платное приложение WebStorm. WebStorm предоставляет множество функций и инструментов, таких как автодополнение кода, инспектирование кода, поддержка отладки приложений, автоматическое форматирование кода и многое другое. WebStorm обладает продвинутым интерфейсом, позволяющим удобно работать с различными библиотеками и фреймворками (например React, VueJS, Angular), а также поддерживает удобную интеграцию с различными инструментами и системами, такими как Git, GitHub, ESLint, Prettier и многими другими. Гибкость и многозадачность приложения WebStorm позволяет прорабатывать все необходимые элементы веб-сайта одновременно и грамотно выстраивать архитектуру приложения.

Главная задача – разработать реактивное веб-приложение (веб-сайт) для прогнозирования исходов спортивных событий, используя такие технологии как HTML, CSS (Tailwind CSS), JavaScript, React, Redux и Router.

Реализуемые функции:

- Возможность выбора видов спорта для пользователя.
- Отображение актуальных матчей, названий и логотипов команд в каждом матче, а также времени и места проведения.
- Подсчет и вывод шансов победы каждой команды в каждом матче.

Каждый процент должен подсчитываться по определенной формуле, используя статистику команды. Выводимый процент дает оценку и отображает вероятность победы той или иной команды.

В качество источника со статистикой команд спортивные порталы и букмекерские конторы используют API. API (Application Program Interface) – это программный интерфейс, позволяющий связывать между собой различные приложения. Создан для упрощения и ускорения разработки. Содержит наборы методов, классов, библиотек и функций.

В этой работе будет использоваться искусственное API, созданное вручную внутри проекта, так как абсолютно все доступные в сети спортивные API являются платными.

Содержание API должно быть реализовано следующим образом:

- В API находится массив объектов. Каждый объект – это матч. Каждый матч содержит в себе: названия обеих команд, логотипы обеих команд (подгружаемые напрямую из интернета), вид спорта, место и дата проведения, проценты побед каждой команды.
- Также в каждый объект записана статистика той и другой команды, а конкретнее их процент побед за все время (WinRate).
- WinRate каждой команды при обновлении страницы рандомизируется.

Структурно веб-приложение должно из себя представлять веб-страницу, где пользовать может выбрать интересующий его вид спорта и перейти на страницу с матчами. После того как пользователь выберет вид спорта, на новой будут отображаться матчи с выводом подробной информации (т.е. команды, дата, место и процент победы).

Файловая структура приложения является типичной структурой для React-приложения.

Папка node-modules содержит все необходимые файлы для работы библиотек (React, Redux, ReduxJSToolkit, Tailwind, React-Router).

В папке /public/ находится файл index.html, в который потом подгружается вся наша «реактивная» структура из папки /src/.

Переходим к папке /src/:

- Папка /assets/ содержит в себе логотип сайта, а также может содержать изображения, содержащиеся на сайте.
- Папка /components/ содержит в себе все необходимые для работы сайта компоненты в формате jsx. Например – компонент MatchItem отвечает за вывод одного конкретного матча, а MatchesList выводит массив компонентов MatchItem с различными данными, подгруженными из MatchesSlice.
- В папке /pages/ содержатся файлы формата jsx для страниц нашего веб-сайта. Например, страница HomePage отображается при загрузке сайта, а MatchesPage «мимикрирует» при загрузке, подстраиваясь под выбранный пользователем вид спорта и подгружая конкретные матчи.
- Папка /store/ хранит в себе несколько папок с файлами конфигурации для «slice'ов» и «thunk'ов».
- Папка utils хранит в себе вспомогательные функции. В нашем случае она одна.
- Файл App.js является основным рабочим файлом, который и является нашим полноценным приложением.
- Файлы fonts.css и preloader.css отвечают за нестандартные шрифты и CSS-классы, которые не входят в базовый комплект Tailwind CSS.
- Остальные файлы (index.js, index.css) отвечают за базовую работу приложения и различных библиотек.

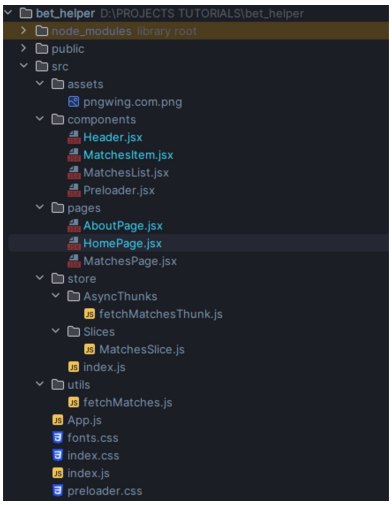

Рисунок 1. Файловая структура проекта

3. Математическая модель приложения

Наш искусственно смоделированный API реализован следующим образом:

- На каждый вид спорта создается отдельный массив объектов, каждый объект является матчем.
- На каждый матч заявлены две команды, у которых есть названия и логотипы (`team*_name`, `team*_logo`).

- Обычно в спортивных API у команд есть определенная статистика, например: процент побед, складывающийся из общего количества матчей и количества побед. В нашем случае мы будем использовать «рандомизированный» процент побед для каждой команды, так как у нас нет доступа к платной статистике.

- Дополним каждый матч местом и датой проведения.

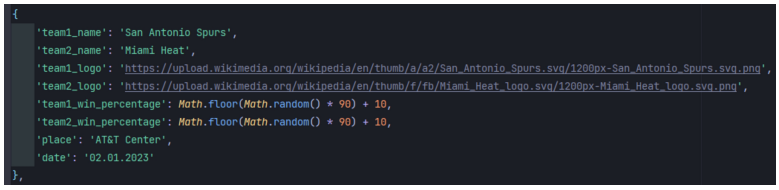

Рисунок 2. Пример объекта «матча»

Следующим шагом будет создание функции запроса данных с сервера. Наше API смоделировано локально, значит будем делать имитацию серверного запроса с помощью Promise.

Наш Promise распознает категорию спорта, который запрашивает пользователь и подгружает с сервера необходимый набор матчей.

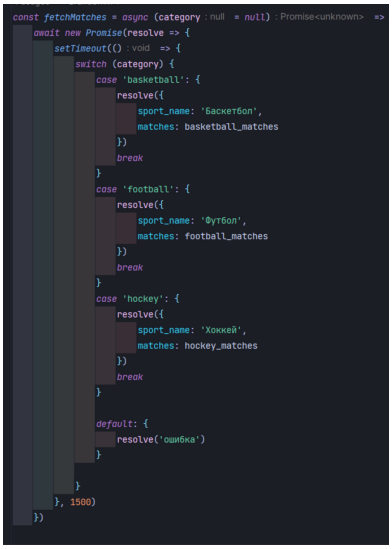

Рисунок 3. Запрос на сервер

После, с помощью Redux будет реализован Slice с подгружаемым массивом объектов, который позволит гибко изменять запрашиваемые данные и использовать их в любом компоненте. Также создан асинхронный «Thunk», с помощью которого проводится проверка работоспособности сервера и успешной загрузки данных.

На этом моменте стоит подробнее описать, как работает тестовый подсчет для процентов победы той или иной команды.

Каждый процент должен подсчитываться по определенной формуле (индивидуальной для каждой БК), используя статистику команды. Выводимый процент дает оценку и отображает вероятность победы той или иной команды.

При разработке тестовой версии веб-сервиса была использована упрощенная формула вероятности победы той или иной команды, где используется только один фактор:

$$p_1 = \frac{w_1}{w_1 + w_2},\ p_2 = \frac{w_2}{w_1 + w_2},$$

где p_i – вероятность победы i-ой команды, w_i – процент побед i-ой команды за все время.

По мере дополнения объектов матчей новыми факторами – формула может улучшаться и повышать точность прогнозирования. В будущем можно добавить такие факторы, как история встреч команд, таблица лиги, домашний/гостевой статус, травмы игроков и прочее. Формула для каждого прогноза или каждого веб-сервиса может быть абсолютно разной.

После того как все события обработаны, пользователь попадает на страницу с матчами по выбранному виду спорта, на котором он получает информацию о предстоящих матчах, местах проведения и шансах победы той или иной команды в каждом матче, которые подсчитываются сразу после загрузки данных с сервера.

4. Тестирование

На основной странице пользователь видит небольшую аннотацию и может выбрать интересующий его вид спорта. К примеру, на рисунке 4, пользователю дается выбор из баскетбола, футбола и хоккея.

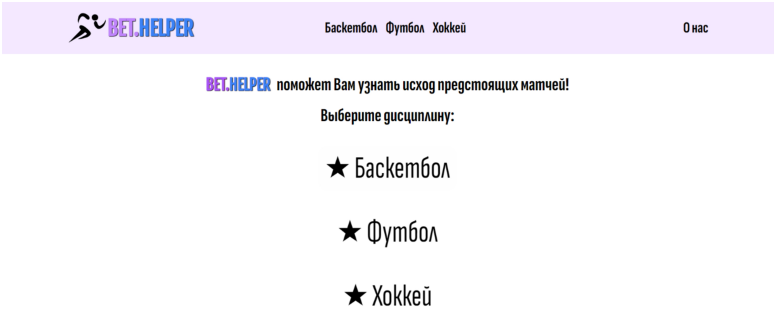

Рисунок 4. Основная страница веб-приложения

Более того, пользователь может в любой момент переключаться между видами спорта и почитать о проекте с помощью вкладок в отдельном меню сверху.

После выбора пользователем одного из видов спорта, подгружается страница с предстоящими матчами выбранного вида спорта. Во время загрузки данных с сервера пользователь будет видеть иконку загрузки, изображенную на рисунке 5.

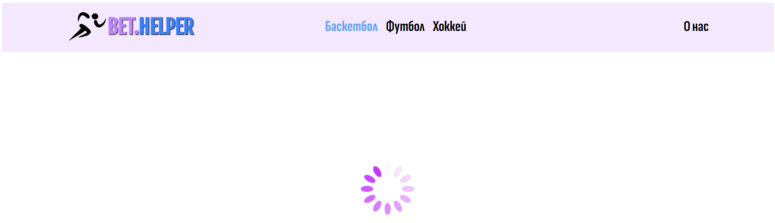

Рисунок 5. Загрузка страницы

После загрузки появится список предстоящих событий. Если, например, пользователь выбрал «Баскетбол», то он увидит предстоящие матчи баскетбольных команд, а также дату, место и процент победы команды, подсчитанные алгоритмами веб-приложением.

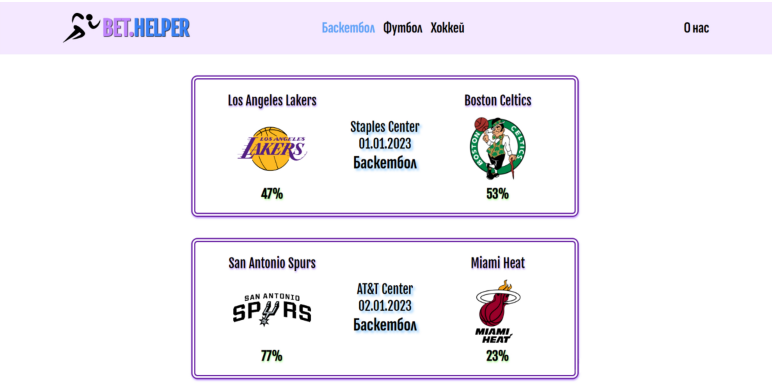

Рисунок 6. Страница с предстоящими матчами

5. Сложности, возникшие в процессе работы над проектом

Основная сложность, возникшая в процессе разработки – суть реализации искусственного API. В силу того, что API спортивных событий являются платными, пришлось реализовать свою искусственную модель данных, которые должны приходить со стороны Back-end'а. Более того, все еще стоит задача над улучшением формулы подсчета процентов победы той или иной команды в спортивном событии. Формула на данный момент является очень примитивной и нуждается в доработке.

Список литературы

1. Антанюк А. Букмекерская контора: что нужно знать до начала игры. — URL: https://legalbet.ru/shkola-bettinga/bukmekerskaya-kontora-chto-nuzhno-znat-do-nachala-igri/. Дата обращения: 02.06.2023.

2. Lindsay Kolowich Cox. Web Design 101: How HTML, CSS, and JavaScript Work. — URL: https://blog.hubspot.com/marketing/web-design-html-css-javascript. Дата обращения: 01.06.2023.

3. Официальная документация React Router. — URL: https://reactrouter.com/en/main. Дата обращения: 07.06.2023.

4. Официальная документация Redux. — URL: https://redux.js.org/. Дата обращения: 06.06.2023.

5. Официальная документация Redux Toolkit. — URL: https://redux-toolkit.js.org/. Дата обращения: 07.06.2023.

6. Официальный сайт React. — URL: https://react.dev. Дата обращения: 08.06.2023.

7. Официальный сайт Tailwind CSS. — URL: https://tailwindcss.com/. Дата обращения: 09.06.2023.

8. Официальный сайт WebStorm. — URL: https://www.jetbrains.com/webstorm/. Дата обращения: 09.06.2023.

9. Resources from Developers to Developers. — URL: https://developer.mozilla.org/ru. Дата обращения: 06.06.2023.

10. Тиленс Томас М. React в действии / Пер. с англ. — СПб. : Питер, 2019.

11. Флэнаган Д. JavaScript. Подробное руководство / Пер. с англ. — СПб. : Символ Плюс, 2008.

12. Хлыст В.В. Разработка web-сайта на основе Html с использованием JavaScript // Научно-практические исследования. — 2020. — № 1-2. — С. 155–161.