**ГОСУДАРСТВЕННОЕ ВЫСШЕЕ УЧЕБНОЕ ЗАВЕДЕНИЕ**

**ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Факультет компьютерных наук и технологий**

**Кафедра программной инженерии**

**Допущен к защите**

**Заведующий каф. ПИ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.И. Федяев**

(подпись, дата) (инициалы, фамилия)

**Пояснительная записка**

**к бакалаврской работе**

**на тему: «Тематический веб-ресурс по предоставлению информации спортивных событий»**

**со спецчастью ««Разработка информационного web-сайта с помощью языка JavaScript»**

**Выполнил: студент 4 курса, группы ПИ-15а**

**направления подготовки (специальности)**

**6.05010302 «Программная инженерия»**

**(шифр и название направления подготовки, специальности)**

**Горбатюк Эдуард Андрееевич**

**( имя, отчество, фамилия студента )**

**Руководитель \_\_\_\_\_\_\_\_\_\_\_Ищенко А. П**

**(подпись, дата) (инициалы, фамилия)**

**Консультант \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**(подпись, дата) (инициалы, фамилия)**

**Консультант \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**(подпись, дата) (инициалы, фамилия)**

**Нормоконтролер \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**(подпись, дата) (инициалы, фамилия)**

**Рецензент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**(подпись, дата) (инициалы, фамилия)**

**Донецк – 2019**

РЕФЕРАТ

*Изм.*

*№ документа*

*Лист*

*Подпись*

*Дата*

*Разработ.*

*Консульт.*

*Руководит.*

*Н. контр.*

*Зав. каф.*

*Буква*

*Лист*

*Листов*

*ФИО*

*А.П. Ищенко*

*О.И. Федяев*

##### ДонНТУ

*Кафедра ПИ*

*группа ИПО-11б*

#### Д05010302.52.02.076ДП

**Разработка информационного web-сайта с помощью языка «JavaScript**

Бакалаврская работа содержит: стр .: ?; рис .: ?; табл .: ?; доп .: ?.

Объектом исследования данной работы является тематический веб-ресурс по предоставлению информации спортивных событий.

Целью данной работы является разработка информационного web-сайта с помощью языка «JavaScript.

Результатом работы является разработанная программа, позволяющая просматривать информацию о спортивных события, а также статистики.

Рассчитана экономическая эффективность внедрения программы.

WEB-САЙТ, ИНТЕРНЕТ, HTML, CSS, JAVASCRIPT, NODEJS, СЕРВЕР, СПОРТИВНЫЕ СОБЫТИЯ, EXPRESS.

*Изм.*

*Лист*

*№ документа*

*Подпись*

*Дата*

#### Д05010302.52.02.076ДП

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 5](#_Toc8737349)

[1 ПОСТАНОВКА ОБЩЕЙ ЗАДАЧИ НА ПРОЕКТИРОВАНИЕ ПРОГРАММНОЙ СИСТЕМЫ 6](#_Toc8737350)

[1.2 Анализ существующих web-сайтов по предоставлению информации спортивных событий. 7](#_Toc8737351)

[2 РАЗРАБОТКА ТРЕБОВАНИЙ К ПРОГРАММНОМУ ПРОДУКТУ 10](#_Toc8737352)

[2.1 Требования к интерфейсу пользователя 10](#_Toc8737353)

[2.2 Системные требования 10](#_Toc8737354)

[2.3 Программные требования 11](#_Toc8737355)

[2.4 Требование к безопасности 12](#_Toc8737356)

[2.5 Общие требования к проекту 13](#_Toc8737357)

[3 ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ПРОЕКТА 14](#_Toc8737358)

[3.1 Архитектура программной системы 14](#_Toc8737359)

[3.2 Разработка шаблона сайта 16](#_Toc8737360)

[3.3 Диаграммы UML 17](#_Toc8737361)

[3.3.2 Диаграмма состояний 19](#_Toc8737362)

[3.3.3 Диаграмма последовательностей 20](#_Toc8737363)

[3.4 Средства разработки 21](#_Toc8737364)

[3.4.1 HTML 21](#_Toc8737365)

[3.4.2 CSS 22](#_Toc8737366)

[3.4.3 JavaScript 24](#_Toc8737367)

[4 ПОСТРОЕНИЕ БАЗ ДАННЫХ ПРОГРАММНОЙ СИСТЕМЫ 25](#_Toc8737368)

[4.1 Описание базы данных 25](#_Toc8737369)

[5 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ 28](#_Toc8737370)

[5.1 Описание основных модулей тематического веб-ресурса по предоставлению информации спортивных событий 28](#_Toc8737371)

[5.1.2 Разработка пользовательского интерфейса 29](#_Toc8737372)

[5.1.4 Возможности для посетителей сайта 33](#_Toc8737373)

[5.2 Описание реализации методов проекта 33](#_Toc8737374)

[5.2.1 Модуль работы «Регистрация» 33](#_Toc8737375)

*Изм.*

*Лист*

*№ документа*

*Подпись*

*Дата*

#### Д05010302.52.02.076ДП

[5.2.2 Модуль работы «Авторизация» и «Сессии» 38](#_Toc8737376)

[6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ ПРОГРАММНОЙ СИСТЕМЫ 41](#_Toc8737377)

[6.1 Назначение программного средства. 41](#_Toc8737378)

[6.2 Инструкция пользователя программным средством. 41](#_Toc8737379)

[7 ТЕСТИРОВАНИЕ ПРОГРАММНОЙ СИСТЕМЫ 47](#_Toc8737380)

[7.1 Выбор метода тестирования 47](#_Toc8737381)

[7.2 Разработка тестов 48](#_Toc8737382)

[ВЫВОДЫ 50](#_Toc8737383)

[ПЕРЕЧЕНЬ ССЫЛОК 52](#_Toc8737384)

[ПРИЛОЖЕНИЕ А 53](#_Toc8737385)

[ПРИЛОЖЕНИЕ Б 54](#_Toc8737386)

[ПРИЛОЖЕНИЕ В 55](#_Toc8737387)

[ПРИЛОЖЕНИЕ Д 87](#_Toc8737388)

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ,

*Изм.*

*Лист*

*№ документа*

*Подпись*

*Дата*

#### Д05010302.52.02.076ДП

СИМВОЛОВ, ЕДИНИЦ И ТЕРМИНОВ

JAVASCRIPT – язык программирования

UML – язык графического описания

СУБД – система управления базами данных

БД – база данных

MONGODB – документированная СУБД

HTML – язык разметки

CSS – каскадные таблицы стилей

NODEJS – программная платформа

EXPRESS – фреймворк веб-приложений NodeJs

ПО – программное обеспечение

# ВВЕДЕНИЕ

**Веб-сайт** - совокупность файлов, документов, отраженных при помощи языка программирования таким образом, чтобы их видели пользователи сети Интернет. Другими словами, сайты включают в себя любую текстовую, графическую, аудио- или видео- визуальную информацию, собранную на странице или нескольких страницах. Определяет уникальность того или иного веб-сайта, как правило, доменное имя.

В**еб-ресурс** - это некий узел или точка, в который включен специальный идентификатор, позволяющий при необходимости с легкостью отыскать нужную страницу на просторах сети Интернет. Чаще всего один домен соответствует одному веб-ресурсу, однако иногда бывает так, что на одном домене располагаются несколько веб-ресурсов, или же один веб-ресурс имеет для себя несколько доменов.

Сайт прямых спортивных трансляций представляет собой сервис онлайн результатов, итоговых результатов матчей с данными о забитых мячах, желтых и красных карточках, статистику очных встреч, а также представление детальной статистики (владение мячом, ударов всего/в створ, штрафные, угловые, фолы), составов, турнирных таблиц всего/дома/в гостях, форму команд. Также для важных матчей проводятся текстовые онлайн трансляции и публикация видео материалов с голами и важными моментами в матчах. На таких ресурсах отображаются вероятность победы той или иной команды. Данный вид веб-ресурсов очень распространен среди тех, кто следит за спортивными событиям.

# 1 ПОСТАНОВКА ОБЩЕЙ ЗАДАЧИ НА ПРОЕКТИРОВАНИЕ ПРОГРАММНОЙ СИСТЕМЫ

1.1 Формирование цели и описание объекта

Объект разработки - Тематический веб-ресурс по предоставлению информации спортивных событий.

Назначение веб-ресурса по предоставлению информации спортивных событий:

а) Представление пользователю онлайн результатов спортивных событий, а также результаты завершившихся матчей и расписание будущих.

б) Регистрация и добавление любимых лиг и клубов в список отслеживаемых.

в) Возможность добавлять любые игры в раздел избранных.

г) Представление статистики и информации отдельных спортивных событий.

Предмет разработки – методы разработки динамических сайтов с помощью редактора исходного кода – Visual Studio Code с использованием языка программирования JavaScript.

Целью выпускной квалификационной работы является разработка информационного web-сайта.

Для достижения цели выпускной квалификационной работы необходимо выполнить следующие задачи:

1. сбор и анализ данных о подобных web-сайтах по предоставлению информации спортивных событий, требований к внешнему виду и функциональности сайта;
2. ознакомление с технологиями редактора исходного кода «Visual Studio Code», языком разметки HTML и языком программирования JavaScript;
3. формирование структуры сайта;
4. разработка проекта дизайна;
5. разработка удобной навигации;
6. рассмотреть вопросы охраны труда при эксплуатации ЭВМ.

## 1.2 Анализ существующих web-сайтов по предоставлению информации спортивных событий.

В данный момент существует множество сайтов, которые предоставляют спортивную информацию. Суть каждого из них состоит в том, что они посредством API получают информацию от специальных систем и отображают ее на своих площадках.

Наиболее популярным из русскоязычных является спортивный сайт «Чемпионат», который предназначен для просмотра новостей о предстоящих и прошедших спортивных событий, а также результатов топ лиг Европы. Предоставляет следующие функции:

1. предоставление новостей, видео и фото материалов;
2. предоставление онлайн результатов популярных видов спорта, а также результаты прошедших событий и расписание будущих;
3. предоставление рейтингов УЕФА и ФИФА;
4. возможность выбора турниров в раздел «Мои турниры»;
5. возможность поиска турниров и любых спортивных событий;
6. возможность сортировать спортивные события и турниры по важности.

Данный сайт предоставляет всю необходимую информацию, чтобы пользователь оставался в курсе всех спортивных событий.

Доступ к сайту осуществляется по адресу - <https://www.championat.com>

Главная страница данного сайта расположена ниже (см. рис. 1.1)

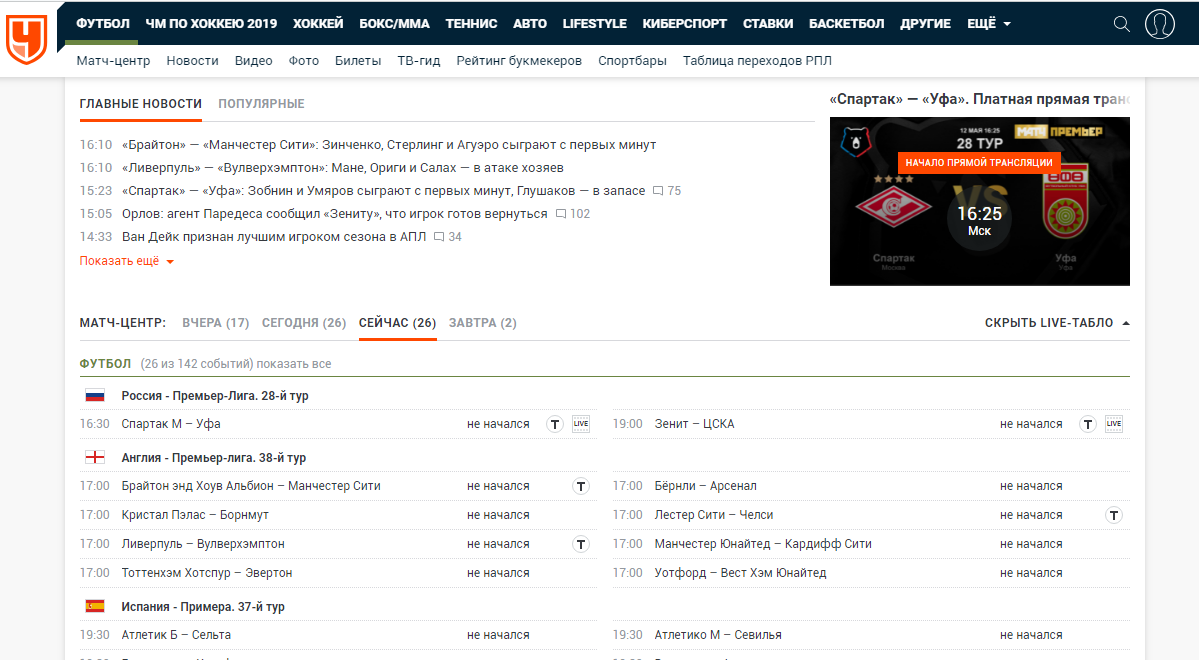


Рисунок 1.1 – Главная страница сайта «Чемпионат»

Еще одним популярным сайтом по предоставлению онлайн результатов спортивных событий является «livescore». Данный сайт предоставляет следующие функции:

1. предоставление онлайн результатов практически всех спортивных событий в мире;
2. предоставление результатов недельной давности, а также расписание будущих на неделю вперед;
3. предоставление текстовых онлайн трансляций;
4. предоставление статистики;
5. предоставление турнирных таблиц в онлайн режиме;
6. возможность добавления игр в раздел «Мои игры» без регистрации;
7. возможность смены языка;
8. возможность изменения временной зоны.

Данный сайт очень прост в использовании, однако предоставляет очень много информации о спортивных событиях.

Доступ к сайту осуществляется по адресу - <https://www.livescore.com>

Главная страница данного сайта расположена ниже (см. рис. 1.2)

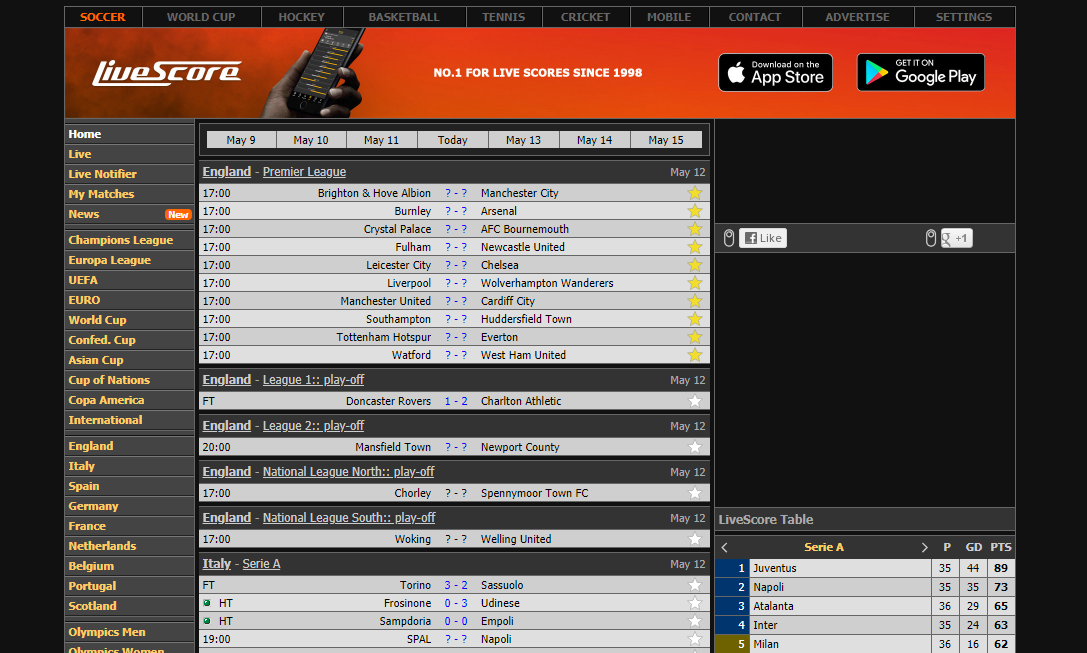


Рисунок 1.2 - Главная страница сайта «livescore»

# 2 РАЗРАБОТКА ТРЕБОВАНИЙ К ПРОГРАММНОМУ ПРОДУКТУ

Тематический веб-ресурс по предоставлению информации спортивных событий представляет собой отображение данных с футбольных матчей в реальном времени, с свою очередь данные поставляются посредством API.

## 2.1 Требования к интерфейсу пользователя

Основные требования к пользовательскому интерфейсу:

1. простота;
2. эстетичный дизайн;
3. изучаемость и запоминаемость;
4. эффективность;
5. минимальное количество ошибок.

Для достижения данных требований требуется следующее:

1. организация навигации:
2. размещение кликабельного логотипа;
3. размещение на каждой странице наименование раздела;
4. необходимо обозначить в меню тот раздел, где находится пользователь.

б) контент:

1. размещение на главной странице актуальной информации;
2. использование шрифтов без засечек и размером 12px;
3. хороший контраст между фоном и текстом.

## 2.2 Системные требования

Работоспособность сайта осуществляется на всех операционных система, в том числе на Linux и Microsoft Windows. Сервер, на котором размещается данный сайт, должен поддерживать Node.js, популярной платформы JavaScript в серверной части. Однако рекомендуется использовать VDS (Virtual Dedicated Server, виртуальный выделенный сервер) на который владелец-администратор может сам установить Node.js и MongoDB. На виртуальных серверах доступны следующие операционные системы:

а) debian;

б) centOS;

в) ubuntu;

г) freeBSD;

д) windows.

Минимальные системные требования:

а) node.js v10.15.3 и новее;

б) mongoDB v3.5.2 и новее.

## 2.3 Программные требования

Клиентская часть:

На клиентском рабочем месте требуется только наличие браузера с поддержкой Ajax. Поддерживаются следующие браузеры:

1. opera;
2. internet explorer;
3. google chrome;
4. mozilla firefox;
5. apple safari;
6. orbitum.

Серверная часть:

Для работы требуются следующие компоненты:

а) bcrypt-nodejs v0.0.3;

б) body-parser v1.19.0;

в) dotenv v8.0.0;

г) ejs v2.6.1;

д) express v4.16.4;

е) gulp v4.0.1;

ж) jquery v3.4.1;

з) mongoose v5.5.5;

и) node-fetch v2.5.0;

к) request v2.88.0.

## 2.4 Требование к безопасности

Требования к надежности и сохранности информации сайта:

1. cервер, на котором расположен cайт, должен обеспечивать ежедневное создание полной или инкрементальной резервной копии базы данных cайта;
2. cервер, на котором расположен cайт, должен иметь резервный канал доступа в сеть интернет и блок бесперебойного питания, обеспечивающие функционирование сайта в течение 2 часов с момента наступления аварийной ситуации [3].

Требования к безопасности и защите информации сайта.

Информация сайта должна быть расположена в закрытой БД, доступ к которой разрешен только из программных модулей сайта и только с использованием пароля доступа:

1. доступ к закрытым областям сайта разрешен только пользователям, прошедшим процедуру регистрации;
2. пароли пользователей должны храниться в виде хэшей;
3. после регистрации, на почту пользователя отправляется письмо с уникальным кодом, который необходим для его активации и полной авторизации.

## 2.5 Общие требования к проекту

Основные требования к проекту:

а) адаптивность сайта;

б) обновление информации каждую минуту;

в) возможность регистрации и авторизации;

г) наличие таких разделов, как:

1) «Мои лиги»;

2) «Мои клубы»;

3) «Страны».

д) проект должен быть написан на языке JavaScript, а также фреймворков для Node.js;

е) программное изделие должно удовлетворять следующим общим требованиям:

1) выполнять контроль над корректным вводом данных;

2) иметь простой и понятный интерфейс;

3) шифрование паролей пользователей;

4) обрабатывать исключительные ситуации и вывод соответствующего сообщения об ошибке.

# 3 ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ПРОЕКТА

Тематический веб-ресурс по предоставлению информации спортивных событий был реализован с помощью таких средств разработки:

а) HTML;

б) CSS;

в) JavaScript.

## 3.1 Архитектура программной системы

Архитектура программной системы представляет собой комплекс решений и исследований в процессе создания, цель которой – облегчить пользователю работу с ним.

Исходя из этого тематический веб-ресурс по предоставлению информации спортивных событий должен иметь удобный интерфейс для пользователя, чтобы он мог увидеть из каких разделов состоит сайт, и соответственно какая информация содержится в каждом из них.

Архитектура сайта состоит из клиентской части, для отображения всей информации на сайте и серверной части, которая отвечает за функционал и реагирует на запросы от клиентской части (см. рис. 3.1).

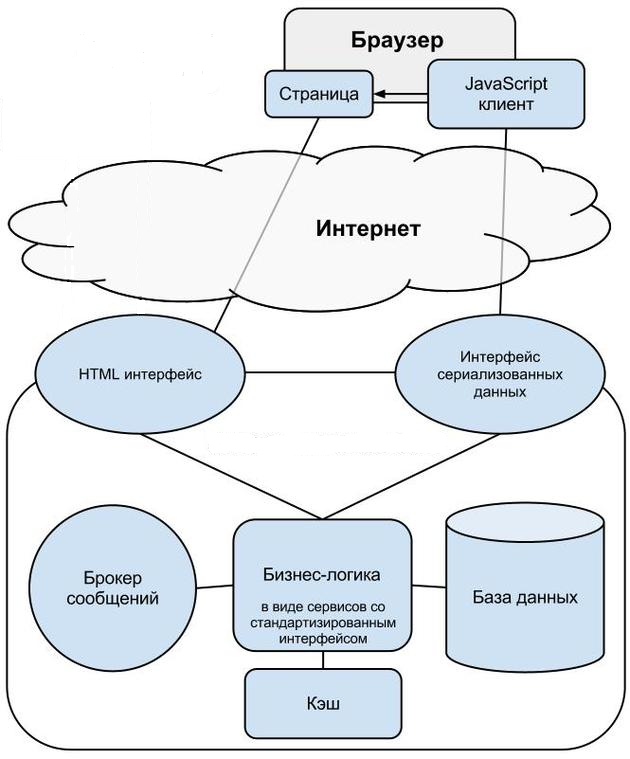


Рисунок 3.1 – Архитектура тематического веб-ресурса по предоставлению информации спортивных событий

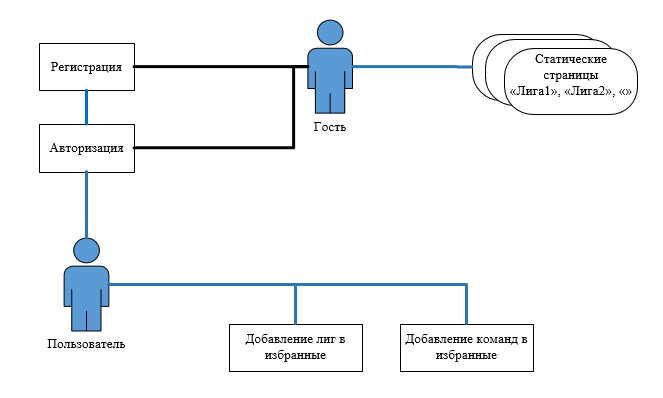


Рисунок 3.2 - Архитектура и БД тематического веб-ресурса по предоставлению информации спортивных событий

Сайт разрабатывался в редакторе исходного кода «Visual Studio Code» на языке JavaScript и с помощью node.js, программной платформы, которая транслирует JavaScript в машинный код. Принципиальных качественных преимуществ у node.js перед остальными языками нет, как и недостатков.

## 3.2 Разработка шаблона сайта

Шаблон сайта требуется для формирования внешнего вида сайта и определения формы отображения информации на сайте, также шаблоны упрощают разработку сайтов (см. рис. 3.3)

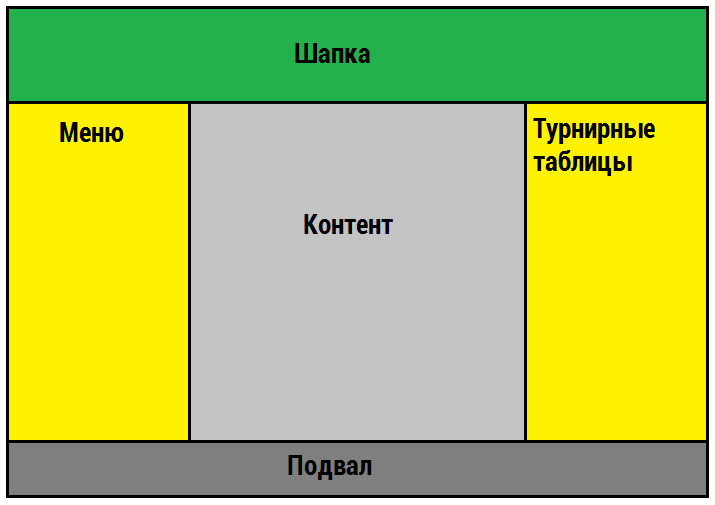


Рисунок 3.3 – Шаблон страницы сайта

Шаблон страницы сайта включает в себя:

а) шапка сайта – размещение кликабельного логотипа, кнопок входа/регистрации и настроек;

б) меню сайта – размещение ссылок на другие страницы сайта;

в) контент страницы – размещение содержимого страницы на которой находится пользователь;

г) левая часть – содержит турнирные таблицы турниров;

д) подвал сайта – содержит информацию правообладателя.

Данный шаблон предназначен для отображения информации для всех страниц сайта, в которых в зависимости от страницы изменяется лишь контент. Контент других страниц содержит в себе:

а) отображение результатов матчей по дням (онлайн, завершенные, будущие);

б) информация о лигах;

в) информация о клубах.

## 3.3 Диаграммы UML

UML - язык [графического](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B7%D1%83%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) описания для [объектного моделирования](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE%D0%B5_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1) в области [разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F), для [моделирования](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) [бизнес-процессов](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81), [системного проектирования](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и отображения организационных структур. UML является языком широкого профиля, это — [открытый стандарт](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%B9_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82), использующий графические обозначения для создания [абстрактной модели](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%82%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C) [системы](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), называемой UML-моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. UML не является языком программирования, но на основании UML-моделей возможна [генерация кода](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4%D0%BE%D0%B3%D0%B5%D0%BD%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D1%8F).

3.3.1 Диаграмма вариантов использования

Основное назначение диаграммы — описание функциональности и поведения, позволяющее [заказчику](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BA%D0%B0%D0%B7%D1%87%D0%B8%D0%BA), [конечному пользователю](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D0%B5%D1%87%D0%BD%D1%8B%D0%B9_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C) и [разработчику](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%81%D1%82) совместно обсуждать проектируемую или существующую [систему](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_(%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80)). При моделировании системы с помощью диаграммы прецедентов [системный аналитик](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D1%82%D0%B8%D0%BA) стремится:

а) чётко отделить систему от её окружения;

б) определить действующих лиц (актёров), их взаимодействие с системой и ожидаемую функциональность системы;

в) определить в глоссарии предметной области понятия, относящиеся к детальному описанию функциональности системы (то есть прецедентов).

Работа над диаграммой может начаться с текстового описания, полученного при работе с заказчиком. При этом нефункциональные требования (например, конкретный язык или система программирования) при составлении модели прецедентов опускаются (для них составляется другой документ).

Диаграмма вариантов использования изображена на рисунке 3.4.

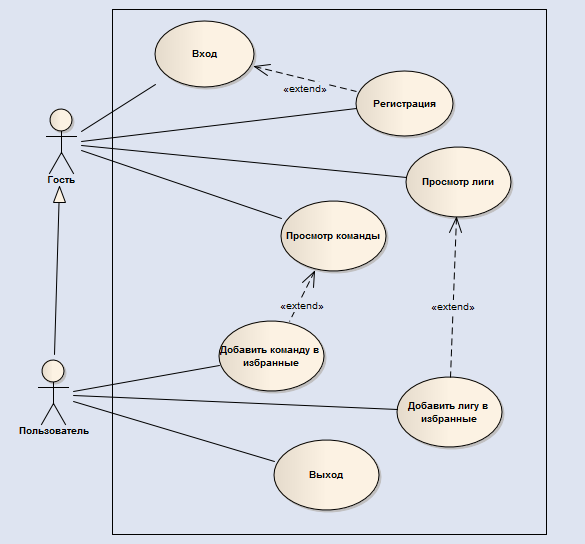


Рисунок 3.4 – Диаграмма вариантов использования

### 3.3.2 Диаграмма состояний

Диаграмма состояний — это, по существу, [диаграмма состояний](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_%D1%81%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D0%B9_(%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%BE%D0%B2)) из теории автоматов со стандартизированными условными обозначениями, которая может определять множество систем от компьютерных программ до [бизнес-процессов](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81).

Диаграмма состояний изображена на рисунке 3.5.

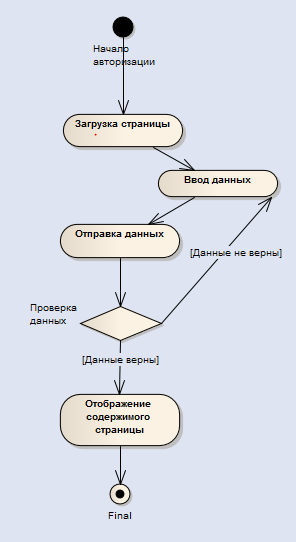


Рисунок 3.5 – Диаграмма состояний

### 3.3.3 Диаграмма последовательностей

Диаграмма последовательности - [диаграмма](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), на которой для некоторого набора объектов на единой временной оси показан жизненный цикл какого-либо определённого объекта (создание-деятельность-уничтожение некой сущности) и взаимодействие актеров (действующих лиц) ИС в рамках какого-либо определённого прецедента (отправка запросов и получение ответов).

Диаграмма последовательностей изображена на рисунке 3.6

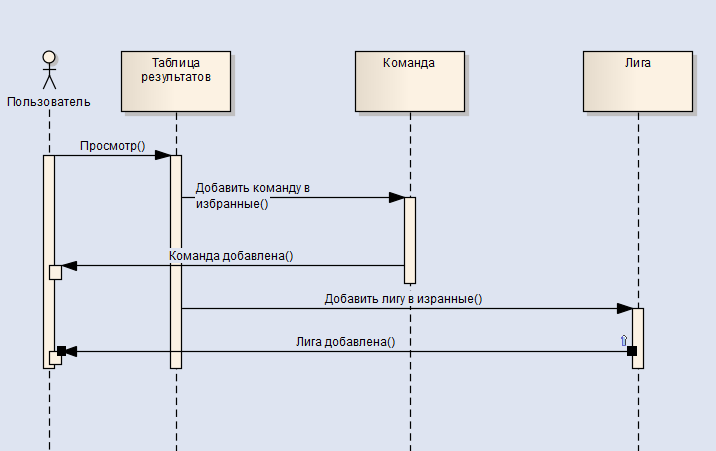


Рисунок 3.6 – Диаграмма последовательностей

## 3.4 Средства разработки

### 3.4.1 HTML

HTML (HyperText Markup Language) — стандартный язык разметки гипертекстовых страниц в Интернете. Есть и другие языки разметки гипертекста, но большая часть страниц сайтов Интернета размечена именно на языке HTML. Такие страницы успешно интерпретируются браузерами, которые отображают их на экранах различных электронных устройств в удобном для человека виде.

HTML является теговым языком разметки гипертекста: чтобы превратить текст в гипертекст, используют разделители (дескрипторы), для краткости названные тегами. Вот пример тега: <strong> — этот открывающий тег обеспечивает вывод текста жирным шрифтом до тех пор, пока не встретится закрывающий тег </strong>.

Начиная создавать сайт с помощью WordPress, достаточно знать, что сайт создаётся на языке HTML, но со временем наверняка придётся хотя бы слегка освоить язык HTML.

Вот для примера достаточно часто встречающаяся ситуация: Вы пишите страницу текста и решили включить в неё фрагмент чужого текста, корректно пометив этот фрагмент, как цитату, чтобы поисковые системы не сочли Вас плагиатором.

Перейдя из режима Визуально в режим Текст, с ужасом обнаруживаете в этом окне HTML-тегов больше, чем самого текста. Это свидетельство того, что текст прогулялся по многим сайтам, на каждом текст пытались дополнить, усилить, а в результате получился жуткий HTML-код.

Можно вручную убрать лишние теги, а лучше скопировать текст в окне Визуально и вставить его в окне Текст - в результате этого простейшего действия все теги будут потеряны, что сэкономит много времени.

### 3.4.2 CSS

**CSS**(англ. Cascading Style Sheets — каскадные таблицы стилей) — технология описания внешнего вида документа, оформленного языком разметки.

Преимущественно используется как средство оформления веб-страниц в формате HTML и XHTML, но может применяться с любыми видами документов в формате XML, включая SVG и XUL.

Каскадные таблицы стилей используются создателями веб-страниц для задания цветов, шрифтов, расположения и других аспектов представления веб-документа. Основной целью разработки CSS являлось разделение содержимого (написанного на HTML или другом языке разметки) и оформления документа (написанного на CSS). Это разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом. Кроме того, CSS позволяет представлять один и тот же документ в различных стилях или методах вывода, таких как экранное представление, печать, чтение голосом (специальным голосовым браузером или программой чтения с экрана), или при выводе устройствами, использующими шрифт Брайля.

Каскадные таблицы стилей (Cascading Style Sheets, CSS) — это стандарт, определяющий представление данных в браузере. Если HTML предоставляет информацию о структуре документа, то таблицы стилей сообщают как он должен выглядеть.

Стиль — это совокупность правил, применяемых к элементу гипертекста и определяющих способ его отображения. Стиль включает все типы элементов дизайна: шрифт, фон, текст, цвета ссылок, поля и расположение объектов на странице.

Таблица стилей — это совокупность стилей, применимых к гипертекстовому документу.

Каскадирование — это порядок применения различных стилей к веб-странице. Браузер, поддерживающий таблицы стилей, будет последовательно применять их в соответствии с приоритетом: сначала связанные, затем внедренные и, наконец, встроенные стили. Другой аспект каскадирования — наследование (inheritance), — означает, что если не указано иное, то конкретный стиль будет применен ко всем дочерним элементами гипертекстового документа. Например, если вы примените определенный цвет текста в теге <div>, то все теги внутри этого блока будут отображаться этим же цветом.

Использование каскадных таблиц дает возможность разделить содержимое и его представление и гибко управлять отображением гипертекстовых документов путем изменения стилей.

### 3.4.3 JavaScript

[JavaScript](https://developer.mozilla.org/ru/docs/%D0%A1%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C/JavaScript) - это полноценный [динамический язык программирования](https://developer.mozilla.org/ru/docs/%D0%A1%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C/Dynamic_programming_language), который применяется к [HTML](https://developer.mozilla.org/ru/docs/%D0%A1%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C/HTML) документу, и может обеспечить динамическую интерактивность на веб-сайтах. Его разработал Brendan Eich, сооснователь проекта Mozilla, Mozilla Foundation и Mozilla Corporation.

JavaScript невероятно универсален. Можно начать с малого, с простых функций, таких как карусели, галереи изображений, изменяющиеся макеты и отклик на нажатие кнопок. Обладая большим опытом, можно создавать игры, анимированную 2D и 3D графику, полномасштабные приложения с базами данных и многое другое!

JavaScript сам по себе довольно компактный, но очень гибкий. Разработчиками написано большое количество инструментов поверх основного языка JavaScript, которые разблокируют огромное количество дополнительных функций с очень небольшим усилием. К ним относятся:

а) Программные интерфейсы приложения ([API](https://developer.mozilla.org/ru/docs/%D0%A1%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C/API)), встроенные в браузеры, обеспечивающие различные функциональные возможности, такие как динамическое создание HTML и установку CSS стилей, захват и манипуляция видеопотоком, работа с веб-камерой пользователя или генерация 3D графики и аудио сэмплов;

б) Сторонние API позволяют разработчикам внедрять функциональность в свои сайты от других разработчиков, таких как Twitter или Facebook;

в) Также можно применить к HTML сторонние фреймворки и библиотеки, что позволит ускорить создание сайтов и приложений.

# 4 ПОСТРОЕНИЕ БАЗ ДАННЫХ ПРОГРАММНОЙ СИСТЕМЫ

## 4.1 Описание базы данных

Для создаваемого сайта вся информация, требуемая для наполнения, поставляется посредством API, однако на сайте есть возможность регистрации и авторизации. Чтобы хранить данные пользователей было решено использовать MongoDB, которая в отличие от реляционных баз данных предлагает документно-ориентированную модель данных, благодаря чему работает быстрее и проста в использовании. Если в традиционном мире SQL есть таблицы, то в мире MongoDB есть коллекции. И если в реляционных БД таблицы хранят однотипные жестко структурированные объекты, то в коллекции могут содержать самые разные объекты, имеющие различную структуру и различный набор свойств.

В итоге была спроектирована база данных с названием EGScoreDB, которая содержит две коллекции:

а) users;

б) sessions.

Для того, чтобы работать с базой данных созданной с помощью MongoDB в нашем проекте, требуется установить Mongoose - это библиотека JavaScript, позволяющая определять схемы со строго-типизированными данными. Mongoose предоставляет огромный набор функциональных возможностей для создания и работы со схемами. На данный момент Mongoose содержит восемь SchemaTypes (типы данных схемы), которые может иметь свойство, сохраняемое в MongoDB. Эти типы следующие:

1. String;
2. Number;
3. Date;
4. Buffer;
5. Boolean;
6. Mixed;
7. ObjectId (\* уникальный идентификатор объекта, первичный ключ, \_id);
8. Array.

После установки всех необходимых компонентов была определена схема в файле user.js (см. рис. 4.1)

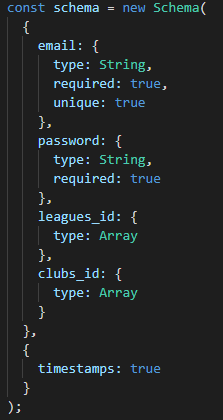


Рисунок 4.1 – Схема User

Данная схема содержит следующие свойства:

а) «email» - почта пользователя; тип данных – String; содержит уникальные данные; обязателен для заполнения;

б) «password» - пароль пользователя; тип данных – String; обязателен для заполнения;

в) «league\_id» - массив идентификаторов избранных лиг; тип данных – Array;

г) «club\_id» - массив идентификаторов избранных лиг; тип данных – Array;

д) «timestamps» - идентификатор, указывающий на добавление в коллекцию даты создания и модификации; тип данных – Boolean.

Данная схема используется при создании модели User.

# 5 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

## 5.1 Описание основных модулей тематического веб-ресурса по предоставлению информации спортивных событий

Общая схема функционирования программы заключается в следующем: при запуске программы пользователь видит главную страницу сайта, содержащий боковое меню с возможностью перехода в другие разделы, раздел с любой из представленных лиг и основную часть страницы, содержащей контент. Пользователь может свободно перемещаться по разделам и знакомиться со всеми имеющимися материалами и пользоваться ими. Схема функционирования сайта показана на рисунке 5.1.

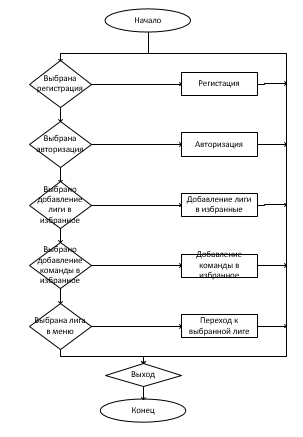


Рисунок 5.1 – Схема функционирования сайта

### 5.1.2 Разработка пользовательского интерфейса

В качестве разработки пользовательского интерфейса был выбран «Процедурно-ориентированные интерфейс»

Процедурно-ориентированные интерфейсы используют традиционную модель взаимодействия с пользователем, основанную на понятиях «процедура» и «операция». В рамках этой модели программное обеспечение предоставляет пользователю возможность выполнения некоторых действий, для которых пользователь определяет соответствующие данные и следствием выполне­ния которых является получение желаемых результатов.

Процедурно-ориентированный интерфейс:

1. обеспечивают пользователю функции, необходимые для выполнения задач;
2. акцент делается на задачи;
3. пиктограммы представляют приложения, окна или операции;
4. содержание папок и справочников отражается с помощью таблицы-списка.

Включают в себя:

1. примитивный;
2. меню;
3. свободная навигация.

В нашем случае используется типы «Меню» и «Свободная навигация»

Интерфейс-меню позволяет пользователю выбирать необходимые операции из специального списка, выводимого ему программой. Эти интер­фейсы предполагают реализацию множества сценариев работы, последовательность действий в которых определяется пользо­вателем.

Интерфейсы со свободной навигацией также называют графи­ческими пользовательскими интерфейсами (GUI — Graphic User Interface) или интерфейсами WYSIWYG (What You See IsWhat You Get — «что видишь, то и получишь»). Эти названия подчер­кивают, что интерфейсы данного типа ориентированы на ис­пользование экрана в графическом режиме с высокой раз­решающей способностью.

Графические интерфейсы поддерживают концепцию ин­терактивного взаимодействия с программным обеспечением, осуществляя визуальную обратную связь с ним и возможность прямого манипулирования объектами и информацией на экране.

В отличие от интерфейса-меню интерфейс со свободной навигацией обеспечивает возможность осуществления любых допустимых в конкретном состоянии операций и событий, доступ к которым возможен через различные интерфейсные компо­ненты.

Например, окна программ, реализующих интерфейс Windows, обычно содержат:

1. меню различных типов: ниспадающее, кнопочное, кон­текстное;
2. разного рода компоненты ввода данных.

Существенной особенностью интерфейсов данного типа явля­ется способность их изменяться в процессе взаимодействия программного обеспечения с пользователем, предлагая выбор только тех операций, которые имеют смысл в конкретной си­туации.

Пользовательский интерфейс представлен в виде взаимосвязанных между собой ejs-страниц, логично разделенных на несколько разделов в зависимости от обрабатываемых данных.

5.1.2.1 Описание страниц сайта

При открытии любой страницы сайта авторизация не требуется, поскольку доступ для посетителей открыт.

Главная страница (index.ejs) содержит шапку сайта, которая включает в себя кликабельный логотип, кнопки регистрации и авторизации, кнопку настройки. Также приведены навигация для других страниц сайта, которые содержат соответствующую разделам информацию. Все эти элементы подключены к остальным страницам сайта с помощью ejs-функции "include", и находятся в корневом каталоге нашего сайта в качестве блоков для страниц. В главной части сайта содержатся результаты текущих игр в режиме онлайн, расписание будущих игр и результаты прошедших матчей (см. рис. 5.2).

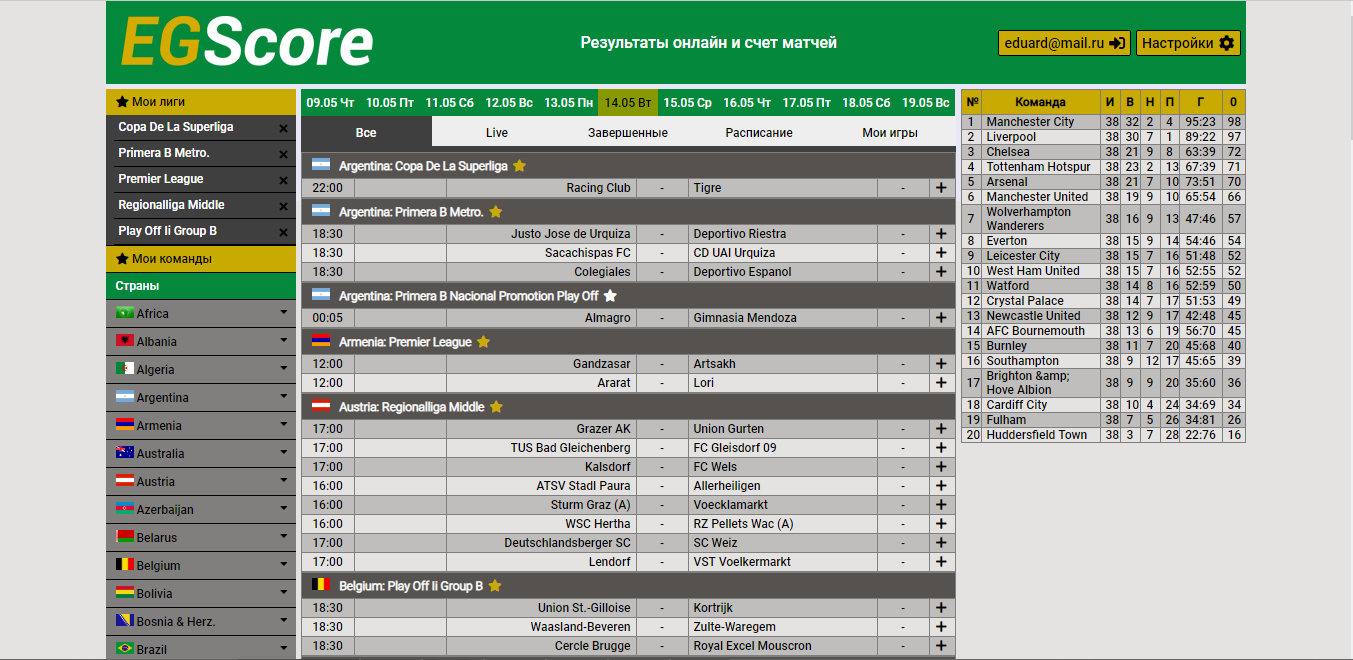


Рисунок 5.2 – Главная страница сайта

Каждая страница сайта содержит меню с переходами на следующие разделы сайта.

Подробнее о наших страницах:

а) раздел «Лига» содержит информацию о выбранной лиге, а именно:

1) результаты;

2) календарь;

3) таблица;

4) команды;

5) архив.

Доступ к данному разделу осуществляется по адресу «egscore.com/”country\_name”/”league\_name ”» (см. рис. 5.3)

Рисунок 5.3 – Страница лиги «Лиги1»(добавлю позже)

б) раздел «команда» содержит информацию о выбранной команде, а именно:

1) результаты;

2) календарь;

3) состав.

Доступ к данному разделу осуществляется по адресу «egscore.com/”country\_name”/”club\_name ”» (см. рис. 5.4)

Рисунок 5.4 – Страница команды «Команда1»(добавлю позже)

в) раздел «матч» содержит информацию о выбранной игре, а именно:

1) обзор матча;

2) статистика;

3) составы;

4) H2H;

5) таблица/сетка.

Доступ к данному разделу осуществляется по адресу «egscore.com/match/?id=”номер выбранного матча”» (см. рис. 5.5)

Рисунок 5.5 – Страница команды «Матч1»(добавлю позже)

### 5.1.4 Возможности для посетителей сайта

Посетитель сайта просмотрев все страницы сайта ознакомится со всей доступной информацией.

## 5.2 Описание реализации методов проекта

### 5.2.1 Модуль работы «Регистрация»

Регистрация на сайте – это действия, направленные на создание личной учетной записи на web-ресурсе, с целью получения доступа к его полному функционалу (будь то расширенные возможности или доступ к страницам Интернет-ресурсов, просматривать которые возможно лишь после регистрации).

То есть, регистрация – это способ входа (или получение возможности входа) на определенный web-ресурс. В большинстве случаев это процесс обязательный, особенно это касается социальных сетей, где возможности пользователей без регистрации ограничены.

При регистрации используется функция jquery.ajax(), которая осуществляет запрос к серверу без перезагрузки страницы. Окно регистрации изображено на рисунке 5.6.

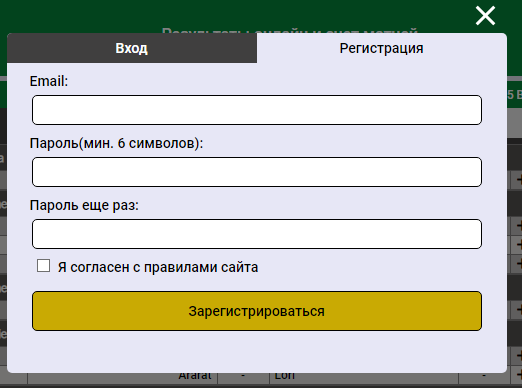


Рисунок 5.5 – Окно регистрации

Ajax-запрос имеет тип «POST» и отправляет данные по адресу «egscore.com/api/auth/register» в формате JSON (см. рис. 5.6)

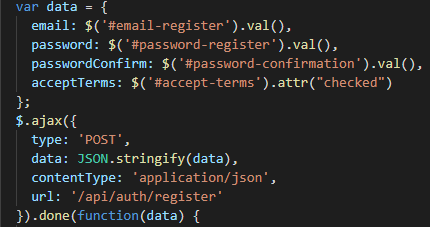


Рисунок 5.6 – Запрос отсылаемый на сервер при нажатии кнопки «Регистрация»

В случае, если поля были неверно заполнены сервер пришлет сообщение об ошибке, которое выведется на экран. (см. рис. 5.7-5.9)

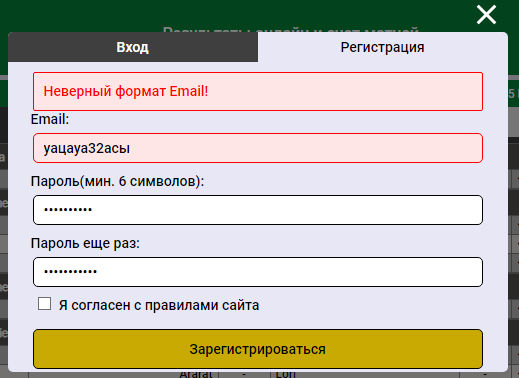


Рисунок 5.7 – Сообщение о неверном формате почты

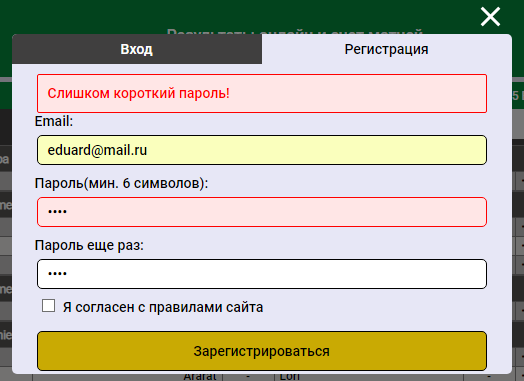


Рисунок 5.8 – Сообщение о коротком пароле

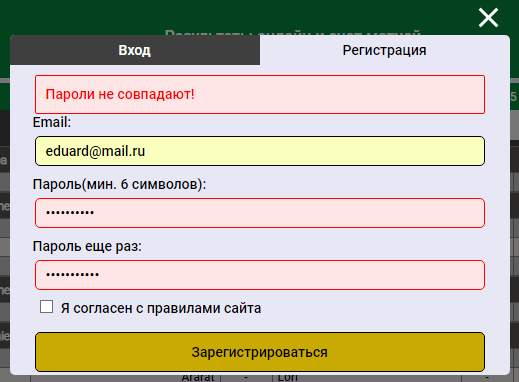


Рисунок 5.9 – Сообщение о не совпадении паролей

После того, как все данные будут правильно введены, на сервере в первую очередь произойдет сравнение введенного email по базе данных, чтобы не допустить повтора почты. Если почта уникальна, пароль хэшируется и шифруется с помощью [UNIX библиотеки bcrypt](https://en.wikipedia.org/wiki/Bcrypt). После чего в коллекцию mongodb добавляется новый пользователь и пользователю приходит сообщение об успешной регистрации (см. рис. 5.10)

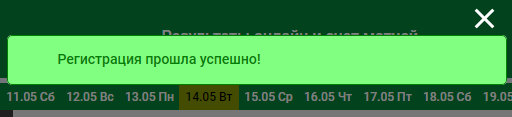


Рисунок 5.10 – Сообщение об успешной регистрации

Алгоритм реализации «Регистрация» представленныйна рис. 5.11.

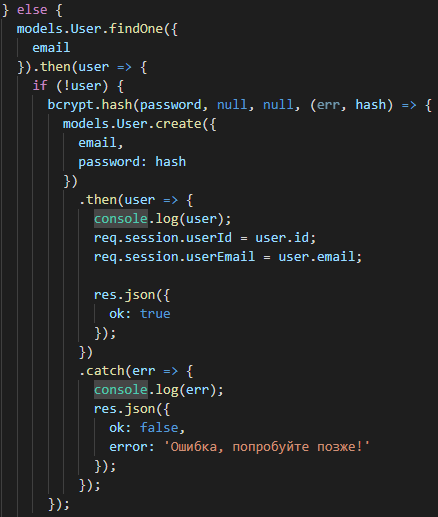


Рисунок 5.11 – Реализация «Регистрации»

### 5.2.2 Модуль работы «Авторизация» и «Сессии»

Авторизацию не следует путать с [аутентификацией](https://ru.wikipedia.org/wiki/%D0%90%D1%83%D1%82%D0%B5%D0%BD%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F) — процедурой проверки [легальности](https://ru.wikipedia.org/wiki/%D0%9B%D0%B5%D0%B3%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9) пользователя или данных, например, проверки соответствия введённого пользователем пароля к учётной записи паролю в базе данных, или проверка цифровой подписи письма по ключу шифрования, или проверка контрольной суммы файла на соответствие заявленной автором этого файла. Авторизация же производит контроль доступа к различным ресурсам системы в процессе работы легальных пользователей после успешного прохождения ими аутентификации.

Авторизация на нашем сайте не имеет большого значения, однако делает использование сайта удобным при повторном посещении. Авторизация проходит следующим образом, после заполнения полей email и пароль и при нажатии на кнопку «вход» на сервер отправляется запрос ajax (тоже самое, что и при регистрации). На сервере проверяется есть ли такой email в базе, если имеется то с помощью функции bcrypt.compare() сравниваются хэши введенного пароля с паролем пользователя. После успешного сравнения пользователь авторизуется и страница перезагрузится.

Пользовательская сессия является более высокоуровневой абстракцией, чем HTTP-сессия. С помощью нее можно не только идентифицировать разных пользователей, но также хранить произвольные данные на каждого пользователя в рамках его сессии. Типичный пример это корзина товаров в интернет-магазине. Для пользовательской сессии не обязательно логиниться (выполнять аутентификацию) на сайте.

Поддержка сессий обычно реализуется с помощью специальных библиотек в рамках используемого фреймворка. В задачи этих библиотек входит:

а) установка соединения, то есть отправка специальной куки, которая содержит идентификатор сессии. Имя этой куки фиксированно и задается на этапе старта приложения;

б) сохранение и извлечение данных из сессии. Этот пункт сильно зависит от используемого фреймворка. В случае express предоставляется специальный объект req.session, в который можно записывать необходимую информацию и читать ее в следующих запросах. Отдельный интерес представляет хранилище данных сессии. Это можно делать в памяти, прямо в куках (в зашифрованном виде), или в различных серверных хранилищах начиная от файлов, заканчивая базами данных;

в) Завершение сессии.

Так как серверная часть у нас написана на фреймворке node.js – express.js, то все данные сессии хранятся в объекте req.session (см. рис. 5.12)

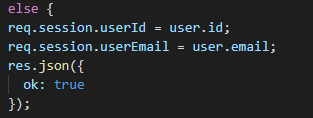


Рисунок 5.12 – Объект req.session

Алгоритм реализации «Авторизация» представленный на рис. 5.13.



Рисунок 5.13 – Реализация «Авторизации»

# 6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ ПРОГРАММНОЙ СИСТЕМЫ

## 6.1 Назначение программного средства.

Данный программный продукт предназначен для просмотра результатов спортивных событий, а также просмотра статистики.

## 6.2 Инструкция пользователя программным средством.

Для того чтобы зайти на сайт вам необходимо:

а) ввести адрес нашего сайта «egscore.com» в браузере;

б) для работы в системе вы можете воспользоваться любым браузером.

Главная страница сайта (см. рис. 6.1)

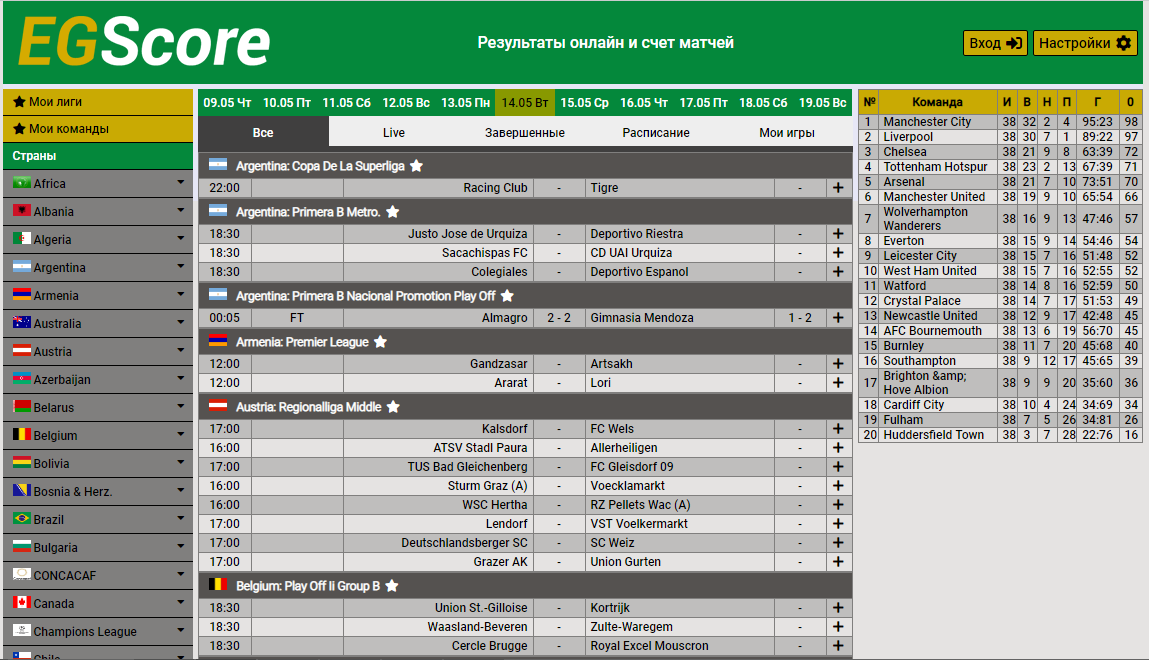


Рисунок 6.1 – Главная страница сайта

После загрузки главной страницы, можно просматривать все спортивные события на текущий день, а также матчи, которые идут в данный момент, расписание будущих матчей и результаты завершившихся матчей. Реализована возможность просматривать результаты игр за последние пять дней и просмотр расписания матчей на пять дней вперед. Если пользователя интересует определенная лига или команда, он может добавить лигу или команду в список избранных, так избранные лиги будут отображаться в самом вверху, а избранные команды во вкладке «Мои игры». Однако если пользователь не выполнил вход на сайт, то перезагрузив страницу избранные лиги и команды исчезнут, чтобы это не происходило требуется пройти регистрацию и вход на сайт и в последующие посещения выбранные лиги и команды не исчезнут (см. рис. 6.2, 6.3, 6.4, 6.5)

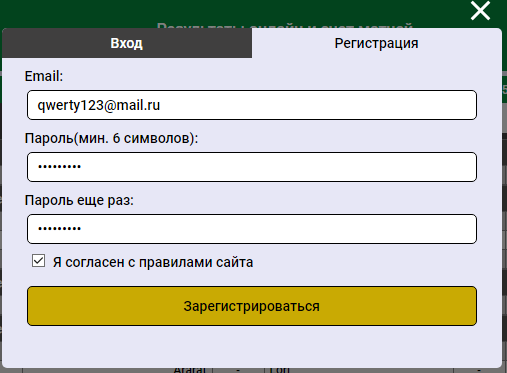


Рисунок 6.2 – Регистрация пользователя

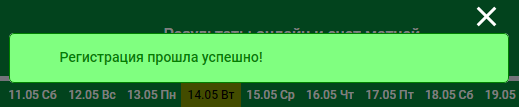


Рисунок 6.3 – Подтверждение регистрации

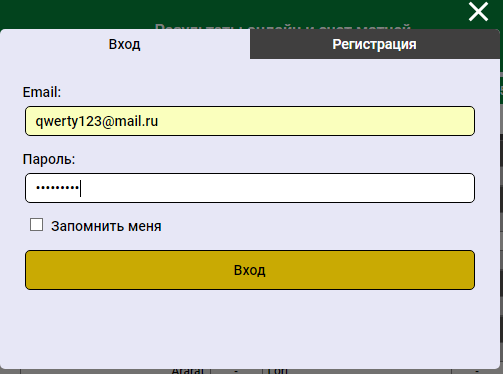


Рисунок 6.4 – Выполнение входа на сайт

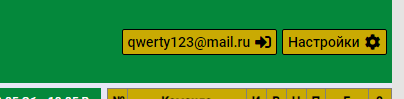


Рисунок 6.5 – Вход выполнен успешно

Чтобы добавить лигу в избранные требуется нажать возле ее названия на звездочку, после чего лига появится в боковом меню (см. рис. 6.6)

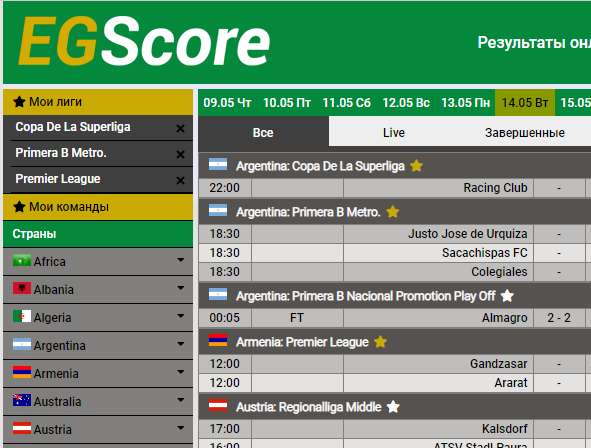


Рисунок 6.6 – Избранные лиги

Чтобы добавить команду в избранные требуется нажать на матч (завершившийся, предстоящий или идущий в данный момент) и на открывшейся странице кликнуть на звездочку возле команды, после чего игры выбранной команды будут отображаться во вкладке «Мои игры» (см. рис. 6.7)

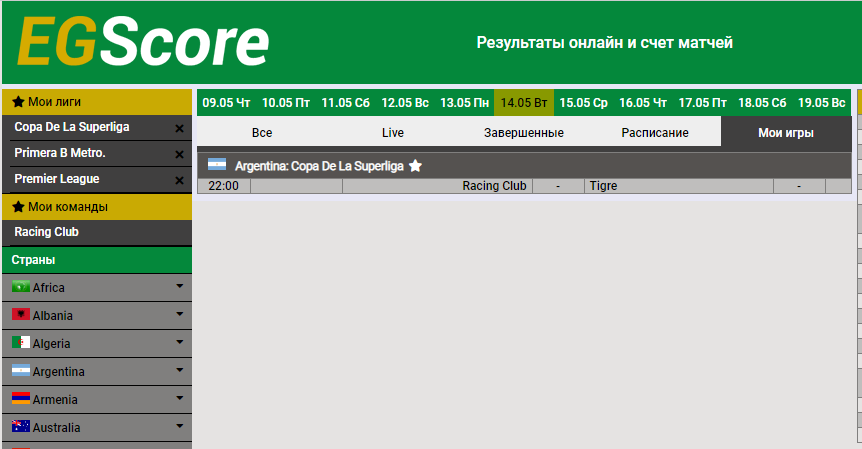


Рисунок 6.7 – Избранные команды

На главной странице для удобства присутствует пять вкладок «Все» - отображение всех матчей на текущий день, «Live» - матчи идущие в данный момент, «Завершенные» - завершенные матчи на текущий день, «Расписание» - матчи, которые еще начались на текущий день, «Мои игры» - матчи добавленные пользователем или матчи избранных команд (см. рис. 6.8, 6.9, 6.10)

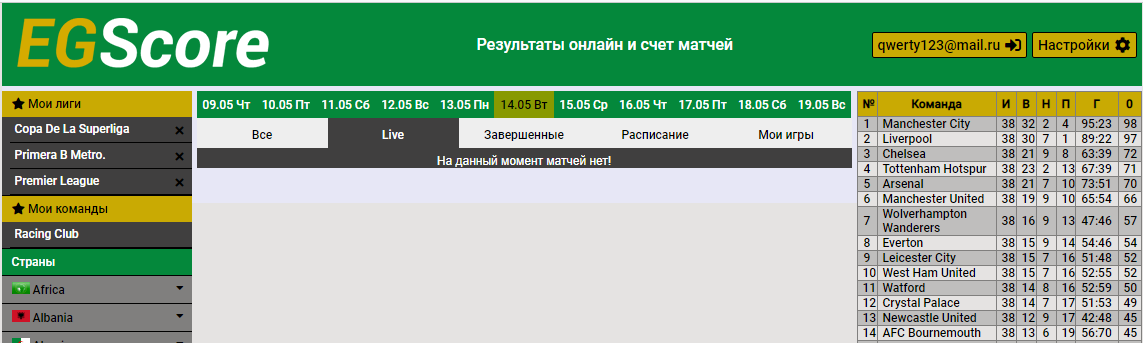


Рисунок 6.8 – Вкладка «Live»

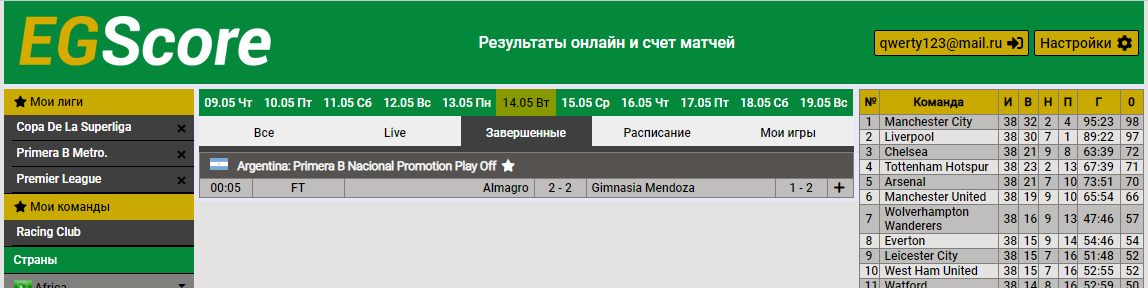


Рисунок 6.9 – Вкладка «Завершенные»

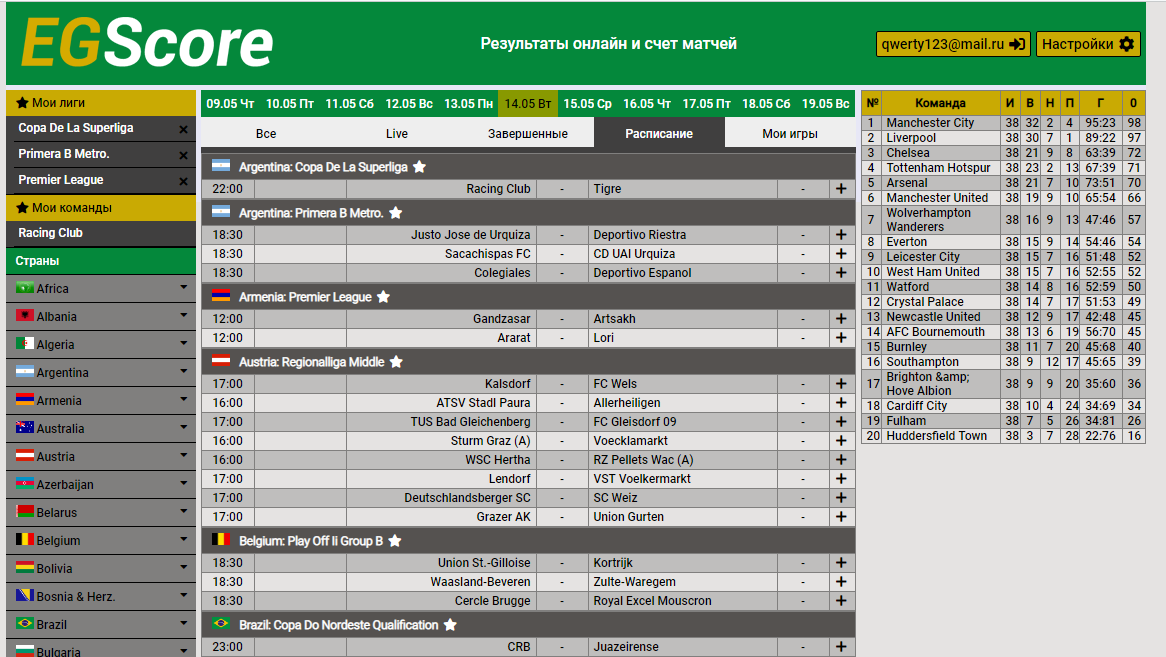


Рисунок 6.10 – Вкладка «Расписание»

# 7 ТЕСТИРОВАНИЕ ПРОГРАММНОЙ СИСТЕМЫ

## 7.1 Выбор метода тестирования

Тестирование (испытание) – установление соответствия программы заданным требованиям и программным документом.

Испытание программы является одним из этапов работ общего процесса разработки программ. Цель тестирования состоит в выявлении наличия ошибок в программе.

Тестирование это необычный процесс, разрушительный, так как его цель заставить программу ошибиться.

Тестирование состоит из следующих этапов:

- проектирование тестов (главный этап);

- выполнение тестов программой на компьютере;

- анализ результатов тестирования.

Основными подходами (стратегиями) к проектированию тестов являются: функциональный (на принципе «черный ящик») и структурный (на принципе «белого ящика»).

Функциональные методы тестирования базируются на рассмотрении программы как чёрного ящика. При этом внутренняя структура программы не учитывается. Тесты составляются по внешним спецификациям программы.

В структурных методах, для построения тестов используется информация о структуре (блок-схема) программы.

К функциональным методом относятся:

- метод функциональных диаграмм;

- метод эквивалентного разбиения;

- метод анализа граничных условий (значений).

Метод функциональных диаграмм основывается на использовании спецификации программы, написанных на естественном языке. Дает не плохие тесты, не страдающие избыточностью, и позволяет обнаружить неполноту и неоднозначность в исходных спецификациях.

Целью метода эквивалентного разбиения является разработка как можно меньшего количества тестов, которые образуют точный набор для всесторонней проверки программы.

Метод анализ граничных условий прост и дает хорошие тесты. У метода существуют такие особенности:

- тесты разрабатывается так, чтобы проверялась каждая граница класса эквивалентности;

- рассматриваются не только пространство входов (входные условия), но и пространство выходных данных, то есть выходные данные эквивалентности.

К структурным методам проектирования тестов относятся;

- метод покрытия операторов;

- метод покрытия переходов;

- метод покрытия условий;

- метод комбинаторного покрытия условий.

Для тестирования нашей программной системы был выбран функциональный метод эквивалентного разбиения.

## 7.2 Разработка тестов

Протестируем модуль информации о новом пользователе.

Сначала выделяем классы эквивалентности. Классы бывают правильными и неправильными.

Правильные классы – входные данные, удовлетворяющие функциональным характеристикам (все функции программы, условия, свойства, описывающие входные данные).

Неправильные классы – входные данные, не удовлетворяющие функциональным характеристикам, то есть все другие значения.

Классы эквивалентности приведены в таблице 7.1.

Таблица 7.1 – Классы эквивалентности

|  |  |  |
| --- | --- | --- |
| Функциональные характеристики | Правильные классы эквивалентности | Неправильные классы эквивалентности |
| Email | Текстовое [30]  9<= Длина <= 30 [22] | Длина > 35 [27]  Длина = 0 [28] |
|  |  |  |
|  |  |  |

...............................................

# ВЫВОДЫ

В результате выполнения дипломного проекта были решены следующие задачи:

1. сбор и анализ данных о подобных программных обеспечениях, требований к внешнему виду и функциональности сайта;
2. ознакомление с технологиями среды разработки «Visual Studio Code», языком разметки HTML и языком программирования JavaScript;
3. формирование структуры сайта;
4. разработка проекта дизайна;
5. рассмотреть вопросы охраны труда при эксплуатации ЭВМ.

Также был создан тематический веб-ресурс по предоставлению информации спортивных событий.

Созданный программный продукт, по сравнению с другими продуктами в исследуемой области, легкий в применении и не требует больших финансовых затрат при его использовании. Следует отметить, что данный проект полезен людям, которые следят за спортивными событиями.

Достоинства:

Главное достоинство данного сайта в том, что он прост в использовании и очень информативен, содержит всю информацию более 400 лиг, а именно:

а) онлайн результаты;

б) живые события (голы / бомбардиры / карты / составы / статистика);

в) результаты;

г) турнирные таблицы;

д) очные встречи команд.

Недостатки:

Недостатков мало, но тем не менее они очень весомы. Главный недостаток заключается в том, что вся информация поставляется посредством API, поэтому при большом количестве спортивных событий наблюдается длительная загрузка страниц сайта. И самое важное для длительного использования информации посредством API требуется платить ежемесячно определенную сумму денег.

# ПЕРЕЧЕНЬ ССЫЛОК

* 1. MongoDB. Устройство базы данных. Режим доступа: <https://metanit.com/nosql/mongodb>
  2. Express – фреймворк веб приложений node.js. Режим доступа: <https://expressjs.com/ru/>
  3. Документация Node.js. Режим доступа: <https://nodejs.org/uk/docs/>
  4. Introduction Bootstrap. Режим доступа: <https://getbootstrap.com/docs/4.3/getting-started/introduction/>
  5. Ejs. Режим доступа: <https://ejs.co/#docs>
  6. Документация по SASS на русском языке. Режим доступа: <https://sass-scss.ru/documentation/>
  7. Википедия – диаграмма прецедентов. Режим доступа: <https://ru.wikipedia.org/wiki/Диаграмма_прецедентов>
  8. Википедия – диаграмма состояний. Режим доступа: <https://ru.wikipedia.org/wiki/Диаграмма_состояний_(UML)>
  9. Википедия – диаграмма последовательностей. Режим доступа: <https://ru.wikipedia.org/wiki/Диаграмма_последовательности>
  10. Векторные иконки. Режим доступа: <https://fontawesome.com>
  11. Сессии Js. Режим доступа: <https://ru.hexlet.io/courses/js-express/lessons/sessions/theory_unit>
  12. Документация npm. Режим доступа: <https://docs.npmjs.com>
  13. Документация JQuery API. Режим доступа: <https://api.jquery.com>
  14. Справочник CSS. Режим доступа: <http://htmlbook.ru>
  15. Маршрутизация в Express. Режим доступа: <https://expressjs.com/ru/guide/routing.html>

# ПРИЛОЖЕНИЕ А

# ПРИЛОЖЕНИЕ Б

# ПРИЛОЖЕНИЕ В

Листинг

Файл auth.js

/\* eslint-disable no-console \*/

const express = require('express');

const router = express.Router();

const bcrypt = require('bcrypt-nodejs');

const models = require('../models');

// POST is reg

router.post('/register', (req, res) => {

const email = req.body.email;

const password = req.body.password;

const passwordConfirm = req.body.passwordConfirm;

const acceptTerms = req.body.acceptTerms;

// eslint-disable-next-line no-useless-escape

var regularMail = /^(([^<>()[\]\\.,;:\s@\"]+(\.[^<>()[\]\\.,;:\s@\"]+)\*)|(\".+\"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;

if (!email || !password || !passwordConfirm) {

const fields = [];

if(!email) fields.push('email')

if(!password) fields.push('password')

if(!passwordConfirm) fields.push('passwordConfirm')

res.json({

ok: false,

error: 'Все поля должны быть заполнены!',

fields

});

} else if (!regularMail.test(email)){

res.json({

ok: false,

error: 'Неверный формат Email!',

fields: ['email']

});

} else if (password.length < 6) {

res.json({

ok: false,

error: 'Слишком короткий пароль!',

fields: ['password']

});

} else if (password !== passwordConfirm) {

res.json({

ok: false,

error: 'Пароли не совпадают!',

fields: ['password', 'passwordConfirm']

});

} else if(acceptTerms != 'checked') {

res.json({

ok: false,

error: 'Вы должны подтвердить, что согласны с правилами!'

});

} else {

models.User.findOne({

email

}).then(user => {

if (!user) {

bcrypt.hash(password, null, null, (err, hash) => {

models.User.create({

email,

password: hash

})

.then(user => {

console.log(user);

req.session.userId = user.id;

req.session.userEmail = user.email;

res.json({

ok: true

});

})

.catch(err => {

console.log(err);

res.json({

ok: false,

error: 'Ошибка, попробуйте позже!'

});

});

});

} else {

res.json({

ok: false,

error: 'Email занят!',

fields: ['email']

});

}

});

}

});

// POST is login

router.post('/login', (req, res) => {

const email = req.body.email;

const password = req.body.password;

//const rememberMe = req.body.rememberMe;

if (!email || !password) {

const fields = [];

if(!email) fields.push('email')

if(!password) fields.push('password')

res.json({

ok: false,

error: 'Все поля должны быть заполнены!',

fields

});

} else {

models.User.findOne({

email

}).then(user=> {

if(!user) {

res.json({

ok: false,

error: 'Неверный логин и пароль!',

fields: ['email', 'password']

});

} else {

bcrypt.compare(password, user.password, function(err,result) {

if(!result) {

res.json({

ok: false,

error: 'Неверный логин и пароль!',

fields: ['email', 'password']

});

} else {

req.session.userId = user.id;

req.session.userEmail = user.email;

res.json({

ok: true

});

}

});

}

})

.catch(err => {

console.log(err);

res.json({

ok: false,

error: 'Ошибка, попробуйте позже!'

});

});

}

});

// GET for logout

router.get('/logout', (req, res) => {

if (req.session) {

req.session.destroy(() => {

res.redirect('/');

});

} else {

res.redirect('/');

}

});

module.exports = router;

Файл index.js

const auth = require('./auth');

module.exports = {

auth

};

Файл footer.ejs

</div>

<div class="col-md-3">

<table class="ttable">

<colgroup>

<col width="5%">

<col width="30%">

<col width="5%">

<col width="5%">

<col width="5%">

<col width="5%">

<col width="11%">

<col width="6%">

</colgroup>

<tr>

<th>№</th>

<th>Команда</th>

<th>И</th>

<th>В</th>

<th>Н</th>

<th>П</th>

<th>Г</th>

<th>0</th>

</tr>

<% for(var i = 0; i < data.ttable.length; i++) {%>

<tr>

<td><%=data.ttable[i].overall\_league\_position%></td>

<td style="text-align: left"><%=data.ttable[i].team\_name%></td>

<td><%=data.ttable[i].overall\_league\_payed%></td>

<td><%=data.ttable[i].overall\_league\_W%></td>

<td><%=data.ttable[i].overall\_league\_D%></td>

<td><%=data.ttable[i].overall\_league\_L%></td>

<td><%=data.ttable[i].overall\_league\_GF%>:<%=data.ttable[i].overall\_league\_GA%></td>

<td><%=data.ttable[i].overall\_league\_PTS%></td>

</tr>

<%}%>

</table>

</div>

</div>

</div>

</div>

<script src="javascripts/tabs.js"></script>

<script>

// var request = new XMLHttpRequest();

// var requestURL = 'https://apifootball.com/api/?action=get\_countries&APIkey=004b26f0bdc9085ae7d6916dc3aa7b7a5c81af51dde4720ab74ea13f1759806a';

// request.open('GET', requestURL);

// request.responseType = 'json';

// request.send();

// request.onload = function (){

// var qwe = request.response;

// console.log(qwe);

// };

/\* Loop through all dropdown buttons to toggle between hiding and showing its dropdown content - This allows the user to have multiple dropdowns without any conflict \*/

var dropdown = document.getElementsByClassName("leftbar-btn");

var i;

for (i = 0; i < dropdown.length; i++) {

dropdown[i].addEventListener("click", function() {

this.classList.toggle("active");

var dropdownContent = this.nextElementSibling;

if (dropdownContent.style.display === "block") {

dropdownContent.style.display = "none";

} else {

dropdownContent.style.display = "block";

}

});

}

</script>

<nav class="modal">

</nav>

<div class="overlay"> </div>

<div class="main-popup">

<div class="popup-header">

<div id="popup-close-button"><a href="#"></a></div>

<div class="buttonContainer">

<button id="sign-in">Вход</button>

<button id="register">Регистрация</button>

</div>

</div>

<div class="popup-content">

<form action="#" class="sign-in" method="post">

<div class="logmess"></div>

<label for="email">Email:</label>

<input type="text" id="email" name="email">

<label for="password">Пароль:</label>

<input type="password" id="password" name="password">

<p class="check-mark">

<input type="checkbox" id="remember-me">

<label for="remember-me">Запомнить меня</label>

</p>

<input type="submit" id="submit-sing-in" value="Вход">

</form>

<form action="#" class="register" method="post">

<div class="regmess"></div>

<label for="email-register">Email:</label>

<input type="text" id="email-register" name="email">

<label for="password-register">Пароль(мин. 6 символов):</label>

<input type="password" id="password-register" name="password">

<label for="password-confirmation">Пароль еще раз:</label>

<input type="password" id="password-confirmation" name="passwordConfirm">

<p class="check-mark">

<input type="checkbox" id="accept-terms">

<label for="accept-terms">Я согласен с правилами сайта</a></label>

</p>

<input type="submit" id="submit-register" value="Зарегистрироваться">

</form>

</div>

</div>

<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js" type="text/javascript"></script>

<script>

$(document).ready(function() {

function getWeekDay(date) {

var days = ['Вс', 'Пн', 'Вт', 'Ср', 'Чт', 'Пт', 'Сб'];

return days[date.getDay()];

}

var $dateofmatch = $('.dateofmatch');

var date = new Date();

for(var i = date.getDate() - 5; i < date.getDate() + 6; i++) {

var startDate = new Date(date.getFullYear(),date.getMonth(), i);

var dd = startDate.getDate();

if (dd < 10) dd = '0' + dd;

var mm = startDate.getMonth() + 1;

if (mm < 10) mm = '0' + mm;

if(i == date.getDate())

$dateofmatch.append('<button style="background-color: #890; color: #000">'+ dd +'.' +mm + ' '+ getWeekDay(startDate) +'</button>');

else

$dateofmatch.append('<button>'+ dd +'.' +mm + ' '+ getWeekDay(startDate) +'</button>');

}

$('.fa-star').on('click', function(e){

var id = e.target.id;

var arr = id.split('\_');

if($(this).hasClass('active')) {

$(this).removeClass('active');

$('li#'+arr[1]+'').remove();

$('hr#'+arr[1]+'').remove();

} else {

$(this).addClass('active');

$('#my\_league').append('<li id=' + arr[1] + '><a href=\"#\">'+ arr[0] +'</a><i class=\"fas fa-times\"></i></li><hr id=' + arr[1] + '>');

}

});

$('.fa-times').on('click', function (e) {

var id = e.target.id;

alert(id);

});

var $overlay = $('.overlay');

var $mainPopUp = $('.main-popup')

var $signIn = $('#sign-in');

var $register = $('#register');

var $formSignIn = $('form.sign-in');

var $formRegister = $('form.register');

function changeUnderline(el) {

$('.underline').css({

"width": el.width(),

"left": el.position().left,

"top": el.position().top + el.outerHeight(true) + 'px'

});

}

$('.auth').on('click', function(){

$('#email').val('');$('#password').val('');$('#email-register').val('');$('#password-register').val('');

$('#password-confirmation').val('');

$('.main-popup').removeClass('hidden');

$('.error, .succes').remove();

$('input').removeClass('errorinput');

$('input').removeClass('errorLog');

$overlay.addClass('visible');

$mainPopUp.addClass('visible');

$signIn.addClass('active');

$register.removeClass('active');

$formRegister.removeClass('move-left');

$formSignIn.removeClass('move-left');

});

$overlay.on('click', function(){

$(this).removeClass('visible');

$mainPopUp.removeClass('visible');

});

$('#popup-close-button a').on('click', function(e){

e.preventDefault();

$overlay.removeClass('visible');

$mainPopUp.removeClass('visible');

});

$signIn.on('click', function(){

$signIn.addClass('active');

$register.removeClass('active');

$formSignIn.removeClass('move-left');

$formRegister.removeClass('move-left');

$('.error, .succes').remove();

$('input').removeClass('errorinput');

});

$register.on('click', function(){

$signIn.removeClass('active');

$register.addClass('active');

$formSignIn.addClass('move-left');

$formRegister.addClass('move-left');

$('.error, .succes').remove();

$('input').removeClass('errorLog');

});

$('.out').on('click', function(e){

e.preventDefault();

$(location).attr('href','/api/auth/logout');

});

$('#submit-register').on('click', function(e){

e.preventDefault();

var data = {

email: $('#email-register').val(),

password: $('#password-register').val(),

passwordConfirm: $('#password-confirmation').val(),

acceptTerms: $('#accept-terms').attr("checked")

};

$.ajax({

type: 'POST',

data: JSON.stringify(data),

contentType: 'application/json',

url: '/api/auth/register'

}).done(function(data) {

$('.error, .succes').remove();

if(!data.ok) {

$('.regmess').prepend('<p class="error">' + data.error + '</p>').show();

if (data.fields) {

data.fields.forEach(function(item) {

$('input[name=' + item + ']').addClass('errorinput');

console.log(item);

});

}

}

else {

$('.main-popup').addClass('hidden');

$('.popup-header').append('<p class="succes">Регистрация прошла успешно!</p>').show();

}

});

});

$('input').on('focus', function() {

$('p.error').remove();

$('input').removeClass('errorinput');

$('input').removeClass('errorLog');

});

$('#submit-sing-in').on('click', function(e){

e.preventDefault();

var data = {

email: $('#email').val(),

password: $('#password').val(),

rememberMe: $('#remember-me').attr("checked")

};

console.log(data.acceptTerms);

$.ajax({

type: 'POST',

data: JSON.stringify(data),

contentType: 'application/json',

url: '/api/auth/login'

}).done(function(data) {

$('.error, .succes').remove();

if(!data.ok) {

$('.logmess').prepend('<p class="error">' + data.error + '</p>').show();

if (data.fields) {

data.fields.forEach(function(item) {

$('input[name=' + item + ']').addClass('errorLog');

console.log(item);

});

}

}

else {

$(location).attr('href', '/');

}

});

});

});

</script>

</body>

</html>

Файл header.ejs

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>EGScore</title>

<link rel="stylesheet" href="stylesheets/bootstrap-grid.css">

<link rel="stylesheet" href="stylesheets/style.css">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.8.1/css/all.css">

</head>

<body>

<div class="container">

<div class="header">

<div class="row align-items-center" >

<div class="col-md-5 logo">

<img src="../images/logo.png" alt="Логотип">

</div>

<div class="col-md-4 string">

<p>Результаты онлайн и счет матчей</p>

</div>

<div class="col-md-3 btns">

<button class="reg">

<p>Настройки</p><i class="fa fa-cog"></i>

</button>

<% if(!user.id) { %>

<button class="auth">

<p>Вход</p><i class="fa fa-sign-in-alt"></i>

</button>

<% } else { %>

<button class="out">

<p><%= user.email %></p><i class="fa fa-sign-in-alt"></i>

</button>

<% } %>

</div>

</div>

</div>

<div class="main">

<div class="row">

<div class="col-md-2">

<div class="left-sidebar">

<button class="leftbar\_btn">

<i class="fa fa-star"></i>

Мои лиги

</button>

<ul class="left-side-menu" id="my\_league">

</ul>

<hr>

<button class="leftbar\_btn">

<i class="fa fa-star"></i>

Мои команды

</button>

<ul class="left-side-menu">

<li><a href="#">Racing Club</a><i class=\"fas fa-times\"></i></li><hr>

</ul>

<hr>

<p>Страны</p>

<hr>

<% for(var i = 0; i < data.cont.length ;i++) {%>

<button class="leftbar-btn country">

<img src="images/<%=data.cont[i].country\_name%>.png">

<%=data.cont[i].country\_name%>

<i class="fas fa-caret-down"></i>

</button>

<ul class="left-sidebar-menu">

<% for(var j = 0; j < data.leagues.length ;j++) {%>

<% if (data.cont[i].country\_name == data.leagues[j].country\_name) {%>

<li><a href="#"><%=data.leagues[j].league\_name%></a><i class="fas fa-plus"></i></li><hr>

<%}}%>

</ul>

<hr>

<%}%>

</div>

</div>

<div class="col-md-7" >

Файл main.ejs

<div class="main">

<div class="row">

<div class="col-md-2">

<div class="left-sidebar">

<button class="leftbar-btn">

<i class="fa fa-star"></i>

Мои лиги (4)

<i class="fa fa-caret-down"></i>

</button>

<ul class="left-sidebar-menu">

<li><i class="fa fa-flag-usa"></i><a href="#">Лига чемпионов</a><i class="fas fa-times"></i></li><hr>

<li><i class="fa fa-flag-usa"></i><a href="#">Премьер-лига</a><i class="fas fa-times"></i></li><hr>

<li><i class="fa fa-flag-usa"></i><a href="#">Примера</a><i class="fas fa-times"></i></li><hr>

<li><i class="fa fa-flag-usa"></i><a href="#">Бундеслига</a><i class="fas fa-times"></i></li>

</ul>

<hr>

<button class="leftbar-btn">

<i class="fa fa-star"></i>

Мои команды(3)

<i class="fa fa-caret-down"></i>

</button>

<ul class="left-sidebar-menu">

<li><i class="fa fa-flag-usa"></i><a href="#">Лига чемпионов</a><i class="fas fa-times"></i></li><hr>

<li><i class="fa fa-flag-usa"></i><a href="#">Премьер-лига</a><i class="fas fa-times"></i></li><hr>

<li><i class="fa fa-flag-usa"></i><a href="#">Примера</a><i class="fas fa-times"></i></li><hr>

<li><i class="fa fa-flag-usa"></i><a href="#">Бундеслига</a><i class="fas fa-times"></i></li>

</ul>

<hr>

<p>Страны</p>

<hr>

<button class="leftbar-btn country">

<i class="fas fa-futbol"></i>

Австрия

<i class="fas fa-angle-right"></i>

</button>

<% for(var i = 0; i < 4 ;i++) {%>

<button class="leftbar-btn country">

<i class="fas fa-futbol"></i>

AAAAAAAAAAAAAAAAAAAAA

<i class="fas fa-angle-right"></i>

</button>

<%} %>

<ul class="left-sidebar-menu">

<li><i class="fa fa-flag-usa"></i><a href="#">Лига чемпионов</a><i class="fas fa-plus"></i></li><hr>

<li><i class="fa fa-flag-usa"></i><a href="#">Премьер-лига</a><i class="fas fa-plus"></i></li><hr>

<li><i class="fa fa-flag-usa"></i><a href="#">Примера</a><i class="fas fa-plus"></i></li><hr>

<li><i class="fa fa-flag-usa"></i><a href="#">Бундеслига</a><i class="fas fa-plus"></i></i></li>

</ul>

<hr>

</div>

</div>

<div class="col-md-7" >

<div class="dateofmatch">

<button onclick="">01.05 Пн</button>

<button onclick="">02.05 Вт</button>

<button onclick="">02.05 Ср</button>

<button onclick="">02.05 Чт</button>

<button onclick="">02.05 Пт</button>

<button onclick="" style="background-color: #c9aa03">02.05 Сб</button>

<button onclick="">02.05 Вс</button>

<button onclick="">02.05 Пн</button>

<button onclick="">02.05 Вт</button>

<button onclick="">02.05 Ср</button>

<button onclick="">02.05 Чт</button>

</div>

<div class="tabContainer">

<div class="buttonContainer">

<button onclick="showPanel(0,'#403f3f')">Все</button>

<button onclick="showPanel(1,'#403f3f')">Live</button>

<button onclick="showPanel(2,'#403f3f')">Завершенные</button>

<button onclick="showPanel(3,'#403f3f')">Расписание</button>

<button onclick="showPanel(4,'#403f3f')">Мои игры</button>

</div>

<div class="tabPanel">

<table id="england">

<colgroup>

<col width="8%">

<col width="14%">

<col width="33%">

<col width="8%">

<col width="33%">

<col width="4%">

</colgroup>

<tr>

<th colspan="6"><i class="fa fa-flag-usa"></i>Чемпионат Европы<i class="fa fa-star"></i></th>

</tr>

<tr>

<td>12:00</td>

<td>Завершен</td>

<td>Республика Беларусь</td>

<td>0:1</td>

<td>Германия</td>

<td><i class="fas fa-plus"></i></td>

</tr>

<tr>

<td>13:00</td>

<td>67'</td>

<td>Швеция</td>

<td>2:2</td>

<td>Россия</td>

<td><i class="fas fa-plus"></i></td>

</tr>

</table>

</div>

<div class="tabPanel"></div>

<div class="tabPanel">Tab 3:Content</div>

<div class="tabPanel">Tab 4:Content</div>

<div class="tabPanel">Tab 4:Content</div>

</div>

</div>

<div class="col-md-3">

<table id="ttable">

<colgroup>

<col width="5%">

<col width="30%">

<col width="5%">

<col width="5%">

<col width="5%">

<col width="5%">

<col width="11%">

<col width="6%">

</colgroup>

<tr>

<th>№</th>

<th>Команда</th>

<th>И</th>

<th>В</th>

<th>Н</th>

<th>П</th>

<th>Г</th>

<th>0</th>

</tr>

<tr>

<td>10</td>

<td>Манчестер Сити</td>

<td>12</td>

<td>14</td>

<td>12</td>

<td>12</td>

<td>12:121</td>

<td>123</td>

</tr>

<tr>

<td>1</td>

<td>Арсенал</td>

<td>12</td>

<td>14</td>

<td>12</td>

<td>12</td>

<td>12:121</td>

<td>123</td>

</tr>

<tr>

<td>1</td>

<td>Манчестер Юнайтед</td>

<td>12</td>

<td>14</td>

<td>12</td>

<td>12</td>

<td>12:121</td>

<td>123</td>

</tr>

<tr>

<td>1</td>

<td>Манчестер Юнайтед</td>

<td>12</td>

<td>14</td>

<td>12</td>

<td>12</td>

<td>12:121</td>

<td>123</td>

</tr>

<tr>

<td>1</td>

<td>Манчестер Юнайтед</td>

<td>12</td>

<td>14</td>

<td>12</td>

<td>12</td>

<td>12:121</td>

<td>123</td>

</tr>

<tr>

<td>1</td>

<td>Манчестер Юнайтед</td>

<td>12</td>

<td>14</td>

<td>12</td>

<td>12</td>

<td>12:121</td>

<td>123</td>

</tr>

<tr>

<td>1</td>

<td>Манчестер Юнайтед</td>

<td>12</td>

<td>14</td>

<td>12</td>

<td>12</td>

<td>12:121</td>

<td>123</td>

</tr>

</table>

</div>

</div>

</div>

Файл error.ejs

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title><%= error.status%></title>

</head>

<body>

<h1><%= message%></h1>

<h2><%= error.status%></h2>

<pre><%= error.stack%></pre>

</body>

</html>

Файл Index.ejs

<% include layout/header.ejs %>

<div class="dateofmatch">

</div>

<div class="tabContainer">

<div class="buttonContainer">

<button onclick="showPanel(0,'#403f3f')">Все</button>

<button onclick="showPanel(1,'#403f3f')">Live</button>

<button onclick="showPanel(2,'#403f3f')">Завершенные</button>

<button onclick="showPanel(3,'#403f3f')">Расписание</button>

<button onclick="showPanel(4,'#403f3f')">Мои игры</button>

</div>

<div class="tabPanel" id="tab1">

<% for (var i = 0; i < data.country\_league.length; i++) {%>

<table class="matches" id="cs">

<colgroup>

<col width="8%">

<col width="14%">

<col width="29%">

<col width="8%">

<col width="29%">

<col width="8%">

<col width="4%">

</colgroup>

<tr>

<th colspan="7"><img src="images/<%=data.country\_league[i].country\_name%>.png"></i><%=data.country\_league[i].country\_name%>: <%=data.country\_league[i].league\_name%><i class="fa fa-star" id="<%=data.country\_league[i].league\_name%>\_<%=data.country\_league[i].league\_id%>"></i></th>

</tr>

<%for(var j = data.count\_match[i]; j < data.count\_match[i + 1]; j++) {%>

<%if ((data.games[j].match\_status != 'FT') && (data.games[j].match\_status != '') && (data.games[j].match\_status != 'AP')) {%>

<tr>

<td><%=data.games[j].match\_time%></td>

<td style="color: red"><%=data.games[j].match\_status%></td>

<td style="text-align: right"><%=data.games[j].match\_hometeam\_name%></td>

<td style="color: red"><%=data.games[j].match\_hometeam\_score%> - <%=data.games[j].match\_awayteam\_score%></td>

<td style="text-align: left"><%=data.games[j].match\_awayteam\_name%></td>

<td style="color: red"><%=data.games[j].match\_hometeam\_halftime\_score%> - <%=data.games[j].match\_awayteam\_halftime\_score%></td>

<td><i class="fas fa-plus"></i></td>

</tr>

<%} else {%>

<tr>

<td><%=data.games[j].match\_time%></td>

<td><%=data.games[j].match\_status%></td>

<td style="text-align: right"><%=data.games[j].match\_hometeam\_name%></td>

<td><%=data.games[j].match\_hometeam\_score%> - <%=data.games[j].match\_awayteam\_score%></td>

<td style="text-align: left"><%=data.games[j].match\_awayteam\_name%></td>

<td><%=data.games[j].match\_hometeam\_halftime\_score%> - <%=data.games[j].match\_awayteam\_halftime\_score%></td>

<td><i class="fas fa-plus"></i></td>

</tr>

<%}}%>

</table>

<% }%>

</div>

<div class="tabPanel">

<%if ( data.live\_con.length != 0) {%>

<% for (var i = 0; i < data.live\_con.length; i++) {%>

<table class="matches" id="cs">

<colgroup>

<col width="8%">

<col width="14%">

<col width="29%">

<col width="8%">

<col width="29%">

<col width="8%">

<col width="4%">

</colgroup>

<tr>

<th colspan="7"><img src="images/<%=data.live\_con[i].country\_name%>.png"></i><%=data.live\_con[i].country\_name%>: <%=data.live\_con[i].league\_name%><i class="fa fa-star"></i></th>

</tr>

<%for(var j = data.live\_matches[i]; j < data.live\_matches[i + 1]; j++) {%>

<tr>

<td><%=data.games\_live[j].match\_time%></td>

<td style="color: red"><%=data.games\_live[j].match\_status%></td>

<td style="text-align: right"><%=data.games\_live[j].match\_hometeam\_name%></td>

<td style="color: red"><%=data.games\_live[j].match\_hometeam\_score%> - <%=data.games\_live[j].match\_awayteam\_score%></td>

<td style="text-align: left"><%=data.games\_live[j].match\_awayteam\_name%></td>

<td style="color: red"><%=data.games\_live[j].match\_hometeam\_halftime\_score%> - <%=data.games\_live[j].match\_awayteam\_halftime\_score%></td>

<td><i class="fas fa-plus"></i></td>

</tr>

<%}%>

</table>

<% } } else {%>

<p>На данный момент матчей нет!</p>

<%}%>

</div>

<div class="tabPanel">

<% for (var i = 2; i < 3; i++) {%>

<table class="matches" id="cs">

<colgroup>

<col width="8%">

<col width="14%">

<col width="29%">

<col width="8%">

<col width="29%">

<col width="8%">

<col width="4%">

</colgroup>

<tr>

<th colspan="7"><img src="images/<%=data.country\_league[i].country\_name%>.png"></i><%=data.country\_league[i].country\_name%>: <%=data.country\_league[i].league\_name%><i class="fa fa-star" id="<%=data.country\_league[i].league\_name%>\_<%=data.country\_league[i].league\_id%>"></i></th>

</tr>

<%for(var j = data.count\_match[i]; j < data.count\_match[i + 1]; j++) {%>

<%if ((data.games[j].match\_status != 'FT') && (data.games[j].match\_status != '') && (data.games[j].match\_status != 'AP')) {%>

<tr>

<td><%=data.games[j].match\_time%></td>

<td style="color: red"><%=data.games[j].match\_status%></td>

<td style="text-align: right"><%=data.games[j].match\_hometeam\_name%></td>

<td style="color: red"><%=data.games[j].match\_hometeam\_score%> - <%=data.games[j].match\_awayteam\_score%></td>

<td style="text-align: left"><%=data.games[j].match\_awayteam\_name%></td>

<td style="color: red"><%=data.games[j].match\_hometeam\_halftime\_score%> - <%=data.games[j].match\_awayteam\_halftime\_score%></td>

<td><i class="fas fa-plus"></i></td>

</tr>

<%} else {%>

<tr>

<td><%=data.games[j].match\_time%></td>

<td><%=data.games[j].match\_status%></td>

<td style="text-align: right"><%=data.games[j].match\_hometeam\_name%></td>

<td><%=data.games[j].match\_hometeam\_score%> - <%=data.games[j].match\_awayteam\_score%></td>

<td style="text-align: left"><%=data.games[j].match\_awayteam\_name%></td>

<td><%=data.games[j].match\_hometeam\_halftime\_score%> - <%=data.games[j].match\_awayteam\_halftime\_score%></td>

<td><i class="fas fa-plus"></i></td>

</tr>

<%}}%>

</table>

<% }%>

</div>

<div class="tabPanel">

<% for (var i = 0; i < data.country\_league.length; i++) {%>

<%if (i != 2) {%>

<table class="matches" id="cs">

<colgroup>

<col width="8%">

<col width="14%">

<col width="29%">

<col width="8%">

<col width="29%">

<col width="8%">

<col width="4%">

</colgroup>

<tr>

<th colspan="7"><img src="images/<%=data.country\_league[i].country\_name%>.png"></i><%=data.country\_league[i].country\_name%>: <%=data.country\_league[i].league\_name%><i class="fa fa-star" id="<%=data.country\_league[i].league\_name%>\_<%=data.country\_league[i].league\_id%>"></i></th>

</tr>

<%for(var j = data.count\_match[i]; j < data.count\_match[i + 1]; j++) {%>

<%if ((data.games[j].match\_status != 'FT') && (data.games[j].match\_status != '') && (data.games[j].match\_status != 'AP')) {%>

<tr>

<td><%=data.games[j].match\_time%></td>

<td style="color: red"><%=data.games[j].match\_status%></td>

<td style="text-align: right"><%=data.games[j].match\_hometeam\_name%></td>

<td style="color: red"><%=data.games[j].match\_hometeam\_score%> - <%=data.games[j].match\_awayteam\_score%></td>

<td style="text-align: left"><%=data.games[j].match\_awayteam\_name%></td>

<td style="color: red"><%=data.games[j].match\_hometeam\_halftime\_score%> - <%=data.games[j].match\_awayteam\_halftime\_score%></td>

<td><i class="fas fa-plus"></i></td>

</tr>

<%} else {%>

<tr>

<td><%=data.games[j].match\_time%></td>

<td><%=data.games[j].match\_status%></td>

<td style="text-align: right"><%=data.games[j].match\_hometeam\_name%></td>

<td><%=data.games[j].match\_hometeam\_score%> - <%=data.games[j].match\_awayteam\_score%></td>

<td style="text-align: left"><%=data.games[j].match\_awayteam\_name%></td>

<td><%=data.games[j].match\_hometeam\_halftime\_score%> - <%=data.games[j].match\_awayteam\_halftime\_score%></td>

<td><i class="fas fa-plus"></i></td>

</tr>

<%}}%>

</table>

<%} }%>

</div>

<div class="tabPanel">

<% for (var i = 0; i < 1; i++) {%>

<table class="matches" id="cs">

<colgroup>

<col width="8%">

<col width="14%">

<col width="29%">

<col width="8%">

<col width="29%">

<col width="8%">

<col width="4%">

</colgroup>

<tr>

<th colspan="7"><img src="images/<%=data.country\_league[i].country\_name%>.png"></i><%=data.country\_league[i].country\_name%>: <%=data.country\_league[i].league\_name%><i class="fa fa-star" id="<%=data.country\_league[i].league\_name%>\_<%=data.country\_league[i].league\_id%>"></i></th>

</tr>

<%for(var j = data.count\_match[i]; j < data.count\_match[i + 1]; j++) {%>

<%if ((data.games[j].match\_status != 'FT') && (data.games[j].match\_status != '') && (data.games[j].match\_status != 'AP')) {%>

<tr>

<td><%=data.games[j].match\_time%></td>

<td style="color: red"><%=data.games[j].match\_status%></td>

<td style="text-align: right"><%=data.games[j].match\_hometeam\_name%></td>

<td style="color: red"><%=data.games[j].match\_hometeam\_score%> - <%=data.games[j].match\_awayteam\_score%></td>

<td style="text-align: left"><%=data.games[j].match\_awayteam\_name%></td>

<td style="color: red"><%=data.games[j].match\_hometeam\_halftime\_score%> - <%=data.games[j].match\_awayteam\_halftime\_score%></td>

<td><i class="fas fa-plus"></i></td>

</tr>

<%} else {%>

<tr>

<td><%=data.games[j].match\_time%></td>

<td><%=data.games[j].match\_status%></td>

<td style="text-align: right"><%=data.games[j].match\_hometeam\_name%></td>

<td><%=data.games[j].match\_hometeam\_score%> - <%=data.games[j].match\_awayteam\_score%></td>

<td style="text-align: left"><%=data.games[j].match\_awayteam\_name%></td>

<td><%=data.games[j].match\_hometeam\_halftime\_score%> - <%=data.games[j].match\_awayteam\_halftime\_score%></td>

<td></td>

</tr>

<%}}%>

</table>

<% }%>

</div>

</div>

<% include layout/footer.ejs %>

Файл \_fonts.scss

@font-face {

font-family: 'RobotoBold';

src: url('../fonts/RobotoBold.eot');

src: url('../fonts/RobotoBold.eot') format('embedded-opentype'),

url('../fonts/RobotoBold.woff2') format('woff2'),

url('../fonts/RobotoBold.woff') format('woff'),

url('../fonts/RobotoBold.ttf') format('truetype'),

url('../fonts/RobotoBold.svg#RobotoBold') format('svg');

}

@font-face {

font-family: 'RobotoLight';

src: url('../fonts/RobotoLight.eot');

src: url('../fonts/RobotoLight.eot') format('embedded-opentype'),

url('../fonts/RobotoLight.woff2') format('woff2'),

url('../fonts/RobotoLight.woff') format('woff'),

url('../fonts/RobotoLight.ttf') format('truetype'),

url('../fonts/RobotoLight.svg#RobotoLight') format('svg');

}

@font-face {

font-family: 'RobotoMedium';

src: url('../fonts/RobotoMedium.eot');

src: url('../fonts/RobotoMedium.eot') format('embedded-opentype'),

url('../fonts/RobotoMedium.woff2') format('woff2'),

url('../fonts/RobotoMedium.woff') format('woff'),

url('../fonts/RobotoMedium.ttf') format('truetype'),

url('../fonts/RobotoMedium.svg#RobotoMedium') format('svg');

}

Файл \_footer.scss

.modal{

font-family: 'RobotoMedium';

font-size: 14px;

}

.container {

width: 100%;

height: 200px;

background-color: #e7e7f6;

border-radius: 5px;

z-index: 2;

}

.overlay {

position: fixed;

left: 0;

top: 0;

width: 100%;

height: 100%;

background-color: rgba(0,0,0,0.5);;

z-index: 10;

opacity: 0;

visibility: hidden;

transition: all .5s ease;

}

.overlay.visible {

opacity: 1;

visibility: visible;

}

.main-popup {

position: fixed;

left: 0;

top: 30px;

margin: 0;

width: 100%;

height: 340px;

background-color: #e7e7f6;

border-radius: 5px;

z-index: 15;

opacity: 0;

visibility: hidden;

transform: translateY(-20px);

transition: all .5s ease;

}

.main-popup.visible {

opacity: 1;

visibility: visible;

transform: translateY(10px);

transition: all .5s ease;

}

@media (min-width: 500px) {

.main-popup {

width: 500px;

left: 50%;

margin: 0 0 0 -250px;

}

}

.popup-header {

position: relative;

padding: 0;

margin: 0;

height: 30px;

width: 100%;

}

#popup-close-button a {

position: absolute;

right: 10px;

top: -30px;

width: 22px;

height: 22px;

}

#popup-close-button a::before {

content: '';

position: absolute;

right: 10px;

top: 0;

width: 3px;

height: 25px;

background-color: #fff;

-webkit-transform: rotate(45deg);

-moz-transform: rotate(45deg);

-ms-transform: rotate(45deg);

-o-transform: rotate(45deg);

transform: rotate(45deg);

}

#popup-close-button a::after {

content: '';

position: absolute;

right: 10px;

top: 0;

width: 3px;

height: 25px;

background-color: #fff;

-webkit-transform: rotate(-45deg);

-moz-transform: rotate(-45deg);

-ms-transform: rotate(-45deg);

-o-transform: rotate(-45deg);

transform: rotate(-45deg);

}

.popup-header ul {

margin: 0;

padding: 0;

}

.popup-header ul li {

text-align: center;

list-style: none;

width: 50%;

float: left;

}

.popup-header ul li a {

display: block;

padding: 20px 0;

margin: 0;

text-decoration: none;

font-family: 'RobotoMedium';

font-size: 14px;

}

#sign-in {

background-color: $color-dark-grey;

color: #fff;

border-radius: 5px 0 0 0;

}

#sign-in.active {

color: #000;

background-color: transparent;

}

#register {

background-color: $color-dark-grey;

color: #fff;

border-radius: 0 5px 0 0;

}

#register.active {

color: #000;

background-color: transparent;

}

.popup-content {

height: 400px;

overflow: hidden;

}

form.sign-in {

position: relative;

top: 25px;

left: 0;

font-family: 'RobotoMedium';

font-size: 14px;

opacity: 1;

-webkit-transition: all .35s;

-moz-transition: all .35s;

-o-transition: all .35s;

transition: all .35s;

}

form.sign-in.move-left {

opacity: 0;

transform: translateX(-450px);

}

form label {

font-family: 'RobotoMedium';

font-size: 14px;

color: #000;

margin-left: 23px;

}

input#submit-sing-in, input#submit-register {

background-color: #c9aa03;

color: #000;

height: 40px;

width: 90%;

margin-left: 5%;

margin-right: 5%;

margin-top: 15px;

padding: 0;

cursor: pointer;

outline: none;

border-radius: 5px;

font-family: 'RobotoMedium';

font-size: 14px;

border: 1px solid black;

transition: 0.5s;

}

input#submit-sing-in:hover, input#submit-register:hover {

background-color: #ffd804;

}

form.register {

position: relative;

top: -211px;

left: 0;

font-size: 14px;

opacity: 0;

transform: translateX(450px);

-webkit-transition: all .35s;

-moz-transition: all .35s;

-o-transition: all .35s;

transition: all .35s;

}

form.register.move-left {

opacity: 1;

transform: translateX(0);

}

form.sign-in input {

border-radius: 5px;

width: 90%;

height: 30px;

margin: 5px 5% 15px 5%;

padding: 10px;

font-family: 'RobotoMedium';

font-size: 14px;

color: #000;

outline: none;

border: 1px solid #000;

&.errorLog {

background-color: lighten(red, 45%);

border-color: red;

}

}

form.register input{

border-radius: 5px;

width: 90%;

height: 30px;

margin: 5px 5% 10px 5%;

padding: 10px;

font-family: 'RobotoMedium';

font-size: 14px;

color: #000;

outline: none;

border: 1px solid #000;

&.errorinput {

background-color: lighten(red, 45%);

border-color: red;

}

}

p.check-mark {

position: relative;

width: 300px;

padding-left: 30px;

text-align: left;

color: #000;

font-family: 'RobotoMedium';

font-size: 14px;

}

p.check-mark input {

border-radius: 0;

width: auto;

height: auto;

margin: 0;

padding: 0;

font-size: 2em;

color: #000;

outline: none;

border: none;

}

p.check-mark label {

margin-left: 5px;

}

.buttonContainer #sign-in, #register {

width: 50%;

font-size: 14px;

padding: 0;

}

.hidden {

height: 50px;

.popup-content, .buttonContainer {

display: none;

}

.popup-header{

height: 50px;

background-color: lighten(green, 50%);

border: 1px solid green;

font-size: 14px;

padding: 15px 50px;

color: green;

border-radius: 5px;

}

}

div.error,

p.error {

border: 1px solid red;

border-radius: 2px;

font-size: 14px;

margin: 0px 25px;

padding: 10px;

background-color: lighten(red, 45%);

border-color: red;

color: red;

}

Файл \_header.scss

.logo {

padding: 15px;

}

.header img:hover {

color: #fff;

}

.timelang {

display:inline;

}

.fa-sign-in-alt, .fa-cog {

font-size: 30px;

color: #c9aa03;

}

.btns p{

font-family: "RobotoMedium";

display: inline;

color: #000;

}

.btns button{

margin-right: 5px;

border: 1px solid #000;

height: 26px;

font-family: "RobotoBold";

font-size: 14px;

border-radius: 2px;

background-color: #c9aa03;

color: #000;

cursor: pointer;

float: right;

transition: 0.5s;

padding: 4px 5px;

}

.fa-sign-in-alt, .fa-cog {

font-size: 16px;

color: #000;

float: right;

padding-left: 5px;

}

button:hover {

color: #000;

background-color: #ffd804;

}

.header {

background: #04883b;

}

.container {

padding: 0 0;

background-color: #fff;

}

.string p {

font-size: 16px;

padding: 6px 0px 6px 0px;

color: #fff;

}

Файл \_main.scss

.left-sidebar-menu li, .left-side-menu li{

list-style-type: none;

padding: 5px;

}

.left-sidebar {

padding-top: 5px;

}

.left-sidebar a, .left-sidebar-menu a{

display: inline-block;

color:#fff;

line-height: 1;

text-decoration:none;

cursor: pointer;

//padding-left: 10px;

}

.left-sidebar a:after, .left-sidebar-menu a:after {

background-color: #ffeb3b;

display: block;

content: "";

height: 2px;

width: 0%;

-webkit-transition: width .2s ease-in-out;

-moz--transition: width .2s ease-in-out;

transition: width .2s ease-in-out;

}

.left-sidebar-menu a:hover:after, .left-sidebar a:hover:after,

a:focus:after {

width: 100%;

}

.left-sidebar p {

background-color: #04883b;

color: #fff;

padding: 6px 8px 6px 10px;

text-decoration: none;

font-family: "RobotoMedium";

font-size: 12px;

}

.leftbar\_btn, .leftbar-btn {

padding: 6px 8px 6px 10px;

text-decoration: none;

font-family: "RobotoMedium";

font-size: 12px;

color: #000000;

display: block;

border: none;

background: $color-gold;

width: 100%;

text-align: left;

outline: none;

transition: 0.5s;

}

.country {

background-color: $color-grey;

}

// .leftbar-btn:hover {

// background-color: #0aee28;

// }

.country:hover {

background-color: #d5d3d1;

}

.left-side-menu {

background-color: $color-dark-grey;

padding-left: 8px;

font-size: 12px;

word-wrap: break-word;

}

.left-sidebar-menu {

display: none;

background-color: $color-dark-grey;

padding-left: 8px;

font-size: 10px;

word-wrap: break-word;

}

.left-sidebar img, .tabContainer img {

width: 18px;

height: 12px;

}

.fa-caret-down{

float: right;

padding-right: 0px;

}

.fa-times, .fa-plus {

float: right;

padding: 3px;

cursor: pointer;

}

.fa-times:hover, .fa-plus:hover {

color: #fff;

transition: 0.5s;

}

.matches .fa-star {

padding-left: 5px;

cursor: pointer;

}

.matches .fa-star:hover {

color: $color-gold;

transition: 0.5s;

}

.active {

color: $color-gold;

}

hr {

border: none;

background-color: #000000;

color: #000000;

height: 1px;

}

.matches, .ttable {

font-family: "RobotoMedium";

font-size: 12px;

border-collapse: collapse;

width: 100%;

table-layout: fixed;

}

.ttable {

margin-top: 5px;

}

.matches img {

margin-right: 10px;

}

.matches td, .matches th, .ttable td, .ttable th {

border: 1px solid #807f7e;

padding: 2px 0px;

text-align: center;

padding-left: 5px;

}

.matches td, .ttable td{

cursor: pointer;

color: #000;

table-layout: fixed;

padding: 0px 5px;

}

.matches tr:nth-child(2n+1){background-color: #e5e3e2;}

.matches tr:nth-child(2n){background-color: #bfbebd;}

.ttable tr:nth-child(2n+1){background-color: #e5e3e2;}

.ttable tr:nth-child(2n){background-color: #bfbebd;}

.matches tr:hover, .ttable tr:hover{background-color: #e5d4cc; transition: 0.5s;}

.matches th {

padding: 5px 10px;

text-align: left;

background-color: #555250;

font-family: "RobotoLight";

color: #fff;

}

.ttable th {

padding: 5px;

text-align: center;

background-color: $color-gold;

color: #000;

}

Файл \_reset.scss

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

textarea, input, button {

outline: none;

}

Файл \_tabs.scss

.dateofmatch{

width: 100%;

padding: 5px;

}

.tabContainer{

width: 100%;

padding: 15px 5px;

}

.buttonContainer{

height: 30px;

}

.buttonContainer button{

width: 20%;

height: 30px;

float: left;

border: none;

outline:none;

cursor: pointer;

padding: 10px;

font-family: "RobotoMedium";

font-size: 12px;

background-color: #eee;

}

.buttonContainer button:hover{

background-color: #d7d4d4;

transition: 0.5s;

}

.tabPanel {

background-color: gray;

color: white;

text-align: center;

box-sizing: border-box;

font-family: sans-serif;

font-size: 12px;

display: none;

padding-top: 13px;

}

.dateofmatch button{

width: 9.09%;

height: 27px;

font-family: "RobotoMedium";

font-size: 12px;

float: left;

border: none;

outline:none;

cursor: pointer;

padding: 4px;

background-color: #04883b;

color: #fff;

}

.dateofmatch button:hover {

transition: 0.5s;

}

Файл \_style.scss

$color-white: #fffdfc;

$color-light-grey: #e5e3e2;

$color-midle-grey: #bfbebd;

$color-grey: #807f7e;

$color-dark-grey: #403f3f;

$color-gold: #c9aa03;

@import 'fonts';

@import 'reset';

@import 'tabs';

html {

font-size: 12px;

font-family: "RobotoMedium";

background-color: $color-light-grey;

}

@import 'header';

@import 'main';

@import 'footer';

Файл app.js

/\* eslint-disable no-console \*/

const express = require('express');

const bodyParser = require('body-parser');

const path = require('path');

const staticAsset = require('static-asset');

const mongoose = require('mongoose');

const session = require('express-session');

const MongoStore = require('connect-mongo')(session);

const fetch = require('node-fetch');

const config = require('./config');

const routes = require('./routes');

// database

mongoose.Promise = global.Promise;

mongoose.set('debug', config.IS\_PRODUCTION);

mongoose.connection

.on('error', error => console.log(error))

.on('close', () => console.log('Database connection closed.'))

.once('open', () => {

const info = mongoose.connections[0];

console.log(`Connected to ${info.host}:${info.port}/${info.name}`);

});

mongoose.connect(config.MONGO\_URL, { useNewUrlParser: true });

// express

const app = express();

app.use(

session({

secret: config.SESSION\_SECRET,

resave: true,

saveUninitialized: false,

store: new MongoStore({

mongooseConnection: mongoose.connection

})

})

);

// sets and uses

app.set('view engine', 'ejs');

app.use(bodyParser.urlencoded({ extended: true }));

app.use(bodyParser.json());

app.use(staticAsset(path.join(\_\_dirname, 'public')));

app.use(express.static(path.join(\_\_dirname, 'public')));

app.use(

'/javascripts',

express.static(path.join(\_\_dirname, 'node\_modules', 'jquery', 'dist'))

);

var request = require('request');

request('https://apifootball.com/api/?action=get\_leagues&APIkey=004b26f0bdc9085ae7d6916dc3aa7b7a5c81af51dde4720ab74ea13f1759806a', function (error, response, body) {

if (!error && response.statusCode == 200) {

leagues = JSON.parse(body).sort(function(obj1, obj2) {

if (obj1.league\_name < obj2.league\_name) return -1;

if (obj1.league\_name > obj2.league\_name) return 1;

return 0;

});

}

})

request('https://apifootball.com/api/?action=get\_countries&APIkey=004b26f0bdc9085ae7d6916dc3aa7b7a5c81af51dde4720ab74ea13f1759806a', function (error, response, body) {

if (!error && response.statusCode == 200) {

cont = JSON.parse(body).sort(function(obj1, obj2) {

if (obj1.country\_name < obj2.country\_name) return -1;

if (obj1.country\_name > obj2.country\_name) return 1;

return 0;

});

}

})

request('https://apifootball.com/api/?action=get\_events&from=2019-05-14&to=2019-05-14&APIkey=004b26f0bdc9085ae7d6916dc3aa7b7a5c81af51dde4720ab74ea13f1759806a', function (error, response, body) {

if (!error && response.statusCode == 200) {

games = JSON.parse(body).sort(

function(a, b) {

if (a.country\_name === b.country\_name) {

return a.league\_name > b.league\_name ? 1 : -1;

}

return a.country\_name > b.country\_name ? 1 : -1;

});

var temp = games[0].league\_name;

for(var i = 0; i < games.length; i++) {

if((i == 0 ) || (temp != games[i].league\_name)) {

country\_league.push(games[i]);

count\_match.push(i);

temp = games[i].league\_name;

}

}

count\_match.push(games.length);

}

})

request('https://apifootball.com/api/?action=get\_events&from=2019-05-14&to=2019-05-14&APIkey=004b26f0bdc9085ae7d6916dc3aa7b7a5c81af51dde4720ab74ea13f1759806a', function (error, response, body) {

if (!error && response.statusCode == 200) {

var end = JSON.parse(body).sort(

function(a, b) {

if (a.country\_name === b.country\_name) {

return a.league\_name > b.league\_name ? 1 : -1;

}

return a.country\_name > b.country\_name ? 1 : -1;

});

for(var b = 0; b < end.length; b++) {

if(end[b].match\_status == 'FT') {

end\_games.push(end[b]);

}

}

var temp = end\_games[0].league\_name;

for(var i = 0; i < end\_games.length; i++) {

if((i == 0 ) || (temp != end\_games[i].league\_name)) {

country\_league.push(end\_games[i]);

count\_end.push(i);

temp = end\_games[i].league\_name;

}

}

count\_end.push(end\_games.length);

}

})

// request('https://apifootball.com/api/?action=get\_events&from=2019-05-10&to=2019-05-10&match\_live=1&APIkey=004b26f0bdc9085ae7d6916dc3aa7b7a5c81af51dde4720ab74ea13f1759806a', function (error, response, body) {

// if (!error && response.statusCode == 200) {

// if(JSON.parse(body).error == 404) {

// console.log(JSON.parse(body).message);

// } else {

// games\_live = JSON.parse(body).sort(

// function(a, b) {

// if (a.country\_name === b.country\_name) {

// return a.league\_name > b.league\_name ? 1 : -1;

// }

// return a.country\_name > b.country\_name ? 1 : -1;

// });

// var temp = games\_live[0].league\_name;

// for(var i = 0; i < games\_live.length; i++) {

// if((i == 0 ) || (temp != games\_live[i].league\_name)) {

// live\_con.push(games\_live[i]);

// live\_matches.push(i);

// temp = games\_live[i].league\_name;

// }

// }

// live\_matches.push(games\_live.length);

// }

// }

// })

function liveMatch() {

fetch('https://apifootball.com/api/?action=get\_events&from=2019-05-14&to=2019-05-14&match\_live=1&APIkey=004b26f0bdc9085ae7d6916dc3aa7b7a5c81af51dde4720ab74ea13f1759806a')

.then(resp => resp.json())

.then(json => {

if(json.error == 404) {

console.log(json.message);

} else {

games\_live = json.sort(

function(a, b) {

if (a.country\_name === b.country\_name) {

return a.league\_name > b.league\_name ? 1 : -1;

}

return a.country\_name > b.country\_name ? 1 : -1;

});

var temp = games\_live[0].league\_name;

for(var i = 0; i < games\_live.length; i++) {

if((i == 0 ) || (temp != games\_live[i].league\_name)) {

live\_con.push(games\_live[i]);

live\_matches.push(i);

temp = games\_live[i].league\_name;

}

}

live\_matches.push(games\_live.length);

}

setTimeout(liveMatch, 60 \* 1000);

}).catch(function (err) {

console.error(err);

setTimeout(liveMatch, 6 \* 1000);

});

}

request('https://apifootball.com/api/?action=get\_standings&league\_id=62&APIkey=004b26f0bdc9085ae7d6916dc3aa7b7a5c81af51dde4720ab74ea13f1759806a', function (error, response, body) {

if (!error && response.statusCode == 200) {

ttable = JSON.parse(body).sort(function(a,b){return a.overall\_league\_position-b.overall\_league\_position;});

}

})

var cont = null;

var leagues = null;

let games = null;

var ttable = null;

var games\_live = null;

var end\_games = [];

var country\_league = [];

var count\_match = [];

var live\_con = [];

var live\_matches = [];

var count\_end = [];

// routers

app.get('/', (req, res) => {

const id = req.session.userId;

const email = req.session.userEmail;

liveMatch();

res.render('index', {

user: {

id,

email,

},

data: {

leagues,

cont,

games,

ttable,

country\_league,

count\_match,

live\_con,

live\_matches,

games\_live

}

});

});

app.use('/api/auth', routes.auth);

//var url = 'https://apifootball.com/api/?action=get\_leagues&APIkey=004b26f0bdc9085ae7d6916dc3aa7b7a5c81af51dde4720ab74ea13f1759806a';

// catch 404 and forward to error handler

app.use((req, res, next) => {

const err = new Error('Not Found');

err.status = 404;

next(err);

});

// error handler

// eslint-disable-next-line no-unused-vars

app.use((error, req, res, next) => {

res.status(error.status || 500);

res.render('error', {

message: error.message,

error: !config.IS\_PRODUCTION ? error : {}

});

});

app.listen(config.PORT, () =>

console.log(`Example app listening on port ${config.PORT}!`)

);

Файл config.js

const dotenv = require('dotenv');

const path = require('path');

const root = path.join.bind(this, \_\_dirname);

dotenv.config({ path: root('.env') });

module.exports = {

PORT: process.env.PORT || 3000,

MONGO\_URL: process.env.MONGO\_URL,

SESSION\_SECRET: process.env.SESSION\_SECRET,

IS\_PRODUCTION: process.env.NODE\_ENV === 'production',

API\_KEY: process.env.API\_KEY

}

Файл gulpfile.js

const gulp = require('gulp');

const sass = require('gulp-sass');

const autoprefixer = require('gulp-autoprefixer');

const cssnano = require('gulp-cssnano');

var paths = {

sass: {

src: './dev/scss/\*\*/\*.{scss,sass}',

dist: './dist/\*.html',

opts: {

}

}

};

gulp.task('scss', function () {

return gulp

.src('dev/scss/\*\*/\*.scss')

.pipe(sass())

.pipe(

autoprefixer(['last 15 versions', '>1%', 'ie 8', 'ie 7'], {

cascade: true

})

)

.pipe(cssnano())

.pipe(gulp.dest('public/stylesheets'))

});

gulp.task('watch:styles', function () {

gulp.watch(paths.sass.src, gulp.series('scss'));

});

gulp.task('watch', gulp.series('scss',

gulp.parallel('watch:styles')

));

gulp.task('default', gulp.series('scss',

gulp.parallel('watch:styles')

));

Файл nodemon.js

{

"ignore": ["node\_modules", "dev", "public"],

"env": {

"NODE\_ENV": "development",

"PORT": 3000,

"MONGO\_URL": "mongodb://localhost/EGScore"

},

"ext": ".js,.json",

"watch": "./"

}

Файл package.json

{

"name": "EduardGorbatyuk.github.io",

"version": "1.0.0",

"description": "MyScore",

"scripts": {

"start": "node app.js",

"gulp": "gulp",

"nodemon": "nodemon app.js",

"dev": "concurrently --kill-others \"npm run gulp\" \"npm run nodemon\""

},

"repository": {

"type": "git",

"url": "git+https://github.com/EduardGorbatyuk/EduardGorbatyuk.github.io.git"

},

"keywords": [],

"author": "",

"license": "ISC",

"bugs": {

"url": "https://github.com/EduardGorbatyuk/EduardGorbatyuk.github.io/issues"

},

"homepage": "https://github.com/EduardGorbatyuk/EduardGorbatyuk.github.io#readme",

"devDependencies": {

"concurrently": "^4.1.0",

"eslint": "^5.16.0",

"eslint-plugin-node": "^8.0.1",

"nodemon": "^1.19.0"

},

"engines": {

"node": ">=6.0.0"

},

"dependencies": {

"bcrypt-nodejs": "0.0.3",

"body-parser": "^1.19.0",

"browser-sync": "^2.26.5",

"connect-mongo": "^2.0.3",

"dotenv": "^8.0.0",

"ejs": "^2.6.1",

"express": "^4.16.4",

"express-session": "^1.16.1",

"gulp": "^4.0.1",

"gulp-autoprefixer": "^6.1.0",

"gulp-concat": "^2.6.1",

"gulp-cssnano": "^2.1.3",

"gulp-nodemon": "^2.4.2",

"gulp-plumber": "^1.2.1",

"gulp-sass": "^4.0.2",

"gulp-uglifyjs": "^0.6.2",

"i18n-abide": "0.0.26",

"jquery": "^3.4.1",

"mongoose": "^5.5.5",

"node-fetch": "^2.5.0",

"request": "^2.88.0",

"static-asset": "^0.6.0",

"xmlhttprequest": "^1.8.0"

}

}

# ПРИЛОЖЕНИЕ Д

ЭКРАННЫЕ ФОРМЫ