**СОФИЙСКИ УНИВЕРСИТЕТ „СВ. КЛИМЕНТ ОХРИДСКИ“**

**ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА**



**ИМЕ НА ПРОЕКТ**

**„VolleyStats-WEB“**

**Изготвил:**

Антони Добренов – ФН: 81454

**Специалност:**

„Компютърни науки“

11.09.2020г София

**Функционални изисквания**

1. *REST API* трябва да преодставя следните възможности на следните групи потребители:
   1. *Statist трябва да може да:*
      1. Да се логва в акаунта си
      2. Да излиза от акаунта си
      3. Да добавя отбор
      4. Да изтрива отбор
      5. Да редактира отбор
      6. Да създава играчи
      7. Да редактира играчи
      8. Да изтрива играчи
      9. Да стартира мачове
      10. Да въвежда статистика
      11. Да преглежда статистика
   2. *Fan трябва да може да:*
      1. Да се логва в акаунта си
      2. Да излиза от акаунта си
      3. Да гледа статистики
   3. *Player трябва да може да:*
      1. Да се логва в акаунта си
      2. Да излиза от акаунта си
      3. Да гледа статистика на отбора си
   4. *Anonymous трябва да може да:*
      1. Да се регистрира, след което вече има акаунт, с който може да се логне

**Нефункционални изисквания**

1. **Сигурност (Security)** - Всеки потребител от горе-посочените роли трябва да има достъп само до своите данни и функционалности, които да ги мениджират, без да може да достъпва функционалност и данни, без необходимата оторизация.
2. **Производителност** - Да се постигне възможно най - висока производителност
3. **Наличност** - Да е налична максимално време, с възможно най - кратки прекъсвания на услугата
4. **Модифицируемост** - Сорс кодът на проекта да е организиран така, че лесно да може да се променят, модули, които могат да търпят чести модификации
5. **Използваемост** - Предоставяне на удобен *WEB* интерфейс за комуникация с REST API частта.

**Използвани технологии и модули**

1. Реализация на *REST API* с *NodeJS* и *ExpressJS*:
   1. **NodeJS v12.14.1**

<https://nodejs.org/en/blog/release/v12.14.1/>

* 1. **Express Framework** - Open source минималистична технология(framework), която предоставя възможност за лесна организация на web приложения по MVC архитектура, използвайки JavaScript.

<https://expressjs.com/>

* 1. **Mongoose** - Библиотека за Обектно моделиране на записи в MongoDB и NodeJS. Предоставя лесно управление на връзките между данните, лесна валидация и методи за репрезентация на данните в JSON формат.

<https://mongoosejs.com/>

* 1. **JSON Web token** - Библиотека, която се използва за оторизация, чрез създаване на JSON Web token, който се използва за логин и валидация на потребителите, вместо сесия. Тя позволява генериране и валидация на JSON Web token.

1. Реализация на *WEB* приложение с *ReactJ* и *React Router:*
   1. ***ReactJS*** *– Font-end* библиотека*(framework)*, позволяваща реализирането *Single page applications SPA*

<https://reactjs.org/>

* 1. **react-router** и **react-router-dom** -Библиотека, която позволява **“рутирането”** между различните части на приложението, когато потребителят въведе определен **URL**, или клика на линк, бутон и т.н.

<https://reactrouter.com/web/guides/quick-start>

* 1. **material-ui** - Библиотека, която предоставя готови компоненти.

**Нетривиални аспекти на системата**

Реализация на бизнес модела на система, в която си взаимодействат потребители с някоя от три роли, Admin, Teacher и Student в контекста на проектираната система специфични за конкретната тематика на решавания проблем.

Системата се състои от две части, сървърна апликация реализирана на база на *ExpressJS + NodeJS сървър* и клиентска част, която е реализирана на база архитектурата - SPA, постигнато чрез използването на библиотеките - *ReactJS* и *React Router.*

Системата е разделена на 4 главни части - функционалност за “регистрация и логване”, функционалност за “създаване на отбор”, функционалност за “добавяне на играчи в този отбор ”, функционалност за “започване на мач и въвеждане на статистика за него”.

**Значими интерфейси**

|  |  |  |
| --- | --- | --- |
| **Метод** | **URL** | **Описание** |
| **Users routes** | | |
| **GET** | **/api/users** | **Връща масив с всички потребители** |
| **GET** | **/api/users/:userId** | **Връща потребителя с избраното ID, ако той съществува, иначе грешка с подходящо съобщение** |
| **POST** | **/api/users/auth/register** | **Позволява на потребители да се регистрират** |
| **POST** | **/api/users/auth/login** | **Позволява на потребители да се логват** |
| **DELETE** | **/api/users/:userId** | **Позволява на потребител по подадено “usedId” да изтрие профил, ако е негов, или ако е с роля “Admin”** |
| **PUT** | **/api/users/:userId** | **Позволява промяна на профил на потребител по подаден “userId”** |
| **PATCH** | **/api/users/auth/passwordchange/:userId/:currentPassword/:newPassword** | **Позволява смяна на паролата на потребител с нова по подадени “userId”, “currentPassword” и “newPassword”** |
| **Courses routes** | | |
| **GET** | **/api/team** | **Връща масив с всички отбори** |
| **GET** | **/api/user/:userId/team** | **Връща масив от всички отбори на даден потребител** |
| **GET** | **/api/user/:userId/team/:teamId** | **Връща конкретен отбор по ID** |
| **POST** | **/api/team** | **Позволява създаването на нов отбор, като връща създадения отбор** |
| **PUT** | **/api/team/:teamId** | **Позволява промяната на отбор по подаденo “teamId”** |
| **DELETE** | **/api/team/:teamId** | **Позволява премахването на отбор по подаденo “teamId”** |
| **GET** | **/api/player** | **Връща масив от всички играчи** |
| **GET** | **/api/team/:teamId/player** | **Връща масив от всички играчи на отбор** |
| **GET** | **/api/team/:teamId/player/:playerId** | **Връща конкретен играч от отбор** |
| **PUT** | **/api/team/:teamId/player/:playerId** | **Редактира конкретен играч от отбор** |
| **DELETE** | **/api/team/:teamId/player/:playerId** | **Премахва конкретен играч от отбор** |
| **POST** | **/api/team/:teamId/match** | **Стартиране на мач с отбор** |
| **GET** | **/api/match** | **Връща всички мачове** |
| **GET** | **/api/team/:teamId/match** | **Връща всички мачове на конкретен отбор** |
| **GET** | **/api/match/:matchId** | **Взима конкретен мач** |
| **PUT** | **/api/match/:matchId** | **Позволява да бъде променян конкретен мач** |
| **DELETE** | **/api/match/:matchId** | **Позволява да бъде изтрит конкретен мач** |

**Инсталация и конфигуриране**

Системата е проектирана и тествана на основа ***NodeJS v12.14.1*** и ***ReactJS v16****.*

За да се стартира сървърната част е ползвана командата “node server.js”.

Преди да може да бъде стартирано приложението е нужно изтеглянето на всички dependency-та описани в секция “Използвани технологии и модули” използвайки ***npm***или ***yarn***. Това може да стане най-лесно, като в главната директория на проекта се изпълни командата “npm isntall”, която автоматично ще инсталира всички модули, които са записани под секциите: “dependencies” и “devDependencies” във файла package.json, който е наличен в репозиторите на проекта.

За да се стартира ReactJS front-end апликацията трябва да бъде изпълнена командата “npm install”, в директорията на web апликацията, която автоматично ще инсталира всички модули, които са записани под секциите: “dependencies” и “devDependencies” във файла package.json, който е наличен в репозиторите на проекта. След това може да се стартира проложението в development сървър с командата “npm start”, изпълнена от директорията на web апликацията. Тя ще стартира сървъра на “<http://localhost:8080>” по подразбиране.

**Потребителска документация**

На потребителя е предоставен удобен интерфейс за работа със системата под формата на *WEB SPA*. Може също да бъде използван всяка една web клиент апликация.

**Заключение**

Реализацията на този проект доведе до запознанството ми с много нови технологии и техните специфики, като например: **ReactJS, react-router, material-ui, Redux,** **NodeJS, JWT архитектура** използване на **Http протокола** за комуникацияи други.

Трудности, срещнати по време на реализацията бяха основно свързани с front-end частта й, поради тази причина, екраните не изглеждат подобаващо.

Плановете за бъдещо развитие са свързани с подобряване на GUI на WEB апликацията, намиране и отстраняване на допуснати грешки и дупки в сигурността, рефакториране на част от кода, добавяне на нови функционалности - като въвеждане на различни статистики, добавяне на снимки на играчите и отборите.

**Източници**

<https://mongoosejs.com/> - Mongoose web page for NodeJS

<https://reactrouter.com/web/guides/quick-start> - react-router-dom documentation web page

<https://reactjs.org/tutorial/tutorial.html> - ReactJS web page

<https://reactjs.org/docs/hooks-overview.html#state-hook>- React JS Hooks

<https://expressjs.com/>- ExpressJS page

<https://nodejs.org/en/docs/> - NodeJS page