СОФИЙСКИ УНИВЕРСИТЕТ „СВ. КЛИМЕНТ ОХРИДСКИ“

ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА

ИМЕ НА ПРОЕКТ

„University courses manager“

Изготвил:

Траян Троев – ФН: 81438

Специалност:

„Компютърни науки“

25.07.2020г София

Функционални изисквания

1. *REST API* трябва да преодставя следните възможности на следните групи потребители:
    1.1. *Админ (Admin) трябва да може да:*
        1.1.1. Да се логва в акаунта си
        1.1.2. Да излиза от акаунта си
        1.1.3. Да добавя курс
        1.1.4. Да изтрива курс
        1.1.5. Да редактира курс
        1.1.6. Да поставя оценка на студент от курс
        1.1.7. Да сменя паролата си

    1.2. *Преподавател (Teacher) трябва да може да:*
        1.2.1. Да се логва в акаунта си
        1.2.2. Да излиза от акаунта си
        1.2.3. Да добавя курс
        1.2.4. Да изтрива СВОЙ курс
        1.2.5. Да редактира СВОЙ курс
        1.2.6. Да премахва студент от СВОЙ курс
        1.2.7. Да поставя оценка на студент от свой курс, или курс в който е лектор
        1.2.8. Да сменя паролата си

    1.3. *Студент (Student) трябва да може да:*
        1.3.1. Да се логва в акаунта си
        1.3.2. Да излиза от акаунта си
        1.3.3. Да се записва за курс, ако той не е започнал
        1.3.4. Да се отписва от курс, ако той не е започнал
        1.3.5. Да сменя паролата си

    1.4.*Анонимен (Anonymous) трябва да може да:*
        1.4.1. 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. **Тестваемост** - Source кода да е организиран, така че лесно да се тестват отделните модули
6. **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*:
   1.1. **NodeJS v12.14.1**

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

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

   https://expressjs.com/

   1.3. **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/

   1.4. **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.
2. Реализация на *WEB* приложение с *ReactJ* и *React Router:*

2.1. ***ReactJS*** – *Font-end* библиотека*(framework)*, позволяваща реализирането *Single page applications SPA*

https://reactjs.org/

2.2. **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

2.3. **formik** - Библиотека за създаване и валидиране на форми, удобна за ползване при реализирането на реакт компонент, представляващ форма

2.4. **secure-ls** - Библиотека позволяваща работата с localStorage на браузъра. Тя предоставя удобно API за тази цел.

2.5. **semantic-ui-react/css** - Библиотека улесняваща реализацията на responsive UI дизайн, чрез предварително wrap-нати компоненти, с предефинирани стилове от *semantic-ui-css*

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 “поставяне на оценка”, функционалност за “записване в курс”, функционалност за управление на “курсове”. Това води до по - лесното дефиниране и изпълнение на *функционалните изисквания*.

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/courses**	**Връща масив с всички курсове**
GET	**/api/courses/:courseId**	**Връща курс по подадено “courseId”**
POST	**/api/courses**	**Позволява създаването на нов курс, като връща създадения курс, както и location хедър със стойността на URI до него**
PUT	**/api/courses/:userId/:courseId**	**Позволява промяната на курс по подадени “userId” и “courseId”, ако курса е на логнатият потребител или логнатият потребител е админ**
PATCH	**/api/courses/transfer/:courseId/:userFrom/:userTo**	**Позволява промяната на собственика на курс по подадени “courseId”, “userFrom”, “user ”, ако курсът е на логнатият потребител или логнатият потребител е админ**
DELETE	**/api/courses/:userId/:courseId**	**Позволява да бъде изтрит курс по даден “userId” и “courseId” на логнатият потребител, ако “userId” е неговото ID или ако логнатият потребител е админ**

		или собственик на курса.
PATCH	**/api/courses/enrol/:courseId/:userId**	**Позволява да бъде записан “студент” в даден курс по подадени “courseId” и ”userId”, преди неговото начало, ако логнатият потребител е този с подаденото “userId” или е собственик на курса, или е админ**
PATCH	**/api/courses/cancelenrolment/:courseId/:userId**	**Позволява да бъде отписан “студент” в даден курс по подадени “courseId” и ”userId”, преди неговото начало, ако логнатият потребител е този с подаденото “userId” или е собственик на курса, или е админ**
Grades routes		
GET	**/api/grades**	**Връща всички оценки**
GET	**/api/grades/:gradeId**	**Връща оценка по подадено “gradeId”**
GET	**/api/grades/byuser/:userId**	**Връща всички оцени на даден потребител по подадено “userId”**
GET	**/api/grades/bycourse/:courseId**	**Връща оценките от даден курс по подадено “courseId”**
GET	**/api/grades/user/:userId/:courseId**	**Връща оценките от**

		даден курс по подадено “courseId”
POST	**/api/grades**	**Позволява да бъде създадена оценка**
PUT	**/api/grades/:userId/:gradeId**	**Позволява да бъде променяна оценка на студент по подадени “userId” и ”gradeId”**
DELETE	**/api/grades/:gradeId**	**Позволява да бъде изтрита оценка на студент по подадено ”gradeId”**
Chat routes		
GET	**/api/chat/messages/:receiver**	**Връща всички съобщения между логнатия потребител и получателя, определен от подаденото ID на получателя, обозначено като “receiver”.**
POST	**/api/chat/message**	**Позволява на логантия потребител да изпрати съобщение**

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:3000” 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, semantic-ui, NodeJS, ExpressJS, JWT архитектура, MVC архитектура**, използване на **Http протокола** за комуникация и други.

Трудности, срещнати по време на реализацията бяха основно свързани с конфигуриране expressJS да работи заедно със socket.io за реализацията на чат функционалност, като front-end частта й, поради тази причина, липсва. В процеса на запознаването ми с socket.io позволяват да бъде предвидено имплементирането и за в бъдеще.

Плановете за бъдещо развитие са свързани с подобряване на GUI на WEB апликацията, намиране и отстраняване на допуснати грешки и дупки

в сигурността, както и разширяване на системата с още функционалности, от които има нужда конкретния бизнес модел.

Източници

https://github.com/iproduct/course-node-express-react/wiki - FMI node-express-react course, by Trayan Iliev, 2019/2020 educational year edition

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

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

https://scotch.io/tutorials/authenticate-a-node-es6-api-with-json-web-tokens - Authenticate a Node ES6 API with JSON Web Tokens by Mabishi Wakio, October 08, 2018

https://medium.com/better-programming/a-practical-guide-for-jwt-authentication-using-nodejs-and-express-d48369e7e6d4 - A Practical Guide for JWT Authentication Using Node.js and Express, by Anshul Goyal, January 1, 2019

https://steveholgado.com/rxjs-chat-app/ - Build a chat app with RxJS and Socket.IO, by Steve Holgado, 2019

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

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

https://formik.org/ - Formik page

https://expressjs.com/- ExpressJS page

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