СОФИЙСКИ УНИВЕРСИТЕТ „СВ. КЛИМЕНТ ОХРИДСКИ“
ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА

ИМЕ НА ПРОЕКТ

„Payments Tracker“

Изготвил:

Станислав Климов – ФН: 61942

Специалност:

„Софтуерно инженерство“

10.09.2020 София

Функционални изисквания

1. *REST API* трябва да предоставя следните възможности на следните групи потребители:
   1.1. *Потребител (User) 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 плащания
      1.1.4. Да изтрива плащания
      1.1.5. Да добавя категории
      1.1.6. Да изтрива категории

   1.2. *Анонимен (Anonymous) трябва да може да:*
      1.2.1. 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 менажира, 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 v12.18.2**

   https://nodejs.org/en/blog/release/v12.18.2/

   1.2. **Typescript v3.5.3**

   https://www.typescriptlang.org/docs/handbook/release-notes/typescript-3-5.html

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

   https://expressjs.com/

   1.4. **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/

   1.5. **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.

1.6. **BCrypt** - Библиотека, която се използва за криптирането на потребителските пароли преди да се запишат в базата.

1.7. **Inversify** - Библиотека, която е inversion of control (IoC) контейнер за TypeScript и JavaScript приложения.

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 кликане на линк, бутон и т.н.
https://reactrouter.com/web/guides/quick-start

2.3. **axios** - Библиотека, която позволява HTTP извиквания за браузъра и node.js

Нетривиални аспекти на системата

Реализация на бизнес модела на система, в която потребители могат да добавят плащания.

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.*

Значими интерфейси

Метод	URL	Описание
Users routes		
POST	**/signup**	Позволява на нерегистрирани потребители да се регистрират в системата
POST	**/login**	Позволява на потребители да се логват в системата
POST	**/logout**	Позволява на потребители да излязат от системата
GET	**/**	Позволява на регистрирани и нерегистрирани потребители да видят началната страница

		на проекта
GET	**/about**	Позволява на регистрирани и нерегистрирани потребители да видят страница с информация на проекта
POST	**/userhome**	Позволява на потребители да видят тяхната лична начална страница, където се намира и техния личен списък с плащания
POST	**/categories**	Позволява промяна да добавя категории за неговия личен списък

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 install”, която автоматично ще инсталира всички модули, които са записани под секциите: “dependencies” и “devDependencies” във файла package.json, който е наличен в репозиторито 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 репозиторитона проекта. След това може да се стартира приложението 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*.

Заключение

Реализацията на този проект доведе до запознанството ми с много нови технологии и техните специфики, като например: **ReactJS, react-router**. Както и до развиването на знанията ми на някои технологии, като например: **NodeJS, ExpressJS, JWT архитектура, MVC архитектура**, използване на **Http протокола** за комуникация и други.

Трудности, срещнати по време на реализацията бяха основно свързани работата с reactJS, поради факта, че никога не съм го използвал в работата си, и по-точно създаване на динамична таблица в която да мога да добавям и трия елементи (и съответно ги записвам базата данни).

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 функционалности, които първоначално бях поставил като цели.

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://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

https://github.com/inversify/InversifyJS - InversifyJS page