

СУ „Св. Климент Охридски“

Документация към проект   
Fintooth, изработен като курсов проект за курса

„Разработка на клиент-сървър (fullstack) приложения с Node.js + Express.js + React.js“

Изработили:

Теодор Йоловски – 81411 Дата:25.07.2020

Теодор Грозданов – 81560

Ростислав Георгиев – 81482

По-голямата част от фунционалността, описана предварително в Project Summary е изпълнена:

Като user, аз мога:  
-да си създам акаунт в сайта  
-да се логвам в сайта при създаден акаунт

-да създавам групи

-да участвам в групи

-да си видя дейностите

-да създавам дейности

-да редактирам дейности

-да трия дейности

-да си добавям сметки

-да трия сметки

-да си си променям потребителското име/паролата

-да си променям имейла

-да си изтрия акаунта в сайта

-да видя графика, представяща промяната по сметките

-да видя дейностите за групата, в която съм

Като админ фунцкионалността е същата, плюс допълнително:

-да трия произволен user

-да редактирам произволен user  
-да добавя user

-да трия произволна група  
-да създам група.

Технологиите, използвани за създаването на проекта са:  
-Front-end:   
 React.js - <https://reactjs.org/>

Redux - <https://redux.js.org/>

Reduc-Saga - <https://redux-saga.js.org/>

Material-UI - <https://material-ui.com/>

Axios -

-Back-end:

Express - <https://expressjs.com/>

Mongoose - <https://mongoosejs.com/>

JWT - <https://jwt.io/>

За реализация на функционалните и нефунционалните изисквания, комуникацията между front-end и back-end става посредство заявка, изпратена асинхронно с axios. Заявката трябва да бъде асинхронна, а Redux store-ът е синхронен. Тук идва ролята на Redux-Saga, която предизвиква заявката и служи за връзка между асинхронната заявка и синхронния Redux store. За авторизация на заявките, използваме JWT – JSON Web Token. В него се кодира обектът {email, nickname, admin, id}. Посредством това кодиране потребителят може да се верифицира, че е той, когато се пращат заявки към базата. По този начин се покачва сигурността. Чрез Express ни е предоставен интерфейс, чрез който ни е улеснен достъпът до базата, която е MongoDB, която е моделирана чрез Mongoose.

Описание на REST service и API протокола за неговото използване:

* GET към <http://localhost:3001/users/> - връща всичките потребители
* GET към <http://localhost:3001/users/:userId/accounts> - връща всичките сметки на потребителя
* GET към <http://localhost:3001/users/:userid> – връща конкретен потребител с това ид
* POST към http://localhost:3001/users/signup/ - регистрира потребител в системата
* POST към <http://localhost:3001/users/login> - потребителят логин в системата
* POST към <http://localhost:3001/users/userid/accounts> - добавяне на сметка
* PATCH към <http://localhost:3001/users/change-password/:userId> – смяна на парола
* PACTH към <http://localhost:3001/users/userid/accounts> - промяна на nickname и имейл
* DELTE към <http://localhost:3001/users/:userid> – изтриване на потребител
* DELTE към http://localhost:3001/users/:userId/accounts/:accId – изтриване на потребител
* GET към http://localhost:3001/groups - връща всичките групи
* GET към http://localhost:3001/groups/groupId - връща конкретна група с това ид
* POST към http://localhost:3001/groups/create - добавяне на нова група
* POST към http://localhost:3001/groups/groupId/add-user - добавяне на потребител към групс
* POST към http://localhost:3001/groups/groupId/add-account - добавяне на сметка към групата
* DELTE към http://localhost:3001/groups/:groupId/remove-user – изтриване на потребител от група
* DELTE към http://localhost:3001/groups/:groupId/accounts/:accId – изтриване на сметка от групата
* GET към http://localhost:3001/activities/user/:id – взимане на дейностите на кнкретен потребител
* GET към http://localhost:3001/ activities/:id – взимане на конкретна дейност
* POST към http://localhost:3001/activities/ user/:id – добавяне на дейност към този потребител
* PATCH към http://localhost:3001/activities/:activityId– промяна на дейност
* DELTE към http://localhost:3001/activities/:activityId – изтриване на дейност
* GET към http://localhost:3001/polls - връща всичките анкети
* GET към http://localhost:3001/ polls /pollId - връща конкретна анкета в
* POST към http://localhost:3001/polls/groupId - добавяне на нова анкета в група
* POST към http://localhost:3001/polls/pollId/comment - добавяне на коментар към анкета
* PATCH към http://localhost:3001/polls/pollId/vote - гласуване към анкета
* DELTE към <http://localhost:3001/polls/::pollId/comment/:commentId> – изтриване на коментар от анкета
* DELTE към <http://localhost:3001/polls/:pollId> – изтриване на анкета от групата

Инсталацията става посредством командите npm install в папките back-end и front-end и след това npm start в същите папки.

За бъдещото развитие на системата сме предвидили разработването на budgets и predictions. С тяхната помощ потребителят по-лесно ще може да си предсказва бъдещите финанси и какво се очаква да бъде разход/приход за следващите периоди от време (ден, седмица, месец, година).

Библиография:

Документация на React - <https://reactjs.org/docs/getting-started.html>

Документация на Redux – <https://redux.js.org/>

Документация на Redux-Saga - <https://redux-saga.js.org/docs/api/>

Документация на Material-UI - <https://material-ui.com/>

Документация на Axios - <https://kapeli.com/cheat_sheets/Axios.docset/Contents/Resources/Documents/index>

Документация на Express - <https://expressjs.com/en/api.html>

Документация на Mongoose - <https://mongoosejs.com/docs/api.html>

Документация на JWT - <https://jwt.io/introduction/>