|  |  |  |
| --- | --- | --- |
|  | **Софийски университет „Св. Кл. Охридски”**  Факултет по математика и информатика  *Магистърска програма РСМТ* |  |

**Предмет: Разработка на клиент-сървър (fullstack) приложения с Node.js + Express.js + React.js**

***Летен семестър, 2020/2021 год.***

**Име на проект: Grader**

*Участници:*

*Здравко Гюров, 26379*

*Йоана Мутазова, 26382*

*Дата: 29.06.2021*

## **1. Функционални и нефункционални изисквания**

## **Функционални изисквания**

### **Регистриране в системата**

### - Системата трябва да дава възможност на нерегистрираните потребителите да се регистрират чрез въвеждане на име, потребителско име, потребителско име в GitHub и парола.

### **Влизане в системата**

### - Потребителите трябва да могат да влизат в системата чрез въвеждане на потребителско име и парола.

### **Преглеждане на курсове**

### - Потребителите и администраторите трябва да могат да разглеждат всички налични курсове.

### **Създаване на курс**

### - Системата трябва да дава възможност на администраторите и преподавателите да създават курс като въведат име на курса, описание и GitHub repository.

### **Промяна на курс**

### - Администраторите и преподавателите трябва да могат да променят информацията за курсовете.

### **Изтриване на курс**

### - Администраторите трябва да могат да изтрият курс.

### **Присъединяване към курс**

### - Потребителите трябва да могат да изпратят заявка за присъединяване към даден курс.

### **Създаване на задание**

### - Администраторите и преподавателите трябва да могат да добавят задания към курсовете като въведат име на заданието, описание и краен срок за предаване.

### **Промяна на задание**

### - Администраторите и преподавателите трябва да могат да променят информацията за дадено задание.

### **Изтриване на задание**

### - Администраторите трябва да могат да изтрият дадено задание.

### **Добавяне на решение към задание**

### - Системата трябва да дава възможност на потребителите да качват решение на дадено задание и след изпълняване на тестове върху него да покаже получения резултат и статуса на решението.

### **Получаване на права**

### - Потребителите трябва да могат да изпратят заявка за получаване на права за извършване на повече дейности в системата.

### **Обработване на заявки**

### - Администраторите трябва да могат да одобряват или отхвърлят заявките на потребителите за присъединяване към курс или за получаване на права.

### **Преглед на потребителите**

### - Администраторите трябва да могат да преглеждат списък на регистрираните в системата потребители като за всеки потребител са налични име, потребителско име, GitHub потребителско име и права за достъп.

## **Нефункционални изисквания**

### **Сигурност (security)**

### В базата от данни паролите не се пазят в чист вид.

### Потребителите, които нямат регистрация в системата не могат да я използват.

* За authentication се използват json web tokens, които лесно могат да се проверят дали са валидни, тоест дали не са били променяни, защото им се нарушава сигнатурата.

### Регистрираните потребители не могат да извършват дейности, за които нямат права.

## **2. Използвани технологии**

<https://www.npmjs.com/package/bcrypt>

<https://www.npmjs.com/package/child_process>

<https://www.npmjs.com/package/cors>

<https://www.npmjs.com/package/express>

<https://www.npmjs.com/package/jsonwebtoken>

<https://www.npmjs.com/package/mongo>

<https://www.npmjs.com/package/nodemon>

<https://www.npmjs.com/package/@material-ui/core>

<https://www.npmjs.com/package/@material-ui/icons>

<https://www.npmjs.com/package/formik>

<https://www.npmjs.com/package/yup>

## **3. Описание на основните endpoint-и**

***/api/users*** - GET, POST

***/api/users/:userId*** - GET, PATCH, DELETE

***/api/login*** - GET

***/api/courses*** - GET, POST

***/api/courses/all*** - GET

***/api/courses/:courseId*** - PATCH, DELETE

***/api/coursesRequests*** - GET, POST

***/api/coursesRequests/courseRequestsAll*** - GET

***/api/coursesRequests/:courseRequestId*** - PATCH

***/api/permissionsRequests*** - GET, POST

***/api/permissionsRequests/all*** - GET

***/api/permissionsRequests/:permissionsRequestId*** - PATCH, DELETE

***/api/assignments*** - GET, POST

***/api/assignments/:assignmentId*** - PATCH, DELETE

***/api/submissions*** - POST

***/api/submissions/all*** - GET

***/api/submissions/:submissionId*** - DELETE

***/api/submissions?:assignmentId*** - GET

## **4. Инсталиране и конфигуриране на системата**

За инсталиране на системата трябва да се пусне mongo база на порт 27017, и docker. Трябва да се стартират двата проекта. В node проекта config/secret.js може да се замени с някой по-секретен. Node сървъра може да се конфигурира от файла server.js в root-a на проекта. За създаване на админ акаунт може да се направи регистрация през UI-a, което ще създаде user с нормални права и после да се добавят директно в базата всички права **ALL\_PEMISSIONS** от models/permissions/permissions.js в node проекта. За да се използва функционалността със submission-ите, трябва да се направи GitHub repo на курса като [това](https://github.com/ZdravkoGyurov/grader-docker-tests), в което се пазят решенията на assignment-ите и са групирани по директории с имена - имената на заданията, и всеки потребител(студент), който иска да submit-не решение трябва да си го е качил в repo като [това](https://github.com/ZdravkoGyurov/grader-docker-solutions), където качва решения на заданията в курса в директории с имената на заданията.

## **5. Потребителска документация**

В UI-a има Navigation Bar, който служи за меню, което съдържа линкове към основните функционалности:

* Sign In - страница за вход в системата, видима и достъпна само ако няма логнат user
* Sign Up - страница за регистрация, видима и достъпна само ако няма логнат user
* Users - страница с всички регистрирани потребители в системата, видима и достъпна само за логнат user с права READ\_USERS, които се дават само на админи
* Courses - страница показваща всички курсове, в които е записан потребителя, видима и достъпна за логнат user с права READ\_COURSES, които се дават на всички
* Browse Courses - страница показваща всички курсове в системата, видима и достъпна за логнат user с права READ\_ALL\_COURSES, които се дават на всички
* Submissions - страница показваща всички submission-и в системата, видима и достъпна за логнат user с права READ\_ALL\_SUBMISSIONS, които се дават на админи
* My Requests - страница с всички request-и/молби за присъединяване в курс в случай, че потребителя е студент или за получаване на права в случай, че потребителя е преподавател, видима и достъпна за логнати потребители с права READ\_COURSESREQUESTS и READ\_PERMISSIONSREQUESTS
* Approve Requests - страница с всички request-и/молби в системата, видима и достъпна за логнати потребители с права READ\_ALL\_COURSESREQUESTS и READ\_ALL\_PERMISSIONSREQUESTS, които се дават на админи

## **6. Бъдещо развитие на системата**

Има доста неща, които могат да се добавят и подобрят. Няколко примера са:

* Поддръжка на няколко езика за тестване (в момента е само Golang)
* Добавяне на Oauth2 authentication с GitHub за още по-лесна интеграция
* Замяна на submission polling-а към backend-a със socket.io events
* Добавяне на допълнителни index-и в базата за оптимизация
* Разкрасяване на UI-a по UX guideline-и

## **7. Използвани материали**

<https://mongodb.github.io/node-mongodb-native/> - node mongo driver documentation

<https://material-ui.com/> - material ui documentation

<http://expressjs.com/> - express documentation

<https://www.npmjs.com/> - documentation for packages