Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Современные платформы программирования (СПП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему

Программное средство «Система элеткронного обучения»

БГУИР КП I–40 01 01 66 ПЗ

Студент: гр. 751006 Киселев Г.А.

Проверил: Голубко Д.В.

Минск, 2020

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Лапицкая Н.В. 2020г.

ЗАДАНИЕ

по курсовому проектированию

Студенту *Киселеву Григорию Александровичу*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Тема работы *Программное средство «Система элеткронного \_\_\_\_\_\_\_\_ обучения»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. Срок сдачи законченной работы *15.05.2020г.*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Исходные данные к работе *Платформа прогрраммирования Node.js, языки программирования Javascript, SQL. Среда разработки Visual Studio Code, MySQL Workbench.*

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

*Введение*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*1 Анализ литературных источников\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2 Постановка задачи\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*3 Разработка программного средства\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*4 Тестирование и проверка работоспособности программного средства\_\_\_\_*

*5 Руководство по установке и использованию программного средства\_\_\_\_\_*

*Заключение \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_*

*Список использованных источников\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Приложения* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

*Схема алгоритма в формате А1*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Консультант по курсовой работе *Голубко Д.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

7.Дата выдачи задания *08.02.2020г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

8. Календарный график работы над проектом на весь период проектирования

(с обозначением сроков выполнения и процентом от общего обьема работы):

*Раздел 1. Введение к 20.03.2020г. – 10 % готовности работы;\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 2 к 30.03.2020г. – 30% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 3 к 05.04.2020г. – 60% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 4 к 20.04.2020г. – 80% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 5. Заключение. Приложения к 03.05.2020г. – 90% готовности работы;*

*оформление пояснительной записки и графического материала к 05.05.2020г. – 100% готовности работы.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Защита курсового проекта с 04.05.2020г. по 15.05.2020г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

РУКОВОДИТЕЛЬ *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Голубко Д.В.*

*(подпись)*

Задание принял к исполнению *Киселев Г.А. 08.02.2020г.*

*(дата и подпись студента)*

Содержание

[Введение 5](#_Toc40337639)

[Анализ литературных источников 6](#_Toc40337640)

[1.1 Анализ существущих приложений. 9](#_Toc40337641)

[2 Постановка задачи 11](#_Toc40337642)

[3 Разработка программного средства 12](#_Toc40337643)

[3.1 Описание алгоритмов 12](#_Toc40337644)

[3.2 Структура данных 13](#_Toc40337645)

[3.2.1 Структура типов основного алгоритма 13](#_Toc40337646)

[3.2.2 Структура данных основного алгоритма 13](#_Toc40337647)

[3.2.3 Структура данных алгоритма getAllTasks 13](#_Toc40337648)

[3.2.4 Структура данных алгоритма getTaskById 14](#_Toc40337649)

[3.2.5 Структура данных алгоритма addNewTask (subject, details) 15](#_Toc40337650)

[3.2.6 Структура данных алгоритма generateToken(user) 15](#_Toc40337651)

[3.3 Схемы алгоритмов решения задачи 16](#_Toc40337662)

[4 Тестирование и проверка работоспособности программного средства 20](#_Toc40337663)

[5 Руководство по установке и использованию программного средства 23](#_Toc40337664)

[5.1 Шаг 1. Установка 23](#_Toc40337665)

[5.2 Шаг 2. Регистрация нового пользователя 23](#_Toc40337666)

[5.3 Шаг 3. Добавление задания 24](#_Toc40337667)

[5.4 Шаг 4. Просмотр заданий учеником 24](#_Toc40337668)

[Заключение 25](#_Toc40337669)

[Список использованной литературы 26](#_Toc40337670)

[Приложение А 27](#_Toc40337671)

Введение

C развитием средств коммуникации наметился повышенный интерес к различным методам дистанционного обучения. Обучаемому нет необходимости тратить время на перемещение к месту занятий, а сам урок можно почитать в любое удобное для себя время, изучив материал полностью или отложив прочтение части на потом. В настоящее время организационные и педагогические возможности дистанционного обучения реализуются с помощью практически всех доступных телекоммуникационных сервисов, таких как электронная почта, электронные журналы, чат, WEB-конференции и т. п.

Сейчас можно встретить большое количество разнообразных продуктов, предназначенных для организации такого сервиса. На рынке программных продуктов существует большое количество программ для дистанционного обучения. В связи с этим встает проблема выбора программного продукта по определенным критериям и для определенного образовательного учреждения. Одним из таких критериев можно выделить способность администратора к быстрому приобретению навыков работы в программе.

# Анализ литературных источников

Дистанционное образование – комплекс образовательных услуг, предоставляемых широким слоям населения в стране и за рубежом с помощью специализированной информационной образовательной среды, базирующейся на средствах обмена учебной информацией на расстоянии (спутниковое телевидение, радио, компьютерная связь и т.п.).

Информационно-образовательная среда ДО представляет собой системно-организованную совокупность:

* средств передачи данных,
* информационных ресурсов,
* протоколов взаимодействия,
* аппаратно- программного и организационно- методического обеспечения,

ориентированную на удовлетворение образовательных потребностей пользователей. ДО является одной из форм непрерывного образования, которое призвано реализовать права человека на образование и получение информации. Итак, дистанционное обучение - новая организация образовательного процесса, базирующаяся на принципе самостоятельного обучения студента. Среда обучения характеризуется тем, что учащиеся в основном, а часто и совсем, отдалены от преподавателя в пространстве и во времени, в то же время они имеют возможность в любой момент поддерживать диалог с помощью средств телекоммуникации.

Реализация дистанционного обучения базируется на информационных технологиях, обеспечивающих:

* доставку обучаемым основного объема изучаемого материала,
* интерактивное взаимодействие обучаемых и преподавателей в процессе обучения,
* предоставление студентам возможности самостоятельной работы по освоению изучаемого учебного материала,
* оценку их знаний и навыков, полученных в процессе обучения.

Термин дистанционное обучение, как правило, связывается с некоторой учебной инфраструктурой (студия учебного телевидения, специализированные узлы компьютерной сети, методические центры, разрабатывающие и распространяющие соответствующие материалы) и относится к учебному заведению, предоставляющему соответствующие услуги, а не к самим учащимся.

* Дистанционное образование с точки зрения обучения относится к способу доставки учебного материала (взаимодействия) в рамках заочного обучения.
* Дистанционное образование с точки зрения учения относится к самостоятельной работе учащегося при любой форме обучения (в том числе и различных системах самообучения).

Дистанционное обучение – это приобретение знаний и умений посредством информации и обучения, включающие в себя все технологии и другие формы обучения на расстоянии.

Электронное обучение

В последние годы на Западе получил широкое распространение термин E-learning, означающий процесс обучения в электронной форме через сеть Интернет с использованием систем управления обучением.

Понятие «электронное обучение» (ЭО) сегодня является расширением термина «дистанционное обучение». ЭО – более широкое понятие, означающее разные формы и способы обучения на основе информационных и коммуникационных технологий (ИКТ).

В настоящее время интерес к электронному обучению неуклонно возрастает. В отечественных вузах разработано большое количество курсов, ориентированных на использование информационно-коммуникационных технологий в обучении.

Особенности дистанционного обучения

Как и всякая специальная форма организации учебно-воспитательного процесса ДО имеет целый ряд специфических особенностей. К их числу обычно относят следующие:

* Гибкость. Обучающиеся, в основном, не посещают регулярных занятий в виде лекций, семинаров. Каждый может уделять учебе столько своих ресурсов (временных, финансовых и пр.), сколько ему лично необходимо для освоения курса, дисциплины и получения необходимых знаний по выбранной специальности.
* Модульность. В основу программ ДО закладывается модульный принцип. Каждая отдельная дисциплина или ряд дисциплин, которые освоены обучающимся, создают целостное представление об определенной предметной области. Это позволяет из набора независимых учебных курсов формировать учебный план, отвечающий индивидуальным или групповым потребностям.
* Параллельность. Обучение может проводиться при совмещении основной профессиональной деятельности с учебой, т.е. "без отрыва от производства".
* Дальнодействие. Расстояние от места нахождения обучающегося до образовательного учреждения (при условии качественной работы связи) не является препятствием для эффективного образовательного процесса.
* Асинхронность. Подразумевает тот факт, что в процессе обучения преподаватель и обучающийся могут реализовывать технологию обучения независимо друг от друга во времени, т. е. по удобному для каждого расписанию и в удобном темпе, однако в пределах некоторых ограничений.
* Охват. Эту особенность иногда называют также “ массовостью”. Количество обучающихся не является критичным параметром. Они имеют доступ ко многим источникам учебной информации (электронным библиотекам, базам данных), а также могут общаться друг с другом и с преподавателем через сети связи или с помощью других средств ИТ.
* Рентабельность. Под этой особенностью подразумевается экономическая эффективность ДО. Средняя оценка зарубежных и отечественных образовательных систем ДО показывает, что они обходятся приблизительно на 10-50% дешевле, в основном за счет более эффективного использования существующих учебных площадей и технических средств ИТ, а также представления более концентрированного и унифицированного содержания учебных материалов и ориентированности технологий ДО на большое количество обучающихся и других факторов.
* Преподаватель. Речь идет о новой роли преподавателя, когда на него возлагаются такие функции, как координирование познавательного процесса, корректировка преподаваемого курса, консультирование, руководство учебными проектами и т.д. Взаимодействие с обучающимися осуществляется, в основном, асинхронно с помощью электронной почты или других систем связи. Допускаются и приветствуются также и очные контакты.
* Обучающийся. Новый смысл подчеркивается самим термином – не обучаемый, а уже обучающийся. Для того чтобы пройти ДО, ему требуется исключительная самоорганизация, трудолюбие и определенный стартовый уровень образования.

## Анализ существущих приложений.

На просторах белоруского интернета было найдено несколько подобных приложений. На сегодняшний день наш университет использует систему электронного обучения на платформе Moodle.

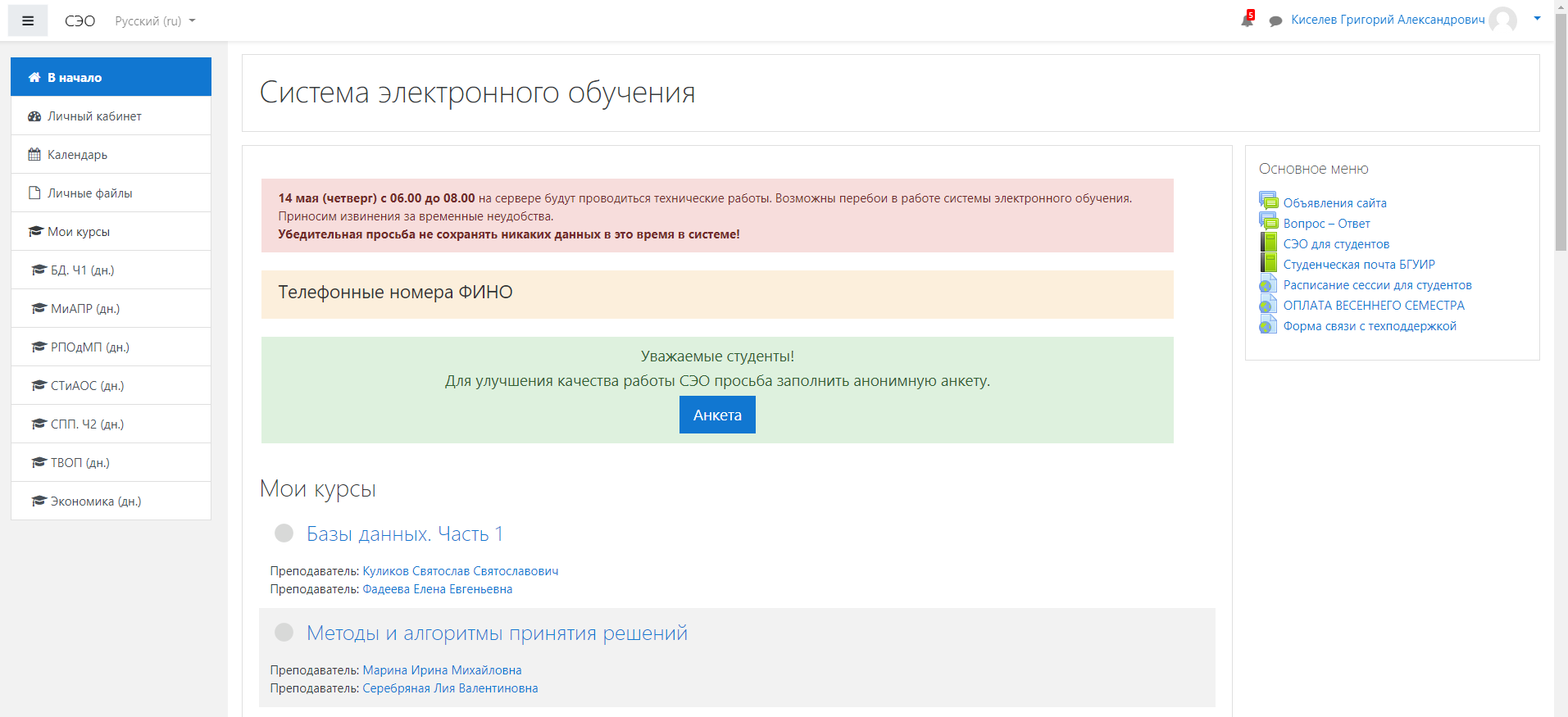


Рисунок 1.1 – Интерфейс сайта СЭО

Также еще один из сайтов системы электронного обучения является

“[elearning.mslu.by](http://elearning.mslu.by/)” Минского Государственного Лингвистического университета. Он предоставляет пользователям возможность выбора упражнений по желаемому языку.



Рисунок 1.2 – Интерфейс сайта “[elearning.mslu.by](http://elearning.mslu.by/)”

# 2 Постановка задачи

Разработать веб-приложение “Система электронного обучения”. Необходимо реализовать отображение пользовательского интерфейса приложения в браузере, сохранение данных в базу данных, авторизацию и аутентификацию с помощью JSON Web Tokens.

При входе на сайт неавторизованный пользователь может зарегистрироваться в качестве студента. После регистрации студент может авторизироваться в системе и просматритвать задания, выданные ему его учителем. Также студенту станет доступна возможность сортировки выданных ему заданий по заданным критериям. Зарегистрироваться в качестве учителя в системе невозможно. Только администратор системы может создавать аккаунты для учителей. Учитель может добавлять, удалять и редактировать задания своих учеников, а также выставлять оценки. Реализовать личную страницу пользователя, на которой будет отображаться его никнейм, роль в системе и все его задания.

Реализовать приложение в виде двух сервисов: backend и frontend. Frontend служит для отрисовки интерфейса в браузере и шлёт запросы для получения информации к backend. Backend проверяет права пользователя, взаимодействует с базой данных и посылает результат frontend-у. Использовать архитектуру REST API.

# 3 Разработка программного средства

## 3.1 Описание алгоритмов

Таблица 1 – Описание основной программы

|  |  |  |  |
| --- | --- | --- | --- |
| № | Наименование алгоритма | Назначение алгоритма | Формальные параметры |
| 1. | authorize | Авторизация пользователя и генерация токена. | login, password – логин и пароль пользователя |
| 2. | getAllTasks | Получить все задания пользователя | user\_id – уникальный идентификатор пользователя |
| 3. | getTaskById | Получить задание по его id | task\_id – уникальный идентификатор задания |
| 4. | addTask | Добавление нового задания | subject – предмет задания,  details – описание задания,  deadline –дедлайн задания,  login – логин студента, который должен выполнить задание |
| 5. | generateToken | Генерация токена для авторизации пользователя | user – авторизирующийся пользователь |

## 3.2 Структура данных

### 3.2.1 Структура типов основного алгоритма

Таблица 2 – Структура типов основного алгоритма

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| User | User {  string login,  string password  } | Хранение данных для аутентификации пользователя |

### 3.2.2 Структура данных основного алгоритма

Таблица 3 – Структура данных основного алгоритма

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Комментарии |
| token | String | Сгенерированный токен | Локальная переменная |
| loginData | User {  string: login,  string: password  } | Хранение логина и пароля пользователя | Формальный параметр |

### 3.2.3 Структура данных алгоритма getAllTasks

Таблица 4 – Структура типов алгоритма getAllTasks

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| TaskEntitity | Task {  number: id; string: subject;  string: details; date: deadline;  number: mark;number: user\_id  } | Хранение данных сущности задания |

Таблица 5 – Структура данных алгоритма getAllTasks

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Комментарии |
| tasks | TaskEntity [] | Массив заданий, который будет возвращён в результате | Локальная переменная |

### 3.2.4 Структура данных алгоритма getTaskById

Таблица 5 – Структура данных алгоритма getTaskById (task\_id)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Комментарии |
| id | int | Id задания, которое необходимо получить | Формальный параметр |
| userId | int | Id пользователя, задание которого необходимо получить | Формальный параметр |

### 3.2.5 Структура данных алгоритма addNewTask (subject, details, deadline)

Таблица 6 – Структура данных алгоритма addNewTask

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Комментарии |
| subject | String | Предмет задания | Формальный параметр |
| details | String | Описание задания | Формальный параметр |
| deadline | Date | Дедлайн задания | Формальный параметр |
| login | String | Логин ученика, выполняющего задание | Локальная переменная |

### 3.2.6 Структура данных алгоритма generateToken(user)

Таблица 7 – Структура данных алгоритма generateToken (User)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Комментарии |
| User | User {  string: login,  string: password  } | Авторизирующийся пользователь | Формальный параметр |
| secretKey | string | Секретный ключ для шифрования | Локальная переменная |
| algorithm | string | Алгоритм шифрования | Локальная переменная |

### 



## 3.3 Схемы алгоритмов решения задачи



Рисунок 3.1 – Схема алгоритма получения заданий



Рисунок 3.2 – Схема получения задания



Рисунок 3.3 – Схема алгоритма addNewTask (subject, details, deadline)



Рисунок 3.4 – Схема алгоритма генерации токена

# 4 Тестирование и проверка работоспособности программного средства

В ходе тестирования были рассмотрены в действии все функциональные возможности веб-приложения “Система электронного обучения”. Было изучено поведение приложения при различных действиях пользователя.

Таблица 9 – Тест 1

|  |  |
| --- | --- |
| Тестовая ситуация | Авторизация с правильными данными |
| Ожидаемый результат: | Переадресация на главную страницу |
| Фактический результат: |  |

Таблица 10 – Тест 2

|  |  |
| --- | --- |
| Тестовая ситуация | Регистрация пользователя |
| Ожидаемый результат: | Успешная регистрация пользователя |
| Фактический результат: | Успешная регистрация пользователя |

Таблица 11 – Тест 3

|  |  |
| --- | --- |
| Тестовая ситуация: | Добавление нового задания |
| Ожидаемый результат: | Новое задание появится в списке заданий |
| Фактический результат: |  |

Таблица 12 – Тест 4

|  |  |
| --- | --- |
| Тестовая ситуация: | Сортировка заданий по дедлайну |
| Ожидаемый результат: | Отсортированные данные |
| Фактический результат: |  |

# 5 Руководство по установке и использованию программного средства

## 5.1 Шаг 1. Установка

Для запуска приложения необходимо наличие субд mysql, запустить backend и frontend. После запуска всех сервисов открыть браузер и перейти по адресу localhost:4200. Появится главная страница с авторизацией пользователя.

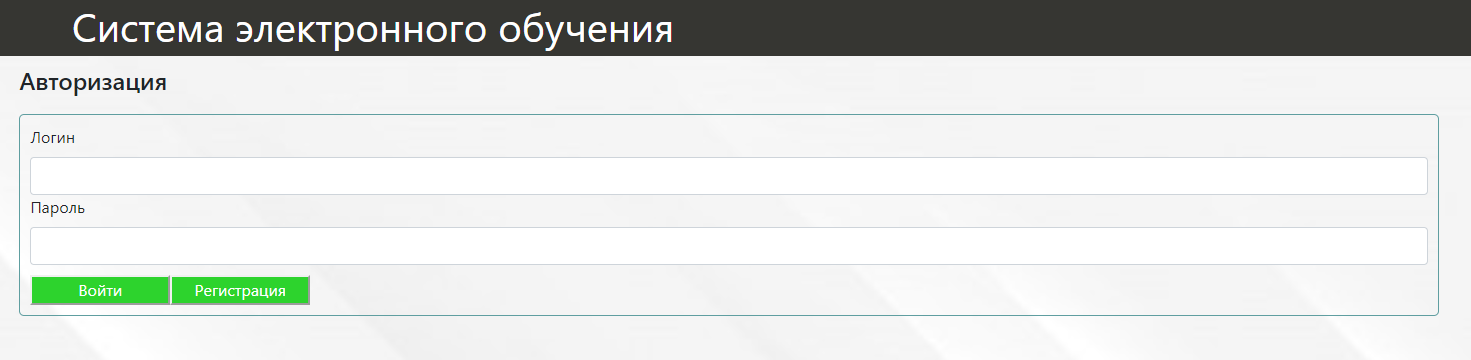


Рисунок 5.1 – Страница авторизации сайта

## 5.2 Шаг 2. Регистрация нового пользователя

После нажатия на кнопку “Регистрация” будет открыта страница с формой для регистрации. Там пользователь может ввсети свои личные данные и зарегистрировать новый аккаунт.

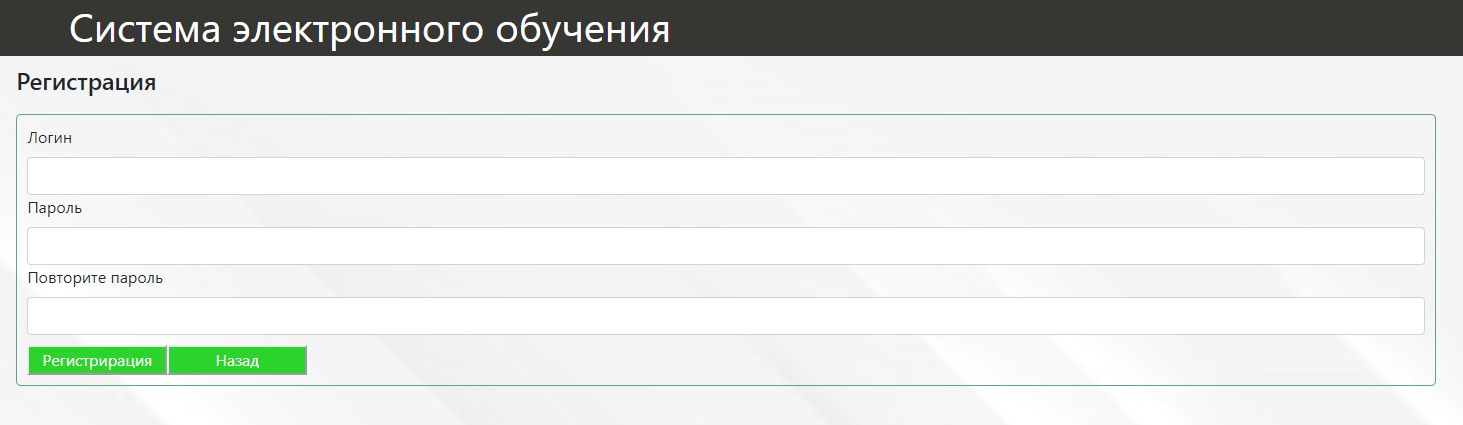


Рисунок 5.2 – Страница регистрации

## 5.3 Шаг 3. Добавление задания

Пользователь в роли учителя может добавить новое задание. Для этого необходимо войти в систему в роли учителя. Откроется список всех заданий учеников. Там нужно нажать на кнопку “Новое задание”, после чего будет открыта страница для добавления поста. На странице добавления задания учитель добавить название предмета, описание задания, его дедлайн и студента, который будет выполнять задание. Для сохранения задания необходимо нажать на кнопку “Сохранить”.

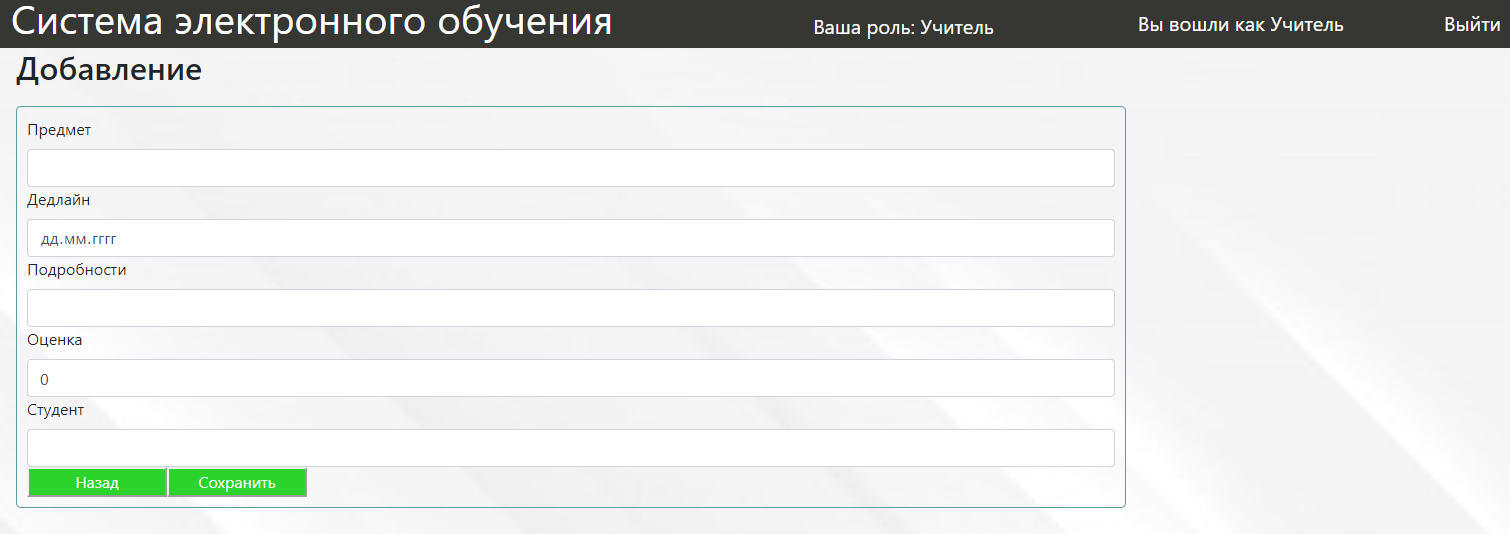


Рисунок 5.3 – Страница добавления задания

## 5.4 Шаг 4. Просмотр заданий учеником

Пользователь в роли ученика может просматривать список своих заданий, выданных учителем, а также сортировать их по выбранным критериям. Для этого необходимо авторизироваться в системе в качестве ученика. Также пользователь может оставлять комментарии под постами. Для этого необходимо ввести текст комментария и нажать на кнопку “Submit”.



Рисунок 5.4 – Список заданий ученика

Заключение

В ходе выполнения курсовой работы мною было разработано веб-приложение “Система электронного обучения”. В процессе разработки использовался язык программирования JavaScript, а также фреймворки Node.js и Angular. В качестве СУБД использовалсь MySQL. Приложение создано в соответствии с архитектурой Rest API.

В разработанном приложении пользователь может зарегистрироваться и затем авторизироваться. Авторизация реализована с помощью JSON Web Tokens. В случае успешной авторизации сервер генерирует токен и выдает его пользователю. Также пользователь может просматривать свои задания и сортировать их по выбранным критериям.

Итогами выполнения данной работы является опыт разработки веб-приложений на языке программирования JavaScript, опыт работы с базами данных, изучение и применение архитектуры Rest API.

Список использованной литературы

[1] Руководство по Node.js [Электронный ресурс] – Режим доступа: [https://metanit.com/web/nodejs](https://metanit.com/web/nodejs/) – Дата доступа: 10.03.2020.

[2] Э. Скотт – SPA Design and Architecture: Understanding Single Page Web Applications, 2015г. – 668с.

[3] Node.js Documentation [Электронный ресурс] – Режим доступа: [https://nodejs.org/ru/docs](https://nodejs.org/ru/docs%20) – Дата доступа: 10.03.2020.

[4] Angular Documentation [Электронный ресурс] – Режим доступа: https://angular.io/docs – Дата доступа: 20.03.2020.

[5] MySQL Documentation [Электронный ресурс] – Режим доступа: https://dev.mysql.com/doc – Дата доступа: 21.03.2020.

Приложение А

(обязательное)

Исходный код программы

authController.js

const jwt = require('jsonwebtoken');

const auth = require('../config/auth');

const mySqlService = require('../config/db');

exports.login = async function(req, res){

let json = JSON.stringify(req.body);

let data = JSON.parse(json);

const connection = mySqlService.getDBConnection();

const sql = `SELECT \* FROM users WHERE \`login\` = '${data.login}' AND password = '${data.password}' LIMIT 1;`;

let result = await connection.promise().query(sql);

//console.log(data);

//console.log(result[0]);

if(result[0].length < 1){

res.status(404).send({

message: 'User not found.'

});

return;

}

let user = result[0][0];

user.token = generationToken(user);

user.password = null;

res.status(200).send(user);

};

exports.registrate = async function(req, res){

let json = JSON.stringify(req.body);

let data = JSON.parse(json);

const connection = mySqlService.getDBConnection();

const sql = `SELECT \* FROM users WHERE login = '${data.login}';`;

let result = await connection.promise().query(sql);

if(result[0].length != 0){

res.status(400).send({

message: 'Such user already exist.'

});

return;

}

const sqlInsert = `INSERT INTO users (login, password, role) VALUES ('${data.login}', '${data.password}', '${data.role}'); ;`;

await connection.promise().query(sqlInsert);

const sqlSelect = `SELECT \* FROM users WHERE login = '${data.login}';`;

let newUser = await connection.promise().query(sqlSelect);

let user = newUser[0][0];

user.token = generationToken(user);

user.password = null;

res.status(200).send(user);

}

let generationToken = (user) => {

return jwt.sign({

userId: user.id

}, auth.secretKey, { algorithm: 'HS256', expiresIn: auth.expires});

};

taskController.js

const mySqlService = require('../config/db');

exports.getAllTasks = async function(req, res){

let user\_uniq\_id = req.headers.user\_id;

var tasks;

var sortType = null;

const connection = mySqlService.getDBConnection();

switch(req.query.sortType){

case 'student': sortType = 'login'; break;

case 'mark':

case 'subject':

case 'details':

case 'deadline': sortType = req.query.sortType; break;

}

user = (await connection.promise().query(`SELECT \* FROM users WHERE id = ?`, [user\_uniq\_id]));

userRole = user[0][0].role;

console.log(req.query.sortType);

if (userRole == "teacher")

{

if(sortType){

tasks = (await connection.promise().query(`SELECT tasks.id, tasks.subject, tasks.details, tasks.deadline, tasks.mark, tasks.user\_id, u.login

FROM tasks

INNER JOIN users u ON u.id = tasks.user\_id ORDER BY \`${sortType}\``));

}

else{

tasks = (await connection.promise().query(`SELECT tasks.id, tasks.subject, tasks.details, tasks.deadline, tasks.mark, tasks.user\_id, u.login

FROM tasks

INNER JOIN users u ON u.id = tasks.user\_id;`));

};

return res.send(tasks[0]);

}

if(sortType){

tasks = (await connection.promise().query(`SELECT \* FROM tasks WHERE user\_id = ? ORDER BY \`${sortType}\``, [user\_uniq\_id]));

}

else{

tasks = (await connection.promise().query("SELECT \* FROM tasks WHERE user\_id = ?", [user\_uniq\_id]));

};

connection.end();

res.send(tasks[0]);

}

exports.createTask = async function(req, res){

if(!req.body) return res.sendStatus(400);

let json = JSON.stringify(req.body);

let data = JSON.parse(json);

const connection = mySqlService.getDBConnection();

const sqlUser = "SELECT \* FROM users WHERE login = ?";

const user = await connection.promise().query(sqlUser, [data.student]);

const id = user[0][0].id;

const sqlInsert = "INSERT INTO tasks (subject, details, deadline, mark, user\_id) VALUES (?, ?, ?, ?, ?);";

let result = await connection.promise().query(sqlInsert, [data.subject, data.details, data.deadline, data.mark, id]);

connection.end();

res.status(200);

res.send({"ok": "OK"});

}

exports.updateTask = async function(req, res){

console.log("123");

if(!req.body) return res.sendStatus(400);

let json = JSON.stringify(req.body);

let data = JSON.parse(json);

const connection = mySqlService.getDBConnection();

const sqlUpdate = "UPDATE tasks SET subject = ?, details = ?, deadline = ?, mark = ? WHERE id = ?";

await connection.promise().query(sqlUpdate, [data.subject, data.details, data.deadline, data.mark, req.params['id']]);

connection.end();

res.status(200);

res.send({"ok": "OK"});

}

exports.deleteTask = async function(req, res){

let json = JSON.stringify(req.body);

let data = JSON.parse(json);

const connection = mySqlService.getDBConnection();

const sqlDelete = "DELETE FROM tasks WHERE id = ?";

await connection.promise().query(sqlDelete, [req.params['id']]);

connection.end();

res.status(200);

res.send({"ok": "OK"});

}

exports.getTaskById = async function(req, res){

if(!req.body) return res.sendStatus(400);

const connection = mySqlService.getDBConnection();

const sqlSelect = "SELECT \* FROM tasks WHERE id = ?";

let result = await connection.promise().query(sqlSelect, [req.params['id']]);

connection.end();

res.send(result[0][0]);

}

index.js

'use strict';

var authService = require('../services/authService');

var authMiddleware = function(req, res, next) {

if (authService.authenticate(req, res)) {

authService.getUserId(req, res);

console.log(req.headers.user\_id);

next();

}

};

module.exports = function(app) {

var controller = require('../controllers/taskController');

var authController = require('../controllers/authController');

console.log("index");

app.route('/tasks').get(authMiddleware, controller.getAllTasks);

app.route('/tasks/:id').get(authMiddleware, controller.getTaskById);

app.route('/task').post(authMiddleware, controller.createTask);

app.route('/task/:id').put(authMiddleware,controller.updateTask);

app.route('/task/:id').delete(authMiddleware, controller.deleteTask);

app.route('/login').post(authController.login);

app.route('/registrate').post(authController.registrate);

};

authService.js

const jwt = require("jsonwebtoken");

const atob = require('atob');

const PRIVATE\_KEY = require('../config/auth').secretKey;

exports.authenticate = function(req, res) {

try {

console.log(req.headers);

var token = req.headers.authorization.replace('Bearer ', '');

return jwt.verify(token, PRIVATE\_KEY);

} catch (e) {

console.log(e);

res.sendStatus(401);

return false;

}

};

exports.getUserId = function(req, res) {

var token = req.headers.authorization.replace('Bearer ', '');

console.log(token, JSON.parse(atob(token.split('.')[1])));

req.headers.user\_id = JSON.parse(atob(token.split('.')[1])).userId;

};

app.js

var express = require('express'),

app = express(),

port = 3005,

bodyParser = require('body-parser');

const cors = require('cors');

app.use(bodyParser.urlencoded({ extended: true }));

app.use(bodyParser.json());

app.use(cors());

var routes = require('./app/routes/index');

routes(app);

app.listen(port, () => {

console.log('Server starts on ' + port);

});

login-form.component.ts

import { Component, OnInit } from '@angular/core';

import { Router, ActivatedRoute } from '@angular/router';

import { first } from 'rxjs/operators';

import { AuthenticationService } from '../services/authentication.service';

import {User} from '../models/user';

@Component({

selector: 'app-login-form',

templateUrl: 'login-form.component.html'

})

export class LoginComponent implements OnInit {

user = new User(null, "", "", "", null);

loading = false;

submitted = false;

returnUrl: string;

error = '';

constructor(

private route: ActivatedRoute,

private router: Router,

private authenticationService: AuthenticationService

) {

if (this.authenticationService.currentUserValue) {

let userId = this.authenticationService.currentUserValue.id;

this.router.navigate([`tasks`]);

}

}

ngOnInit() {

}

onSubmit(user: User) {

this.submitted = true;

this.loading = true;

this.authenticationService.login(user)

.pipe(first())

.subscribe(

data => {

let userId = data.id;

this.router.navigate([`tasks`]);

},

error => {

this.error = error;

console.log(error);

this.loading = false;

});

}

onRegistrate(user: User) {

this.submitted = true;

this.loading = true;

this.authenticationService.registrate(user)

.pipe(first())

.subscribe(

data => {

this.router.navigate(['/login']);

},

error => {

this.error = error;

this.loading = false;

});

}

};

reg-form.component.ts

import { Component, OnInit } from '@angular/core';

import { Router, ActivatedRoute } from '@angular/router';

import { first } from 'rxjs/operators';

import { AuthenticationService } from '../services/authentication.service';

import {User} from '../models/user';

import { regUser } from '../models/regUser';

@Component({

selector: 'app-reg-form',

templateUrl: 'reg-form.component.html'

})

export class RegComponent implements OnInit {

user = new regUser("", "", "");

loading = false;

submitted = false;

returnUrl: string;

error = '';

constructor(

private route: ActivatedRoute,

private router: Router,

private authenticationService: AuthenticationService

) {

if (this.authenticationService.currentUserValue) {

this.router.navigate(['/login']);

}

}

ngOnInit() {

}

onRegistrate(user: regUser) {

this.submitted = true;

this.loading = true;

this.authenticationService.registrate(new User(null, user.login, user.password, "student", null))

.pipe(first())

.subscribe(

data => {

this.router.navigate(['/login']);

},

error => {

this.error = error;

this.loading = false;

});

}

}

auth.guard.ts

import { Injectable } from '@angular/core';

import { Router, CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot } from '@angular/router';

import { AuthenticationService } from './authentication.service';

@Injectable({ providedIn: 'root' })

export class AuthGuard implements CanActivate {

constructor(

private router: Router,

private authenticationService: AuthenticationService

) { }

canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot) {

const currentUser = this.authenticationService.currentUserValue;

if (currentUser) {

return true;

}

console.log("no user");

this.router.navigate(['/login'], { queryParams: { returnUrl: state.url } });

return false;

}

}

authentication.service.ts

import { Injectable } from '@angular/core';

import { HttpClient, HttpParams, HttpHeaders } from '@angular/common/http';

import { BehaviorSubject, Observable } from 'rxjs';

import { map } from 'rxjs/operators';

import { environment } from '../../../environments/environment';

import { User } from '../models/user';

import { regUser } from '../models/regUser';

@Injectable({ providedIn: 'root' })

export class AuthenticationService {

private currentUserSubject: BehaviorSubject<User>;

public currentUser: Observable<User>;

private url = environment.apiUrl+ "login/";

private reg = environment.apiUrl+ "registrate/";

private headers: HttpHeaders = new HttpHeaders({

'Content-Type': 'application/x-www-form-urlencoded',

});

constructor(private http: HttpClient) {

this.currentUserSubject = new BehaviorSubject<User>(JSON.parse(localStorage.getItem('currentUser')));

this.currentUser = this.currentUserSubject.asObservable();

}

public get currentUserValue(): User {

return this.currentUserSubject.value;

}

login(user: User) {

let form = this.init(user);

return this.http.post<any>(`${this.url}`, form.toString(), {headers: this.headers})

.pipe(map(user => { console.log(user);

localStorage.setItem('currentUser', JSON.stringify(user));

this.currentUserSubject.next(user);

return user;

}));

}

registrate(user: User) {

let form = this.init(user);

return this.http.post<any>(`${this.reg}`, form.toString(), {headers: this.headers})

.pipe(map(user => {

localStorage.setItem('currentUser', JSON.stringify(user));

this.currentUserSubject.next(user);

return user;

}));

}

logout() {

localStorage.removeItem('currentUser');

this.currentUserSubject.next(null);

}

init(user: User) {

let form = new HttpParams()

.set(`id`, user.id !== null ? user.id.toString() : null)

.set(`login`, user.login)

.set(`password`, user.password)

.set(`token`, user.token)

.set(`role`, user.role)

return form;

}

}

task-form.component.ts

import { Component, OnInit } from '@angular/core';

import { ActivatedRoute, Router } from '@angular/router';

import { Task } from '../models/task';

import { TaskService } from '../services/task.service';

@Component({

selector: 'app-task-form',

templateUrl: './task-form.component.html'

})

export class TaskFormComponent implements OnInit {

task = new Task(0, '', '', '', 0, '');

existed = false;

constructor(

private route: ActivatedRoute,

private router: Router,

private taskService: TaskService,

) {

}

ngOnInit() {

this.route.params.subscribe(p => {

if (p['id'] === undefined) {

return;

}

this.taskService.getTask(p['id']).subscribe(h => this.initTask(h));

this.existed = true;

});

}

initTask(h: Task){

this.task = h;

console.log(h);

this.task.deadline = this.task.deadline.split('T')[0];

console.log(this.task);

}

navigateToTasks() {

this.router.navigate([`tasks`])

}

onCancel() {

this.navigateToTasks();

}

onSubmit(task: Task) {

if(this.existed)

this.taskService.updateTask(task).subscribe(c => this.router.navigate([`tasks`]));

else

this.taskService.addTask(task).subscribe(c => this.router.navigate([`tasks`]));

}

}

task-list.component.ts

import { Component, OnInit } from '@angular/core';

import { Task } from '../models/task';

import { TaskService } from '../services/task.service';

import { CommonModule } from '@angular/common';

import { BrowserModule } from '@angular/platform-browser';

@Component({

selector: 'app-customer-item',

templateUrl: './task-list.component.html',

})

export class TaskListComponent implements OnInit {

tasks: Task[];

today: number;

userId: number;

userRole: string;

constructor(private taskService: TaskService) { this.userRole = (JSON.parse(localStorage.getItem('currentUser'))).role; }

ngOnInit() {

this.getAllTasks();

//console.log(this.tasks);

}

getAllTasks() {

this.userId = JSON.parse(localStorage.getItem('currentUser')).id;

this.taskService.getTasks().subscribe(tasks => this.tasks = tasks) ;

}

sortTasks(sortType: string){

this.taskService.getSortedTasks(sortType).subscribe(tasks => this.tasks = tasks);

}

onDelete(task\_id: number){

this.taskService.deleteTask(task\_id).subscribe(tasks => this.getAllTasks());

}

}

task.service.ts

import { HttpClient, HttpHeaders, HttpParams } from '@angular/common/http';

import { Injectable } from '@angular/core';

import { environment } from '../../../environments/environment';

import { Observable, onErrorResumeNext } from 'rxjs';

import { Task } from '../models/task';

@Injectable({

providedIn: 'root'

})

export class TaskService {

private headers: HttpHeaders = new HttpHeaders({

'Content-Type': 'application/x-www-form-urlencoded',

});

private url = environment.apiUrl;

private singleUrl = environment.apiUrl + '/task/';

constructor(private http: HttpClient) { }

getTasks(): Observable<Array<Task>> {

return this.http.get<Array<Task>>(`${this.url}tasks/`);

}

getSortedTasks(sortType: string): Observable<Array<Task>> {

return this.http.get<Array<Task>>(`${this.url}tasks?sortType=${sortType}`);

}

getTask(taskId: number): Observable<Task> {

return this.http.get<Task>(`${this.url}tasks/${taskId}`);

}

addTask(task: Task) : Observable<Task> {

console.log();

const form = this.init(task);

return this.http.post<Task>(`${this.url}task/`, form.toString(), {headers: this.headers});

}

updateTask(task: Task): Observable<Task>{

console.log(task);

const form = this.init(task);

return this.http.put<Task>(`${this.url}task/${task.id}`, form.toString(), {headers: this.headers});

}

deleteTask(taskId: number): Observable<any> {

return this.http.delete<any>(`${this.url}task/${taskId}`);

}

init( task: Task) {

let form = new HttpParams()

.set(`id`, task.id === null ? null : task.id.toString())

.set(`deadline`, task.deadline)

.set(`details`, task.details)

.set(`subject`, task.subject)

.set(`mark`, task.mark.toString())

.set(`student`, task.student)

return form;

}

}

app-routing.module.ts

import { NgModule } from '@angular/core';

import { RouterModule, Routes } from '@angular/router';

import { TaskListComponent} from './task-manager/lists/task-list.component';

import { TaskFormComponent} from './task-manager/forms/task-form.component';

import { LoginComponent } from './login-manager/forms/login-form.component';

import { AuthGuard } from './login-manager/services/auth.guard';

import { RegComponent } from './login-manager/forms/reg-form.component';

const routes: Routes = [

{ path: '', redirectTo: 'tasks', pathMatch: 'full' },

{ path: 'tasks', component: TaskListComponent, canActivate: [AuthGuard]},

{ path: 'task', component: TaskFormComponent },

{ path: 'task/:id', component: TaskFormComponent },

{ path: 'login', component: LoginComponent },

{ path: 'registration', component: RegComponent },

];

@NgModule({

imports: [ RouterModule.forRoot(routes) ],

exports: [ RouterModule ]

})

export class AppRoutingModule { }

app.component.ts

import { Component } from '@angular/core';

import { AuthenticationService } from './login-manager/services/authentication.service';

import { User } from './login-manager/models/user';

import { Router } from '@angular/router';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

currentUser: User;

constructor(

private router: Router,

private authenticationService: AuthenticationService

) {

this.authenticationService.currentUser.subscribe(x => this.currentUser = x);

}

logout() {

this.authenticationService.logout();

this.router.navigate(['/login']);

}

}

app.module.ts

import { HttpClient, HttpClientModule } from '@angular/common/http';

import { HTTP\_INTERCEPTORS } from '@angular/common/http';

import { NgModule } from '@angular/core';

import { FormsModule } from '@angular/forms';

import { BrowserModule } from '@angular/platform-browser';

import { NgbModule } from '@ng-bootstrap/ng-bootstrap';

import { AppRoutingModule } from './app-routing.module';

import { AppComponent } from './app.component';

import { TaskService } from './task-manager/services/task.service';

import { TaskListComponent } from './task-manager/lists/task-list.component';

import { TaskFormComponent } from './task-manager/forms/task-form.component';

import { LoginComponent } from './login-manager/forms/login-form.component';

import { AuthenticationService } from './login-manager/services/authentication.service';

import { ErrorInterceptor } from './helpers/error.interceptor';

import { JwtInterceptor } from './helpers/jwt.interceptor';

import { RegComponent } from './login-manager/forms/reg-form.component';

@NgModule({

declarations: [

AppComponent,

TaskListComponent,

TaskFormComponent,

LoginComponent,

RegComponent

],

imports: [

BrowserModule,

FormsModule,

NgbModule.forRoot(),

HttpClientModule,

AppRoutingModule,

],

providers: [

HttpClient,

TaskService,

AuthenticationService,

{ provide: HTTP\_INTERCEPTORS, useClass: JwtInterceptor, multi: true },

{ provide: HTTP\_INTERCEPTORS, useClass: ErrorInterceptor, multi: true },

],

bootstrap: [AppComponent]

})

export class AppModule { }

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Обозначение | | | | Наименование | | | | Дополнительные сведения | | | |
|  | | | | Текстовые документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| БГУИР КП 1–40 01 01 066 ПЗ | | | | Пояснительная записка | | | | 40 с. | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | | Графические документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| БГУИР 751006 066 СП | | | | Алгоритм работы метода getTaskById Схема алгоритма | | | | Формат А1 | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  |  |  |  |  | БГУИР КП 1-40 01 01 066 Д1 | | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| Изм. | Л. | № докум. | Подп. | Дата | Веб-приложение “Система электронного обучения”  Ведомость курсового  проекта |  | | | | Лист | Листов |
| Разраб. | | Киселев Г.А. |  |  |  |  |  | |  | 40 | 40 |
| Пров. | | Голубко Д.В. |  |  |  | Кафедра ПОИТ  гр. 751006 | | | | | |