**Приложение А. Текст программы**

АННОТАЦИЯ

В данном программном документе «Текст программы» приведен исходный текст программного изделия для информационной системы «Разработка WEB-платформы для онлайн-обучения «EasyAssign»».

В документе представлен раздел “Текст программы” с подразделами:

1. Наименование программы
2. Область применения программы
3. Модули
4. Код программы

В подразделе “Наименование программы” содержится наименование программы, которая была создана для данного приложения.

В подразделе “Область применения программы” содержится описание того, для чего была предназначена программа.

В подразделе “Модули” содержится таблица с названием модуля, его описанием, количеством строк и размерами файла к КБайтах.

В подразделе “Код программы” содержится код каждого модуля, представленных в разделе “Модули”.

СОДЕРЖАНИЕ

[1. ТЕКСТ ПРОГРАММЫ 3](#_Toc136545371)

[1.1. Наименование программы 3](#_Toc136545372)

[1.2. Область применения программы 3](#_Toc136545373)

[1.3. Модули 3](#_Toc136545374)

[1.4. Код программы 10](#_Toc136545375)

## ТЕКСТ ПРОГРАММЫ

## Наименование программы

Наименование – приложение «EasyAssign»

## Область применения программы

Потенциальными пользователями системы являются:

Студенты: Основными пользователями платформы будут студенты университета. Они смогут получать доступ к онлайн-курсам, материалам, заданиям, тестам и другим обучающим ресурсам. Студенты смогут изучать материалы в удобное для них время и темпе, задавать вопросы преподавателям и получать обратную связь.

Преподаватели: Преподаватели будут активно использовать платформу для создания и организации учебного контента. Они смогут загружать лекции, презентации, видеоматериалы, создавать задания и тесты, а также выставлять оценки и предоставлять обратную связь студентам. Преподаватели также смогут взаимодействовать с учащимися через коммуникационные инструменты платформы.

Таким образом, WEB-платформа для онлайн-обучения будет использоваться студентами и преподавателями университета для улучшения процесса обучения и обеспечения эффективного взаимодействия между всеми участниками образовательного процесса.

## Модули

В Таблице 1 представлены модули сайта.

Таблица - Модули

| № | Модули | Описание | Количество строк кода | Размер (в Кбайтах) |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
|  | App.tsx | Роутинг всего приложения | 29 | 2 |
|  | components\AccountInfo.tsx | Информация об аккаунте | 80 | 4 |
|  | components\AddGroupToClass.tsx | Добавление группы на курс | 143 | 7 |
|  | components\AddUserToClass.tsx | Добавление пользователя на курс | 278 | 13 |
|  | components\AuthError.tsx | Страница «Главная» если пользователь не авторизовал или не имеет курсов или групп | 12 | 1 |
|  | components\ChangeAccount.tsx | Изменение аккаунта | 113 | 5 |
|  | components\ClassroomLenta.tsx | Вкладка «Лента» на странице курса | 198 | 10 |
|  | components\ClassroomTask.tsx | Вкладка «Задания» на странице курса | 110 | 5 |
|  | components\ClassroomUsers.tsx | Вкладка «Пользователи» на странице курса | 126 | 6 |
|  | components\CourseCard.tsx | Карточка курса | 85 | 4 |
|  | components\CourseMap.tsx | Список курсов | 10 | 1 |
|  | components\FileInput.tsx | Текстовое поле с типом файл | 64 | 2 |
|  | components\Footer.tsx | Блок нижней части страницы | 32 | 2 |
|  | components\FreeUserMap.tsx | Список свободных пользователей, не состоящих на курсе | 10 | 1 |
|  | components\FreeUsersGroupCard.tsx | Карточка свободных пользователей, не состоящих на курсе | 95 | 5 |
|  | components\FreeUsersGroupMap.tsx | Список свободных пользователей, не состоящих в группе | 10 | 1 |
|  | components\GroupsCard.tsx | Карточка свободных пользователей, не состоящих в группе | 56 | 3 |
|  | components\GroupsMap.tsx | Список групп | 10 | 1 |
|  | components\HeaderHome.tsx | Заголовок страницы «Главная» | 56 | 3 |
|  | components\HeaderHomeNoneAuth.tsx | Заголовок страницы «Главная», если пользователь не авторизован | 29 | 1 |
|  | components\MainHomeIsAuth.tsx | Страница «Главная», если пользователь не авторизован | 94 | 4 |
|  | components\ModalAddToTheme.tsx | Модальное окно с добавлением задания в тему | 107 | 4 |
|  | components\ModalAddUserToGroup.tsx | Модальное окно с добавлением пользователя в группу | 125 | 5 |
|  | components\ModalCreateClass.tsx | Модальное окно с созданием и редактированием курса | 169 | 7 |
|  | components\ModalCreateGroup.tsx | Модальное окно с созданием и редактированием группы | 92 | 3 |
|  | components\ModalCreateTask.tsx | Модальное окно с созданием и редактированием задания | 256 | 11 |
|  | components\ModalFileComplementary.tsx | Модальное окно с добавлением файлов с решением | 82 | 3 |
|  | components\ModalImage.tsx | Модальное окно с добавлением картинок в приложение | 83 | 3 |
|  | components\ModalTheme.tsx | Модальное окно с созданием и редактированием темы | 97 | 4 |
|  | components\ModalUsersGroups.tsx | Модальное окно с списком участников группы | 116 | 5 |
|  | components\Sidebar.tsx | Боковое меню | 106 | 4 |
|  | components\SmallCourseCard.tsx | Маленькая карточка курса, выводящаяся в боковом меню | 20 | 1 |
|  | components\SmallCoursesMap.tsx | Список курса, выводящийся в боковом меню | 10 | 1 |
|  | components\StudentTaskIsForm.tsx | Вывод одного задания, в котором есть форма для сдачи | 308 | 14 |
|  | components\TaskCard.tsx | Карточка задания | 103 | 5 |
|  | components\TaskMap.tsx | Список заданий | 16 | 1 |
|  | components\ThemeCard.tsx | Карточка темы | 54 | 3 |
|  | components\ThemeMap.tsx | Список тем | 10 | 1 |
|  | components\UserCard.tsx | Карточка пользователя | 77 | 3 |
|  | components\UserMap.tsx | Список пользователей | 10 | 1 |
|  | http/index.ts | Связующее звено между backend и frontend | 30 | 1 |
|  | middleware\colorToRGBA.ts | Перевод цветов в разные форматы | 16 | 1 |
|  | middleware\image.ts | Проверка картинки на доступность | 10 | 1 |
|  | model\AuthResponse.ts | Модель авторизации | 7 | 1 |
|  | model\Complementary.ts | Модель решения | 8 | 1 |
|  | model\Course.ts | Модель курса | 10 | 1 |
|  | model\Group.ts | Модель группы | 4 | 1 |
|  | model\Member.ts | Модель участников курса | 5 | 1 |
|  | model\Task.ts | Модель задания | 11 | 1 |
|  | model\Theme.ts | Модель темы | 4 | 1 |
|  | model\User.ts | Модель пользователя | 9 | 1 |
|  | pages\Account.tsx | Страница «Аккаунт» | 92 | 5 |
|  | pages\ChangePassword.tsx | Страница «Смена пароля» | 116 | 5 |
|  | pages\Classroom.tsx | Страница курса | 145 | 6 |
|  | pages\Home.tsx | Страница «Главная» | 53 | 2 |
|  | pages\Login.tsx | Страница «Авторизация» | 100 | 5 |
|  | pages\Registration.tsx | Страница «Регистрация» | 168 | 8 |
|  | pages\ResetPassword.tsx | Страница «Сброс пароля» | 61 | 3 |
|  | pages\Task.tsx | Страница «Задание» | 46 | 2 |
|  | service\AuthService.ts | Управление авторизацией | 15 | 1 |
|  | service\ComplementaryService.ts | Управление  информацией о решении | 12 | 1 |
|  | service\CourseService.ts | Управление  информацией о курсе | 31 | 2 |
|  | service\GroupService.ts | Управление информацией о группе | 35 | 2 |
|  | service\MemberService.ts | Управление информацией об участниках | 28 | 2 |
|  | service\TaskService.ts | Управление информацией о заданиях | 36 | 2 |
|  | service\ThemeService.ts | Управление информацией о темах | 20 | 1 |
|  | service\UserService.ts | Управление пользовательской информацией | 15 | 1 |
|  | store\store.ts | Центральное хранилище данными | 626 | 20 |
|  | controller\auth-controller.ts | Обработка запросов, связанных с авторизацией | 64 | 3 |
|  | controller\class-controller.ts | Обработка запросов, связанных с курсом | 105 | 4 |
|  | controller\complementary-controller.ts | Обработка запросов, связанных с решением | 24 | 1 |
|  | controller\group-controller.ts | Обработка запросов, связанных с группой | 97 | 3 |
|  | controller\member-controller.ts | Обработка запросов, связанных с участником | 106 | 4 |
|  | controller\task-controller.ts | Обработка запросов, связанных с заданием | 117 | 4 |
|  | controller\theme-controller.ts | Обработка запросов, связанных с темой | 44 | 2 |
|  | controller\user-controller.ts | Обработка запросов, связанных с пользователем | 90 | 4 |
|  | exception\api-error.ts | Обработчик ошибок | 21 | 1 |
|  | index.ts | Начальная страница backend, запускающая его | 39 | 2 |
|  | middleware\auth-middleware.ts | Промежуточное программное обеспечение, связанное с аутентификацией | 22 | 1 |
|  | middleware\error-middleware.ts | Промежуточное программное обеспечение, связанное с выводом ошибок | 9 | 1 |
|  | middleware\prisma-middleware.ts | Промежуточное программное обеспечение, связанное с ORM Prisma.io | 4 | 1 |
|  | middleware\upload-middleware.ts | Промежуточное программное обеспечение, связанное с загрузкой файлов на сервер | 26 | 1 |
|  | prisma\schema.prisma | Файл ORM Prisma.io | 153 | 5 |
|  | router\auth-router.ts | Роутинг пользователя | 21 | 1 |
|  | router\class-router.ts | Роутинг курса | 14 | 1 |
|  | router\complementary-router.ts | Роутинг решения | 10 | 1 |
|  | router\group-router.ts | Роутинг группы | 15 | 1 |
|  | router\member-router.ts | Роутинг участника | 14 | 1 |
|  | router\task-router.ts | Роутинг задания | 14 | 1 |
|  | router\theme-router.ts | Роутинг темы | 11 | 1 |
|  | router\user-router.ts | Роутинг пользователя | 16 | 1 |
|  | router.ts | Общий роутинг | 24 | 1 |
|  | service\auth-service.ts | Код с функциями авторизации | 185 | 6 |
|  | service\class-service.ts | Код с функциями курса | 315 | 11 |
|  | service\complementary-service.ts | Код с функциями решения | 51 | 2 |
|  | service\file-service.ts | Код с функциями файла | 19 | 1 |
|  | service\group-service.ts | Код с функциями группы | 285 | 9 |
|  | service\mail-service.ts | Код с функциями почты | 170 | 7 |
|  | service\member-service.ts | Код с функциями участника | 379 | 13 |
|  | service\task-service.ts | Код с функциями задания | 425 | 15 |
|  | service\theme-service.ts | Код с функциями темы | 85 | 3 |
|  | service\token-service.ts | Код с функциями токена | 86 | 3 |
|  | service\user-service.ts | Код с функциями пользователя | 254 | 8 |

## Код программы

* + 1. App.tsx

import React from 'react';

import {Routes, Route} from 'react-router-dom'

import Login from './pages/Login';

import Registration from './pages/Registration';

import ResetPassword from './pages/ResetPassword';

import ChangePassword from './pages/ChangePassword';

import Home from './pages/Home';

import { observer } from 'mobx-react-lite';

import Account from './pages/Account';

import Classroom from './pages/Classroom';

import Task from './pages/Task';

function App() {

return (

<Routes>

<Route path="/" element={<Home/>}/>

<Route path="/signin" element={<Login/>}/>

<Route path="/signup" element={<Registration/>}/>

<Route path="/resetpassword" element={<ResetPassword/>}/>

<Route path="/changepassword/:id" element={<ChangePassword/>}/>

<Route path="/account" element={<Account/>}/>

<Route path="/c/:id" element={<Classroom/>}/>

<Route path="/t/:id" element={<Task/>}/>

</Routes>

);

}

export default observer(App);

* + 1. components\AccountInfo.tsx

import { observer } from 'mobx-react-lite'

import React, {useContext, useState} from 'react'

import { Context } from '..'

import {FiEdit2} from 'react-icons/fi'

import { useNavigate } from 'react-router-dom'

import { ToastContainer, toast } from 'react-toastify'

import ModalImage from './ModalImage'

import { API\_URL } from '../http'

const AccountInfo:React.FC = () => {

const {store} = useContext(Context)

const navigate = useNavigate()

const [openModal, setOpenModal] = useState(false)

const handleLogout = async()=>{

store.logout().then(()=>{

navigate('/signin')

})

}

const handleChangePassword = async()=>{

store.resetPassword(store.user.email).then(()=>{

toast.success('Для смены пароля перейдите по ссылке, которая была выслана на почту', {

position: "top-right",

autoClose: 5000,

hideProgressBar: false,

closeOnClick: true,

pauseOnHover: true,

draggable: true,

progress: undefined,

theme: "light",

});

})

}

const handleCloseModal = () => {

setOpenModal(false);

}

return (

<div className="account-info">

<ToastContainer />

<div className="cont-account-info">

<div className="cnt-photo">

<div className="cont-account">

{!store.user.file&&(

<div className="img-icon" style={{background:store.user.colorProfile}}>

<div style={{paddingTop:'10px'}}>

<p>{store.user.email[0]}</p>

</div>

</div>

)}

{store.user.file&&(

<div style={{ width: "100px", height: "100px", borderRadius: "50%", overflow: "hidden" }}>

<img

src={API\_URL + 'images/' + store.user.file}

alt="avatar"

style={{ width: "100%", height: "100%", objectFit: "cover" }}

/>

</div>

)}

</div>

<div className="addphoto" onClick={()=>setOpenModal(true)}>

<FiEdit2 style={{paddingTop:'9px', color:'white', fontSize:'12px'}}/>

</div>

{openModal&&(

<ModalImage show={openModal} onClose={handleCloseModal} classid=''/>

)}

</div>

<div className="welcome-zone">

<h1>Добро пожаловать, {store.user.lastname} {store.user.surname} {store.user.name}</h1>

<div className="btn-cont">

<button className='logout-btn' onClick={handleLogout}>Выйти из аккаунта</button>

</div>

<div className="btn-cont">

<button className='change-pass-btn' onClick={handleChangePassword}>Сменить пароль</button>

</div>

</div>

</div>

</div>

)

}

export default observer(AccountInfo)

* + 1. components\AddGroupToClass.tsx

import React, { useRef, useEffect, useState, useContext } from 'react';

import '../style/Modal.scss';

import { Context } from '..';

import { observer } from 'mobx-react-lite';

import { PropagateLoader } from 'react-spinners';

import { ToastContainer } from 'react-toastify';

import { BsPlus } from 'react-icons/bs';

import { useParams } from 'react-router-dom';

type ModalProps = {

show: boolean;

onClose: () => void;

};

const Modal: React.FC<ModalProps> = ({ show, onClose}) => {

const {store} = useContext(Context)

const id = useParams()

const modalRef = useRef<HTMLDivElement>(null);

const [loading, setLoading] = useState<boolean>(true)

const [group, setGroup] = useState<any>()

const [groupOld, setGroupOld] = useState<any>()

const [searchValue, setSearchValue] = useState('');

useEffect(() => {

if (show) {

store.getAllGroup(store.user.id).then((data)=>{

if(data){

setGroup(data?.data)

setGroupOld(data?.data)

}

setLoading(false)

})

const handleOutsideClick = (event: MouseEvent) => {

if (modalRef.current && !modalRef.current.contains(event.target as Node)) {

onClose();

}

};

document.addEventListener('mousedown', handleOutsideClick);

return () => {

document.removeEventListener('mousedown', handleOutsideClick);

};

}

// eslint-disable-next-line react-hooks/exhaustive-deps

}, []);

const handleClose = () => {

onClose();

};

const handleAdd = (group:string) =>{

if(id.id)

store.addGroupToClass(group, id.id).then((data)=>{

if(data)

window.location.reload()

})

}

const handleInputChange = (e: React.ChangeEvent<HTMLInputElement>) => {

const value = e.target.value;

setSearchValue(value);

const filteredUsers = group.filter((user: { [s: string]: unknown; } | ArrayLike<unknown>) =>

// eslint-disable-next-line array-callback-return

Object.values(user).some((field:any) =>

field.toString().toLowerCase().includes(value.toLowerCase())

)

);

if(value==="")

setGroup(groupOld)

else

setGroup(filteredUsers);

};

return (

<div className='group-modal'>

<div className="modal-backdrop">

<div className="modal" ref={modalRef} style={{height:'470px', color:'black'}}>

<div className="modal-header" style={{marginTop:'0px'}}>

<h1 style={{fontSize:'28px', width:'12rem', marginTop:'-10px'}}>Участники</h1>

<button className="modal-close" onClick={handleClose} style={{marginTop:'0.8rem'}}>

&times;

</button>

</div>

<div className="modal-body">

{loading?(

<div style={{ display: 'flex', justifyContent: 'center', alignItems: 'center', height: '100vh' }}>

<PropagateLoader color="#335DFF" />

</div>

):(

<div className="modal-body-cont" style={{margin:'0 auto', height:'19rem'}}>

<div>

{group.length!==0&&(

<div className="continput" style={{width:'100%', marginLeft:'-5px'}}>

<p>Поиск:</p>

<input type="text"

placeholder='&#x1F50E;&#xFE0E;'

className='inputlog'

autoComplete='false'

value={searchValue}

onChange={handleInputChange}/>

</div>

)}

{group.length===0&&(

<div className="continput" style={{width:'30rem', marginLeft:'-5px'}}>

<p>Поиск:</p>

<input type="text"

placeholder='&#x1F50E;&#xFE0E;'

className='inputlog'

autoComplete='false'

value={searchValue}

onChange={handleInputChange}/>

</div>

)}

<div className="freeusers-cont" style={{marginLeft:'10px'}}>

{group&&(

<>

{group.map((user:any)=>(

<div className="card-users" style={{margin:'0 auto', marginTop:'20px'}}>

<ToastContainer />

<div className="course-card" style={{width:'27rem', margin:'0 auto', marginBottom:'10px'}}>

<div className="course-card\_\_content" style={{width:'97%'}}>

<h2 style={{fontSize:'16px', width:'100%'}} className="course-card\_\_title">{user.title}</h2>

</div>

<BsPlus onClick={()=>handleAdd(user.id)}/>

</div>

</div>

))}

</>

)}

</div>

</div>

</div>

)}

</div>

<div className="button-cont">

<button style={{background:'#9f9da3', marginLeft:'8rem'}} onClick={handleClose}>Отмена</button>

</div>

</div>

</div>

</div>

);

};

export default observer(Modal);

* + 1. components\AddUserToClass.tsx

import React, { useRef, useEffect, useState, useContext } from 'react';

import '../style/Modal.scss';

import { Context } from '..';

import { observer } from 'mobx-react-lite';

import { PropagateLoader } from 'react-spinners';

import { ToastContainer } from 'react-toastify';

import { API\_URL } from '../http';

import { BsPlus } from 'react-icons/bs';

import checkFileAvailability from '../middleware/image';

import { RxCross2 } from 'react-icons/rx';

type ModalProps = {

show: boolean;

onClose: () => void;

id:any;

role:string;

};

const Modal: React.FC<ModalProps> = ({ show, onClose, id, role }) => {

const {store} = useContext(Context)

const modalRef = useRef<HTMLDivElement>(null);

const [loading, setLoading] = useState<boolean>(true)

const [selectedUser, setSelectedUser] = useState<any>([]);

const [users, setUsers] = useState<any>()

const [usersOld, setUsersOld] = useState<any>()

const [searchValue, setSearchValue] = useState('');

const fetchData = async () => {

try {

const data = await store.freeUsers(id);

if (data?.data) {

setUsersOld(data?.data);

const images = await Promise.all(

data.data.map(async (user: any) => {

const fileUrl = API\_URL + 'images/' + user.file;

const isFileAvailable = await checkFileAvailability(fileUrl);

return {

images: isFileAvailable,

id: user.id,

};

})

);

const userData = await Promise.all(

data.data.map(async(user:any)=>{

const check = images.find((file: any) => file.id === user.id)

if(check?.images){

return {

email: user.email,

colorProfile: user.colorProfile,

file: user.file,

id: user.id,

lastname: user.lastname,

name: user.name,

surname: user.surname

};

}

else{

return {

email: user.email,

colorProfile: user.colorProfile,

file: '',

id: user.id,

lastname: user.lastname,

name: user.name,

surname: user.surname

};

}

})

)

setUsers(userData)

setLoading(false);

}

} catch (error) {

console.error(error);

setLoading(false);

}

};

useEffect(() => {

if (show) {

fetchData();

const handleOutsideClick = (event: MouseEvent) => {

if (modalRef.current && !modalRef.current.contains(event.target as Node)) {

onClose();

}

};

document.addEventListener('mousedown', handleOutsideClick);

return () => {

document.removeEventListener('mousedown', handleOutsideClick);

};

}

// eslint-disable-next-line react-hooks/exhaustive-deps

}, []);

const handleDivClick = (email:string) => {

// Добавление новых данных в массив

const newData = [...selectedUser, email];

setSelectedUser(newData);

};

const handleClose = () => {

onClose();

};

const handleAdd = () =>{

store.addUsersToClass(selectedUser, id, role).then((data)=>{

if(data)

window.location.reload()

})

}

const handleInputChange = (e: React.ChangeEvent<HTMLInputElement>) => {

const value = e.target.value;

setSearchValue(value);

const filteredUsers = users.filter((user: { [s: string]: unknown; } | ArrayLike<unknown>) =>

// eslint-disable-next-line array-callback-return

Object.values(user).some((field:any) =>

field.toString().toLowerCase().includes(value.toLowerCase())

)

);

if(value==="")

setUsers(usersOld)

else

setUsers(filteredUsers);

};

const handleRemoveClick = (email: string) => {

const updatedArray = selectedUser.filter((item:any) => item !== email);

setSelectedUser(updatedArray);

};

return (

<div className='group-modal'>

<div className="modal-backdrop">

<div className="modal" ref={modalRef} style={{height:'470px', color:'black'}}>

<div className="modal-header" style={{marginTop:'0px'}}>

<h1 style={{fontSize:'28px', width:'12rem', marginTop:'-10px'}}>Участники</h1>

<button className="modal-close" onClick={handleClose} style={{marginTop:'0.8rem'}}>

&times;

</button>

</div>

<div className="modal-body">

{loading?(

<div style={{ display: 'flex', justifyContent: 'center', alignItems: 'center', height: '100vh' }}>

<PropagateLoader color="#335DFF" />

</div>

):(

<div className="modal-body-cont" style={{margin:'0 auto', height:'19rem'}}>

<div>

{users.length!==0&&(

<>

{selectedUser.length<3?(

<div className="continput" style={{width:'100%', marginLeft:'-5px'}}>

<p>Поиск:</p>

<input type="text"

placeholder='&#x1F50E;&#xFE0E;'

className='inputlog'

autoComplete='false'

value={searchValue}

onChange={handleInputChange}/>

</div>

):(

<div className="continput" style={{marginLeft:'60px'}}>

<p>Поиск:</p>

<input type="text" placeholder='&#x1F50E;&#xFE0E;' className='inputlog' autoComplete='false'/>

</div>

)}

</>

)}

{users.length===0&&(

<div className="continput" style={{width:'30rem', marginLeft:'-5px'}}>

<p>Поиск:</p>

<input type="text"

placeholder='&#x1F50E;&#xFE0E;'

className='inputlog'

autoComplete='false'

value={searchValue}

onChange={handleInputChange}/>

</div>

)}

{selectedUser.length<3?(

<div className="selected-users" style={{marginLeft:'-5px'}}>

{selectedUser&&(

<>

{selectedUser.map((user:any)=>(

<div className="selected-users\_\_card">

<p>{user}</p>

<RxCross2 style={{marginTop:'10.5px',

fontSize:'12px'}} onClick={()=>handleRemoveClick(user)}/>

</div>

))}

</>

)}

</div>

):(

<div className="selected-users" style={{marginLeft:'60px'}}>

{selectedUser&&(

<>

{selectedUser.map((user:any)=>(

<div className="selected-users\_\_card">

<p>{user}</p>

<RxCross2 style={{marginTop:'10.5px',

fontSize:'12px'}} onClick={()=>handleRemoveClick(user)}/>

</div>

))}

</>

)}

</div>

)}

<div className="freeusers-cont" style={{marginLeft:'10px'}}>

{users&&(

<>

{users.map((user:any)=>(

<div className="card-users" style={{margin:'0 auto', marginTop:'20px'}}>

<ToastContainer />

<div className="course-card" style={{width:'27rem', margin:'0 auto', marginBottom:'10px'}}>

<div className="course-card\_\_icon">

{user?.file ? (

<img

src={API\_URL + 'images/' + user.file}

alt=""

style={{

width: '49px',

height: '49px',

objectFit: 'cover',

borderRadius: '50%',

}}

/>

) : (

<div

style={{

background: user.colorProfile,

width: '49px',

height: '49px',

borderRadius: '50%',

textAlign:'center'

}}

>

<p

style={{

color: 'white',

marginTop: '7px'

}}

>

{user.email[0].toUpperCase()}

</p>

</div>

)}

</div>

<div className="course-card\_\_content" style={{width:'90%'}}>

<h2 style={{fontSize:'16px', width:'100%'}} className="course-card\_\_title">{user.surname} {user.name} {user.lastname}</h2>

<p className="course-card\_\_subtitle" style={{color:'black'}}>{user.email}</p>

</div>

{!selectedUser.includes(user.email) ? (

<BsPlus style={{fontSize:'20px'}} onClick={()=>handleDivClick(user.email)} />

) : null}

</div>

</div>

))}

</>

)}

</div>

</div>

</div>

)}

</div>

<div className="button-cont">

<button style={{background:'#9f9da3'}} onClick={handleClose}>Отмена</button>

<button style={{background:'#335DFF'}} onClick={handleAdd}>Пригласить</button>

</div>

</div>

</div>

</div>

);

};

export default observer(Modal);

* + 1. components\AuthError.tsx

import React from 'react'

export const AuthError = () => {

return (

<div className="err-cnt">

<div className="err-main">

<h1>Ни одного курса</h1>

<img src="https://cdn.dribbble.com/users/902546/screenshots/6233250/11\_copy.png" alt="error" className="err-img" />

</div>

</div>

)

}

* + 1. components\ChangeAccount.tsx

import { observer } from 'mobx-react-lite'

import React, {useContext, useState, useEffect} from 'react'

import { Context } from '..'

import { FiEdit2 } from 'react-icons/fi'

import { API\_URL } from '../http'

import ModalImage from './ModalImage'

import { ToastContainer, toast } from 'react-toastify'

import { useNavigate } from 'react-router-dom'

const ChangeAccount = () => {

const {store} = useContext(Context)

const navigation = useNavigate()

const [surname, setSurname] = useState('')

const [name, setName] = useState('')

const [lastname, setLastname] = useState('')

const [email, setEmail] = useState('')

const [emailNow, setEmailNow] = useState('')

const [openModal, setOpenModal] = useState(false)

useEffect(() => {

setSurname(store.user.surname)

setName(store.user.name)

setLastname(store.user.lastname)

setEmail(store.user.email)

setEmailNow(store.user.email)

// eslint-disable-next-line react-hooks/exhaustive-deps

}, [])

const handleCloseModal = () => {

setOpenModal(false);

}

const handleChangeInfo=async()=>{

store.editProfile(store.user.id, email, surname, name, lastname).then((data)=>{

if(data){

if(email!==emailNow){

store.logout().then(()=>navigation('/signin'))

}

else{

window.location.reload()

}

}

if(store.error){

toast.error(store.error, {

position: "top-right",

autoClose: 5000,

hideProgressBar: false,

closeOnClick: true,

pauseOnHover: true,

draggable: true,

progress: undefined,

theme: "light",

});

}

})

}

return (

<div className="change-account-info">

<div className="change-acc-cont">

<div className="edit-photo">

<ToastContainer />

{!store.user.file&&(

<div className="img-icon" style={{background:store.user.colorProfile}}>

<div style={{paddingTop:'10px'}}>

<p>{store.user.email[0]}</p>

</div>

</div>

)}

{store.user.file&&(

<div style={{ width: "100px", height: "100px", borderRadius: "50%", overflow: "hidden" }}>

<img

src={API\_URL + 'images/' + store.user.file}

alt="avatar"

style={{ width: "100%", height: "100%", objectFit: "cover" }}

/>

</div>

)}

<div className="addphoto" onClick={()=>setOpenModal(true)}>

<FiEdit2 style={{paddingTop:'9px', color:'white', fontSize:'12px'}}/>

</div>

{openModal&&(

<ModalImage show={openModal} onClose={handleCloseModal} classid=''/>

)}

</div>

<div className="continput">

<p>Фамилия:</p>

<input type="text" placeholder='Фамилия' className='inputlog' autoComplete='false'

onChange={(e)=>setSurname(e.target.value)} value={surname}/>

</div>

<div className="continput">

<p>Имя:</p>

<input type="text" placeholder='Имя' className='inputlog' autoComplete='false'

onChange={(e)=>{setName(e.target.value)}} value={name}/>

</div>

<div className="continput">

<p>Отчество:</p>

<input type="text" placeholder='Отчество' className='inputlog' autoComplete='false'

onChange={(e)=>{setLastname(e.target.value)}} value={lastname}/>

</div>

<div className="continput">

<p>Адрес электронной почты:</p>

<input type="email" placeholder='Email' className='inputlog' autoComplete='false'

onChange={(e)=>{setEmail(e.target.value)}} value={email}/>

</div>

<div className="btn-cnt">

<div className="btn">

<button onClick={handleChangeInfo}>Изменить</button>

</div>

</div>

</div>

</div>

)

}

export default observer(ChangeAccount)

* + 1. components\ClassroomLenta.tsx

/\* eslint-disable react-hooks/exhaustive-deps \*/

import { observer } from 'mobx-react-lite'

import React, {useContext, useEffect, useState} from 'react'

import { Context } from '..'

import { useNavigate, useParams } from 'react-router-dom'

import { PropagateLoader } from 'react-spinners'

import { AiFillInfoCircle, AiOutlineInfoCircle } from 'react-icons/ai'

import {BsThreeDotsVertical } from 'react-icons/bs'

import ModalCreateClass from './ModalCreateClass'

import ModalImage from './ModalImage'

import checkFileAvailability from '../middleware/image'

import { API\_URL } from '../http'

import { ToastContainer, toast } from 'react-toastify'

import ModalCreateTask from './ModalCreateTask'

import { TaskMap } from './TaskMap'

const ClassroomLenta = () => {

const {store} = useContext(Context)

const [course, setCourse] = useState<any>()

const [task, setTask] = useState<any>()

const [loading, setLoading] = useState<boolean>(true)

const [openModal, setOpenModal] = useState(false)

const [openModalTask, setOpenModalTask] = useState(false)

const [checkFile, setCheckFile] = useState<boolean>(false)

const [info, setInfo] = useState<boolean>(false)

const [role, setRole] = useState<any>()

const id = useParams()

const navigation = useNavigate()

useEffect(() => {

if(id.id)

store.getCourseOne(id.id, store.user.id).then((data)=>{

if(id.id)

store.viewTasks(id.id).then((tsk:any)=>{

setCourse(data?.data.member[0].class)

setTask(tsk?.data)

setRole(data?.data.member[0].role)

setLoading(false)

})

}).finally(()=>{

handleCheckImage()

})

}, [])

setTimeout(() => {

handleCheckImage()

}, 1000);

const [openModalClass, setOpenModalClass] = useState(false)

const handleCloseClassModal = () => {

setOpenModalClass(false);

}

const handleCheckImage = async() =>{

const fileUrl = API\_URL + 'images/'+course.cover;

const isFileAvailable = await checkFileAvailability(fileUrl);

setCheckFile(isFileAvailable);

}

const handleCloseModal = () => {

setOpenModal(false);

}

const handleCloseModalTask = () => {

setOpenModalTask(false);

}

const handleAddArchive = () =>{

if(id.id)

store.addArchive(id.id).then((data)=>{

if(data)

navigation('/')

})

}

if(loading){

return(

<div style={{width:'100%', paddingTop:'10rem', paddingBottom:'20rem'}}>

<div style={{display:'flex', width:'100%'}}>

<div style={{margin:'0 auto', marginTop:'5rem', textAlign:'center'}}>

<PropagateLoader color="#335DFF" style={{marginLeft:'-15px'}}/>

</div>

</div>

</div>

)

}

const handleCopyLink = () =>{

navigator.clipboard.writeText(`http://localhost:3000/c/${course.id}`);

toast.success('Ссылка скопирована в буфер обмена', {

position: "top-right",

autoClose: 5000,

hideProgressBar: false,

closeOnClick: true,

pauseOnHover: true,

draggable: true,

progress: undefined,

theme: "light",

});

}

return (

<div className="lenta">

<div className="lenta\_\_cont">

<div className="lenta\_\_cont\_\_cover" style={{background: course.decor}}>

{checkFile&&(

<img src={API\_URL + 'images/' + course.cover} alt="" className='image-cover'/>

)}

<div className="lenta\_\_cont\_\_cover\_\_info">

<div className="lenta\_\_cont\_\_cover\_\_info\_\_three-dots">

{role.name==='teacher'&&(

<div className="dropdown">

<button className='dropdown\_\_button\_\_normal'>

<BsThreeDotsVertical style={{color:'white', marginTop:'18px'}}/>

</button>

<div className="dropdown\_\_content">

<div className="dropdownline" onClick={()=>setOpenModal(true)}>Изменить обложку</div>

<div className="dropdownline" onClick={()=>setOpenModalClass(true)}>Изменить информацию</div>

<div className="dropdownline" onClick={handleAddArchive}>Архивировать</div>

</div>

{openModalClass&&(

<ModalCreateClass show={openModalClass} onClose={handleCloseClassModal} course={course}/>

)}

{openModal&&(

<ModalImage show={openModal} onClose={handleCloseModal} classid={id.id}/>

)}

</div>

)}

</div>

{role.name==='teacher'?(

<div className="lenta\_\_cont\_\_cover\_\_info\_\_flex">

<div className="lenta\_\_cont\_\_cover\_\_info\_\_flex\_\_title">

<h1>{course.title}</h1>

<p>{course.chapter}</p>

</div>

{!info&&(

<AiFillInfoCircle className='lenta\_\_cont\_\_cover\_\_info\_\_flex\_\_circle'

onClick={()=>setInfo(!info)}/>

)}

{info&&(

<AiOutlineInfoCircle className='lenta\_\_cont\_\_cover\_\_info\_\_flex\_\_circle'

onClick={()=>setInfo(!info)}/>

)}

</div>

):(

<div className="lenta\_\_cont\_\_cover\_\_info\_\_flex" style={{marginTop:'13rem'}}>

<div className="lenta\_\_cont\_\_cover\_\_info\_\_flex\_\_title">

<h1>{course.title}</h1>

<p>{course.chapter}</p>

</div>

{!info&&(

<AiFillInfoCircle className='lenta\_\_cont\_\_cover\_\_info\_\_flex\_\_circle'

onClick={()=>setInfo(!info)}/>

)}

{info&&(

<AiOutlineInfoCircle className='lenta\_\_cont\_\_cover\_\_info\_\_flex\_\_circle'

onClick={()=>setInfo(!info)}/>

)}

</div>

)}

</div>

</div>

{info&&(

<div className="lenta\_\_cont\_\_subinfo" style={{border:`1px solid ${course.decor}`}}>

<div className="lenta\_\_cont\_\_subinfo\_\_welcomelink">

<h1 style={{color:course.decor}}>Пригласительная ссылка: </h1>

<p className='lenta\_\_cont\_\_subinfo\_\_welcomelink\_\_id'>{course.id}</p>

<button className='lenta\_\_cont\_\_subinfo\_\_welcomelink\_\_copy'

style={{background:course.decor}}

onClick={handleCopyLink}>COPY</button>

<ToastContainer />

</div>

<div className="lenta\_\_cont\_\_subinfo\_\_subject">

<h1 style={{color:course.decor}}>Предмет: </h1>

<p>{course.subject}</p>

</div>

<div className="lenta\_\_cont\_\_subinfo\_\_chapter">

<h1 style={{color:course.decor}}>Раздел: </h1>

<p>{course.chapter}</p>

</div>

<div className="lenta\_\_cont\_\_subinfo\_\_audience">

<h1 style={{color:course.decor}}>Аудитория: </h1>

<p>{course.audience}</p>

</div>

</div>

)}

<div className="lenta\_\_cont\_\_task-content" style={{marginBottom:'15px'}}>

{role.name==='teacher'&&(

<button className="lenta\_\_cont\_\_task-content\_\_btn-add"

style={{background:course.decor}} onClick={()=>setOpenModalTask(true)}>

<p className='lenta\_\_cont\_\_task-content\_\_btn-add\_\_bs-plus'>+</p> Добавить задание

</button>

)}

{openModalTask&&(

<ModalCreateTask show={openModalTask} onClose={handleCloseModalTask} course\_id={id.id} task=''/>

)}

<TaskMap task={task} role={role}/>

</div>

</div>

</div>

)

}

export default observer(ClassroomLenta)

* + 1. components\ClassroomTask.tsx

import { observer } from 'mobx-react-lite'

import React, { useContext, useEffect, useState } from 'react'

import { PropagateLoader } from 'react-spinners'

import { Context } from '..'

import { useParams } from 'react-router-dom'

import Select from 'react-select'

import ModalTheme from './ModalTheme'

import { ThemeMap } from './ThemeMap'

import { TaskMap } from './TaskMap'

const ClassroomTask = () => {

const {store} = useContext(Context)

const [loading, setLoading] = useState<boolean>(true)

const [course, setCourse] = useState<any>()

const id = useParams()

const [task, setTask] = useState<any>()

const [role, setRole] = useState<any>()

const [allTask, setAllTask] = useState<any>()

const [options, setOptions] = useState<any>()

const [selected, setSelected] = useState<any>()

const [allTheme, setAllTheme] = useState<any>()

useEffect(() => {

if(id.id)

store.getCourseOne(id.id, store.user.id).then((data)=>{

if(id.id)

store.viewAllTheme(id.id).then((themeData)=>{

if(id.id)

store.allTaskWithoutTheme(id.id).then((allTaskData)=>{

if(id.id)

store.allTaskWithTheme(id.id).then((taskData)=>{

if(themeData){

const options: { value: string; label: string }[] = [

{ value: '--', label: 'Все темы' },

...themeData.data.map(item => ({

value: item.id,

label: item.title

}))

];

setSelected(options[0])

setOptions(options)

}

setTask(taskData?.data)

setAllTheme(taskData?.data)

setCourse(data?.data.member[0].class)

setRole(data?.data.member[0].role)

setAllTask(allTaskData?.data)

setLoading(false)

})

})

})

})

}, [id.id, store])

const handleChange = (option: any) => {

setSelected(option);

if(option.value==="--"){

setTask(allTheme)

}else if(option.value!=="--"){

const selectedData = allTheme.filter((item: { id: any }) => item.id === option.value);

setTask(selectedData)

}

};

const [openModalTheme, setOpenModalTheme] = useState(false)

const handleCloseThemeModal = () => {

setOpenModalTheme(false);

}

if(loading){

return(

<div style={{width:'100%', paddingTop:'10rem', paddingBottom:'20rem'}}>

<div style={{display:'flex', width:'100%'}}>

<div style={{margin:'0 auto', marginTop:'5rem', textAlign:'center'}}>

<PropagateLoader color="#335DFF" style={{marginLeft:'-15px'}}/>

</div>

</div>

</div>

)

}

return (

<div className="classroom-task">

<div className="classroom-task\_\_body">

<div className="classroom-task\_\_body\_\_button-cont">

{role.name==='teacher'&&(

<div className="button">

<button style={{background:course.decor}} onClick={()=>setOpenModalTheme(true)}>Добавить тему</button>

</div>

)}

{openModalTheme&&(

<ModalTheme show={openModalTheme} onClose={handleCloseThemeModal} theme='' course={id.id}/>

)}

<Select

options={options}

className='select'

defaultValue={options[0]}

value={selected}

onChange={handleChange}/>

</div>

{selected.value==='--'&&(

<div className="classroom-task\_\_body\_\_task-body-no-theme">

<TaskMap task={allTask} role={role}/>

</div>

)}

<div className="classroom-task\_\_body\_\_theme-body">

<ThemeMap theme={task} role={role}/>

</div>

</div>

</div>

)

}

export default observer(ClassroomTask)

* + 1. components\ClassroomUsers.tsx

import { observer } from 'mobx-react-lite'

import React, {useContext, useEffect, useState} from 'react'

import { Context } from '..'

import { useParams } from 'react-router-dom'

import { PropagateLoader } from 'react-spinners'

import {HiOutlineUserAdd} from 'react-icons/hi'

import { UserMap } from './UserMap'

import AddUserToClass from './AddUserToClass'

import AddGroupToClass from './AddGroupToClass'

const ClassroomUsers = () => {

const {store} = useContext(Context)

const id = useParams()

const [course, setCourse] = useState<any>()

const [role, setRole] = useState<any>()

const [teacher, setTeacher] = useState<any>()

const [student, setStudent] = useState<any>()

const [loading, setLoading] = useState<boolean>(true)

const [openModalUsersTeacher, setOpenModalUsersTeacher] = useState(false)

const [openModalUsersStudent, setOpenModalUsersStudent] = useState(false)

const [openModalGroup, setOpenModalGroup] = useState(false)

useEffect(() => {

if(id.id)

store.getCourseOne(id.id, store.user.id).then((data)=>{

if(id.id)

store.viewTeacher(id.id).then((teacherData)=>{

if(id.id)

store.viewStudent(id.id).then((studentData)=>{

setTeacher(teacherData?.data)

setStudent(studentData?.data)

setCourse(data?.data.member[0].class)

setRole(data?.data.member[0].role)

setLoading(false)

})

})

})

}, [id.id, store])

if(loading){

return(

<div style={{width:'100%', paddingTop:'10rem', paddingBottom:'20rem'}}>

<div style={{display:'flex', width:'100%'}}>

<div style={{margin:'0 auto', marginTop:'5rem', textAlign:'center'}}>

<PropagateLoader color="#335DFF" style={{marginLeft:'-15px'}}/>

</div>

</div>

</div>

)

}

const handleCloseModalClassTeacher = () => {

setOpenModalUsersTeacher(false);

}

const handleCloseModalClassStudent = () => {

setOpenModalUsersStudent(false);

}

const handleAddGroupToClassroom = () => {

setOpenModalGroup(false);

}

const handleRemoveStudent = () => {

if(id.id)

store.viewStudent(id.id).then((studentData)=>{

setStudent(studentData?.data)

})

};

const handleRemoveTeacher = () => {

if(id.id)

store.viewTeacher(id.id).then((teacherData)=>{

setTeacher(teacherData?.data)

})

};

return (

<div className="classroom-users">

<div className="classroom-users\_\_teacher-body">

<div className="classroom-users\_\_teacher-body\_\_header"

style={{color:course.decor, borderBottom:`2px solid ${course.decor}`}}>

<h1 style={{width:'98%'}}>Преподаватели</h1>

{role.name==="teacher"&&(

<HiOutlineUserAdd

style={{fontSize:'24px', marginTop:'35px'}}

onClick={()=>setOpenModalUsersTeacher(true)}/>

)}

</div>

{teacher&&(

<UserMap userList={teacher} author={course} onRemoveUser={handleRemoveTeacher}role={role} />

)}

{openModalUsersTeacher&&(

<AddUserToClass show={openModalUsersTeacher} onClose={handleCloseModalClassTeacher} id={id.id} role='teacher'/>

)}

</div>

<div className="classroom-users\_\_teacher-body">

<div className="classroom-users\_\_teacher-body\_\_header"

style={{color:course.decor, borderBottom:`2px solid ${course.decor}`}}>

<h1 style={{width:'100%'}}>Учащиеся</h1>

{role.name==="teacher"&&(

<div className="dropdown">

<button className='dropdown\_\_button\_\_normal' style={{marginLeft:'10px'}}>

<HiOutlineUserAdd style={{fontSize:'24px',

marginTop:'35px',

color:course.decor,

paddingLeft:'10px'}}/>

</button>

<div className="dropdown\_\_content">

<div className="dropdownline" onClick={()=>setOpenModalUsersStudent(true)}>Добавить учащегося</div>

<div className="dropdownline" onClick={()=>setOpenModalGroup(true)}>Добавить группу</div>

</div>

{openModalUsersStudent&&(

<AddUserToClass show={openModalUsersStudent} onClose={handleCloseModalClassStudent} id={id.id} role='student'/>

)}

{openModalGroup&&(

<AddGroupToClass show={openModalGroup} onClose={handleAddGroupToClassroom}/>

)}

</div>

)}

</div>

{student&&(

<UserMap userList={student} author={course} onRemoveUser={handleRemoveStudent} role={role}/>

)}

</div>

</div>

)

}

export default observer(ClassroomUsers)

* + 1. components\CourseCard.tsx

import React, {useState, useContext} from 'react'

import { BsThreeDotsVertical } from 'react-icons/bs'

import { API\_URL } from '../http'

import ModalCreateClass from './ModalCreateClass'

import { Context } from '..'

import { useNavigate } from 'react-router-dom'

import checkFileAvailability from '../middleware/image'

export const CourseCard = (courses:any) => {

const navigation = useNavigate()

const {store} = useContext(Context)

const [checkFile, setCheckFile] = useState<boolean>(false)

const [checkAva, setCheckAva] = useState<boolean>(false)

const [openModalClass, setOpenModalClass] = useState(false)

const handleCloseModal = () => {

setOpenModalClass(false);

}

const handleAddArchive = () =>{

store.addArchive(courses.courses.id).then((data)=>{

if(data)

window.location.reload()

})

}

setTimeout(() => {

handleCheckImage()

handleCheckAva()

}, 1000);

const handleCheckImage = async() =>{

const fileUrl = API\_URL + 'images/'+courses.courses.cover;

const isFileAvailable = await checkFileAvailability(fileUrl);

setCheckFile(isFileAvailable);

}

const handleCheckAva = async() =>{

const fileUrl = API\_URL + 'images/'+courses.courses.user.file;

const isFileAvailable = await checkFileAvailability(fileUrl);

setCheckAva(isFileAvailable);

}

return (

<div className="courser-card">

<div className="courser-card\_\_header" style={{background:courses.courses.decor, overflow:'hidden',

position:'relative'}}>

{checkFile&&(

<img src={API\_URL + 'images/' + courses.courses.cover}

alt="" className='image-cover'/>

)}

<div className="courser-card\_\_header\_\_info"

style={{zIndex:'0'}} onClick={()=>navigation(`/c/${courses.courses.id}`)}>

<h1 className="courser-card\_\_header\_\_info\_\_title">{courses.courses.title}</h1>

<p className="courser-card\_\_header\_\_info\_\_subtitle">{courses.courses.chapter}</p>

</div>

{store.user.id===courses.courses.user\_id&&(

<div className="dropdown" style={{position:'absolute', right:'0'}}>

<button className='dropdown\_\_button\_\_normal'>

<BsThreeDotsVertical style={{color:'white', marginTop:'18px'}}/>

</button>

<div className="dropdown\_\_content" style={{marginLeft:'-145px'}}>

<div className="dropdownline" onClick={()=>setOpenModalClass(true)}>Изменить</div>

<div className="dropdownline" onClick={handleAddArchive}>Архивировать</div>

</div>

{openModalClass&&(

<ModalCreateClass show={openModalClass} onClose={handleCloseModal} course={courses.courses}/>

)}

</div>

)}

</div>

<div className="courser-card\_\_body">

<p className="courser-card\_\_body\_\_author">

{courses.courses.user.surname} {courses.courses.user.name} {courses.courses.user.lastname}

</p>

<div className="courser-card\_\_body\_\_avatar" style={{overflow: "hidden",

background:courses.courses.user.colorProfile, textAlign:'center'}}>

{!checkAva?(

<p style={{color:'white', fontSize:'24px', marginTop:'15px'}}>{courses.courses.user.email[0].toUpperCase()}</p>

):(

<img src={API\_URL + 'images/' + courses.courses.user.file} alt=""

style={{width:'60.5px', height:'60.5px', objectFit:'cover'}} />

)}

</div>

</div>

</div>

)

}

* + 1. components\CourseMap.tsx

import React from 'react'

import { CourseCard } from './CourseCard'

export const CourseMap = (courses:any) => {

return (

courses.courses.map((course:any)=>(

<CourseCard courses={course}/>

))

)

}

* + 1. components\FileInput.tsx

import React, { useState } from 'react';

import '../style/FileInput.scss';

type Props = {

multiple?: boolean;

accept?: string;

onChange: (event: React.ChangeEvent<HTMLInputElement>) => void;

onDelete?: (index: number) => void;

};

const FileInput: React.FC<Props> = ({

multiple,

accept,

onChange,

onDelete,

}) => {

const [fileNames, setFileNames] = useState<string[]>([]);

const handleFileChange = (event: React.ChangeEvent<HTMLInputElement>) => {

const newFileNames = [...fileNames];

for (let i = 0; i < event.target.files!.length; i++) {

newFileNames.push(event.target.files![i].name);

}

setFileNames(newFileNames);

onChange(event);

};

const handleFileDelete = (index: number) => {

const newFileNames = [...fileNames];

newFileNames.splice(index, 1);

setFileNames(newFileNames);

if (onDelete) {

onDelete(index);

}

};

return (

<div className="file-input">

<input

type="file"

multiple={multiple}

accept={accept}

onChange={handleFileChange}

/>

<ul className="file-list">

{fileNames.map((fileName, index) => (

<li key={index} className="file-item">

{fileName}

{onDelete && (

<button

className="file-delete"

onClick={() => handleFileDelete(index)}

>

X

</button>

)}

</li>

))}

</ul>

</div>

);

};

export default FileInput;

* + 1. components\Footer.tsx

import React from 'react'

import logo from '../image/logo.svg'

import {FaTelegramPlane, FaGithubAlt} from 'react-icons/fa'

import {IoMail} from 'react-icons/io5'

import '../style/Footer.scss'

export const Footer = () => {

return (

<div className='footer'>

<div className="footer-cont">

<div className="left-cont-footer" style={{width:'100%'}}>

<p>Обучение с <img src={logo} alt="logo" className='logo-image-footer'/> становится более интересным и увлекательным, а также более <br/>

эффективным и доступным для всех ;)

</p>

</div>

<div className="right-cont-footer" style={{width:'10%'}}>

<div className="icons-cont-footer" style={{width:'10rem'}}>

<a href="https://t.me/rbwitch">

<FaTelegramPlane style={{marginLeft:'1px'}}/>

</a>

<a href="https://github.com/PaiNaiP">

<FaGithubAlt style={{marginLeft:'1rem'}}/>

</a>

<a href="mailto:isip\_e.i.batygina@mpt.ru">

<IoMail style={{marginLeft:'1.2rem'}}/>

</a>

</div>

<p style={{marginTop:'-10px'}}>©EasyAssign2023</p>

</div>

</div>

</div>

)

}

* + 1. components\FreeUserMap.tsx

import React from 'react'

import { FreeUsersGroupCard } from './FreeUsersGroupCard'

export const FreeUserMap = (users:any) => {

return (

users.users.map((user:any)=>(

<FreeUsersGroupCard users={user} group={users.group} onRemoveUser={users.onRemoveUser} cross={users.cross}/>

))

)

}

* + 1. components\FreeUsersGroupCard.tsx

/\* eslint-disable react-hooks/exhaustive-deps \*/

import React, {useContext, useEffect, useState} from 'react'

import { API\_URL } from '../http'

import { BsPlus } from 'react-icons/bs'

import { Context } from '..'

import { RxCross2 } from 'react-icons/rx'

import { ToastContainer, toast } from 'react-toastify'

export const FreeUsersGroupCard = (users:any) => {

const {store} = useContext(Context)

const [user, setUser] = useState<any>()

const [loading, setLoading] = useState<boolean>(true)

const [marginLeft, setMarginLeft] = useState<string>('')

const handleAddUserToGroup=()=>{

store.addUserToGroup(users.users.id, users.group).then((data)=>{

if(data?.data)

toast.success(`Пользователю ${user.surname} ${user.name} ${user.lastname} отправлено приглашение на почту` , {

position: "top-right",

autoClose: 5000,

hideProgressBar: false,

closeOnClick: true,

pauseOnHover: true,

draggable: true,

progress: undefined,

theme: "light",

});

})

}

const handleDelete = () =>{

store.deleteUserFromGroup(users.users.id, users.group)

.then((data: any) => {

if (data) {

// Если удаление успешно, обновить список пользователей без удаленного пользователя

// Получаем текущий список пользователей

const currentUsers = users;

const updatedUsers: any[] = [];

for (let i = 0; i < currentUsers.length; i++) {

const user = currentUsers[i];

if (user.id !== users.users.id) {

updatedUsers.push(user);

}

}

// Вызов функции userList.onRemoveUser() с обновленным списком пользователей

users.onRemoveUser(updatedUsers);

}

});

}

useEffect(() => {

if(users.users.surname){

setUser(users.users)

setMarginLeft('50px')

setLoading(false)

}

else if(users.users.user.id)

{

setUser(users.users.user)

setMarginLeft('10px')

setLoading(false)

}

}, [])

if(loading){

return(

<div>...loading</div>

)

}

return (

<div className="card-users" style={{marginLeft:marginLeft, marginTop:'20px'}}>

<ToastContainer />

<div className="course-card" style={{width:'27rem', margin:'0 auto'}}>

<div className="course-card\_\_icon">

{users.users.file&&(

<img src={API\_URL + 'images/' + users.users.file} alt="" style={{width:'49px', height:'49px', objectFit:'cover', borderRadius:'50%'}}/>

)}

{!users.users.file&&(

<div style={{background:users.users.colorProfile, width:'49px', height:'49px', borderRadius:'50%', textAlign:'center'}}>

<p style={{color:'white', marginTop:'7px'}} >{users.users.email[0].toUpperCase()}</p>

</div>

)}

</div>

<div className="course-card\_\_content" style={{width:'90%'}}>

<h2 style={{fontSize:'16px', width:'100%'}} className="course-card\_\_title">{users.users.surname} {users.users.name} {users.users.lastname}</h2>

<p className="course-card\_\_subtitle" style={{color:'black'}}>{users.users.email}</p>

</div>

{!users.cross&&(

<BsPlus style={{fontSize:'20px'}} onClick={handleAddUserToGroup}/>

)}

{users.cross&&(

<RxCross2 style={{fontSize:'20px'}} onClick={handleDelete}/>

)}

</div>

</div>

)

}

* + 1. components\FreeUsersGroupMap.tsx

import React from 'react'

import { FreeUsersGroupCard } from './FreeUsersGroupCard'

export const FreeUsersGroupMap = (users:any) => {

return (

users.users.map((user:any)=>(

<FreeUsersGroupCard users={user.user} group={users.group} onRemoveUser={users.onRemoveUser} cross={users.cross}/>

))

)

}

* + 1. components\GroupsCard.tsx

import React, {useState, useContext} from 'react'

import { BsThreeDotsVertical } from 'react-icons/bs'

import { TiGroup } from 'react-icons/ti'

import ModalCreateGroup from './ModalCreateGroup'

import { Context } from '..'

import ModalUsersGroups from './ModalUsersGroups'

export const GroupsCard = (groups:any) => {

const {store} = useContext(Context)

const [openModalGroup, setOpenModalGroup] = useState(false)

const [openModalUsers, setOpenModalUsers] = useState(false)

const handleCloseModalGroup = () => {

setOpenModalGroup(false);

}

const handleCloseModelUsers = () => {

setOpenModalUsers(false);

}

const handleDeleteGroup = () =>{

store.deleteGroupe(groups.groups.id).then((data)=>{

if(data)

window.location.reload()

})

}

return (

<div className="card-group" style={{width:'20rem'}}>

<div className="course-card-cont">

<div className="course-card\_\_icon">

<TiGroup/>

</div>

<div className="course-card\_\_content" style={{width:'47%'}}>

<h2 className="course-card\_\_title" style={{width:'100%'}}>{groups.groups.title}</h2>

</div>

<div className="dropdown">

<button className='dropdown\_\_button\_\_normal'>

<BsThreeDotsVertical/>

</button>

<div className="dropdown\_\_content">

<div className="dropdownline" onClick={()=>setOpenModalGroup(true)}>Изменить группу</div>

<div className="dropdownline" onClick={handleDeleteGroup}>Удалить группу</div>

<div className="dropdownline" onClick={()=>setOpenModalUsers(true)}>Участники</div>

</div>

{openModalGroup&&(

<ModalCreateGroup show={openModalGroup} onClose={handleCloseModalGroup} titleGroup={groups.groups.title} idGroup={groups.groups.id}/>

)}

{openModalUsers&&(

<ModalUsersGroups show={openModalUsers} onClose={handleCloseModelUsers} id={groups.groups.id}/>

)}

</div>

</div>

</div>

)

}

* + 1. components\GroupsMap.tsx

import React from 'react'

import { GroupsCard } from './GroupsCard'

export const GroupsMap = (groups:any) => {

return (

groups.groups.map((group:any)=>(

<GroupsCard groups={group}/>

))

)

}

* + 1. components\HeaderHome.tsx

import { observer } from 'mobx-react-lite'

import React, {useContext, useState} from 'react'

import logo from '../image/logo.svg'

import { Context } from '..'

import {RxHamburgerMenu} from 'react-icons/rx'

import { useNavigate } from 'react-router-dom'

import { API\_URL } from '../http'

import Sidebar from './Sidebar'

export const HeaderHome = () => {

const {store} = useContext(Context)

const navigate = useNavigate()

const [openDrawer, setOpenDrawer] = useState(false)

const handleCloseDrawer = () => {

setOpenDrawer(false);

}

return (

<header className="header">

<div className="header\_\_left">

<RxHamburgerMenu className='hamburmenu' onClick={()=>setOpenDrawer(true)}/>

{openDrawer&&(

<Sidebar isOpen={openDrawer} onClose={handleCloseDrawer}/>

)}

<button className="header\_\_menu-btn">

<i className="fas fa-bars"></i>

</button>

<h1 className="header\_\_title">

<img src={logo} alt="" style={{width:'90%'}}/>

</h1>

</div>

<div className="header\_\_right" onClick={()=>navigate('/account')}>

<button className="header\_\_button header\_\_button--account">

<span className="header\_\_username">{store.user.email}</span>

</button>

<button className="header\_\_button header\_\_button--notifications">

{!store.user.file&&(

<div className="img-icon" style={{background:store.user.colorProfile}}>

<div style={{paddingTop:'3px'}}>

<p>{store.user.email[0]}</p>

</div>

</div>

)}

{store.user.file&&(

<div style={{ width: "60px", height: "60px", borderRadius: "50%", overflow: "hidden" }}>

<img

src={API\_URL + 'images/' + store.user.file}

alt="avatar"

style={{ width: "100%", height: "100%", objectFit: "cover" }}

/>

</div>

)}

</button>

</div>

</header>

)

}

export default observer(HeaderHome)

* + 1. components\HeaderHomeNoneAuth.tsx

import React from 'react'

import logo from '../image/logo.svg'

import { useNavigate } from 'react-router-dom'

export const HeaderHomeNoneAuth = () => {

const navigate = useNavigate()

return (

<header className="header">

<div className="header\_\_left">

<button className="header\_\_menu-btn">

<i className="fas fa-bars"></i>

</button>

<h1 className="header\_\_title">

<img src={logo} alt="" style={{width:'90%'}}/>

</h1>

</div>

<div className="header\_\_right">

<button className="header\_\_button header\_\_button--notifications">

<i className="fas fa-bell"></i>

</button>

<button className="header\_\_button header\_\_button--account">

<i className="fas fa-user-circle"></i>

<span onClick={()=>navigate('/signin')}>Авторизоваться</span>

</button>

</div>

</header>

)

}

* + 1. components\MainHomeIsAuth.tsx

/\* eslint-disable react-hooks/exhaustive-deps \*/

import { observer } from 'mobx-react-lite'

import React, {useState, useEffect, useContext} from 'react'

import { Context } from '..'

import { BsPlus } from 'react-icons/bs'

import ModalCreateClass from './ModalCreateClass'

import '../style/Dropdown.scss'

import { PropagateLoader } from 'react-spinners'

import ModalCreateGroup from './ModalCreateGroup'

import { GroupsMap } from './GroupsMap'

import { CourseMap } from './CourseMap'

import { AuthError } from './AuthError'

const MainHomeIsAuth = () => {

const {store} = useContext(Context)

const [openModalClass, setOpenModalClass] = useState(false)

const [openModalGroup, setOpenModalGroup] = useState(false)

const [courseInfo, setCourseInfo] = useState<any>()

const [group, setGroup] = useState<any>()

const [loading, setLoading] = useState<boolean>(true)

useEffect(() => {

store.checkAuth().then(()=>{

store.getAllClass(store.user.id).then((data)=>{

store.getAllGroup(store.user.id).then((dataGroup)=>{

setCourseInfo(data?.data)

setGroup(dataGroup?.data)

setLoading(false)

})

})

})

}, [])

if(loading){

return(

<div style={{width:'100%', paddingTop:'10rem', paddingBottom:'20rem'}}>

<div style={{display:'flex', width:'100%'}}>

<div style={{margin:'0 auto', marginTop:'5rem', textAlign:'center'}}>

<PropagateLoader color="#335DFF" style={{marginLeft:'-15px'}}/>

</div>

</div>

</div>

)

}

const handleCloseModal = () => {

setOpenModalClass(false);

}

const handleCloseModalGroup = () => {

setOpenModalGroup(false);

}

return (

<div className="main">

<div className="main\_\_content-home">

<div className="main\_\_content-home\_\_cont" style={{display:'flex'}}>

<div className="dropdown">

<button className="dropbtn">

<BsPlus/>

</button>

<div className="dropdown-content">

<div className="dropdownline" onClick={()=>setOpenModalClass(true)}>Создать курс</div>

<div className="dropdownline" onClick={()=>setOpenModalGroup(true)}>Создать группу</div>

</div>

{openModalClass&&(

<ModalCreateClass show={openModalClass} onClose={handleCloseModal} course=''/>

)}

{openModalGroup&&(

<ModalCreateGroup show={openModalGroup} onClose={handleCloseModalGroup} titleGroup='' idGroup=''/>

)}

</div>

</div>

{group&&(

<div className="courses" style={{display:'flex', margin:'0 auto'}}>

<div className="courses-div" style={{margin: '0 auto'}}>

<GroupsMap groups={group} />

</div>

</div>

)}

<div style={{margin:'0 auto', display:'flex', width:'100%'}}>

<div style={{margin:'0 auto', width:'100%'}} className='courseCont'>

{courseInfo&&(

<div className="course" style={{margin:'0 auto'}}>

<div className="course\_\_cont" style={{margin:'0 auto'}}>

<CourseMap courses={courseInfo} />

</div>

</div>

)}

</div>

</div>

{courseInfo.length===0 && group.length===0&&(

<AuthError/>

)}

</div>

</div>

)

}

export default observer(MainHomeIsAuth)

* + 1. components\ModalAddToTheme.tsx

import React, { useRef, useEffect, useState, useContext } from 'react';

import '../style/Modal.scss';

import { Context } from '..';

import { observer } from 'mobx-react-lite';

import { PropagateLoader } from 'react-spinners';

import Select from 'react-select'

type ModalProps = {

show: boolean;

onClose: () => void;

course:string;

task:any;

};

const Modal: React.FC<ModalProps> = ({ show, onClose, course, task}) => {

const {store} = useContext(Context)

const modalRef = useRef<HTMLDivElement>(null);

const [loading, setLoading] = useState<boolean>(true)

const [options, setOptions] = useState<any>()

const [selected, setSelected] = useState<any>()

useEffect(() => {

store.viewAllTheme(course).then((data)=>{

const option = data?.data.map((item: { id: any; title: any; }) => ({

value: item.id,

label: item.title

}));

setOptions(option)

setLoading(false)

})

const handleOutsideClick = (event: MouseEvent) => {

if (modalRef.current && !modalRef.current.contains(event.target as Node)) {

onClose();

}

};

if (show) {

document.addEventListener('mousedown', handleOutsideClick);

}

return () => {

document.removeEventListener('mousedown', handleOutsideClick);

};

// eslint-disable-next-line react-hooks/exhaustive-deps

}, []);

const handleClose = () => {

onClose();

};

const handleChange = (option: any) => {

setSelected(option);

};

const handleAddTheme = ()=>{

store.addTheme(task.id, selected.value).then((data)=>{

if(data)

window.location.reload()

})

}

return (

<div className='theme-modal'>

<div className="modal-backdrop">

<div className="modal" ref={modalRef} style={{height:'300px'}}>

<div className="modal-header">

<div>

<h1>Добавление темы</h1>

<p style={{marginTop:'-10px', color:'#6D6875'}}>{task.title}</p>

</div>

<button className="modal-close" onClick={handleClose} style={{marginTop:'1rem', marginLeft:'-67px'}}>

&times;

</button>

</div>

<div className="modal-body">

{loading&&(

<div style={{width:'100%', paddingTop:'2rem', paddingBottom:'20rem'}}>

<div style={{display:'flex', width:'100%'}}>

<div style={{margin:'0 auto', marginTop:'5rem', textAlign:'center'}}>

<PropagateLoader color="#335DFF" style={{marginLeft:'-15px'}}/>

</div>

</div>

</div>

)}

{!loading&&(

<>

<div className="continput">

<p>Темы:</p>

<Select

options={options}

className='select'

defaultValue={options[0]}

onChange={handleChange}

value={selected}/>

</div>

<div className="btn-cnt" >

<button className='cancel' onClick={handleClose}>Отменить</button>

<button className='save' onClick={handleAddTheme}>Сохранить</button>

</div>

</>

)}

</div>

</div>

</div>

</div>

);

};

export default observer(Modal);

* + 1. components\ModalAddUserToGroup.tsx

import React, { useRef, useEffect, useState, useContext } from 'react';

import '../style/Modal.scss';

import { Context } from '..';

import { observer } from 'mobx-react-lite';

import { PropagateLoader } from 'react-spinners';

import { FreeUserMap } from './FreeUserMap';

type ModalProps = {

show: boolean;

onClose: () => void;

id:string;

};

const Modal: React.FC<ModalProps> = ({ show, onClose, id }) => {

const {store} = useContext(Context)

const modalRef = useRef<HTMLDivElement>(null);

const [loading, setLoading] = useState<boolean>(true)

const [users, setUsers] = useState<any>()

const [searchValue, setSearchValue] = useState('');

const [usersOld, setUsersOld] = useState<any>();

useEffect(() => {

store.viewFreeUsersInGroup(id).then((data)=>{

if(data?.data){

setUsers(data?.data)

setUsersOld(data?.data)

}

setLoading(false)

})

const handleOutsideClick = (event: MouseEvent) => {

if (modalRef.current && !modalRef.current.contains(event.target as Node)) {

onClose();

}

};

if (show) {

document.addEventListener('mousedown', handleOutsideClick);

}

return () => {

document.removeEventListener('mousedown', handleOutsideClick);

};

// eslint-disable-next-line react-hooks/exhaustive-deps

}, []);

const handleClose = () => {

onClose();

};

const handleInputChange = (e: React.ChangeEvent<HTMLInputElement>) => {

const value = e.target.value;

setSearchValue(value);

const filteredUsers = users.filter((user: { [s: string]: unknown; } | ArrayLike<unknown>) =>

// eslint-disable-next-line array-callback-return

Object.values(user).some((field:any) =>

field.toString().toLowerCase().includes(value.toLowerCase())

)

);

if(value==="")

setUsers(usersOld)

else

setUsers(filteredUsers);

};

return (

<div className='group-modal'>

<div className="modal-backdrop">

<div className="modal" ref={modalRef} style={{height:'470px'}}>

<div className="modal-header" style={{marginTop:'0px'}}>

<h1 style={{fontSize:'28px', width:'12rem'}}>Участники</h1>

<button className="modal-close" onClick={handleClose} style={{marginTop:'0.8rem'}}>

&times;

</button>

</div>

<div className="modal-body">

<div className="modal-body-cont" style={{margin:'0 auto', height:'19rem', display:'flex'}}>

{loading&&(

<div style={{margin:'0 auto', marginTop:'8rem'}}>

<PropagateLoader

color="#335DFF"

style={{marginLeft:'-5px'}}

className='loaderone'

/>

</div>

)}

{!loading&&(

<div>

{users.length===0?(

<div style={{width:'30rem'}} className="continput">

<p style={{marginLeft:'-7px'}}>Поиск:</p>

<input type="text" style={{marginLeft:'-7px'}}

placeholder='&#x1F50E;&#xFE0E;'

className='inputlog'

autoComplete='false'

value={searchValue}

onChange={handleInputChange}/>

</div>

):users&&(

<div className="continput">

<p>Поиск:</p>

<input type="text"

placeholder='&#x1F50E;&#xFE0E;'

className='inputlog'

autoComplete='false'

value={searchValue}

onChange={handleInputChange}/>

</div>

)}

<div className="freeusers-cont">

<FreeUserMap users={users} group={id} cross={false}/>

</div>

</div>

)}

</div>

</div>

<div className="button-cont">

<button style={{background:'#9f9da3', marginLeft:'8rem'}} onClick={handleClose}>Отмена</button>

</div>

</div>

</div>

</div>

);

};

export default observer(Modal);

* + 1. components\ModalCreateClass.tsx

import React, { useRef, useEffect, useState, useContext } from 'react';

import '../style/Modal.scss';

import {BsCheck} from 'react-icons/bs'

import {BiPlus} from 'react-icons/bi'

import { ChromePicker } from 'react-color'

import { Context } from '..';

import { observer } from 'mobx-react-lite';

interface Color {

code: string;

}

const colors: Color[] = [

{ code: "#AEB5C8" },

{ code: "#F92929" },

{ code: "#FA6E42" },

{ code: "#FBCD57" },

{ code: "#82BF5C" },

{ code: "#657DFC" },

{ code: "#89F0EA" },

{ code: "#A957FB" },

{ code: "#2A2E31" },

{ code: "#EC59B1" },

{ code: "#FB576B" },

];

type ModalProps = {

show: boolean;

onClose: () => void;

course:any;

};

const Modal: React.FC<ModalProps> = ({ show, onClose, course }) => {

const {store} = useContext(Context)

const modalRef = useRef<HTMLDivElement>(null);

const [colorz, setColorz] = useState<any>();

const [edit, setEdit] = useState<boolean>(false);

const [colorHex, setColorHex] = useState<any>();

const [colorPicker, setColorPicker] = useState<boolean>(false);

const [titleClass, setTitleClass] = useState<string>()

const [chapter, setChapter] = useState<string>()

const [object, setObject] = useState<string>()

const [audience, setAudience] = useState<string>()

useEffect(() => {

handleRandomColorClick()

setColorHex('#EBB893')

if(course.decor){

setColorz(course.decor)

setTitleClass(course.title)

setChapter(course.chapter)

setObject(course.subject)

setAudience(course.audience)

setEdit(true)

}

const handleOutsideClick = (event: MouseEvent) => {

if (modalRef.current && !modalRef.current.contains(event.target as Node)) {

onClose();

}

};

if (show) {

document.addEventListener('mousedown', handleOutsideClick);

}

return () => {

document.removeEventListener('mousedown', handleOutsideClick);

};

// eslint-disable-next-line react-hooks/exhaustive-deps

}, []);

const handleColorChange = (color: any) => {

setColorHex(color.hex);

setColorz(color.hex)

};

const handleClose = () => {

onClose();

};

const handleRandomColorClick = () => {

const randomIndex = Math.floor(Math.random() \* colors.length);

setColorz(colors[randomIndex]);

};

const handleCreateClass=async()=>{

if(edit){

if(titleClass&&chapter&&object&&audience)

store.editCourse(titleClass, chapter, object, audience, course.id, colorz).then((data)=>{

if(data)

window.location.reload()

})

}

else{

if(titleClass&&chapter&&object&&audience)

store.createClass(titleClass, chapter, object, audience, colorz, store.user.id).then((data)=>{

if(data)

window.location.reload()

})

}

}

return (

<div className='class-modal'>

<div className="modal-backdrop">

<div className="modal" ref={modalRef}>

<div className="modal-header">

{edit&&(

<h1 style={{color:'black'}}>Изменить курс</h1>

)}

{!edit&&(

<h1>Создать курс</h1>

)}

<button className="modal-close" onClick={handleClose} style={{marginTop:'0.8rem'}}>

&times;

</button>

</div>

<div className="modal-body">

<div className="continput">

<p>Название курса:</p>

<input type="text" placeholder='Название' className='inputlog' autoComplete='false'

value={titleClass} onChange={(e)=>setTitleClass(e.target.value)}/>

</div>

<div className="continput">

<p>Раздел:</p>

<input type="text" placeholder='Раздел' className='inputlog' autoComplete='false'

value={chapter} onChange={(e)=>setChapter(e.target.value)}/>

</div>

<div className="continput">

<p>Предмет:</p>

<input type="text" placeholder='Предмет' className='inputlog' autoComplete='false'

value={object} onChange={(e)=>setObject(e.target.value)}/>

</div>

<div className="continput">

<p>Аудитория:</p>

<input type="text" placeholder='Аудитория' className='inputlog' autoComplete='false'

value={audience} onChange={(e)=>setAudience(e.target.value)}/>

</div>

<div className="continput">

<p>Цвет темы:</p>

<div className="colors">

<div style={{background:colorHex}} className="colorBtn" onClick={()=>setColorPicker(!colorPicker)}>

<BiPlus style={{marginLeft:'2.5px', marginTop:'2.5px'}}/>

</div>

{colors.map((item) => (

<div key={item.code} style={{background:item.code}} className="color"

onClick={()=>setColorz(item.code)}>

{colorz?.code===item.code&&(

<BsCheck style={{marginLeft:'2px', marginTop:'2.5px'}}/>

)}

{colorz===item.code&&(

<BsCheck style={{marginLeft:'2px', marginTop:'2.5px'}}/>

)}

</div>

))}

</div>

{colorPicker&&(

<div style={{marginTop:'10px'}}>

<ChromePicker color={colorHex} onChange={handleColorChange} />

</div>

)}

</div>

<div className="btn-cnt" >

<button className='cancel' onClick={handleClose}>Отменить</button>

<button className='save' onClick={handleCreateClass}>Сохранить</button>

</div>

</div>

</div>

</div>

</div>

);

};

export default observer(Modal);

* + 1. components\ModalCreateGroup.tsx

import React, { useRef, useEffect, useState, useContext } from 'react';

import '../style/Modal.scss';

import { Context } from '..';

import { observer } from 'mobx-react-lite';

type ModalProps = {

show: boolean;

onClose: () => void;

titleGroup: string;

idGroup:string;

};

const Modal: React.FC<ModalProps> = ({ show, onClose, titleGroup, idGroup }) => {

const {store} = useContext(Context)

const modalRef = useRef<HTMLDivElement>(null);

const [title, setTitle] = useState<string>()

const [id, setId] = useState<string>()

useEffect(() => {

setTitle(titleGroup)

setId(idGroup)

const handleOutsideClick = (event: MouseEvent) => {

if (modalRef.current && !modalRef.current.contains(event.target as Node)) {

onClose();

}

};

if (show) {

document.addEventListener('mousedown', handleOutsideClick);

}

return () => {

document.removeEventListener('mousedown', handleOutsideClick);

};

// eslint-disable-next-line react-hooks/exhaustive-deps

}, []);

const handleClose = () => {

onClose();

};

const handleCreateClass=async()=>{

if(id){

if(title)

store.editGroupe(title, id).then((data)=>{

if(data)

window.location.reload()

})

}else{

if(title)

store.createGroupe(title, store.user.id).then((data)=>{

if(data)

window.location.reload()

})

}

}

return (

<div className='group-modal'>

<div className="modal-backdrop">

<div className="modal" ref={modalRef} style={{height:'300px'}}>

<div className="modal-header">

<div style={{width:'82%'}}>

{id&&(

<h1 style={{width:'100%'}}>Изменить группу</h1>

)}

{!id&&(

<h1 style={{width:'100%'}}>Создать группу</h1>

)}

</div>

<button className="modal-close" onClick={handleClose} style={{marginTop:'2.4rem'}}>

&times;

</button>

</div>

<div className="modal-body">

<div className="continput">

<p>Название группы:</p>

<input type="text" placeholder='Название' className='inputlog' autoComplete='false'

value={title} onChange={(e)=>setTitle(e.target.value)}/>

</div>

</div>

<div className="button-cont" style={{marginLeft:'23rem'}}>

<button style={{background:'#9f9da3'}} onClick={handleClose}>Отмена</button>

<button style={{background:'#335DFF'}} onClick={handleCreateClass}>Сохранить</button>

</div>

</div>

</div>

</div>

);

};

export default observer(Modal);

* + 1. components\ModalCreateTask.tsx

/\* eslint-disable array-callback-return \*/

import React, { useRef, useEffect, useState, useContext } from 'react';

import '../style/Modal.scss';

import { Context } from '..';

import { observer } from 'mobx-react-lite';

import ReactQuill from 'react-quill';

import 'react-quill/dist/quill.snow.css';

import DatePicker from 'react-datepicker';

import 'react-datepicker/dist/react-datepicker.css';

import { AiOutlineFileText, AiOutlinePlus } from 'react-icons/ai';

import { RxCross2 } from 'react-icons/rx';

import { PropagateLoader } from 'react-spinners';

type ModalProps = {

show: boolean;

onClose: () => void;

course\_id:any;

task:any;

};

const Modal: React.FC<ModalProps> = ({ show, onClose, course\_id, task }) => {

const {store} = useContext(Context)

const [loading, setLoading] = useState<boolean>(true)

const modalRef = useRef<HTMLDivElement>(null);

const [edit, setEdit] = useState<boolean>(false);

const [title, setTitle] = useState<string>()

const [point, setPoint] = useState<string>()

const [content, setContent] = useState('');

const fileInputRef = useRef<HTMLInputElement>(null);

const [selectedFile, setSelectedFile] = useState<any>([]);

const [oldFiles, setOldFiles] = useState<any>([]);

const [isForm, setIsForm] = useState<boolean>(false)

const [id, setId] = useState<string>('')

const handleFileSelected = (event: React.ChangeEvent<HTMLInputElement>) => {

const selectedFiles = event.target.files as FileList;

setSelectedFile((prev: File[]) => {

for (let i = 0; i < selectedFiles?.length; i++) {

prev.push(selectedFiles[i])

}

return [...prev]; // Если реагировать не будет, то return [ ...prev ]

})

};

function removeItemByName(name: string) {

const filteredFiles = oldFiles.filter((fileName: any) => fileName.name !== name);

setOldFiles(filteredFiles)

setSelectedFile((prevItems: any[]) => prevItems.filter(item => item.name !== name));

}

const modules = {

toolbar: [

[{ 'header': [1, 2, false] }],

['bold', 'italic', 'underline', 'strike', 'code'],

[{ 'color': [] }, { 'background': [] }],

[{ 'font': [] }],

[{ 'align': [] }],

[{ 'list': 'ordered' }, { 'list': 'bullet' }],

['link', 'image'],

['clean']

]

};

useEffect(() => {

if(task.id){

const k: any[] =[]

store.viewFiles(task.files).then((data)=>{

// eslint-disable-next-line array-callback-return

data?.data.map((dat:any)=>{

k.push(dat[0])

})

setOldFiles(k)

setLoading(false)

setEdit(true)

setTitle(task.title)

setContent(task.description)

setPoint(task.point)

setSelectedFile(task.files)

setIsForm(task.isForm)

setSelectedFile(k)

setId(task.id)

})

if(task.deadlineDatetime!=='1970-01-01T00:00:00.000Z'){

const dateDeadline = new Date(task.deadlineDatetime)

setDate(dateDeadline)

}

}

else{

setLoading(false)

}

const handleOutsideClick = (event: MouseEvent) => {

if (modalRef.current && !modalRef.current.contains(event.target as Node)) {

onClose();

}

};

if (show) {

document.addEventListener('mousedown', handleOutsideClick);

}

return () => {

document.removeEventListener('mousedown', handleOutsideClick);

};

// eslint-disable-next-line react-hooks/exhaustive-deps

}, []);

const formats = [

'header',

'bold', 'italic', 'underline', 'strike', 'code',

'color', 'background', 'font', 'align',

'list', 'bullet',

'link', 'image'

];

function handleChange(value: string) {

setContent(value);

}

const handleClose = () => {

onClose();

};

const [date, setDate] = useState<Date | null>(null);

const handleChangeDate = (date: Date | null) => {

setDate(date);

};

function handleCheckboxChange(event: React.ChangeEvent<HTMLInputElement>) {

setIsForm(event.target.checked);

}

const handleOnClick = () =>{

if(id){

store.updateTask(id, oldFiles, selectedFile, String(date), String(title), content, String(point), String(isForm)).then((data)=>{

if(data)

window.location.reload()

})

}else{

store.addTask(selectedFile, String(date), String(title), content, String(point), course\_id, String(isForm), store.user.id).then((data)=>{

if(data)

window.location.reload()

})

}

}

return (

<div className='task-modal'>

<div className="modal-backdrop">

<div className="modal" ref={modalRef} style={{position:'static'}}>

<div className="modal-header">

{edit&&(

<h1 style={{color:'black'}}>Изменить задание</h1>

)}

{!edit&&(

<h1>Создать задание</h1>

)}

<button className="modal-close" onClick={handleClose} style={{marginTop:'0.5rem', marginLeft:'-60px'}}>

&times;

</button>

</div>

<div className="modal-body" style={{ display: 'flex', flexDirection: 'column', marginTop: '-20px', overflow: 'auto' }}>

{loading ? (

<div style={{ width: '100%', paddingTop: '10rem', paddingBottom: '20rem' }}>

<div style={{ display: 'flex', width: '100%' }}>

<div style={{ margin: '0 auto', marginTop: '5rem', textAlign: 'center' }}>

<PropagateLoader color="#335DFF" style={{ marginLeft: '-15px' }} />

</div>

</div>

</div>

) : (

<>

<div className="continput">

<p>Название:</p>

<input type="text" placeholder='Название' className='inputlog' autoComplete='false'

value={title} onChange={(e)=>setTitle(e.target.value)}/>

</div>

<div className="continput">

<p>Инструкции:</p>

<ReactQuill

value={content}

onChange={handleChange}

modules={modules}

formats={formats}

style={{width:'100%'}}/>

</div>

<div className="continput">

<p>Добавить форму для сдачи:</p>

<input type="checkbox" checked={isForm} onChange={handleCheckboxChange}

style={{width:'20px', height:'20px', marginTop:'-15px'}}/>

</div>

<div className="continput">

<p>Баллы:</p>

<input type="number" placeholder='Баллы' className='inputlog' autoComplete='false'

value={point} onChange={(e)=>setPoint(e.target.value)}/>

</div>

<div className="continput">

<p>Дата сдачи:</p>

<DatePicker

id="dateTimePicker"

className="inputlog"

selected={date}

onChange={handleChangeDate}

showTimeSelect

dateFormat="Pp"

/>

</div>

<div className="continput">

<p>Дополнительные файлы: </p>

<div className="add-photo" style={{width:'100%', height:'60px'}}>

<div className="file-input" style={{marginTop:'10px'}}>

<label htmlFor="file-input" style={{width:'30px', height:'30px'}}>

<AiOutlinePlus style={{fontSize:'18px', paddingBottom:'2px'}}/>

</label>

<input

ref={fileInputRef}

id="file-input"

type="file"

onChange={handleFileSelected}

accept='\*'

multiple

/>

</div>

</div>

{selectedFile&&( //проверка на то пуст ли файл

<div style={{width:'100%'}}>

{selectedFile.map((t:any)=>(

<div className="body-file"

style={{display:'flex',

border:'2px solid #AEB5C8',

marginTop:'10px',

width:'100%',

borderRadius:'5px',

background:'#AEB5C8',

color:'white'}}>

<AiOutlineFileText className='file' style={{fontSize:'20px',

marginTop:'16px', marginRight:'10px'}}/>

<p style={{width:'90%'}}>{t.name}</p>

<RxCross2 className='cross'

style={{marginTop:'20px'}}

onClick={()=>removeItemByName(t.name)}/>

</div>

))}

</div>

)}

</div>

<div className="btn-cnt" >

<button className='cancel' onClick={handleClose}>Отменить</button>

<button className='save' onClick={handleOnClick}>Сохранить</button>

</div>

</>

)}

</div>

</div>

</div>

</div>

);

};

export default observer(Modal);

* + 1. components\ModalFileComplementary.tsx

import React, { useRef, useEffect, useState, useContext } from 'react';

import '../style/Modal.scss';

import {IoIosClose} from 'react-icons/io'

import { Context } from '..';

import { FileUploader } from "react-drag-drop-files";

type ModalProps = {

show: boolean;

onClose: () => void;

classid:any;

};

const Modal: React.FC<ModalProps> = ({ show, onClose, classid }) => {

const {store} = useContext(Context)

const [selectedFile, setSelectedFile] = useState<any>()

const modalRef = useRef<HTMLDivElement>(null);

const handleChange = (file: React.SetStateAction<null>) => {

if (file) {

setSelectedFile(file);

}

console.log(file)

};

useEffect(() => {

const handleOutsideClick = (event: MouseEvent) => {

if (modalRef.current && !modalRef.current.contains(event.target as Node)) {

onClose();

}

};

if (show) {

document.addEventListener('mousedown', handleOutsideClick);

}

return () => {

document.removeEventListener('mousedown', handleOutsideClick);

};

}, [show, onClose]);

const handleClose = () => {

onClose();

};

const handleAddAvatar = () =>{

if(classid){

store.editCover(classid, selectedFile).then((data)=>{

if(data)

window.location.reload()

})

}

else{

store.editAva(store.user.email, selectedFile).then((data)=>{

if(data)

window.location.reload()

})

}

}

return (

<div className='modalimage'>

{show && (

<div className="modal-backdrop">

<div className="modal" ref={modalRef} style={{position:'relative', height:'25rem'}}>

<div className="modal-header">

<h1 style={{color:'black'}}>Добавление файла</h1>

<button className="modal-close" onClick={handleClose}>

<IoIosClose/>

</button>

</div>

<div className="modal-body" style={{margin:'0 auto'}}>

<div className="file-upld" style={{margin:'0 auto', marginBottom:'60px'}}>

<FileUploader handleChange={handleChange} name="file" multiple={true}/>

</div>

</div>

<div className="btn-cnt" style={{position:'absolute', right:0}}>

<button className='cancel' onClick={handleClose}>Отменить</button>

</div>

</div>

</div>

)}

</div>

);

};

export default Modal;

* + 1. components\ModalImage.tsx

import React, { useRef, useEffect, useState, useContext } from 'react';

import '../style/Modal.scss';

import {IoIosClose} from 'react-icons/io'

import { Context } from '..';

import { FileUploader } from "react-drag-drop-files";

type ModalProps = {

show: boolean;

onClose: () => void;

classid:any;

};

const fileTypes = ["JPG", "PNG", "GIF"];

const Modal: React.FC<ModalProps> = ({ show, onClose, classid }) => {

const {store} = useContext(Context)

const [selectedFile, setSelectedFile] = useState<any>()

const modalRef = useRef<HTMLDivElement>(null);

const handleChange = (file: React.SetStateAction<null>) => {

if (file) {

setSelectedFile(file);

}

};

useEffect(() => {

const handleOutsideClick = (event: MouseEvent) => {

if (modalRef.current && !modalRef.current.contains(event.target as Node)) {

onClose();

}

};

if (show) {

document.addEventListener('mousedown', handleOutsideClick);

}

return () => {

document.removeEventListener('mousedown', handleOutsideClick);

};

}, [show, onClose]);

const handleClose = () => {

onClose();

};

const handleAddAvatar = () =>{

if(classid){

store.editCover(classid, selectedFile).then((data)=>{

if(data)

window.location.reload()

})

}

else{

store.editAva(store.user.email, selectedFile).then((data)=>{

if(data)

window.location.reload()

})

}

}

return (

<div className='modalimage'>

{show && (

<div className="modal-backdrop">

<div className="modal" ref={modalRef} style={{position:'relative', height:'25rem'}}>

<div className="modal-header">

<h1 style={{color:'black'}}>Изменение фотографии</h1>

<button className="modal-close" onClick={handleClose}>

<IoIosClose/>

</button>

</div>

<div className="modal-body" style={{margin:'0 auto'}}>

<div className="file-upld" style={{margin:'0 auto', marginBottom:'60px'}}>

<FileUploader handleChange={handleChange} name="file" types={fileTypes} />

</div>

</div>

<div className="btn-cnt" style={{position:'absolute', right:0}}>

<button className='cancel' onClick={handleClose}>Отменить</button>

<button className='save' onClick={handleAddAvatar}>Сохранить</button>

</div>

</div>

</div>

)}

</div>

);

};

export default Modal;

* + 1. components\ModalTheme.tsx

import React, { useRef, useEffect, useState, useContext } from 'react';

import '../style/Modal.scss';

import { Context } from '..';

import { observer } from 'mobx-react-lite';

type ModalProps = {

show: boolean;

onClose: () => void;

theme:any;

course:any;

};

const Modal: React.FC<ModalProps> = ({ show, onClose, theme, course}) => {

const {store} = useContext(Context)

const modalRef = useRef<HTMLDivElement>(null);

const [title, setTitle] = useState<string>()

useEffect(() => {

setTitle(theme.title)

const handleOutsideClick = (event: MouseEvent) => {

if (modalRef.current && !modalRef.current.contains(event.target as Node)) {

onClose();

}

};

if (show) {

document.addEventListener('mousedown', handleOutsideClick);

}

return () => {

document.removeEventListener('mousedown', handleOutsideClick);

};

// eslint-disable-next-line react-hooks/exhaustive-deps

}, []);

const handleClose = () => {

onClose();

};

const handleCreateTheme=async()=>{

if(!theme){

if(title)

store.createTheme(title, course).then((data)=>{

if(data)

window.location.reload()

})

}else{

if(title)

store.updateTheme(theme.id, title).then((data)=>{

if(data)

window.location.reload()

})

}

}

return (

<div className='group-modal'>

<div className="modal-backdrop">

<div className="modal" ref={modalRef} style={{height:'300px', color:'black'}}>

<div className="modal-header">

{theme&&(

<>

<h1 style={{width:'38%'}}>Изменить тему</h1>

<button className="modal-close" onClick={handleClose}

style={{marginTop:'2rem', marginLeft:'-30px'}}>

&times;

</button>

</>

)}

{!theme&&(

<>

<h1>Создать тему</h1>

<button className="modal-close" onClick={handleClose} style={{marginTop:'0.8rem'}}>

&times;

</button>

</>

)}

</div>

<div className="modal-body">

<div className="continput">

<p>Название темы:</p>

<input type="text" placeholder='Название' className='inputlog' autoComplete='false'

value={title} onChange={(e)=>setTitle(e.target.value)}/>

</div>

</div>

<div className="button-cont" style={{marginLeft:'23rem'}}>

<button style={{background:'#9f9da3'}} onClick={handleClose}>Отмена</button>

<button style={{background:'#335DFF'}} onClick={handleCreateTheme}>Сохранить</button>

</div>

</div>

</div>

</div>

);

};

export default observer(Modal);

* + 1. components\ModalUsersGroups.tsx

import React, { useRef, useEffect, useState, useContext } from 'react';

import '../style/Modal.scss';

import { Context } from '..';

import { observer } from 'mobx-react-lite';

import { BsPlus } from 'react-icons/bs';

import { PropagateLoader } from 'react-spinners';

import ModalAddUserToGroup from './ModalAddUserToGroup';

import { FreeUsersGroupMap } from './FreeUsersGroupMap';

type ModalProps = {

show: boolean;

onClose: () => void;

id:string;

};

const Modal: React.FC<ModalProps> = ({ show, onClose, id }) => {

const [openModalGroup, setOpenModalGroup] = useState(false)

const {store} = useContext(Context)

const modalRef = useRef<HTMLDivElement>(null);

const [loading, setLoading] = useState<boolean>(true)

const [users, setUsers] = useState<any>()

useEffect(() => {

store.viewUsersInGroup(id).then((data)=>{

if(data?.data){

setUsers(data.data)

setLoading(false)

}

})

const handleOutsideClick = (event: MouseEvent) => {

if (modalRef.current && !modalRef.current.contains(event.target as Node)) {

onClose();

}

};

if (show) {

document.addEventListener('mousedown', handleOutsideClick);

}

return () => {

document.removeEventListener('mousedown', handleOutsideClick);

};

// eslint-disable-next-line react-hooks/exhaustive-deps

}, []);

const handleCloseModalGroup = () => {

setOpenModalGroup(false);

}

const handleClose = () => {

onClose();

};

const handleRemoveUser = () => {

store.viewUsersInGroup(id).then((user)=>{

if(user?.data)

setUsers(user?.data)

})

};

return (

<div className='group-modal'>

<div className="modal-backdrop">

<div className="modal" ref={modalRef} style={{height:'470px'}}>

<div className="modal-header" style={{marginTop:'-30px', marginLeft:'10px'}}>

<h1>Участники</h1>

<button className="modal-close" onClick={handleClose} style={{marginTop:'2.3rem'}}>

&times;

</button>

</div>

<div className="modal-body" style={{overflowX:'hidden'}}>

<div className="model-body-cont" style={{height:'18rem', marginLeft:'10px', width:'100%', display:'flex'}}>

<div>

<BsPlus style={{paddingLeft:'3rem', fontSize:'30px', color:'#335DFF'}}

className='btn-add-user'

onClick={()=>setOpenModalGroup(true)}/>

</div>

{loading&&(

<div style={{margin:'0 auto', marginTop:'8rem'}}>

<PropagateLoader

color="#335DFF"

style={{marginLeft:'-60px'}}

className='loader'

/>

</div>

)}

{!loading&&(

<div style={{marginTop:'20px'}}>

{!users&&(

<div style={{margin:'0 auto', textAlign:'center'}}>

<h1 >Ни одного пользователя в группе</h1>

<img style={{width:'15rem'}}

src="https://cdn.dribbble.com/userupload/5389624/file/original-f271ff07d00d9b9e526c261921927cae.jpg?compress=1&resize=1024x1050"

alt="not found" />

</div>

)}

{users&&(

<div style={{marginLeft:'-75px'}}>

<FreeUsersGroupMap users={users} group={id} onRemoveUser={handleRemoveUser} cross={true}/>

</div>

)}

</div>

)}

</div>

{openModalGroup&&(

<ModalAddUserToGroup show={openModalGroup} onClose={handleCloseModalGroup} id={id}/>

)}

</div>

<div className="button-cont">

<button style={{background:'#9f9da3', marginLeft:'8rem'}} onClick={handleClose}>Отмена</button>

</div>

</div>

</div>

</div>

);

};

export default observer(Modal);

* + 1. components\Sidebar.tsx

import React, { useState, useEffect, useRef, useContext } from "react";

import "../style/Sidebar.scss";

import { AiFillHome } from "react-icons/ai";

import { BsArchiveFill } from "react-icons/bs";

import { useNavigate } from "react-router-dom";

import { Context } from "..";

import { PropagateLoader } from "react-spinners";

import { SmallCoursesMap } from "./SmallCoursesMap";

interface SidebarProps {

isOpen: boolean;

onClose: () => void;

}

const Sidebar: React.FC<SidebarProps> = ({ isOpen, onClose }) => {

const {store} = useContext(Context)

const navigation = useNavigate()

const [isClosing, setIsClosing] = useState(false);

const [coursesStudent, setCoursesStudent] = useState<any>()

const [coursesTeacher, setCoursesTeacher] = useState<any>()

const [loading, setLoading] = useState(true);

const sidebarRef = useRef<HTMLDivElement>(null);

useEffect(() => {

store.checkAuth().then(()=>{

store.getClassStudent(store.user.id).then((studentData)=>{

store.getClassTeacher(store.user.id).then((teacherData)=>{

setCoursesStudent(studentData?.data)

setCoursesTeacher(teacherData?.data)

setLoading(false)

})

})

})

}, [store])

const handleClose = () => {

setIsClosing(true);

setTimeout(() => {

onClose();

setIsClosing(false);

}, 300);

};

const handleContentClick = (event: React.MouseEvent<HTMLDivElement, MouseEvent>) => {

event.stopPropagation();

};

useEffect(() => {

const handleClickOutside = (event: MouseEvent) => {

if (sidebarRef.current && !sidebarRef.current.contains(event.target as Node)) {

handleClose();

}

};

document.addEventListener("mousedown", handleClickOutside);

return () => {

document.removeEventListener("mousedown", handleClickOutside);

};

// eslint-disable-next-line react-hooks/exhaustive-deps

}, []);

return (

<div className={`sidebar ${isOpen ? "open" : ""} ${isClosing ? "closing" : ""}`} ref={sidebarRef}>

<div className="sidebar-content" onClick={handleContentClick}>

<div className="home-cont" onClick={()=>navigation('/')}>

<AiFillHome style={{paddingTop:'18px', paddingRight:'10px', paddingLeft:'10px'}}/>

<p>Курсы</p>

</div>

{loading?(

<div style={{ display: 'flex', justifyContent: 'center', alignItems: 'center', height: '100vh' }}>

<PropagateLoader color="#335DFF" />

</div>

):(

<div>

<p style={{textAlign:'left'}}>Курсы, слушателем я являюсь:</p>

{!coursesStudent.length?(

<div className="not-found">

<img src="https://cdn.dribbble.com/users/902546/screenshots/6233250/11\_copy.png" alt=""

style={{width:'20rem', marginTop:'5rem'}}/>

<p style={{color:'black'}}>Ни одного не найдено</p>

</div>

):(

<div>

<SmallCoursesMap courses={coursesStudent}/>

</div>

)}

{coursesTeacher.length&&(

<div>

<p style={{textAlign:'left'}}>Курсы, которые я преподаю:</p>

<SmallCoursesMap courses={coursesTeacher}/>

</div>

)}

</div>

)}

<div className="archive-cont">

<BsArchiveFill style={{paddingTop:'18px', paddingRight:'10px', paddingLeft:'10px'}}/>

<p>Архив</p>

</div>

</div>

<div className="sidebar-overlay" onClick={handleClose} />

</div>

);

};

export default Sidebar;

* + 1. components\SmallCourseCard.tsx

import React from 'react'

export const SmallCourseCard = (courses:any) => {

return (

<a href={`/c/${courses.courses.id}`} style={{textDecoration:'none'}}>

<div className="course-card-small">

<div className="course-card-small\_\_body">

<div className="course-card-small\_\_body\_\_avatar"

style={{background:courses.courses.decor}}>

<p style={{color:'white'}}>{courses.courses.title[0].toUpperCase()}</p>

</div>

<div className="course-card-small\_\_body\_\_info">

<h1>{courses.courses.title}</h1>

<p style={{textAlign:'left', marginLeft:'15px'}}>{courses.courses.chapter}</p>

</div>

</div>

</div>

</a>

)

}

* + 1. components\SmallCoursesMap.tsx

import React from 'react'

import { SmallCourseCard } from './SmallCourseCard'

export const SmallCoursesMap = (courses:any) => {

return (

courses.courses.map((course:any)=>(

<SmallCourseCard courses={course}/>

))

)

}

* + 1. components\StudentTaskIsForm.tsx

import { observer } from 'mobx-react-lite'

import React, { useContext, useEffect, useState } from 'react'

import { AiOutlineFileText } from 'react-icons/ai'

import { Context } from '..'

import { useParams } from 'react-router-dom'

import { PropagateLoader } from 'react-spinners'

import { Prism as SyntaxHighlighter } from 'react-syntax-highlighter';

import { vscDarkPlus } from 'react-syntax-highlighter/dist/esm/styles/prism';

import { decode } from 'html-entities';

import { API\_URL } from '../http'

import checkFileAvailability from '../middleware/image'

import { BiSend } from 'react-icons/bi'

import { BsFillFileTextFill, BsPlus } from 'react-icons/bs'

import hexToRgb from '../middleware/colorToRGBA'

import ModalFileComplementary from './ModalFileComplementary'

const StudentTaskIsForm = () => {

const [openModalFile, setOpenModalFile] = useState(false)

const {store} = useContext(Context)

const id = useParams()

const [isHovered, setIsHovered] = useState(false);

const [course, setCourse] = useState<any>()

const [task, setTask] = useState<any>()

const [loading, setLoading] = useState<boolean>(true)

const [checkFile, setCheckFile] = useState<boolean>(false)

const [complementary, setComplementary] = useState<any>()

const [dateNow, setDateNow] = useState<any>()

useEffect(() => {

if(id.id)

store.viewTaskOne(id.id, store.user.id).then((data)=>{

if(data?.data.role.name==='student'){

if(id.id)

store.viewComplementaryOne(id.id, store.user.id).then((compData)=>{

setComplementary(compData?.data)

setCourse(data?.data.class)

setTask(data?.data.class.task[0])

handleCheckImage()

const currentDate = new Date(Date.now());

setDateNow(currentDate.toISOString())

setLoading(false)

})

}else{

setCourse(data?.data.class)

setTask(data?.data.class.task[0])

handleCheckImage()

setLoading(false)

}

})

// eslint-disable-next-line react-hooks/exhaustive-deps

}, [id.id, store])

const handleCheckImage = async() =>{

const fileUrl = API\_URL + 'images/'+store.user.file;

const isFileAvailable = await checkFileAvailability(fileUrl);

setCheckFile(isFileAvailable);

}

const currentDate = new Date();

let myDate = new Date()

if(task){

myDate = new Date(task.date);

}

function formatTime(date: Date) {

const hours = date.getHours().toString().padStart(2, '0');

const minutes = date.getMinutes().toString().padStart(2, '0');

return `${hours}:${minutes}`;

}

function formatDate(date: Date) {

const options: Intl.DateTimeFormatOptions = {

day: 'numeric',

month: 'short'

};

return date.toLocaleDateString('ru-RU', options);

}

let displayText: string;

if (

myDate.getDate() === currentDate.getDate() &&

myDate.getMonth() === currentDate.getMonth() &&

myDate.getFullYear() === currentDate.getFullYear()

) {

displayText = formatTime(myDate);

} else {

displayText = formatDate(myDate);

}

const currentDateDeadline = new Date();

let myDateDeadline = new Date()

if(task){

myDateDeadline = new Date(task.deadlineDatetime);

}

function formatTimeDeadline(date: Date) {

const hours = date.getHours().toString().padStart(2, '0');

const minutes = date.getMinutes().toString().padStart(2, '0');

return `${hours}:${minutes}`;

}

function formatDateDeadline(date: Date) {

const options: Intl.DateTimeFormatOptions = {

day: 'numeric',

month: 'short'

};

return date.toLocaleDateString('ru-RU', options);

}

let displayTextDeadline: string;

if (

myDateDeadline.getDate() === currentDateDeadline.getDate() &&

myDateDeadline.getMonth() === currentDateDeadline.getMonth() &&

myDateDeadline.getFullYear() === currentDateDeadline.getFullYear()

) {

displayTextDeadline = formatTimeDeadline(myDateDeadline);

} else {

displayTextDeadline = formatDateDeadline(myDateDeadline);

}

const removeCodeTags = (htmlString: string) => {

return htmlString.replace(/<code>.\*?<\/code>/gs, '');

};

if(loading){

return(

<div style={{width:'100%', paddingTop:'10rem', paddingBottom:'20rem'}}>

<div style={{display:'flex', width:'100%'}}>

<div style={{margin:'0 auto', marginTop:'5rem', textAlign:'center'}}>

<PropagateLoader color="#335DFF" style={{marginLeft:'-15px'}}/>

</div>

</div>

</div>

)

}

let modifiedHtmlString = ''

if(task){

modifiedHtmlString = removeCodeTags(task.description);

}

const extractCode = (htmlString: string) => {

const regex = /<code>(.\*?)<\/code>/s;

const match = htmlString.match(regex);

return match ? match[1] : '';

};

let codeString = ''

if(task){

codeString = decode(extractCode(task.description)); // Decode the HTML entities

}

const buttonStyle = {

backgroundColor: isHovered ? hexToRgb(course.decor, 0.2) : 'transparent',

border:`1px solid ${isHovered? course.decor : '#E2E4EB'}`,

color:course.decor

};

const handleEditStatus = () =>{

store.editComplementaryStatus(complementary.id, complementary.status).then((data)=>{

if(data)

window.location.reload()

})

}

const handleCloseModalFile = () => {

setOpenModalFile(false);

}

return (

<div className="task-body">

<div className="task-body\_\_body">

<div className="task-body\_\_body\_\_subleft">

<div className="task-body\_\_body\_\_subleft\_\_circle" style={{background:course.decor}}>

<AiOutlineFileText style={{marginTop:'13px', marginLeft:'2px'}}/>

</div>

</div>

<div className="task-body\_\_body\_\_left">

<div className="task-body\_\_body\_\_left\_\_header">

<h1 style={{color:course.decor}}>{task.title}</h1>

</div>

<div className="task-body\_\_body\_\_left\_\_subheader">

<p>{task.member.user.surname} {task.member.user.name} {task.member.user.lastname}</p>

<p>•</p>

<p>{displayText}</p>

</div>

<div className="task-body\_\_body\_\_left\_\_point"

style={{borderBottom:`2px solid ${course.decor}`}}>

{task.point!==0?(

<p style={{width:'85%'}}>{task.point} баллов</p>

):(

<div style={{width:'85%'}}></div>

)}

{task.deadlineDatetime&&(

<p style={{whiteSpace:'nowrap'}}>Срок сдачи: {displayTextDeadline}</p>

)}

</div>

{task.description&&(

<div className="task-body\_\_body\_\_left\_\_decription"

style={{borderBottom:`1px solid ${course.decor}`, paddingBottom:'10px'}}>

<div dangerouslySetInnerHTML={{ \_\_html: modifiedHtmlString }} />

{codeString&&(

<SyntaxHighlighter language="html" style={vscDarkPlus}>

{codeString}

</SyntaxHighlighter>

)}

<div className="task-body\_\_body\_\_left\_\_description\_\_files">

{task.files.map((file:any)=>(

<div className='task-body\_\_body\_\_left\_\_description\_\_files\_\_file'>

<BsFillFileTextFill style={{paddingTop:'12px'}}/>

<a href={`${API\_URL}files/${file[0].id}`}

style={{padding:'10px'}}>{file[0].name}</a>

</div>

))}

</div>

</div>

)}

<p style={{fontWeight:'600'}}>Комментарии</p>

<div className="task-body\_\_body\_\_left\_\_comment-input">

<div className="task-body\_\_body\_\_left\_\_comment-input\_\_image-cont"

style={{background:store.user.colorProfile}}>

{checkFile?(

<img src={API\_URL + 'images/'+store.user.file}

style={{objectFit:'cover', width:'40px'}}alt="" />

):(

<div>

<p style={{marginTop:'8px'}}>{store.user.email[0].toUpperCase()}</p>

</div>

)}

</div>

<input type="text"

placeholder='Добавить комментарий'

style={{border:`1px solid ${course.decor}`}}/>

<BiSend style={{color:course.decor, fontSize:'28px', marginTop:'5px'}} />

</div>

</div>

{complementary&&(

<div className="task-body\_\_body\_\_right">

<div className="task-body\_\_body\_\_right\_\_body">

{!Boolean(complementary.status)&&task.deadlineDatetime<dateNow &&(

<div className="task-body\_\_body\_\_right\_\_header">

<h1>Мои задания</h1>

<p style={{color:'#e63946',

paddingRight:'10px',

marginTop:'-20px',

fontWeight:'600'}}>Пропущен срок сдачи</p>

</div>

)}

{!Boolean(complementary.status)&&task.deadlineDatetime>dateNow&&(

<div className="task-body\_\_body\_\_right\_\_header" style={{display:'flex'}}>

<h1>Мои задания</h1>

<p>Не сдано</p>

</div>

)}

{Boolean(complementary.status)&&task.deadlineDatetime<dateNow&&(

<div className="task-body\_\_body\_\_right\_\_header">

<h1>Мои задания</h1>

<p style={{color:'#9A9A9A',

paddingRight:'10px',

paddingLeft:'27px',

marginTop:'-20px',

fontWeight:'600'}}>Сдано с опозданием</p>

</div>

)}

{Boolean(complementary.status)&&task.deadlineDatetime>dateNow&&(

<div className="task-body\_\_body\_\_right\_\_header" style={{display:'flex'}}>

<h1>Мои задания</h1>

<p style={{color:'#9A9A9A'}}>Сдано</p>

</div>

)}

{!Boolean(complementary.status)&&complementary.file.length===0&&(

<>

<div style={{display:'flex', margin:'0 auto'}}>

<div className='task-body\_\_body\_\_right\_\_add-file'

style={buttonStyle}

onMouseEnter={() => setIsHovered(true)}

onMouseLeave={() => setIsHovered(false)} onClick={()=>setOpenModalFile(true)}>

<BsPlus style={{marginTop:'15px', fontSize:'24px',

fontWeight:'700'}}/>

<p>Добавить файл</p>

</div>

</div>

<div style={{background:course.decor}}

className='task-body\_\_body\_\_right\_\_pass'

onClick={handleEditStatus}>Отметить как выполненное</div>

</>

)}

{Boolean(complementary.status)&&complementary.file.length===0&&(

<>

<div className='task-body\_\_body\_\_right\_\_not-file'>

Задание не прикреплено

</div>

<div className='task-body\_\_body\_\_right\_\_cancel-btn'

style={buttonStyle}

onMouseEnter={() => setIsHovered(true)}

onMouseLeave={() => setIsHovered(false)}

onClick={handleEditStatus}>

Открепить решение

</div>

</>

)}

</div>

</div>

)}

{openModalFile&&(

<ModalFileComplementary show={openModalFile} onClose={handleCloseModalFile} classid={undefined}/>

)}

</div>

</div>

)

}

export default observer(StudentTaskIsForm)

* + 1. components\TaskCard.tsx

import { observer } from 'mobx-react-lite'

import React, {useContext, useState} from 'react'

import { BsFileEarmarkTextFill, BsThreeDotsVertical } from 'react-icons/bs'

import ModalCreateTask from './ModalCreateTask';

import { Context } from '..';

import ModalAddToTheme from './ModalAddToTheme';

import { useNavigate } from 'react-router-dom';

const TaskCard = (task:any) => {

const {store} = useContext(Context)

const currentDate = new Date();

const myDate = new Date(task.task.date);

const [openModalTask, setOpenModalTask] = useState(false)

const [openModalTheme, setOpenModalTheme] = useState(false)

const navigation = useNavigate()

function formatTime(date: Date) {

const hours = date.getHours().toString().padStart(2, '0');

const minutes = date.getMinutes().toString().padStart(2, '0');

return `${hours}:${minutes}`;

}

function formatDate(date: Date) {

const options: Intl.DateTimeFormatOptions = {

day: 'numeric',

month: 'short'

};

return date.toLocaleDateString('ru-RU', options);

}

let displayText: string;

if (

myDate.getDate() === currentDate.getDate() &&

myDate.getMonth() === currentDate.getMonth() &&

myDate.getFullYear() === currentDate.getFullYear()

) {

displayText = formatTime(myDate);

} else {

displayText = formatDate(myDate);

}

const handleCloseModalTask = () => {

setOpenModalTask(false);

}

const handleCloseModalTheme = () => {

setOpenModalTheme(false);

}

const handleDeleteTask=()=>{

store.deleteTask(task.task.id).then(()=>{

window.location.reload()

})

}

const handleDeleteFromTheme=()=>{

store.deleteFromTheme(task.task.id).then((data)=>{

if(data)

window.location.reload()

})

}

return (

<div className="task-card" style={{border:`1px solid ${task.task.class.decor}`}}>

<div className="task-card\_\_body">

<div className="task-card\_\_body\_\_circle" style={{background:task.task.class.decor}}>

<BsFileEarmarkTextFill className="task-card\_\_body\_\_circle\_\_file"/>

</div>

<div className="task-card\_\_body\_\_info" onClick={()=>navigation(`/t/${task.task.id}`)}>

<h1>Пользователь {task.task.member.user.surname} {task.task.member.user.name} {task.task.member.user.lastname} добавил задание: {task.task.title}</h1>

<p>{displayText}</p>

</div>

{task.role&&(

<>

{task.role.name==="teacher"&&(

<div className="dropdown" style={{marginTop:'5px'}}>

<button className='dropdown\_\_button\_\_normal'>

<BsThreeDotsVertical style={{color:task.task.class.decor, marginTop:'18px'}}/>

</button>

<div className="dropdown\_\_content">

<div className="dropdownline" onClick={()=>setOpenModalTask(true)}>Изменить</div>

{!task.task.theme\_id&&(

<div className="dropdownline" onClick={()=>setOpenModalTheme(true)}>Добавить тему</div>

)}

{task.task.theme\_id&&(

<div className="dropdownline" onClick={handleDeleteFromTheme}>Удалить тему</div>

)}

<div className="dropdownline" onClick={handleDeleteTask}>Удалить</div>

</div>

{openModalTask&&(

<ModalCreateTask show={openModalTask} onClose={handleCloseModalTask} task={task.task} course\_id={task.task.class.id}/>

)}

{openModalTheme&&(

<ModalAddToTheme show={openModalTheme} onClose={handleCloseModalTheme} course={task.task.class.id} task={task.task}/>

)}

</div>

)}

</>

)}

</div>

</div>

)

}

export default observer(TaskCard)

* + 1. components\TaskMap.tsx

import React from 'react'

import TaskCard from './TaskCard'

export const TaskMap = (task:any) => {

return (

<>

{task.task.lenght!==0&&(

<>

{ task.task.map((tsk:any)=>(

<TaskCard task={tsk} role={task.role} />

))}

</>

)}

</>

)

}

* + 1. components\ThemeCard.tsx

import React, { useContext, useState } from 'react'

import { BsThreeDotsVertical } from 'react-icons/bs'

import { TaskMap } from './TaskMap'

import ModalTheme from './ModalTheme'

import { Context } from '..'

export const ThemeCard = (theme:any) => {

const {store} = useContext(Context)

const [openModalTheme, setOpenModalTheme] = useState(false)

const handleCloseThemeModal = () => {

setOpenModalTheme(false);

}

const handleDeleteTheme = () =>{

store.deleteTheme(theme.th.id).then((data)=>{

if(data)

window.location.reload()

})

}

return (

<div className="th-card">

<div className="theme-card" style={{color:theme.th.class.decor,

borderBottom:`2px solid ${theme.th.class.decor}`}}>

<div className="theme-card\_\_body">

<h1>{theme.th.title}</h1>

{theme.role&&(

<>

{theme.role.name==="teacher"&&(

<div className="dropdown">

<button className='dropdown\_\_button\_\_normal'>

<BsThreeDotsVertical style={{color:theme.th.class.decor, marginTop:'2.3rem'}}/>

</button>

<div className="dropdown\_\_content">

<div className="dropdownline" onClick={()=>setOpenModalTheme(true)}>Изменить</div>

<div className="dropdownline" onClick={handleDeleteTheme}>Удалить</div>

</div>

{openModalTheme&&(

<ModalTheme show={openModalTheme} onClose={handleCloseThemeModal} theme={theme.th} course=''/>

)}

</div>

)}

</>

)}

</div>

</div>

{theme.th.task.lenght!==0&&(

<div className="body-task-theme">

<TaskMap task={theme.th.task}/>

</div>

)}

</div>

)

}

* + 1. components\ThemeMap.tsx

import React from 'react'

import { ThemeCard } from './ThemeCard'

export const ThemeMap = (theme:any) => {

return (

theme.theme.map((th:any)=>(

<ThemeCard th={th} role={theme.role}/>

))

)

}

* + 1. components\UserCard.tsx

import { observer } from 'mobx-react-lite'

import React, {useState, useEffect, useContext} from 'react'

import { API\_URL } from '../http';

import checkFileAvailability from '../middleware/image';

import { RxCross2 } from 'react-icons/rx';

import { Context } from '..';

const UserCard = (userList:any) => {

const [checkFile, setCheckFile] = useState<any>()

const {store} = useContext(Context)

// eslint-disable-next-line react-hooks/exhaustive-deps

const handleCheckImage = async() =>{

const fileUrl = API\_URL + 'images/'+userList.userList.user.file;

const isFileAvailable = await checkFileAvailability(fileUrl);

setCheckFile(isFileAvailable);

}

useEffect(() => {

handleCheckImage()

}, [handleCheckImage])

const handleDelete = () =>{

store.deleteFromClass(userList.userList.user.id, userList.author.id)

.then((data: any) => {

if (data) {

// Если удаление успешно, обновить список пользователей без удаленного пользователя

// Получаем текущий список пользователей

const currentUsers = userList.author;

const updatedUsers: any[] = [];

for (let i = 0; i < currentUsers.length; i++) {

const user = currentUsers[i];

if (user.id !== userList.user.id) {

updatedUsers.push(user);

}

}

// Вызов функции userList.onRemoveUser() с обновленным списком пользователей

userList.onRemoveUser(updatedUsers);

}

});

}

return (

<div className="user-card" style={{border:`1px solid ${userList.author.decor}`}}>

<div className="user-card\_\_body">

<div className="img-icon"

style={{background:userList.userList.user.colorProfile,

color:'white', width:'50px', height:'50px'}}>

{!checkFile&&(

<div>

<p style={{marginTop:'7px'}}>{userList.userList.user.email[0].toUpperCase()}</p>

</div>

)}

{checkFile&&(

<img src={API\_URL + 'images/'+userList.userList.user.file}

alt=""

style={{objectFit:'cover', width:'50px'}}/>

)}

</div>

<div className="user-card\_\_body\_\_info">

<h1>{userList.userList.user.surname} {userList.userList.user.name} {userList.userList.user.lastname}</h1>

<p>{userList.userList.user.email}</p>

</div>

{userList.role&&(

<>

{userList.role.name==="teacher"&&(

<>

{userList.author.user\_id!==userList.userList.user.id&&(

<RxCross2 style={{color:userList.author.decor, marginTop:'30px', fontSize:'24px'}} onClick={handleDelete}/>

)}

</>

)}

</>

)}

</div>

</div>

)

}

export default observer(UserCard)

* + 1. components\UserMap.tsx

import React from 'react'

import UserCard from './UserCard'

export const UserMap = (userList:any) => {

return (

userList.userList.map((user:any)=>(

<UserCard userList={user} author={userList.author} onRemoveUser={userList.onRemoveUser} role={userList.role}/>

))

)

}

* + 1. http\index.ts

import axios from 'axios';

import { AuthRespose } from '../model/AuthResponse';

export const API\_URL = `http://localhost:5000/`

const api = axios.create({

withCredentials: true,

baseURL: API\_URL

})

api.interceptors.request.use((config)=>{

config.headers.Authorization = `Bearer ${localStorage.getItem('token')}`

return config

})

api.interceptors.response.use((config)=>{

return config

}, async(error)=>{

const originalRequest = error.config

if(error.response.status === 401 && !error.config.\_isRetry){

originalRequest.\_isRetry = true

try {

const response = await axios.get<AuthRespose>(`${API\_URL}auth/refresh`, {withCredentials:true})

localStorage.setItem('token', response.data.accessToken)

return api.request(originalRequest)

} catch (e) {

console.log('НЕ АВТОРИЗОВАН')

}

}

throw error

})

export default api

* + 1. middleware\colorToRGBA.ts

export default function hexToRgb(hex: string, alpha: number): string {

const shorthandRegex = /^#?([a-f\d])([a-f\d])([a-f\d])$/i;

hex = hex.replace(shorthandRegex, (\_, r, g, b) => r + r + g + g + b + b);

const result = /^#?([a-f\d]{2})([a-f\d]{2})([a-f\d]{2})$/i.exec(hex);

if (!result) {

throw new Error('Invalid HEX color format');

}

const rgb = [

parseInt(result[1], 16),

parseInt(result[2], 16),

parseInt(result[3], 16),

];

return `rgba(${rgb[0]}, ${rgb[1]}, ${rgb[2]}, ${alpha})`;

}

* + 1. middleware\image.ts

async function checkFileAvailability(url: string): Promise<boolean> {

try {

const response = await fetch(url, { method: 'HEAD' });

return response.ok;

} catch (error) {

console.error(error);

return false;

}

}

export default checkFileAvailability

* + 1. model\AuthResponse.ts

import { User } from "./User"

export interface AuthRespose{

accessToken: string

refreshToken: string

user: User

}

* + 1. model\Complementary.ts

export interface Complementary{

file: string[]

status:boolean

mark:any

datetime:any

task\_id:string

member\_id:string

id:string

}

* + 1. model\Course.ts

export interface Course{

title:string

chapter: string

subject:string

audience: string

cover: string

decor:string

isArchive:boolean

user\_id:string

id: string

}

* + 1. model\Group.ts

export interface Group{

title:string

user\_id:string

id:string

}

* + 1. model\Member.ts

export interface Member{

class\_id:string

user\_id:string

role\_id:string

id:string

}

* + 1. model\Task.ts

export interface Task{

id:string

files:string[]

deadlineDatetime:string[]

title:string

description:string

point:number

isForm:boolean

class\_id:string

theme\_id:string

user\_id:string

}

* + 1. model\Theme.ts

export interface Theme{

title:string

class\_id:string

id:string

}

* + 1. model\User.ts

export interface User{

surname:string

name: string

lastname:string

email: string

isActived: boolean

file:string

colorProfile:string

id: string

}

* + 1. pages\Account.tsx

/\* eslint-disable jsx-a11y/alt-text \*/

import { observer } from 'mobx-react-lite'

import React, { useContext, useEffect, useState } from 'react'

import { PacmanLoader } from 'react-spinners'

import { Context } from '..'

import { RxHamburgerMenu } from 'react-icons/rx'

import logo from '../image/logo.svg'

import '../style/Account.scss'

import AccountInfo from '../components/AccountInfo'

import { Footer } from '../components/Footer'

import ChangeAccount from '../components/ChangeAccount'

import { API\_URL } from '../http'

import { useNavigate } from 'react-router-dom'

import Sidebar from '../components/Sidebar'

const Account = () => {

const [loading, setLoading] = useState<boolean>(true)

const [changeAccount, setChangeAccount] = useState<boolean>(false)

const {store} = useContext(Context)

const navigation = useNavigate()

const [openDrawer, setOpenDrawer] = useState(false)

useEffect(() => {

store.checkAuth().then(()=>{

setLoading(false)

})

})

if(loading){

return(

<div style={{display:'flex'}}>

<div style={{margin:'0 auto', marginTop:'22%'}}>

<PacmanLoader color="#335DFF" />

</div>

</div>

)

}

const handleCloseDrawer = () => {

setOpenDrawer(false);

}

return (

<div className='account'>

<div className='header-account'>

<div className="cont-headerisauth-home">

<div className="logo-isauth-cont">

<div className="burg-menu-cont" onClick={()=>setOpenDrawer(true)}>

<RxHamburgerMenu/>

</div>

{openDrawer&&(

<Sidebar isOpen={openDrawer} onClose={handleCloseDrawer}/>

)}

<img src={logo} alt="EasyAssign" className='logo-img' onClick={()=>navigation('/')}/>

</div>

{!changeAccount&&(

<div className="info-account">

<p style={{borderBottom:'3px solid #335DFF', cursor:'pointer', marginBottom:'-1px', color:'#335DFF'}}>Главная</p>

<p style={{marginLeft:'25px', cursor:'pointer', color:'#AEB5C8'}} onClick={()=>setChangeAccount(true)}>Личная информация</p>

</div>

)}

{changeAccount&&(

<div className="info-account">

<p style={{marginBottom:'-1px', cursor:'pointer', color:'#AEB5C8'}} onClick={()=>setChangeAccount(false)}>Главная</p>

<p style={{borderBottom:'3px solid #335DFF', cursor:'pointer',marginBottom:'-1px', marginLeft:'25px', color:'#335DFF'}}>Личная информация</p>

</div>

)}

<div className="cont-account">

<p>{store.user.email}</p>

{!store.user.file&&(

<div className="img-icon" style={{background:store.user.colorProfile}}>

<p>{store.user.email[0]}</p>

</div>

)}

{store.user.file&&(

<div style={{ width: "60px", height: "60px", borderRadius: "50%", overflow: "hidden", marginLeft:'10px'}}>

<img

src={API\_URL + 'images/' + store.user.file}

alt="avatar"

style={{ width: "100%", height: "100%", objectFit: "cover" }}

/>

</div>

)}

</div>

</div>

</div>

{!changeAccount&&(

<AccountInfo />

)}

{changeAccount&&(

<ChangeAccount/>

)}

<Footer/>

</div>

)

}

export default observer(Account)

* + 1. pages\ChangePassword.tsx

import { observer } from 'mobx-react-lite'

import React, {useContext, useState} from 'react'

import { useNavigate, useParams } from 'react-router-dom';

import { Context } from '..';

import welcome from '../image/welcome.svg'

import {BiHide, BiShow} from 'react-icons/bi'

import '../style/ChangePassword.scss'

import { ToastContainer, toast } from 'react-toastify';

import 'react-toastify/dist/ReactToastify.css';

const ChangePassword = () => {

const [showPassword, setShowPassword] = useState(false);

const [password, setPassword] = useState('');

const [showPassword2, setShowPassword2] = useState(false);

const [password2, setPassword2] = useState('');

const params = useParams();

const navigate = useNavigate()

const {store} = useContext(Context)

const handlePasswordChange = (event: React.ChangeEvent<HTMLInputElement>) => {

setPassword(event.target.value);

};

const toggleShowPassword = () => {

setShowPassword(!showPassword);

};

const repeatHandlePasswordChange = (event: React.ChangeEvent<HTMLInputElement>) => {

setPassword2(event.target.value);

};

const reapeatToggleShowPassword = () => {

setShowPassword2(!showPassword2);

};

const handleChangePassword=()=>{

if(password===password2){

if(params.id)

store.changePassword(password, params.id).then((data)=>{

if(data)

navigate('/signin')

if(store.error)

// eslint-disable-next-line array-callback-return

toast.error(store.error, {

position: "top-right",

autoClose: 5000,

hideProgressBar: false,

closeOnClick: true,

pauseOnHover: true,

draggable: true,

progress: undefined,

theme: "light",

});

})

}

else{

toast.error('Пароли не совпадают', {

position: "top-right",

autoClose: 5000,

hideProgressBar: false,

closeOnClick: true,

pauseOnHover: true,

draggable: true,

progress: undefined,

theme: "light",

});

}

}

return (

<div className="login-bod">

<div className="cont-welcome-image">

<img className='welcome-img' src={welcome} alt="welcomeimg" />

</div>

<div className="cont-chng-form">

<div className="cnt-chgpass">

<h1>Смена пароля</h1>

<ToastContainer />

<div className="continput">

<p>Пароль:</p>

<div style={{display:'flex'}} className='inputlog'>

<input

type={showPassword ? 'text' : 'password'}

value={password}

onChange={handlePasswordChange}

style={{border:'none', width:'92%'}}

placeholder='Пароль'

/>

<button onClick={toggleShowPassword} style={{background:'none', border:'none', fontSize:'20px', marginTop:'5px', color:'#6D6875'}}>

{showPassword ? <BiHide/> : <BiShow/>}

</button>

</div>

</div>

<div className="continput">

<p>Повторить пароль:</p>

<div style={{display:'flex'}} className='inputlog'>

<input

type={showPassword2 ? 'text' : 'password'}

value={password2}

onChange={repeatHandlePasswordChange}

style={{border:'none', width:'92%'}}

placeholder='Пароль'

/>

<button onClick={reapeatToggleShowPassword} style={{background:'none', border:'none', fontSize:'20px', marginTop:'5px', color:'#6D6875'}}>

{showPassword2 ? <BiHide/> : <BiShow/>}

</button>

</div>

</div>

<div className="btn-cont">

<div className="btn-auth">

<button onClick={handleChangePassword}>Сменить пароль</button>

</div>

</div>

</div>

</div>

</div>

)

}

export default observer(ChangePassword)

* + 1. pages\Classroom.tsx

import { observer } from 'mobx-react-lite'

import React, {useContext, useState, useEffect} from 'react'

import logo from '../image/logo.svg'

import { Context } from '..'

import { useNavigate, useParams } from 'react-router-dom'

import { RxHamburgerMenu } from 'react-icons/rx'

import { API\_URL } from '../http'

import '../style/Classroom.scss'

import Sidebar from '../components/Sidebar'

import { PacmanLoader } from 'react-spinners'

import ClassroomLenta from '../components/ClassroomLenta'

import { Footer } from '../components/Footer'

import ClassroomTask from '../components/ClassroomTask'

import ClassroomUsers from '../components/ClassroomUsers'

const Classroom = () => {

const navigate = useNavigate()

const [openDrawer, setOpenDrawer] = useState(false)

const [loading, setLoading] = useState<boolean>(true)

const {store} = useContext(Context)

const [lenta, setLenta] = useState<boolean>(true)

const [task, setTask] = useState<boolean>(false)

const [users, setUsers] = useState<boolean>(false)

const id = useParams()

const [course, setCourse] = useState<any>()

useEffect(() => {

store.checkAuth().then(()=>{

if(id.id)

store.getCourseOne(id.id, store.user.id).then((data)=>{

setCourse(data?.data.member[0].class)

setLoading(false)

})

})

})

if(loading){

return(

<div style={{display:'flex'}}>

<div style={{margin:'0 auto', marginTop:'22%'}}>

<PacmanLoader color="#335DFF" />

</div>

</div>

)

}

const handleCloseDrawer = () => {

setOpenDrawer(false);

}

const activeLenta = () =>{

setLenta(true)

setUsers(false)

setTask(false)

}

const activeTask = () =>{

setTask(true)

setLenta(false)

setUsers(false)

}

const activeUsers = () =>{

setUsers(true)

setLenta(false)

setTask(false)

}

return (

<div className="classroom">

<div className="header-class">

<header className="header-classroom">

<div className="header-classroom\_\_left">

<RxHamburgerMenu className='hamburmenu' onClick={()=>setOpenDrawer(true)}/>

{openDrawer&&(

<Sidebar isOpen={openDrawer} onClose={handleCloseDrawer}/>

)}

<button className="header-classroom\_\_menu-btn">

<i className="fas fa-bars"></i>

</button>

<h1 className="header-classroom\_\_title">

<img src={logo} alt="" style={{width:'90%'}}/>

</h1>

</div>

<div className="header-classroom\_\_center" style={{marginTop:'22px', color:'black'}}>

{lenta&&(

<p className="header-classroom\_\_center\_\_lenta" style={{color:course.decor, borderBottom:`2px solid ${course.decor}`,

paddingBottom:'22px'}}>Лента</p>

)}

{!lenta&&(

<p className="header-classroom\_\_center\_\_lenta" onClick={activeLenta}>Лента</p>

)}

{task&&(

<p className="header-classroom\_\_center\_\_task" style={{color:course.decor, borderBottom:`2px solid ${course.decor}`,

paddingBottom:'22px'}}>Задания</p>

)}

{!task&&(

<p className="header-classroom\_\_center\_\_task" onClick={activeTask}>Задания</p>

)}

{users&&(

<p className="header-classroom\_\_center\_\_users" style={{color:course.decor, borderBottom:`2px solid ${course.decor}`,

paddingBottom:'22px'}}>Пользователи</p>

)}

{!users&&(

<p className="header-classroom\_\_center\_\_users" onClick={activeUsers}>Пользователи</p>

)}

</div>

<div className="header-classroom\_\_right" onClick={()=>navigate('/account')}>

<button className="header-classroom\_\_button header-classroom\_\_button--account">

<span className="header-classroom\_\_username">{store.user.email}</span>

</button>

<button className="header-classroom\_\_button header-classroom\_\_button--notifications">

{!store.user.file&&(

<div className="img-icon" style={{background:store.user.colorProfile}}>

<div style={{paddingTop:'3px'}}>

<p>{store.user.email[0]}</p>

</div>

</div>

)}

{store.user.file&&(

<div style={{ width: "60px", height: "60px", borderRadius: "50%", overflow: "hidden" }}>

<img

src={API\_URL + 'images/' + store.user.file}

alt="avatar"

style={{ width: "100%", height: "100%", objectFit: "cover" }}

/>

</div>

)}

</button>

</div>

</header>

</div>

<div className="content">

{lenta&&(

<ClassroomLenta/>

)}

{task&&(

<ClassroomTask/>

)}

{users&&(

<ClassroomUsers/>

)}

</div>

<div className="footer">

<Footer/>

</div>

</div>

)

}

export default observer(Classroom)

* + 1. pages\Home.tsx

import { observer } from 'mobx-react-lite'

import React, {useState, useEffect, useContext} from 'react'

import HeaderHome from '../components/HeaderHome'

import { Footer } from '../components/Footer'

import '../style/Home.scss'

import PacmanLoader from "react-spinners/PacmanLoader";

import { Context } from '..'

import '../style/Dropdown.scss'

import MainHomeIsAuth from '../components/MainHomeIsAuth'

import { HeaderHomeNoneAuth } from '../components/HeaderHomeNoneAuth'

import { AuthError } from '../components/AuthError'

const Home = () => {

const [loading, setLoading] = useState<boolean>(true)

const {store} = useContext(Context)

useEffect(() => {

store.checkAuth().then(()=>{

setLoading(false)

})

})

if(loading){

return(

<div style={{display:'flex'}}>

<div style={{margin:'0 auto', marginTop:'22%'}}>

<PacmanLoader color="#335DFF" />

</div>

</div>

)

}

return (

<div className="home">

<div className="header-cont">

{store.isAuth&&(

<HeaderHome />

)}

{!store.isAuth&&(

<HeaderHomeNoneAuth/>

)}

</div>

<div className="content">

{store.isAuth&&(

<MainHomeIsAuth/>

)}

{!store.isAuth&&(

<AuthError/>

)}

</div>

<div className="footer">

<Footer/>

</div>

</div>

)

}

export default observer(Home)

* + 1. pages\Login.tsx

import React, {useState, useContext} from 'react'

import welcome from '../image/welcome.svg'

import '../style/Login.scss'

import {BiShow, BiHide} from 'react-icons/bi'

import {useNavigate} from 'react-router-dom'

import { Context } from '..'

import { ToastContainer, toast } from 'react-toastify';

import 'react-toastify/dist/ReactToastify.css';

import { observer } from 'mobx-react-lite';

const Login = () => {

const [showPassword, setShowPassword] = useState(false);

const [password, setPassword] = useState('');

const [email, setEmail] = useState('')

const navigation = useNavigate()

const {store} = useContext(Context)

const handlePasswordChange = (event: React.ChangeEvent<HTMLInputElement>) => {

setPassword(event.target.value);

};

const toggleShowPassword = () => {

setShowPassword(!showPassword);

};

const handleLogin = () =>{

store.login(email, password).then((data)=>{

if(data)

navigation('/')

if(store.error){

toast.error(store.error, {

position: "top-right",

autoClose: 5000,

hideProgressBar: false,

closeOnClick: true,

pauseOnHover: true,

draggable: true,

progress: undefined,

theme: "light",

});

}

else if(!store.user.isActived){

toast.error('Аккаунт не подтвержден', {

position: "top-right",

autoClose: 5000,

hideProgressBar: false,

closeOnClick: true,

pauseOnHover: true,

draggable: true,

progress: undefined,

theme: "light",

});

}

})

}

return (

<div className="login-bod">

<div className="cont-welcome-image">

<img className='welcome-img' src={welcome} alt="welcomeimg" />

</div>

<div className="cont-auth-form">

<div className="cnt-aut">

<h1>Авторизация</h1>

<ToastContainer />

<div className="continput">

<p>Адрес электронной почты:</p>

<input type="email" placeholder='Email' className='inputlog' autoComplete='false'

onChange={(e)=>setEmail(e.target.value)} value={email}/>

</div>

<div className="continput">

<p>Пароль:</p>

<div style={{display:'flex'}} className='inputlog'>

<input

type={showPassword ? 'text' : 'password'}

value={password}

onChange={handlePasswordChange}

style={{border:'none', width:'92%'}}

/>

<button onClick={toggleShowPassword} style={{background:'none', border:'none', fontSize:'20px', marginTop:'5px', color:'#6D6875'}}>

{showPassword ? <BiHide/> : <BiShow/>}

</button>

</div>

</div>

<div className="change-password">

<p>Забыли пароль?</p>

<a href="/resetpassword">Вспомнить</a>

</div>

<div className="btn-cont">

<div className="btn-auth">

<button onClick={handleLogin}>Авторизоваться</button>

</div>

<div className="btn-reg">

<button onClick={()=>navigation('/signup')}>Зарегистрироваться</button>

</div>

</div>

</div>

</div>

</div>

)

}

export default observer(Login)

* + 1. pages\Registration.tsx

import React, {useState, useContext, useEffect} from 'react'

import welcome from '../image/welcome.svg'

import {BiHide, BiShow} from 'react-icons/bi'

import '../style/Registration.scss'

import {useNavigate} from 'react-router-dom'

import { Context } from '..'

import { ToastContainer, toast } from 'react-toastify';

import 'react-toastify/dist/ReactToastify.css';

import { observer } from 'mobx-react-lite'

const getRandomHexColor = (): string => {

const letters = "0123456789ABCDEF";

let color = "#";

for (let i = 0; i < 6; i++) {

color += letters[Math.floor(Math.random() \* 16)];

}

return color;

};

const Registration = () => {

const [showPassword, setShowPassword] = useState(false);

const [password, setPassword] = useState('');

const [showPassword2, setShowPassword2] = useState(false);

const [password2, setPassword2] = useState('');

const [surname, setSurname] = useState('')

const [name, setName] = useState('')

const [lastname, setLastname] = useState('')

const [email, setEmail] = useState('')

const [color, setColor] = useState<string>(getRandomHexColor());

const navigate = useNavigate()

const {store} = useContext(Context)

useEffect(() => {

setColor(getRandomHexColor());

}, [])

const handlePasswordChange = (event: React.ChangeEvent<HTMLInputElement>) => {

setPassword(event.target.value);

};

const toggleShowPassword = () => {

setShowPassword(!showPassword);

};

const repeatHandlePasswordChange = (event: React.ChangeEvent<HTMLInputElement>) => {

setPassword2(event.target.value);

};

const reapeatToggleShowPassword = () => {

setShowPassword2(!showPassword2);

};

const handleRegistration=()=>{

if(password===password2){

store.registration(surname, name, lastname, email, password, color).then((data)=>{

if(data){

toast.success('Для подтверждения регистрации подтвердите почту', {

position: "top-right",

autoClose: 5000,

hideProgressBar: false,

closeOnClick: true,

pauseOnHover: true,

draggable: true,

progress: undefined,

theme: "light",

});

}

if(store.error){

toast.error(store.error, {

position: "top-right",

autoClose: 5000,

hideProgressBar: false,

closeOnClick: true,

pauseOnHover: true,

draggable: true,

progress: undefined,

theme: "light",

});

}

})

}

else{

toast.error('Пароли не совпадают', {

position: "top-right",

autoClose: 5000,

hideProgressBar: false,

closeOnClick: true,

pauseOnHover: true,

draggable: true,

progress: undefined,

theme: "light",

});

}

}

return (

<div className="login-bod" style={{overflowY:'hidden'}}>

<div className="cont-welcome-image">

<img className='welcome-img' src={welcome} alt="welcomeimg" />

</div>

<div className="cont-auth-form">

<div className="cnt-reg">

<h1>Регистрация</h1>

<ToastContainer />

<div className="continput">

<p>Фамилия:</p>

<input type="text" placeholder='Фамилия' className='inputlog' autoComplete='false'

onChange={(e)=>setSurname(e.target.value)} value={surname}/>

</div>

<div className="continput">

<p>Имя:</p>

<input type="text" placeholder='Имя' className='inputlog' autoComplete='false'

onChange={(e)=>{setName(e.target.value)}} value={name}/>

</div>

<div className="continput">

<p>Отчество:</p>

<input type="text" placeholder='Отчество' className='inputlog' autoComplete='false'

onChange={(e)=>{setLastname(e.target.value)}} value={lastname}/>

</div>

<div className="continput">

<p>Адрес электронной почты:</p>

<input type="email" placeholder='Email' className='inputlog' autoComplete='false'

onChange={(e)=>{setEmail(e.target.value)}} value={email}/>

</div>

<div className="continput">

<p>Пароль:</p>

<div style={{display:'flex'}} className='inputlog'>

<input

type={showPassword ? 'text' : 'password'}

value={password}

onChange={handlePasswordChange}

style={{border:'none', width:'92%'}}

placeholder='Пароль'

/>

<button onClick={toggleShowPassword} style={{background:'none', border:'none', fontSize:'20px', marginTop:'5px', color:'#6D6875'}}>

{showPassword ? <BiHide/> : <BiShow/>}

</button>

</div>

</div>

<div className="continput">

<p>Повторить пароль:</p>

<div style={{display:'flex'}} className='inputlog'>

<input

type={showPassword2 ? 'text' : 'password'}

value={password2}

onChange={repeatHandlePasswordChange}

style={{border:'none', width:'92%'}}

placeholder='Пароль'

/>

<button onClick={reapeatToggleShowPassword} style={{background:'none', border:'none', fontSize:'20px', marginTop:'5px', color:'#6D6875'}}>

{showPassword2 ? <BiHide/> : <BiShow/>}

</button>

</div>

</div>

<div className="btn-cont">

<div className="btn-auth">

<button onClick={handleRegistration}>Зарегистрироваться</button>

</div>

<div className="btn-reg">

<button onClick={()=>navigate('/signin')}>Авторизоваться</button>

</div>

</div>

</div>

</div>

</div>

)

}

export default observer(Registration)

* + 1. pages\ResetPassword.tsx

import React, {useState, useContext} from 'react'

import welcome from '../image/welcome.svg'

import '../style/ResetPassword.scss'

import { Context } from '..'

import { ToastContainer, toast } from 'react-toastify';

import 'react-toastify/dist/ReactToastify.css';

import { observer } from 'mobx-react-lite';

const ResetPassword = () => {

const [email, setEmail] = useState('')

const {store} = useContext(Context)

const handleResetPassword = ()=>{

store.resetPassword(email).then((data)=>{

if(data)

toast.success('Для смены пароля перейдите по ссылке, которая была выслана на почту', {

position: "top-right",

autoClose: 5000,

hideProgressBar: false,

closeOnClick: true,

pauseOnHover: true,

draggable: true,

progress: undefined,

theme: "light",

});

if(store.error)

toast.error(store.error, {

position: "top-right",

autoClose: 5000,

hideProgressBar: false,

closeOnClick: true,

pauseOnHover: true,

draggable: true,

progress: undefined,

theme: "light",

});

})

}

return (

<div className="login-bod">

<div className="cont-welcome-image">

<img className='welcome-img' src={welcome} alt="welcomeimg" />

</div>

<div className="cont-resetpass-form">

<div className="cnt-aut">

<h1>Сброс пароля</h1>

<ToastContainer />

<div className="continput">

<p>Адрес электронной почты:</p>

<input type="email" placeholder='Email' className='inputlog' autoComplete='false'

value={email} onChange={(e)=>setEmail(e.target.value)}/>

</div>

<div className="btn-cont">

<div className="btn-auth">

<button onClick={handleResetPassword}>Сбросить пароль</button>

</div>

</div>

</div>

</div>

</div>

)

}

export default observer(ResetPassword)

* + 1. pages\Task.tsx

import { observer } from 'mobx-react-lite'

import React, { useContext, useEffect, useState } from 'react'

import { Context } from '..'

import { useParams } from 'react-router-dom'

import HeaderHome from '../components/HeaderHome'

import { PacmanLoader } from 'react-spinners'

import { Footer } from '../components/Footer'

import StudentTaskIsForm from '../components/StudentTaskIsForm'

import '../style/Task.scss'

const Task = () => {

const {store} = useContext(Context)

const id = useParams()

const [role, setRole] = useState<any>()

const [loading, setLoading] = useState<boolean>(true)

useEffect(() => {

store.checkAuth().then(()=>{

if(id.id)

store.viewTaskOne(id.id, store.user.id).then((data)=>{

setRole(data?.data.role)

setLoading(false)

})

})

}, [id.id, store])

if(loading){

return(

<div style={{display:'flex'}}>

<div style={{margin:'0 auto', marginTop:'22%'}}>

<PacmanLoader color="#335DFF" />

</div>

</div>

)

}

return (

<>

{role.name==="student"&&(

<>

<HeaderHome />

</>

)}

<StudentTaskIsForm/>

<Footer />

</>

)

}

export default observer(Task)

* + 1. service\AuthService.ts

import api from "../http";

import {AxiosResponse} from 'axios'

import { AuthRespose } from "../model/AuthResponse";

export default class AuthService{

static async login(email: string, password: string): Promise<AxiosResponse<AuthRespose>>{

return await api.post<AuthRespose>('auth/login', {email, password})

}

static async registration(surname:string, name:string, lastname:string, email: string, password: string, color:string): Promise<AxiosResponse<AuthRespose>>{

return await api.post<AuthRespose>('auth/registration', {surname, name, lastname, email, password, color})

}

static async logout(): Promise<void>{

return await api.post('auth/logout')

}

}

* + 1. service\ComplementaryService.ts

import { AxiosResponse } from "axios";

import { Complementary } from "../model/Complementary";

import api from "../http";

export default class ComplementaryService{

static async getOne(task\_id: string, user\_id: string): Promise<AxiosResponse<Complementary>>{

return await api.post<Complementary>('complementary/getOne', {task\_id, user\_id})

}

static async editStatus(id: string, status: string): Promise<AxiosResponse<Complementary>>{

return await api.post<Complementary>('complementary/editStatus', {id, status})

}

}

* + 1. service\CourseService.ts

import api from "../http";

import {AxiosResponse} from 'axios'

import { Course } from "../model/Course";

export default class CourseService{

static async createClass(title:string, chapter:string, subject:string, audience:string, decor:string, user\_id:string): Promise<AxiosResponse<Course>>{

return await api.post<Course>('class/create', {title, chapter, subject, audience, decor, user\_id})

}

static async viewClassList(user:string): Promise<AxiosResponse<Course>>{

return await api.post<Course>('class/getAll', {user})

}

static async editClass(title:string, chapter:string, subject:string, audience:string, welcomeLink:string, decor:string): Promise<AxiosResponse<Course>>{

return await api.post<Course>('class/edit', {title, chapter, subject, audience, welcomeLink, decor})

}

static async addArchive(id:string): Promise<AxiosResponse<Course>>{

return await api.post<Course>('class/addArchive', {id})

}

static async getOne(welcomelink:string, user:string): Promise<AxiosResponse<any>>{

return await api.post<any>('class/getOne', {welcomelink, user})

}

static async getClassStudent(user:string): Promise<AxiosResponse<Course>>{

return await api.post<Course>('class/getClassStudent', {user})

}

static async getClassTeacher(user:string): Promise<AxiosResponse<Course>>{

return await api.post<Course>('class/getClassTeacher', {user})

}

}

* + 1. service\GroupService.ts

import api from "../http";

import {AxiosResponse} from 'axios'

import { Group } from "../model/Group";

export default class GroupService{

static async create(title: string, user\_id: string): Promise<AxiosResponse<Group>>{

return await api.post<Group>('group/create', {title, user\_id})

}

static async viewAll(user\_id: string): Promise<AxiosResponse<Group>>{

return await api.post<Group>('group/getAll', {user\_id})

}

static async edit(title: string, id: string): Promise<AxiosResponse<Group>>{

return await api.post<Group>('group/edit', {title, id})

}

static async delete(id: string): Promise<AxiosResponse<Group>>{

return await api.post<Group>('group/delete', {id})

}

static async viewUsers(id: string): Promise<AxiosResponse<any>>{

return await api.post<any>('group/viewUsers', {id})

}

static async viewFreeUsers(id: string): Promise<AxiosResponse<Group>>{

return await api.post<Group>('group/viewFreeUsers', {id})

}

static async addUserToGroup(user: string, group:string): Promise<AxiosResponse<Group>>{

return await api.post<Group>('group/addUserToGroup', {user, group})

}

static async deleteUserFromGroup(user: string, group:string): Promise<AxiosResponse<Group>>{

return await api.post<Group>('group/deleteUserFromGroup', {user, group})

}

}

* + 1. service\MemberService.ts

import api from "../http";

import {AxiosResponse} from 'axios'

import { Member } from "../model/Member";

export default class MemberService{

static async viewTeacher(class\_id:string): Promise<AxiosResponse<Member[]>>{

return await api.post<Member[]>('member/viewTeacher', {class\_id})

}

static async viewStudent(class\_id:string): Promise<AxiosResponse<Member[]>>{

return await api.post<Member[]>('member/viewStudent', {class\_id})

}

static async freeUsers(id:string): Promise<AxiosResponse<Member[]>>{

return await api.post<Member[]>('member/viewFreeUsers', {id})

}

static async addUsersToClass(users:any, class\_id:string, role\_name:string): Promise<AxiosResponse<Member[]>>{

return await api.post<Member[]>('member/addUsersToClass', {users, class\_id, role\_name})

}

static async deleteFromClass(user\_id:string, class\_id:string): Promise<AxiosResponse<Member[]>>{

return await api.post<Member[]>('member/deleteFromClass', {user\_id, class\_id})

}

static async addGroupToClass(group\_id:string, class\_id:string): Promise<AxiosResponse<Member[]>>{

return await api.post<Member[]>('member/addGroupToClass', {group\_id, class\_id})

}

}

* + 1. service\TaskService.ts

import api from "../http";

import {AxiosResponse} from 'axios'

import { Task } from "../model/Task";

export default class TaskService{

static async viewTasks(class\_id:string): Promise<AxiosResponse<Task[]>>{

return await api.post<Task[]>('task/view', {class\_id})

}

static async viewFiles(files:string): Promise<AxiosResponse<any>>{

return await api.post<any>('task/viewFile', {files})

}

static async delete(id:string): Promise<AxiosResponse<Task[]>>{

return await api.post<Task[]>('task/delete', {id})

}

static async addTheme(id:string, theme\_id:string): Promise<AxiosResponse<Task[]>>{

return await api.post<Task[]>('task/addTheme', {id, theme\_id})

}

static async deleteTheme(id:string): Promise<AxiosResponse<Task[]>>{

return await api.post<Task[]>('task/deleteTheme', {id})

}

static async allTaskWithoutTheme(class\_id:string): Promise<AxiosResponse<Task[]>>{

return await api.post<Task[]>('task/allTaskWithoutTheme', {class\_id})

}

static async allTaskWithTheme(class\_id:string): Promise<AxiosResponse<Task[]>>{

return await api.post<Task[]>('task/allTaskWithTheme', {class\_id})

}

static async viewTaskOne(id:string, user\_id:string): Promise<AxiosResponse<any>>{

return await api.post<any>('task/viewTaskOne', {id, user\_id})

}

}

* + 1. service\ThemeService.ts

import api from "../http";

import {AxiosResponse} from 'axios'

import { Theme } from "../model/Theme";

export default class ThemeService{

static async create(title:string, class\_id:string): Promise<AxiosResponse<Theme[]>>{

return await api.post<Theme[]>('theme/create', {title, class\_id})

}

static async viewAll(class\_id:string): Promise<AxiosResponse<Theme[]>>{

return await api.post<Theme[]>('theme/viewAll', {class\_id})

}

static async update(id:string,title:string): Promise<AxiosResponse<Theme[]>>{

return await api.post<Theme[]>('theme/update', {id, title})

}

static async delete(id:string): Promise<AxiosResponse<Theme[]>>{

return await api.post<Theme[]>('theme/delete', {id})

}

}

* + 1. service\UserService.ts

import api from "../http";

import {AxiosResponse} from 'axios'

import { User } from "../model/User";

export default class UserService{

static async resetPassword(email:string): Promise<AxiosResponse<User[]>>{

return await api.post<User[]>('user/resetPassword', {email})

}

static async changePassword(password:string,passwordLink:any): Promise<AxiosResponse<User[]>>{

return await api.post<User[]>('user/changePassword', {password, passwordLink})

}

static async editUser(id:string, email:string, surname:string, name:string, lastname:string): Promise<AxiosResponse<User[]>>{

return api.post<User[]>('user/editProfile', {id, email, surname, name, lastname})

}

}

* + 1. store\store.ts

import { makeAutoObservable } from "mobx"

import { User } from "../model/User"

import AuthService from "../service/AuthService"

import axios from "axios"

import { AuthRespose } from "../model/AuthResponse"

import { API\_URL } from "../http"

import UserService from "../service/UserService"

import CourseService from "../service/CourseService"

import GroupService from "../service/GroupService"

import TaskService from "../service/TaskService"

import ThemeService from "../service/ThemeService"

import MemberService from "../service/MemberService"

import ComplementaryService from "../service/ComplementaryService"

export default class Store{

user = {} as User

error? = []

isAuth = false

isLoading = false

constructor(){

makeAutoObservable(this)

}

setAuth(bool:boolean){

this.isAuth = bool

}

setUser(user: User){

this.user = user

}

setLoading(bool: boolean){

this.isLoading = bool

}

setError(error?: []){

this.error = error

}

async login(email: string, password:string){

try {

this.setError([])

const {data} = await AuthService.login(email, password)

localStorage.setItem('token', data.accessToken)

this.setAuth(true)

this.setUser(data.user)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async registration(surname:string, name:string, lastname:string, email: string, password:string, color:string){

try {

this.setError([])

const {data} = await AuthService.registration(surname, name, lastname, email, password, color)

localStorage.setItem('token', data.accessToken)

this.setAuth(true)

this.setUser(data.user)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async logout(){

try {

await AuthService.logout()

localStorage.removeItem('token')

this.setAuth(false)

this.setUser({} as User)

} catch (e) {

console.log(e)

}

}

async checkAuth(){

this.setLoading(true)

try {

const response = await axios.get<AuthRespose>(`${API\_URL}auth/refresh`, {withCredentials:true})

if(localStorage.getItem('token'))

{this.setAuth(true)

this.setUser(response.data.user)}

} catch (e) {

console.log(e)

} finally{

this.setLoading(false)

}

}

async resetPassword(email:string){

try {

this.setError([])

const data = await UserService.resetPassword(email)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async changePassword(password:string, passwordLink:string){

try {

this.setError([])

const data = await UserService.changePassword(password, passwordLink)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async editAva(email:string, image:any){

try {

const formData = new FormData()

formData.append('testImage', image)

formData.append('email', email)

const data = await axios.post(API\_URL+'user/editPhoto', formData)

return data

} catch (e) {

}

}

async editProfile(id:string, email:string, surname:string, name:string, lastname:string){

try {

this.setError([])

const data = await UserService.editUser(id, email, surname, name, lastname)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async createClass(title:string, chapter:string, subject:string, audience:string, decor:string, user\_id:string){

try {

this.setError([])

const data = await CourseService.createClass(title, chapter, subject, audience, decor, user\_id)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async getAllClass(user\_id:string){

try {

this.setError([])

const data = await CourseService.viewClassList(user\_id)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async createGroupe(title:string, user\_id:string){

try {

this.setError([])

const data = await GroupService.create(title, user\_id)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async getAllGroup(user\_id:string){

try {

this.setError([])

const data = await GroupService.viewAll(user\_id)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async editGroupe(title:string, id:string){

try {

this.setError([])

const data = await GroupService.edit(title, id)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async deleteGroupe(id:string){

try {

this.setError([])

const data = await GroupService.delete(id)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async viewUsersInGroup(id:string){

try {

this.setError([])

const data = await GroupService.viewUsers(id)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async viewFreeUsersInGroup(id:string){

try {

this.setError([])

const data = await GroupService.viewFreeUsers(id)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async addUserToGroup(user:string, group:string){

try {

this.setError([])

const data = await GroupService.addUserToGroup(user, group)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async deleteUserFromGroup(user:string, group:string){

try {

this.setError([])

const data = await GroupService.deleteUserFromGroup(user, group)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async editCourse(title:string, chapter:string, subject:string, audience:string, welcomeLink:string, decor:string){

try {

this.setError([])

const data = await CourseService.editClass(title, chapter, subject, audience, welcomeLink, decor)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async addArchive(id:string){

try {

this.setError([])

const data = await CourseService.addArchive(id)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async getCourseOne(welcomelink:string, user:string){

try {

this.setError([])

const data = await CourseService.getOne(welcomelink, user)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async editCover(welcomelink:string, image:any){

try {

const formData = new FormData()

formData.append('testImage', image)

formData.append('welcomelink', welcomelink)

const data = await axios.post(API\_URL+'class/editCover', formData)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async addTask(files:any, deadlineDatetime:string, title:string, description:string, point:string, class\_id:string, isForm:string, user\_id:string){

try {

const formData = new FormData()

if(files){

files.forEach((element: any) => {

formData.append('files', element)

});

}

formData.append('deadlineDatetime', deadlineDatetime)

formData.append('title', title)

formData.append('description', description)

formData.append('point', point)

formData.append('class\_id', class\_id)

formData.append('isForm', isForm)

formData.append('user\_id', user\_id)

const data = await axios.post(API\_URL+'task/create', formData)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async viewTasks(class\_id:string){

try {

this.setError([])

const data = await TaskService.viewTasks(class\_id)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async viewFiles(files:any){

try {

this.setError([])

const data = await TaskService.viewFiles(files)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async updateTask(id:string, oldFiles:any, files:any, deadlineDatetime:string, title:string, description:string, point:string, isForm:string){

try {

const formData = new FormData()

if(files){

files.forEach((element: any) => {

formData.append('files', element)

});

}

if(oldFiles){

oldFiles.forEach((old:any)=>{

formData.append('oldFiles', old.id)

})

}

formData.append('id', id)

formData.append('deadlineDatetime', deadlineDatetime)

formData.append('title', title)

formData.append('description', description)

formData.append('point', point)

formData.append('isForm', isForm)

const data = await axios.post(API\_URL+'task/update', formData)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async deleteTask(id:string){

try {

this.setError([])

const data = await TaskService.delete(id)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async createTheme(title:string, class\_id:string){

try {

this.setError([])

const data = await ThemeService.create(title, class\_id)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async viewAllTheme(class\_id:string){

try {

this.setError([])

const data = await ThemeService.viewAll(class\_id)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async addTheme(id:string, theme\_id:string){

try {

this.setError([])

const data = await TaskService.addTheme(id, theme\_id)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async deleteFromTheme(id:string){

try {

this.setError([])

const data = await TaskService.deleteTheme(id)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async allTaskWithoutTheme(class\_id:string){

try {

this.setError([])

const data = await TaskService.allTaskWithoutTheme(class\_id)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async allTaskWithTheme(class\_id:string){

try {

this.setError([])

const data = await TaskService.allTaskWithTheme(class\_id)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async updateTheme(id:string, title:string){

try {

this.setError([])

const data = await ThemeService.update(id, title)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async deleteTheme(id:string){

try {

this.setError([])

const data = await ThemeService.delete(id)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async viewTeacher(class\_id:string){

try {

this.setError([])

const data = await MemberService.viewTeacher(class\_id)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async viewStudent(class\_id:string){

try {

this.setError([])

const data = await MemberService.viewStudent(class\_id)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async freeUsers(id:string){

try {

this.setError([])

const data = await MemberService.freeUsers(id)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async addUsersToClass(users:any, class\_id:string, role\_name:string){

try {

this.setError([])

const data = await MemberService.addUsersToClass(users, class\_id, role\_name)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async deleteFromClass(user\_id:string, class\_id:string){

try {

this.setError([])

const data = await MemberService.deleteFromClass(user\_id, class\_id)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async addGroupToClass(group\_id:string, class\_id:string){

try {

this.setError([])

const data = await MemberService.addGroupToClass(group\_id, class\_id)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async getClassStudent(user:string){

try {

this.setError([])

const data = await CourseService.getClassStudent(user)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async getClassTeacher(user:string){

try {

this.setError([])

const data = await CourseService.getClassTeacher(user)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async viewTaskOne(id:string, user\_id:string){

try {

this.setError([])

const data = await TaskService.viewTaskOne(id, user\_id)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async viewComplementaryOne(task\_id:string, user\_id:string){

try {

this.setError([])

const data = await ComplementaryService.getOne(task\_id, user\_id)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

async editComplementaryStatus(id:string, status:string){

try {

this.setError([])

const data = await ComplementaryService.editStatus(id, status)

return data

} catch (e) {

console.log(e)

let error: any = e;

this.setError(error.response.data.message)

}

}

}

1.4.69. controller\auth-controller.ts

import {validationResult} from 'express-validator'

import { ApiError } from '../exception/api-error'

import AuthService from '../service/auth-service'

class AuthController{

async login(req:any, res:any, next:any){

try {

const {email, password} = req.body

const userData = await AuthService.login(email, password)

res.cookie('refreshToken', userData.refreshToken, {maxAge: 30 \* 24 \* 60 \* 60 \* 1000, httpOnly: true})

return res.send(userData)

} catch (e) {

next(e)

}

}

async registration(req:any, res:any, next:any){

try {

const errors = validationResult(req)

if(!errors.isEmpty()){

console.log(errors)

return next(ApiError.BadRequest('Пароль должен содержать минимум 6 символов и максимум 32', Object(errors)))

}

const {email, password, surname, name, lastname, color} = req.body

const userData = await AuthService.registration( email, password, name, surname, lastname, color )

res.cookie('refreshToken', userData.refreshToken, {maxAge: 30 \* 24 \* 60 \* 60 \* 1000, httpOnly: true})

return res.json(userData)

} catch (e) {

next(e)

}

}

async activate(req:any, res:any, next:any){

try {

const activationLink = req.params.link;

await AuthService.activate(activationLink)

return res.redirect(process.env.CLIENT\_URL+`/signin`)

} catch (e) {

next(e)

}

}

async logout(req:any, res:any, next:any){

try {

const {id} = req.user

const token = await AuthService.logout(id)

res.clearCookie('refreshToken')

return res.json(token)

} catch (e) {

next(e)

}

}

async refresh(req:any, res:any, next:any){

try {

const {refreshToken} = req.cookies

const userData = await AuthService.refresh(refreshToken)

res.cookie('refreshToken', userData.refreshToken, {maxAge: 30 \* 24 \* 60 \* 60 \* 1000, httpOnly: true})

return res.json(userData)

} catch (e) {

next(e)

}

}

}

export default new AuthController()

1.4.70. controller\class-controller.ts

import { upload } from "../middleware/upload-middleware"

import classService from "../service/class-service"

class ClassController{

async createClass(req:any, res:any, next:any){

try {

const {title, chapter, subject, audience, user\_id, decor} = req.body

const classData = classService.createClass(title, chapter, subject, audience, user\_id, decor)

return res.send(await classData)

} catch (e) {

next(e)

}

}

async getClass(req:any, res:any, next:any){

try {

const {user} = req.body

const classData = await classService.getClass(user)

return res.send(classData)

} catch (e) {

next(e)

}

}

async getClassOne(req:any, res:any, next:any){

try {

const {welcomelink, user} = req.body

const classData = await classService.getClassOne(welcomelink, user)

return res.send(classData)

} catch (e) {

next(e)

}

}

async getAuthorInfo(req:any, res:any, next:any){

try {

const {user} = req.body

const classData = await classService.getAuthorInfo(user)

return res.send(classData)

} catch (e) {

next(e)

}

}

async editCover(req:any, res:any, next:any){

try {

upload(req, res, async(err:any)=>{

if(err){

next(err)

}

else if(req.file)

{

const {welcomelink} = req.body

const userData = await classService.edit\_avatar(req.file.filename, welcomelink)

return res.send(userData)

}

else {

next('Файл отсутствует')

}

})

} catch (e) {

next(e)

}

}

async updateClass(req:any, res:any, next:any){

try {

const {title, chapter, subject, audience, welcomeLink, decor} = req.body

const classData = await classService.updateClass(title, chapter, subject, audience, welcomeLink, decor)

return res.send(classData)

} catch (e) {

next(e)

}

}

async addArchive(req:any, res:any, next:any){

try {

const {id} = req.body

const classData = await classService.addArchive(id)

return res.send(classData)

} catch (e) {

next(e)

}

}

async getClassStudent(req:any, res:any, next:any){

try {

const {user} = req.body

const classData = await classService.getClassStudent(user)

return res.send(classData)

} catch (e) {

next(e)

}

}

async getClassTeacher(req:any, res:any, next:any){

try {

const {user} = req.body

const classData = await classService.getClassTeacher(user)

return res.send(classData)

} catch (e) {

next(e)

}

}

}

export default new ClassController()

1.4.71. controller\complementary-controller.ts

import complementaryService from "../service/complementary-service"

class ComplementaryController{

async getOne(req:any, res:any, next:any){

try {

const {task\_id, user\_id} = req.body

const classData = await complementaryService.getOne(task\_id, user\_id)

return res.send(classData)

} catch (e) {

next(e)

}

}

async editStatusWithoutFiles(req:any, res:any, next:any){

try {

const {id, status} = req.body

const classData = await complementaryService.editStatusWithoutFiles(id, status)

return res.send(classData)

} catch (e) {

next(e)

}

}

}

export default new ComplementaryController()

1.4.72. controller\group-controller.ts

import groupService from "../service/group-service"

class GroupController{

async createGroup(req:any, res:any, next:any){

try {

const {title, user\_id} = req.body

const groupData = await groupService.createGroup(title, user\_id)

return res.send(groupData)

} catch (e) {

next(e)

}

}

async editGroup(req:any, res:any, next:any){

try {

const {title, id} = req.body

const groupData = await groupService.editGroup(title, id)

return res.send(groupData)

} catch (e) {

next(e)

}

}

async deleteGroup(req:any, res:any, next:any){

try {

const {id} = req.body

const groupData = await groupService.deleteGroup(id)

return res.send(groupData)

} catch (e) {

next(e)

}

}

async viewAllUsersInGroup(req:any, res:any, next:any){

try {

const {id} = req.body

const groupData = await groupService.viewUsersGroup(id)

return res.send(groupData)

} catch (e) {

next(e)

}

}

async getGroups(req:any, res:any, next:any){

try {

const {user\_id} = req.body

const groupData = await groupService.getGroups(user\_id)

return res.send(groupData)

} catch (e) {

next(e)

}

}

async viewFreeUsers(req:any, res:any, next:any){

try {

const {id} = req.body

const groupData = await groupService.viewFreeUsers(id)

return res.send(groupData)

} catch (e) {

next(e)

}

}

async addUserToGroup(req:any, res:any, next:any){

try {

const {user, group} = req.body

const groupData = await groupService.sendWelcomeLink(user, group)

return res.send(groupData)

} catch (e) {

next(e)

}

}

async deleteUserFromGroup(req:any, res:any, next:any){

try {

const {user, group} = req.body

const groupData = await groupService.deleteUserFromGroup(user, group)

return res.send(groupData)

//TODO: Autintification about delete

} catch (e) {

next(e)

}

}

async activateUserToGroup(req:any, res:any, next:any){

try {

const group = req.params.link

const user = req.params.id

await groupService.addUserToGroup(user, group)

return res.redirect(process.env.CLIENT\_URL + '/')

} catch (e) {

next(e)

}

}

}

export default new GroupController()

1.4.73. controller\member-controller.ts

import memberService from "../service/member-service"

class MemberController{

async getAllUsers(req:any, res:any, next:any){

try {

const {welcomeLink} = req.body

const userData = await memberService.getAllUsers(welcomeLink)

return res.status(200).send(userData)

} catch (e) {

next(e)

}

}

async checkRoleUser(req:any, res:any, next:any){

try {

const {user, welcomeLink} = req.body

const userData = await memberService.checkRoleUser(user, welcomeLink)

return res.send(userData)

} catch (e) {

next(e)

}

}

async sendActiveMail(req:any, res:any, next:any){

try {

const {user\_id, class\_id, role} = req.body

const memberData = await memberService.sendMail(user\_id, class\_id, role)

return res.send(memberData)

} catch (e) {

next(e)

}

}

async add(req:any, res:any, next:any){

try {

const welcomelink = req.params.link

const role = req.params.role

const user = req.params.id

await memberService.add(user, welcomelink, role)

return res.redirect(process.env.CLIENT\_URL + '/')

} catch (e) {

next(e)

}

}

async viewTeacher(req:any, res:any, next:any){

try {

const {class\_id} = req.body

const memberData = await memberService.viewTeacher(class\_id)

return res.send(memberData)

} catch (e) {

next(e)

}

}

async viewStudent(req:any, res:any, next:any){

try {

const {class\_id} = req.body

const memberData = await memberService.viewStudents(class\_id)

return res.send(memberData)

} catch (e) {

next(e)

}

}

async freeUsers(req:any, res:any, next:any){

try {

const {id} = req.body

const memberData = await memberService.viewFreeUsers(id)

return res.send(memberData)

} catch (e) {

next(e)

}

}

async addToClass(req:any, res:any, next:any){

try {

const {users, class\_id, role\_name} = req.body

const memberData = await memberService.addUserToClass(users, class\_id, role\_name)

return res.send(memberData)

} catch (e) {

next(e)

}

}

async deleteFromClass(req:any, res:any, next:any){

try {

const {user\_id, class\_id} = req.body

const memberData = await memberService.deleteUser(user\_id, class\_id)

return res.send(memberData)

} catch (e) {

next(e)

}

}

async addGroupToClass(req:any, res:any, next:any){

try {

const {group\_id, class\_id} = req.body

const memberData = await memberService.addGroup(group\_id, class\_id)

return res.send(memberData)

} catch (e) {

next(e)

}

}

}

export default new MemberController()

1.4.74. controller\task-controller.ts

import { uploadFiles } from "../middleware/upload-middleware"

import taskService from "../service/task-service"

class TaskController{

async createFiles(req:any, res:any, next:any){

try {

uploadFiles(req, res, async(err:any)=>{

if(err){

next(err)

}

else if(req.files)

{

const {deadlineDatetime, title, description, point, class\_id, isForm, user\_id} = req.body

const userData = await taskService.create(req.files, deadlineDatetime, title, description, point, class\_id, isForm, user\_id)

return res.send(userData)

}

else {

next(err)

}

})

} catch (e) {

next(e)

}

}

async viewTasks(req:any, res:any, next:any){

try {

const {class\_id} = req.body

const taskData = await taskService.viewTasks(class\_id)

res.send(taskData)

} catch (e) {

next(e)

}

}

async update(req:any, res:any, next:any){

try {

uploadFiles(req, res, async(err:any)=>{

if(err){

next(err)

}

else if(req.files)

{

const {id,oldFiles, deadlineDatetime, title, description, point, isForm} = req.body

const userData = await taskService.update(id, req.files, oldFiles, deadlineDatetime, title, description, point, isForm)

return res.send(userData)

}

else {

next(err)

}

})

} catch (e) {

next(e)

}

}

async delete(req:any, res:any, next:any){

try {

const {id} = req.body

const taskData = await taskService.delete(id)

res.send(taskData)

} catch (e) {

next(e)

}

}

async addTheme(req:any, res:any, next:any){

try {

const {id, theme\_id} = req.body

const taskData = await taskService.addTheme(id, theme\_id)

res.send(taskData)

} catch (e) {

next(e)

}

}

async deleteTheme(req:any, res:any, next:any){

try {

const {id} = req.body

const taskData = await taskService.deleteTheme(id)

res.send(taskData)

} catch (e) {

next(e)

}

}

async allTaskWithoutTheme(req:any, res:any, next:any){

try {

const {class\_id} = req.body

const taskData = await taskService.allTaskWithoutTheme(class\_id)

res.send(taskData)

} catch (e) {

next(e)

}

}

async allTaskWithTheme(req:any, res:any, next:any){

try {

const {class\_id} = req.body

const taskData = await taskService.allTaskWithTheme(class\_id)

res.send(taskData)

} catch (e) {

next(e)

}

}

async viewTaskOne(req:any, res:any, next:any){

try {

const {id, user\_id} = req.body

const taskData = await taskService.viewTaskOne(id, user\_id)

res.send(taskData)

} catch (e) {

next(e)

}

}

}

export default new TaskController()

1.4.72. controller\theme-controller.ts

import themeService from "../service/theme-service"

class ThemeController{

async create(req:any, res:any, next:any){

try {

const {title, class\_id} = req.body

const themeData = await themeService.create(title, class\_id)

res.send(themeData)

} catch (e) {

next(e)

}

}

async viewAll(req:any, res:any, next:any){

try {

const {class\_id} = req.body

const themeData = await themeService.viewAll(class\_id)

res.send(themeData)

} catch (e) {

next(e)

}

}

async update(req:any, res:any, next:any){

try {

const {id, title} = req.body

const themeData = await themeService.update(id, title)

res.send(themeData)

} catch (e) {

next(e)

}

}

async delete(req:any, res:any, next:any){

try {

const {id} = req.body

const themeData = await themeService.delete(id)

res.send(themeData)

} catch (e) {

next(e)

}

}

}

export default new ThemeController()

1.4.73. controller\user-controller.ts

import { validationResult } from 'express-validator'

import userService from '../service/user-service'

import {ApiError} from '../exception/api-error'

import { upload } from '../middleware/upload-middleware'

class UserController{

async editUserInfo(req:any, res:any, next:any){

try {

const {surname, name, lastname, email, id} = req.body

const userData = await userService.edit(id, email, surname, name, lastname)

res.cookie('refreshToken', userData.refreshToken, {maxAge: 30 \* 24 \* 60 \* 60 \* 1000, httpOnly: true})

return res.json(userData)

} catch (e) {

next(e)

}

}

async resetPassword(req:any, res:any, next:any){

try {

const errors = validationResult(req)

if(!errors.isEmpty()){

return next(ApiError.BadRequest('Ошибка валидации'))

}

const {email} = req.body

const userData = await userService.reset(email)

return res.json(userData)

} catch (e) {

next(e)

}

}

async activate\_password(req:any, res:any, next:any){

try {

const passwordLink = req.params.link;

await userService.activate\_password(passwordLink)

return res.redirect(process.env.CLIENT\_URL + `/changepassword/${passwordLink}`)

} catch (e) {

next(e)

}

}

async editAvatar(req:any, res:any, next:any){

try {

upload(req, res, async(err:any)=>{

if(err){

console.log(err)

}

else if(req.file)

{

const {email} = req.body

const userData = await userService.edit\_avatar(req.file.filename, email)

return res.json(userData)

}

else {

return res.json('Файл отсутствует')

}

})

} catch (e) {

next(e)

}

}

async changePassword(req:any, res:any, next:any){

try {

const {password, passwordLink} = req.body

const errors = validationResult(req)

if(!errors.isEmpty()){

console.log(errors)

return next(ApiError.BadRequest('Пароль должен содержать минимум 6 символов и максимум 32', Object(errors)))

}

const updatedUser = await userService.changePassword(password, passwordLink)

return res.json(updatedUser)

} catch (e) {

next(e)

}

}

async getUserInfo(req:any, res:any, next:any){

try {

const {id} = req.body

const userData = userService.getUserInfo(id)

return res.send(await userData)

} catch (e) {

next(e)

}

}

}

export default new UserController()

1.4.74. exception\api-error.ts

export class ApiError extends Error {

status : number;

errors: any[];

constructor(status:number, message:string, errors=[]){

super(message)

this.status = status

this.errors = errors

Object.setPrototypeOf(this, ApiError.prototype)

}

static UnauthorizedError(){

return new ApiError(401, 'Пользователь не авторизован')

}

static BadRequest(message:string, errors = []){

return new ApiError(400, message)

}

}

1.4.75. index.ts

import express from 'express'

const PORT = process.env.PORT || 5000

import cors from 'cors'

import \* as dotenv from 'dotenv';

const app = express()

import {router} from './router'

import errorMiddleware from './middleware/error-middleware'

import bodyParser from 'body-parser'

import prisma from './middleware/prisma-middleware';

app.use(bodyParser.json())

app.use(express.json())

app.use('/', router)

dotenv.config();

app.use(cors(

{

credentials: true,

origin: 'http://localhost:3000'

}

))

app.use(errorMiddleware)

app.use('/images', express.static('uploads/images'))

app.use('/files', express.static('uploads/files'))

const start = async()=>{

try {

app.listen(PORT, ()=>console.log('server work on 5000'))

await prisma.role.createMany({

data: [

{ name: 'student'},

{ name: 'teacher'},

],

skipDuplicates: true,

})

} catch (e) {

console.log(e)

}

}

start()

1.4.76. middleware\auth-middleware.ts

import {ApiError} from '../exception/api-error'

import tokenService from '../service/token-service'

export default function(req:any, res:any, next:any){

try {

const authorizationHeader = req.headers.authorization

if(!authorizationHeader){

return next(ApiError.UnauthorizedError())

}

const accessToken = authorizationHeader.split(' ')[1]

if(!accessToken){

return next(ApiError.UnauthorizedError())

}

const userData = tokenService.validateAccessToken(accessToken)

if(!userData){

return next(ApiError.UnauthorizedError())

}

req.user = userData

next()

} catch (e) {

return next(ApiError.UnauthorizedError())

}

}

1.4.77. middleware\error-middleware.ts

import {ApiError} from '../exception/api-error'

import express, { Request, Response, NextFunction } from 'express';

export default function errorMiddleware(err:any, req:Request, res:Response, next:Function){

if(err instanceof ApiError){

console.log(err)

return res.status(err.status).json({message: err.message, errors:err.errors})

}

return res.status(500).json({message:'Непридвиденная ошибка'})

}

1.4.78. middleware\prisma-middleware.ts

import {PrismaClient} from "@prisma/client"

const prisma = new PrismaClient()

export default prisma

1.4.79. middleware\upload-middleware.ts

import multer from 'multer'

import \* as uuid from 'uuid'

const Storage = multer.diskStorage({

destination:'uploads/images',

filename: (req, file, cb)=>{

cb(null, `${uuid.v4()}.${file.originalname.split('.').at(-1)}`)

},

})

const StorageFiles = multer.diskStorage({

destination:'uploads/files',

filename: (req, file, cb)=>{

cb(null, `${uuid.v4()}.${file.originalname.split('.').at(-1)}`)

},

})

const upload = multer({

storage: Storage

}).single('testImage')

const uploadFiles = multer({

storage: StorageFiles

}).array('files', 10)

export {upload, uploadFiles}

1.4.80. prisma\schema.prisma

// This is your Prisma schema file,

// learn more about it in the docs: https://pris.ly/d/prisma-schema

generator client {

provider = "prisma-client-js"

}

datasource db {

provider = "postgresql"

url = env("DATABASE\_URL")

}

model User{

id String @id @default(uuid()) @db.Uuid

surname String?

name String?

lastname String?

email String? @unique

password String?

isActived Boolean

activationLink String?

passwordLink String?

file String?

colorProfile String?

tokens Token[]

class Class[]

group Group[]

memberGroup MemberGroup[]

member Member[]

}

model File{

id String @id

name String

}

model Role{

id String @id @default(uuid()) @db.Uuid

name String @unique

member Member[]

}

model Token{

id String @id @default(uuid()) @db.Uuid

refreshToken String

user User? @relation(fields: [user\_id], references: [id])

user\_id String? @db.Uuid

}

model Class{

id String @id @default(uuid()) @db.Uuid

title String

chapter String?

subject String?

audience String?

cover String?

decor String?

isArchive Boolean

user User? @relation(fields: [user\_id], references: [id])

user\_id String? @db.Uuid

task Task[]

member Member[]

theme Theme[]

}

model Group{

id String @id @default(uuid()) @db.Uuid

title String

user User? @relation(fields: [user\_id], references: [id])

user\_id String? @db.Uuid

memberGroup MemberGroup[]

}

model MemberGroup{

id String @id @default(uuid()) @db.Uuid

user User? @relation(fields: [user\_id], references: [id])

group Group? @relation(fields: [group\_id], references: [id])

user\_id String? @db.Uuid

group\_id String? @db.Uuid

}

model Member{

id String @id @default(uuid()) @db.Uuid

user User? @relation(fields: [user\_id], references: [id])

class Class? @relation(fields: [class\_id], references: [id])

role Role? @relation(fields: [role\_id], references: [id])

user\_id String? @db.Uuid

class\_id String? @db.Uuid

role\_id String? @db.Uuid

task Task[]

taskComment TaskComment[]

complementary Complementary[]

complementaryComment ComplementaryComment[]

}

model Task{

id String @id @default(uuid()) @db.Uuid

files String[]

deadlineDatetime DateTime?

title String

description String?

date DateTime?

point Int @default(0)

isForm Boolean

class Class? @relation(fields: [class\_id], references: [id])

class\_id String? @db.Uuid

member Member? @relation(fields: [member\_id], references: [id])

member\_id String? @db.Uuid

theme Theme? @relation(fields: [theme\_id], references: [id])

theme\_id String? @db.Uuid

complementary Complementary[]

taskComment TaskComment[]

}

model Complementary{

id String @id @default(uuid()) @db.Uuid

file String[]

status Boolean @default(false)

mark Int?

datetime DateTime?

task Task? @relation(fields: [task\_id], references: [id])

task\_id String? @db.Uuid

member Member? @relation(fields: [member\_id], references: [id])

member\_id String? @db.Uuid

complementaryComment ComplementaryComment[]

}

model TaskComment{

id String @id @default(uuid()) @db.Uuid

text String?

datetime DateTime?

task Task? @relation(fields: [task\_id], references: [id])

task\_id String? @db.Uuid

member Member? @relation(fields: [member\_id], references: [id])

member\_id String? @db.Uuid

}

model ComplementaryComment{

id String @id @default(uuid()) @db.Uuid

text String?

datetime DateTime?

complementary Complementary? @relation(fields: [complementary\_id], references: [id])

complementary\_id String? @db.Uuid

member Member? @relation(fields: [member\_id], references: [id])

member\_id String? @db.Uuid

}

model Theme{

id String @id @default(uuid()) @db.Uuid

title String?

class Class? @relation(fields: [class\_id], references: [id])

class\_id String? @db.Uuid

task Task[]

}

1.4.81. router\auth-router.ts

import { Router } from 'express';

import AuthController from '../controller/auth-controller';

import cookieParser from 'cookie-parser'

import cors from 'cors'

import { body } from 'express-validator';

import authMiddleware from '../middleware/auth-middleware';

export const authRouter = Router();

authRouter.use(cookieParser())

authRouter.use(cors(

{

credentials: true,

origin: 'http://localhost:3000'

}

))

authRouter.post('/registration',body('password').isLength({min:6, max:32}),

AuthController.registration)

authRouter.get('/activate/:link', AuthController.activate)

authRouter.post('/login', AuthController.login)

authRouter.post('/logout', authMiddleware, AuthController.logout)

authRouter.get('/refresh', AuthController.refresh)

1.4.82. router\class-router.ts

import { Router } from 'express';

import classController from '../controller/class-controller';

export const classRouter = Router();

classRouter.post('/create', classController.createClass)

classRouter.post('/getAll', classController.getClass)

classRouter.post('/getOne', classController.getClassOne)

classRouter.post('/authorInfo', classController.getAuthorInfo)

classRouter.post('/editCover', classController.editCover)

classRouter.post('/edit', classController.updateClass)

classRouter.post('/addArchive', classController.addArchive)

classRouter.post('/getClassStudent', classController.getClassStudent)

classRouter.post('/getClassTeacher', classController.getClassTeacher)

1.4.83. router\complementary-router.ts

import { Router } from 'express';

import complementaryController from '../controller/complementary-controller';

export const complementaryRouter = Router();

complementaryRouter.post('/getOne', complementaryController.getOne)

complementaryRouter.post('/editStatus', complementaryController.editStatusWithoutFiles)

1.4.84. router\group-router.ts

import { Router } from 'express';

import groupController from '../controller/group-controller';

export const groupRouter = Router();

groupRouter.post('/create', groupController.createGroup)

groupRouter.post('/edit', groupController.editGroup)

groupRouter.post('/delete', groupController.deleteGroup)

groupRouter.post('/viewUsers', groupController.viewAllUsersInGroup)

groupRouter.post('/getAll', groupController.getGroups)

groupRouter.post('/viewFreeUsers', groupController.viewFreeUsers)

groupRouter.post('/addUserToGroup', groupController.addUserToGroup)

groupRouter.post('/deleteUserFromGroup', groupController.deleteUserFromGroup)

groupRouter.get('/:link/:id', groupController.activateUserToGroup)

1.4.85. router\member-router.ts

import { Router } from 'express';

import memberController from '../controller/member-controller';

export const memberRouter = Router();

memberRouter.post('/viewAll', memberController.getAllUsers)

memberRouter.post('/checkRole', memberController.checkRoleUser)

memberRouter.post('/sendMail', memberController.sendActiveMail)

memberRouter.get('/:link/:role/:id', memberController.add)

memberRouter.post('/viewTeacher', memberController.viewTeacher)

memberRouter.post('/viewStudent', memberController.viewStudent)

memberRouter.post('/viewFreeUsers', memberController.freeUsers)

memberRouter.post('/addUsersToClass', memberController.addToClass)

memberRouter.post('/deleteFromClass', memberController.deleteFromClass)

memberRouter.post('/addGroupToClass', memberController.addGroupToClass)

1.4.86. router\task-router.ts

import { Router } from 'express';

import taskController from '../controller/task-controller';

export const taskRouter = Router();

taskRouter.post('/create', taskController.createFiles)

taskRouter.post('/view', taskController.viewTasks)

taskRouter.post('/update', taskController.update)

taskRouter.post('/delete', taskController.delete)

taskRouter.post('/addTheme', taskController.addTheme)

taskRouter.post('/deleteTheme', taskController.deleteTheme)

taskRouter.post('/allTaskWithoutTheme', taskController.allTaskWithoutTheme)

taskRouter.post('/allTaskWithTheme', taskController.allTaskWithTheme)

taskRouter.post('/viewTaskOne', taskController.viewTaskOne)

1.4.87. router\theme-router.ts

import { Router } from 'express';

import themeController from '../controller/theme-controller';

export const themeRouter = Router();

themeRouter.post('/create', themeController.create)

themeRouter.post('/viewAll', themeController.viewAll)

themeRouter.post('/update', themeController.update)

themeRouter.post('/delete', themeController.delete)

1.4.88. router\user-router.ts

import { Router } from 'express';

import userController from '../controller/user-controller';

import cookieParser from 'cookie-parser'

import {body, validationResult} from 'express-validator'

export const userRouter = Router();

userRouter.use(cookieParser())

userRouter.post('/editProfile', userController.editUserInfo)

userRouter.post('/resetPassword', userController.resetPassword)

userRouter.get('/activatePassword/:link', userController.activate\_password)

userRouter.post('/editPhoto', userController.editAvatar)

userRouter.post('/changePassword',

body('password').isLength({min:6, max:32}),

userController.changePassword)

userRouter.post('/getUserInfo', userController.getUserInfo)

1.4.89. router.ts

import { Router } from 'express';

import { authRouter } from './router/auth-router';

import { userRouter } from './router/user-router';

import { classRouter } from './router/class-router';

import { memberRouter } from './router/member-router';

import { taskRouter } from './router/task-router';

import cors from 'cors'

import { groupRouter } from './router/group-router';

import { themeRouter } from './router/theme-router';

import { complementaryRouter } from './router/complementary-router';

export const router = Router();

router.use(cors(

{

credentials: true,

origin: 'http://localhost:3000'

}

))

router.use('/auth/', authRouter)

router.use('/user/', userRouter)

router.use('/class/', classRouter)

router.use('/member/', memberRouter)

router.use('/task/', taskRouter)

router.use('/group/', groupRouter)

router.use('/theme/', themeRouter)

router.use('/complementary/', complementaryRouter)

1.4.90. service\auth-service.ts

import bcrypt from 'bcrypt'

import \* as uuid from 'uuid'

import MailService from './mail-service'

import TokenService from './token-service'

import {ApiError} from '../exception/api-error'

import prisma from '../middleware/prisma-middleware'

class AuthService{

async registration(email:string, password:string, name:string, surname:string, lastname:string, color:string){

const userCandiet = await prisma.user.findFirst(

{where:{

email:email

},

select: {

surname: true,

name: true,

lastname:true,

email:true,

passwordLink:true,

activationLink:true,

isActived:true,

file:true,

id:true

},

}, )

if(userCandiet){

throw ApiError.BadRequest('Пользователь с таким email уже существует')

}

const hashPassword = await bcrypt.hash(password, 4)

const activationLink = uuid.v4();

if(!surname)

throw ApiError.BadRequest('Введите фамилию')

if(!name)

throw ApiError.BadRequest('Введите имя')

if(!email)

throw ApiError.BadRequest('Введите почту')

if(!password)

throw ApiError.BadRequest('Введите пароль')

await prisma.user.create({

data:{

surname:surname,

name:name,

lastname:lastname,

password:hashPassword,

activationLink:activationLink,

email:email,

isActived:false,

passwordLink:'',

file:'',

colorProfile:color

},

})

await MailService.sendActivationMail(email,`${process.env.API\_URL}/auth/activate/${activationLink}`)

const userData = await prisma.user.findFirst({

where:{

email:email

},

select: {

surname: true,

name: true,

lastname:true,

email:true,

passwordLink:true,

activationLink:true,

isActived:true,

file:true,

id:true

},

})

const user = userData

const tokens = TokenService.generateTokens({...userData})

if(userData){

if(tokens.refreshToken)

await TokenService.saveToken(userData.id, tokens.refreshToken)

}

return {

...tokens,

user

}

}

async activate(activationLink:string){

const user = await prisma.user.findFirst({where:{

activationLink:activationLink

}})

if(!user){

throw ApiError.BadRequest('Некорректная ссылка активации')

}

await prisma.user.update({

where:{

id:user.id

},

data:{

isActived:true,

},

select: {

surname: true,

name: true,

lastname:true,

email:true,

passwordLink:true,

activationLink:true,

isActived:true,

file:true,

id:true

},

})

}

async login(email:string, password:string){

let isPassEquals:any;

const userCandiet = await prisma.user.findFirst({where:{

email:email

}})

if(!userCandiet){

throw ApiError.BadRequest('Пользователь не найден')

}

if(userCandiet.password){

isPassEquals = await bcrypt.compare(password, userCandiet.password)

}

if(!isPassEquals)

throw ApiError.BadRequest('Неверный пароль')

const user = await prisma.user.findFirst({

where:{

email:email

},

select: {

surname: true,

name: true,

lastname:true,

email:true,

passwordLink:true,

activationLink:true,

isActived:true,

file:true,

id:true

},

})

const tokens = TokenService.generateTokens({...user})

if(user)

await TokenService.saveToken(user.id, tokens.refreshToken)

return {

...tokens,

user

}

}

async logout(refreshToken:string){

const token = await TokenService.removeToken(refreshToken)

return token

}

async refresh(refreshToken:string){

if(!refreshToken){

throw ApiError.UnauthorizedError()

}

const userData = TokenService.validateRefreshToken(refreshToken)

await TokenService.findToken(refreshToken)

const user = await prisma.user.findFirst({

where:{

id:Object(userData).id

},

select: {

surname: true,

name: true,

lastname:true,

email:true,

passwordLink:true,

activationLink:true,

isActived:true,

file:true,

id:true,

colorProfile:true

},

})

const tokens = TokenService.generateTokens({...user})

if(user)

await TokenService.saveToken(user.id, tokens.refreshToken)

return {

...tokens,

user

}

}

}

export default new AuthService()

1.4.91. service\class-service.ts

import \* as uuid from 'uuid'

import {ApiError} from '../exception/api-error'

import prisma from '../middleware/prisma-middleware';

import fs from 'fs'

import { promisify } from 'util'

const unlinkAsync = promisify(fs.unlink)

class ClassService{

async createClass(title:string, chapter:string, subject:string, audience:string, author:string, decor:string){

const userCandiet = await prisma.user.findFirst({

where:{

id:author

},

select: {

id:true,

email: true

},

})

if(!userCandiet){

throw ApiError.BadRequest('Пользователь с таким email не найден')

}

let dec='';

if(Object(decor).code)

dec=Object(decor).code

else

dec=decor

const classData = await prisma.class.create({

data:{

title:title,

chapter:chapter,

subject:subject,

audience:audience,

user\_id:userCandiet.id,

cover:'',

decor: dec,

isArchive:false

},

})

const role = await prisma.role.findFirst({

where:{

name:'teacher'

}

})

await prisma.member.create({

data:{

class\_id:classData.id,

role\_id:role?.id,

user\_id: userCandiet.id

},

})

return classData

}

async getClass(user:string){

const userCandiet = await prisma.user.findFirst({

where:{

id:user,

},

select: {

surname: true,

name: true,

lastname:true,

email:true,

passwordLink:true,

activationLink:true,

isActived:true,

file:true,

id:true

},

})

if(!userCandiet){

throw ApiError.BadRequest('Пользователь с таким email не найден')

}

const memberData = await prisma.member.findMany({

where: {

user\_id: user,

},

include: {

user: true,

class: {

include:{

user:true

}

},

},

});

const classData = memberData.map(member => {

if (member.class && !member.class.isArchive) {

return member.class;

}

return null;

}).filter(Boolean);

if (classData == null) return []

if (classData instanceof Array) return classData

return [classData]

}

async getClassOne(welcomelink:string, user:string){

const userCandiet = await prisma.user.findFirst({

where:{

id:user

},

select: {

surname: true,

name: true,

lastname:true,

email:true,

passwordLink:true,

activationLink:true,

isActived:true,

file:true,

id:true

},

})

if(!userCandiet){

throw ApiError.BadRequest('Пользователь с таким email не найден')

}

const classCandiet = await prisma.class.findFirst({

where:{

id:welcomelink

}

})

if(!classCandiet){

throw ApiError.BadRequest('Курс не найден')

}

const member = await prisma.member.findMany({

where:{

user\_id:user,

class\_id:welcomelink

},

include: {

role: true,

class:{

include:{

user:{

select:{

surname: true,

name: true,

lastname:true,

email:true,

passwordLink:true,

activationLink:true,

isActived:true,

file:true,

id:true

}

}

}

}

},

})

return {

member

}

}

async getAuthorInfo(user:string){

const userCandiet = await prisma.class.findFirst({

where:{

user\_id:user

}

})

if(!userCandiet){

throw ApiError.BadRequest('Пользователь с таким email не найден')

}

const author = await prisma.user.findFirst({

where:{

id:user

},

select:{

email:true,

surname:true,

name:true,

lastname:true,

file:true

}

})

return author

}

async edit\_avatar(cover:string, welcomelink:string){

const classCandiet = await prisma.class.findFirst({

where:{

id:welcomelink

}

})

if(!classCandiet){

throw ApiError.BadRequest('Пользователь с таким email не найден')

}

if(classCandiet.cover){

fs.access('uploads\\images\\'+classCandiet.cover, fs.constants.F\_OK, async(err) => {

if (!err) {

await unlinkAsync('uploads\\images\\'+classCandiet.cover)

}

});

}

const data = await prisma.class.update({

where:{

id:welcomelink

},

data: {

cover:cover

},

})

return data

}

async updateClass(title:string, chapter:string, subject:string, audience:string, welcomelink:string, decor:string){

const data = await prisma.class.update({

where:{

id:welcomelink

},

data: {

title:title,

chapter:chapter,

subject:subject,

audience:audience,

decor:decor

},

}).catch(()=>{

throw ApiError.BadRequest('Ошибка с заполнением данных')

})

return data

}

async addArchive(id:string){

const data = await prisma.class.update({

where:{

id:id

},

data:{

isArchive:true

}

}).catch(()=>{

throw ApiError.BadRequest('Ошибка с заполнением данных')

})

return data

}

async getClassStudent(user:string){

const userCandiet = await prisma.user.findFirst({

where:{

id:user,

},

select: {

surname: true,

name: true,

lastname:true,

email:true,

passwordLink:true,

activationLink:true,

isActived:true,

file:true,

id:true

},

})

if(!userCandiet){

throw ApiError.BadRequest('Пользователь с таким email не найден')

}

const memberData = await prisma.member.findMany({

where: {

user\_id: user,

},

include: {

user: true,

class: {

include:{

user:true

}

},

},

});

const classData = memberData

.filter(member => member.class && !member.class.isArchive && member.class.user && member.class.user.id !== user)

.map(member => member.class);

if (classData == null) return []

if (classData instanceof Array) return classData

return [classData]

}

async getClassTeacher(user:string){

const userCandiet = await prisma.user.findFirst({

where:{

id:user,

},

select: {

surname: true,

name: true,

lastname:true,

email:true,

passwordLink:true,

activationLink:true,

isActived:true,

file:true,

id:true

},

})

if(!userCandiet){

throw ApiError.BadRequest('Пользователь с таким email не найден')

}

const classData = await prisma.class.findMany({

where: {

user\_id: user,

isArchive:false

},

include: {

user: true,

},

});

if (classData == null) return []

if (classData instanceof Array) return classData

return [classData]

}

}

export default new ClassService()

1.4.92. service\complementary-service.ts

import { ApiError } from "../exception/api-error"

import prisma from "../middleware/prisma-middleware"

import fileService from "./file-service"

class ComplementaryService{

async getOne(task\_id:string, user\_id:string){

const task = await prisma.task.findFirst({

where:{

id:task\_id

},

include:{

class:true

}

})

if(!task)

throw ApiError.BadRequest('Задание не найдено')

const member = await prisma.member.findFirst({

where:{

user\_id:user\_id,

class\_id:task.class?.id

}

})

if(!member)

throw ApiError.BadRequest('Участник не найден')

const complementary = await prisma.complementary.findFirst({

where:{

member\_id: member.id,

task\_id:task\_id

}

})

if(complementary?.file){

const files = await fileService.viewFiles(complementary?.file)

complementary.file = files;

}

return complementary

}

async editStatusWithoutFiles(id:string, status:string){

const complementary = await prisma.complementary.update({

where:{

id:id

},

data:{

status:Boolean(!status)

}

})

return complementary

}

}

export default new ComplementaryService()

1.4.93. service\file-service.ts

import prisma from "../middleware/prisma-middleware"

class FileService{

async viewFiles(files:any){

const filesArray = await Promise.all(files.map(async(file:any)=>{

const fileData = await prisma.file.findMany({

where:{

id:file

}

})

return fileData

}))

if (filesArray == null) return []

if (filesArray instanceof Array) return filesArray

return [filesArray]

}

}

export default new FileService()

1.4.94. service\group-service.ts

import { ApiError } from "../exception/api-error"

import prisma from "../middleware/prisma-middleware"

import mailService from "./mail-service"

class GroupService{

async createGroup(title:string, user\_id:string){

const userCandiet = await prisma.user.findFirst({

where:{

id:user\_id

},

select: {

id:true,

email: true

},

})

if(!userCandiet){

throw ApiError.BadRequest('Пользователь с таким email не найден')

}

const group = await prisma.group.create({

data:{

title:title,

user\_id:user\_id

}

}).catch((err)=>{

console.log(err)

throw ApiError.BadRequest('Ошибка создания')

})

return group

}

async editGroup(title:string, id:string){

const userCandiet = await prisma.group.findFirst({

where:{

id:id

},

})

if(!userCandiet){

throw ApiError.BadRequest('Группа не найдена')

}

const group = await prisma.group.update({

where:{

id:id

},

data:{

title:title

}

}).catch((err)=>{

console.log(err)

throw ApiError.BadRequest('Ошибка изменения')

})

return group

}

//TODO: check title for unique

async deleteGroup(id:string){

const userCandiet = await prisma.group.findFirst({

where:{

id:id

},

})

if(!userCandiet){

throw ApiError.BadRequest('Группа не найдена')

}

const group = await prisma.group.delete({

where:{

id:id

}

}).catch((err)=>{

console.log(err)

throw ApiError.BadRequest('Ошибка изменения')

})

return group

}

async getGroups(user\_id:string){

const userCandiet = await prisma.user.findFirst({

where:{

id:user\_id

},

select: {

id:true,

email: true

},

})

if(!userCandiet){

throw ApiError.BadRequest('Пользователь с таким email не найден')

}

const group = await prisma.group.findMany({

where:{

user\_id:user\_id

}

})

if (group == null) return []

if (group instanceof Array) return group

return [group]

}

async viewFreeUsers(id:string){

const groupCandiet = await prisma.group.findFirst({

where:{

id:id

}

})

if(!groupCandiet){

throw ApiError.BadRequest('Группа не найдена')

}

const memberCandiet = await prisma.memberGroup.findMany({

where:{

group\_id:id

}

})

const userIds = memberCandiet.map((candidate) => candidate.user\_id);

const filteredUserIds = userIds.filter(id => id !== null) as string[];

const groupCandidateUserIds = groupCandiet?.user\_id ? [groupCandiet.user\_id] : [];

const freeUsers = await prisma.user.findMany({

where: {

isActived:true,

id: {

notIn: [...filteredUserIds, ...groupCandidateUserIds],

},

},

select:{

email:true,

surname:true,

name:true,

lastname:true,

file:true,

colorProfile:true,

id:true

}

});

if (freeUsers == null) return []

if (freeUsers instanceof Array) return freeUsers

return [freeUsers]

}

async sendWelcomeLink(user:string, group:string){

const groupCandiet = await prisma.group.findFirst({

where:{

id:group

}

})

if(!groupCandiet){

throw ApiError.BadRequest('Группа не найдена')

}

const userCandiet = await prisma.user.findFirst({

where:{

id:user

},

select: {

id:true,

email: true

},

})

if(!userCandiet){

throw ApiError.BadRequest('Пользователь с таким email не найден')

}

const memberCandiet = await prisma.memberGroup.findFirst({

where:{

user\_id:user,

group\_id:group

},

})

if(memberCandiet){

throw ApiError.BadRequest('Пользователь уже состоит в группе')

}

if(userCandiet.email)

await mailService.sendGroupActivationWelcomeLink(userCandiet?.email,`${process.env.API\_URL}/group/${group}/${user}`, groupCandiet.title)

return 'done'

}

async addUserToGroup(user:string, group:string){

const groupCandiet = await prisma.group.findFirst({

where:{

id:group

}

})

if(!groupCandiet){

throw ApiError.BadRequest('Группа не найдена')

}

const userCandiet = await prisma.user.findFirst({

where:{

id:user

},

select: {

id:true,

email: true

},

})

if(!userCandiet){

throw ApiError.BadRequest('Пользователь с таким email не найден')

}

const memberCandiet = await prisma.memberGroup.findFirst({

where:{

user\_id:user,

group\_id:group

},

select: {

id:true,

},

})

if(memberCandiet){

throw ApiError.BadRequest('Пользователь уже состоит в группе')

}

const memberGroup = await prisma.memberGroup.create({

data:{

user\_id:user,

group\_id: group

}

})

return memberGroup

}

async deleteUserFromGroup(user:string, group:string){

const groupCandiet = await prisma.group.findFirst({

where:{

id:group

}

})

if(!groupCandiet){

throw ApiError.BadRequest('Группа не найдена')

}

const userCandiet = await prisma.user.findFirst({

where:{

id:user

},

select: {

id:true,

email: true

},

})

if(!userCandiet){

throw ApiError.BadRequest('Пользователь с таким email не найден')

}

const memberCandiet = await prisma.memberGroup.findFirst({

where:{

user\_id:user,

group\_id:group

},

select: {

id:true,

},

})

if(!userCandiet){

throw ApiError.BadRequest('Пользователь с таким email не найден')

}

const memberGroup = await prisma.memberGroup.delete({

where:{

id:memberCandiet?.id

}

})

return memberGroup

}

async viewUsersGroup(id:string){

const groupCandiet = await prisma.group.findFirst({

where:{

id:id

}

})

if(!groupCandiet){

throw ApiError.BadRequest('Группа не найдена')

}

const members = await prisma.memberGroup.findMany({

where:{

group\_id:id

},

include:{

user:true

}

})

if (members == null) return []

if (members instanceof Array) return members

return [members]

}

}

export default new GroupService()

1.4.95. service\mail-service.ts

import \* as nodemailer from 'nodemailer'

import \* as dotenv from 'dotenv';

dotenv.config();

class MailService {

private transporter: nodemailer.Transporter;

constructor(){

this.transporter = nodemailer.createTransport({

host: 'smtp.gmail.com',

port: 465,

secure: true,

auth:{

user: process.env.SMTP\_USER,

pass: process.env.SMTP\_PASSWORD

}

})

}

async sendActivationMail(to:any, link:any){

await this.transporter.sendMail({

from: process.env.SMTP\_USER,

to,

subject: 'Активация аккаунта на ' + process.env.API\_URL,

text: '',

html:

`<div>

<img src="https://psv4.userapi.com/c236331/u405955725/docs/d20/a664b160e457/Frame\_64.png?extra=pcaUU4pSu4jmKl70vWHFtZSLSXU1blT9s-5JTBs4QSt5a8J\_tlHPFgPByRggFeooJyidXPOe1-j2pXEFdeF0cyWnb5lQWT-e9oU5Ajeez8YBDdOCrMo5elsLfhGej2qnMsHrSkkco6beEH987gjygVY3TA"

alt = "logo">

<center>

<h1>Для активации аккаунта нажмите на кнопку: </h1>

<a href="${link}">

<button>Активировать аккаунт</button>

</a>

</center>

</div>

<style>

button {

background: #335DFF;

border-radius: 7px;

border: none;

padding: 15px;

color:white;

font-family: 'Source Sans Pro';

font-style: normal;

font-weight: 600;

font-size: 18px;

}

button:active {

background: #526adf;

}

</style>

`

})

}

async sendActivationPasswordMail(to:string, link:string){

await this.transporter.sendMail({

from: process.env.SMTP\_USER,

to,

subject: 'Смена пароля на ' + process.env.API\_URL,

text: '',

html:

`<div>

<img src="https://psv4.userapi.com/c236331/u405955725/docs/d20/a664b160e457/Frame\_64.png?extra=pcaUU4pSu4jmKl70vWHFtZSLSXU1blT9s-5JTBs4QSt5a8J\_tlHPFgPByRggFeooJyidXPOe1-j2pXEFdeF0cyWnb5lQWT-e9oU5Ajeez8YBDdOCrMo5elsLfhGej2qnMsHrSkkco6beEH987gjygVY3TA"

alt = "logo">

<center>

<h1>Для смены пароля нажмите на кнопку: </h1>

<a href="${link}">

<button>Сменить пароль</button>

</a>

</center>

</div>

<style>

button {

background: #335DFF;

border-radius: 7px;

border: none;

padding: 15px;

color:white;

font-family: 'Source Sans Pro';

font-style: normal;

font-weight: 600;

font-size: 18px;

}

button:active {

background: #526adf;

}

</style>

`

})

}

async sendActivationWelcomeLink(to:string, link:string, title:string){

await this.transporter.sendMail({

from: process.env.SMTP\_USER,

to,

subject: `Приглашение на курс ${title} на ${process.env.API\_URL}`,

text: '',

html:

`<div>

<img src="https://psv4.userapi.com/c236331/u405955725/docs/d20/a664b160e457/Frame\_64.png?extra=pcaUU4pSu4jmKl70vWHFtZSLSXU1blT9s-5JTBs4QSt5a8J\_tlHPFgPByRggFeooJyidXPOe1-j2pXEFdeF0cyWnb5lQWT-e9oU5Ajeez8YBDdOCrMo5elsLfhGej2qnMsHrSkkco6beEH987gjygVY3TA"

alt = "logo">

<center>

<h1>Вас пригласили на курс ${title} на ${process.env.API\_URL}</h1>

<p>Чтобы вступить нажмите на кнопку:</p>

<a href="${link}">

<button>Вступить на курс</button>

</a>

</center>

</div>

<style>

button {

background: #335DFF;

border-radius: 7px;

border: none;

padding: 15px;

color:white;

font-family: 'Source Sans Pro';

font-style: normal;

font-weight: 600;

font-size: 18px;

}

button:active {

background: #526adf;

}

</style>

`

})

}

async sendGroupActivationWelcomeLink(to:string, link:string, title:string){

await this.transporter.sendMail({

from: process.env.SMTP\_USER,

to,

subject: `Приглашение в группу ${title} на ${process.env.API\_URL}`,

text: '',

html:

`<div>

<img src="https://psv4.userapi.com/c236331/u405955725/docs/d20/a664b160e457/Frame\_64.png?extra=pcaUU4pSu4jmKl70vWHFtZSLSXU1blT9s-5JTBs4QSt5a8J\_tlHPFgPByRggFeooJyidXPOe1-j2pXEFdeF0cyWnb5lQWT-e9oU5Ajeez8YBDdOCrMo5elsLfhGej2qnMsHrSkkco6beEH987gjygVY3TA"

alt = "logo">

<center>

<h1>Вас пригласили в группу ${title} на ${process.env.API\_URL}</h1>

<p>Чтобы вступить нажмите на кнопку:</p>

<a href="${link}">

<button>Вступить в группу</button>

</a>

</center>

</div>

<style>

button {

background: #335DFF;

border-radius: 7px;

border: none;

padding: 15px;

color:white;

font-family: 'Source Sans Pro';

font-style: normal;

font-weight: 600;

font-size: 18px;

}

button:active {

background: #526adf;

}

</style>

`

})

}

}

export default new MailService()

1.4.96. service\member-service.ts

import { ApiError } from "../exception/api-error"

import prisma from "../middleware/prisma-middleware"

import mailService from "./mail-service"

class MemberService{

async getAllUsers(welcomeLink:string){

const classData = await prisma.member.findMany({

where:{

class\_id:welcomeLink

},

select:{

user\_id:true

}

}).catch(()=>{

throw ApiError.BadRequest('Курс не найден')

})

const freeUsers = await prisma.user.findMany({

where:{

isActived:true,

NOT:{

id:{in:Object(classData).user\_id}

}

},

select: {

surname: true,

name: true,

lastname:true,

email:true,

passwordLink:true,

activationLink:true,

isActived:true,

file:true,

id:true

},

}).catch(()=>{

throw ApiError.BadRequest('Ошибка вывода')

})

if (freeUsers == null) return []

if (freeUsers instanceof Array) return freeUsers

return [freeUsers]

}

async checkRoleUser(user:string, welcomeLink:string){

const role = await prisma.member.findMany({

where:{

user\_id:user,

class\_id:welcomeLink

},

include:{

role:true

}

}).catch(()=>{

throw ApiError.BadRequest('Ошибка вывода')

})

if (role == null) return []

if (role instanceof Array) return role

return [role]

}

async sendMail(user\_id:string, class\_id:string, roleTitle:string){

const role = await prisma.role.findFirst({

where:{

name:roleTitle

}

}).catch(()=>{

throw ApiError.BadRequest('Роль не найдена')

})

const user = await prisma.user.findFirst({

where:{

id:user\_id

}

}).catch(()=>{

throw ApiError.BadRequest('Пользователь не найден')

})

const classData = await prisma.class.findFirst({

where:{

id:class\_id

}

}).catch(()=>{

throw ApiError.BadRequest('Курс не найден')

})

if(user?.email && classData?.id && role?.id && user.id && classData.title){

await mailService.sendActivationWelcomeLink(user?.email,`${process.env.API\_URL}/member/${classData?.id}/${role.id}/${user.id}`, classData.title)

}

return 'Done'

}

async add(user\_id:string, class\_id:string, role\_id:string){

const member = await prisma.member.findFirst({

where:{

user\_id:user\_id,

class\_id:class\_id

}

})

if(member){

throw ApiError.BadRequest('Пользователь уже на курсе')

}

const memberData = await prisma.member.create({

data:{

role\_id:role\_id,

user\_id:user\_id,

class\_id:class\_id

}

}).catch(()=>{

throw ApiError.BadRequest('Ошибка создания')

})

const role = await prisma.role.findFirst({

where:{

id:role\_id

}

})

if(role?.name==="student"){

const tasks = await prisma.task.findMany({

where:{

class\_id:class\_id

}

})

tasks.map(async(task)=>{

const complementaryData = await prisma.complementary.findFirst({

where:{

task\_id:task.id,

member\_id:user\_id

}

})

if(!complementaryData){

await prisma.complementary.create({

data:{

task\_id:task.id,

member\_id:memberData.id

}

})

}

})

}

return memberData

}

async viewTeacher(class\_id:string){

const role = await prisma.role.findFirst({

where:{

name:'teacher'

}

}).catch(()=>{

throw ApiError.BadRequest('Роль не найдена')

})

await prisma.member.findMany({

where:{

class\_id:class\_id

}

}).catch(()=>{

throw ApiError.BadRequest('Курс не найден')

})

const member = await prisma.member.findMany({

where:{

class\_id:class\_id,

role\_id:String(role?.id)

},

include:{

class:true,

user:{

select:{

email:true,

surname:true,

name:true,

lastname:true,

file:true,

colorProfile:true,

id:true

}

}

}

})

if (member == null) return []

if (member instanceof Array) return member

return [member]

}

async viewStudents(class\_id:string){

const role = await prisma.role.findFirst({

where:{

name:'student'

}

}).catch(()=>{

throw ApiError.BadRequest('Роль не найдена')

})

await prisma.member.findMany({

where:{

class\_id:class\_id

}

}).catch(()=>{

throw ApiError.BadRequest('Курс не найден')

})

const member = await prisma.member.findMany({

where:{

class\_id:class\_id,

role\_id:String(role?.id)

},

include:{

class:true,

user:{

select:{

email:true,

surname:true,

name:true,

lastname:true,

file:true,

colorProfile:true,

id:true

}

}

}

})

if (member == null) return []

if (member instanceof Array) return member

return [member]

}

async viewFreeUsers(id:string){

const groupCandiet = await prisma.class.findFirst({

where:{

id:id

}

})

if(!groupCandiet){

throw ApiError.BadRequest('Группа не найдена')

}

const memberCandiet = await prisma.member.findMany({

where:{

class\_id:id

}

})

const userIds = memberCandiet.map((candidate) => candidate.user\_id);

const filteredUserIds = userIds.filter(id => id !== null) as string[];

const freeUsers = await prisma.user.findMany({

where: {

isActived:true,

id: {

notIn: filteredUserIds,

},

},

select:{

email:true,

surname:true,

name:true,

lastname:true,

file:true,

colorProfile:true,

id:true

}

});

if (freeUsers == null) return []

if (freeUsers instanceof Array) return freeUsers

return [freeUsers]

}

async addUserToClass(users:any, class\_id:any, role\_name:string){

users.map(async(user:any)=>{

const role = await prisma.role.findFirst({

where:{

name:role\_name

}

}).catch(()=>{

throw ApiError.BadRequest('Роль не найдена')

})

const userData = await prisma.user.findFirst({

where:{

email:user

}

}).catch(()=>{

throw ApiError.BadRequest('Пользователь не найден')

})

const classData = await prisma.class.findFirst({

where:{

id:class\_id

}

}).catch(()=>{

throw ApiError.BadRequest('Курс не найден')

})

if(userData?.email && classData?.id && role?.id && userData.id && classData.title){

await mailService.sendActivationWelcomeLink(userData?.email,`${process.env.API\_URL}/member/${classData?.id}/${role.id}/${userData.id}`, classData.title)

}

})

return 'Done'

}

async deleteUser(user\_id:string, class\_id:string){

const classData = await prisma.class.findFirst({

where:{

id:class\_id

}

})

if(!classData)

throw ApiError.BadRequest('Курс не найден')

const userData = await prisma.user.findFirst({

where:{

id:user\_id

}

})

if(!userData)

throw ApiError.BadRequest('Пользователь не найден')

const memberData = await prisma.member.findFirst({

where:{

user\_id:user\_id,

class\_id:class\_id

}

})

if(!memberData)

throw ApiError.BadRequest('Пользователь не найден')

const member = await prisma.member.delete({

where:{

id:memberData.id

}

})

return member

}

async addGroup(group\_id:string, class\_id:string){

const group = await prisma.group.findFirst({

where:{

id:group\_id

}

})

if(!group)

throw ApiError.BadRequest('Группа не найдена')

const memberGroup = await prisma.memberGroup.findMany({

where:{

group\_id:group\_id

}

})

if(!memberGroup)

throw ApiError.BadRequest('Участник группы не найден')

memberGroup.map(async(mmbr)=>{

const role = await prisma.role.findFirst({

where:{

name:'student'

}

}).catch(()=>{

throw ApiError.BadRequest('Роль не найдена')

})

const userData = await prisma.user.findFirst({

where:{

id:String(mmbr.user\_id)

}

}).catch(()=>{

throw ApiError.BadRequest('Пользователь не найден')

})

const classData = await prisma.class.findFirst({

where:{

id:class\_id

}

}).catch(()=>{

throw ApiError.BadRequest('Курс не найден')

})

const findMember = await prisma.member.findFirst({

where:{

user\_id:mmbr.user\_id,

class\_id:class\_id

}

})

if(!findMember){

if(userData?.email && classData?.id && role?.id && userData.id && classData.title){

await mailService.sendActivationWelcomeLink(userData?.email,`${process.env.API\_URL}/member/${classData?.id}/${role.id}/${userData.id}`, classData.title)

}

}

})

return 'Done'

}

}

export default new MemberService()

1.4.97. service\task-service.ts

import { ApiError } from "../exception/api-error"

import prisma from "../middleware/prisma-middleware"

import fs from 'fs'

import { promisify } from 'util'

import transliteration from 'transliteration'

import fileService from "./file-service"

const unlinkAsync = promisify(fs.unlink)

class TaskService{

async create(files:any, deadlineDatetime:string, title:string, description:string, point: string | undefined = undefined, class\_id:string, isForm:boolean, user\_id:string){

const date = new Date()

const isoDate = date.toISOString();

if(deadlineDatetime!=='null'){

const dateDeadline = new Date(deadlineDatetime)

deadlineDatetime = dateDeadline.toISOString()

}

else{

const dateDeadline = new Date(0)

deadlineDatetime = dateDeadline.toISOString()

}

if (!point) {

point = '0';

}

const pointNumber: number = parseInt(point) || 0;

const filesArray = await Promise.all(files.map(async(file:any)=>{

const decodedText: string = Buffer.from(file.originalname, 'binary').toString('utf-8');

const fileData = await prisma.file.create({

data:{

id:file.filename,

name: decodedText

}

})

return fileData.id

}))

const memberData = await prisma.member.findFirst({

where:{

class\_id: class\_id,

user\_id:user\_id

}

})

const task = await prisma.task.create({

data:{

deadlineDatetime: deadlineDatetime,

title:title,

description:description,

date:isoDate,

point:Number(pointNumber),

class\_id:class\_id,

files:filesArray,

isForm:Boolean(isForm),

member\_id:memberData?.id

}

})

const role = await prisma.role.findFirst({

where:{

name:'student'

},

select: {

id:true,

},

})

const member = await prisma.member.findMany({

where:{

class\_id:class\_id,

role\_id:role?.id

}

})

if(Object(member).id){

await prisma.complementary.create({

data:{

task\_id:task.id,

member\_id:Object(member).id

}

})

}

else{

Object(member).map(async(data:any)=>{

await prisma.complementary.create({

data:{

task\_id:task.id,

member\_id:data.id

}

})

})

}

return task

}

async viewTasks(class\_id:string){

const tasks = await prisma.task.findMany({

where:{

class\_id: class\_id

},

include:{

member: {

include:{

user:{

select:{

email:true,

surname:true,

name:true,

lastname:true,

file:true,

colorProfile:true,

id:true

}

}

}

},

class:true,

},

orderBy: {

date: 'desc', // Сортировка по полю даты в порядке убывания (от новых к старым)

},

}).catch(()=>{

throw ApiError.BadRequest('Ошибка вывода')

})

if (tasks == null) return []

if (tasks instanceof Array) return tasks

return [tasks]

}

async update(id:string, files:any, oldFiles:string[], deadlineDatetime:string, title:string, description:string, point: string | undefined = undefined, isForm:boolean){

const oldTask = await prisma.task.findFirst({

where:{

id:id

},

select:{

files:true

}

})

// console.log(files)

if(oldTask){

await Promise.all(oldTask.files.map(async(file:any)=>{

if (Array.isArray(oldFiles)) {

oldFiles.map((old:string)=>{

if(old!==file){

file = files.filter((fileName: string[]) => fileName !== oldFiles);

}

})

}else{

if(oldFiles!==file){

file = files.filter((fileName: string[]) => fileName !== oldFiles);

}

}

if(file.length!==0){

fs.access('uploads\\files\\'+file, fs.constants.F\_OK, async(err) => {

if (!err) {

await unlinkAsync('uploads\\files\\'+file)

}

});

await prisma.file.delete({

where:{

id:file

}

}).catch(()=>{

console.log('Уже удалено')

})

}

}))

}

if(deadlineDatetime!=='null'){

const dateDeadline = new Date(deadlineDatetime)

deadlineDatetime = dateDeadline.toISOString()

}

else{

const dateDeadline = new Date(0)

deadlineDatetime = dateDeadline.toISOString()

}

if (!point) {

point = '0';

}

const pointNumber: number = parseInt(point) || 0;

const filesArray = await Promise.all(files.map(async(file:any)=>{

const decodedText: string = Buffer.from(file.originalname, 'binary').toString('utf-8');

const fileData = await prisma.file.create({

data:{

id:file.filename,

name: decodedText

}

})

return fileData.id

}))

if (Array.isArray(oldFiles)) {

oldFiles.map((old:string)=>{

filesArray.push(old)

})

}else{

filesArray.push(oldFiles)

} const task = await prisma.task.update({

where:{

id:id

},

data:{

deadlineDatetime: deadlineDatetime,

title:title,

description:description,

point:Number(pointNumber),

files:filesArray,

isForm:Boolean(isForm),

}

})

return task

}

async delete(id:string){

const oldTask = await prisma.task.findFirst({

where:{

id:id

},

select:{

files:true

}

})

if(oldTask){

await Promise.all(oldTask.files.map(async(file:any)=>{

if(file){

fs.access('uploads\\files\\'+file, fs.constants.F\_OK, async(err) => {

if (!err) {

await unlinkAsync('uploads\\files\\'+file)

}

});

await prisma.file.delete({

where:{

id:file

}

}).catch(()=>{

console.log('Уже удалено')

})

}

}))

}

const task = await prisma.task.delete({

where:{

id:id

}

})

return task

}

async addTheme(id:string, theme\_id:string){

const themeData = await prisma.theme.findFirst(({

where:{

id:theme\_id

}

}))

if(!themeData)

throw ApiError.BadRequest('Тема не найдена')

const task = await prisma.task.update({

where:{

id:id

},

data:{

theme\_id:theme\_id

}

}).catch(()=>{

throw ApiError.BadRequest('Ошибка добавления')

})

return task

}

async deleteTheme(id:string){

const task = await prisma.task.update({

where:{

id:id

},

data:{

theme\_id:null

}

}).catch(()=>{

throw ApiError.BadRequest('Ошибка добавления')

})

return task

}

async allTaskWithoutTheme(class\_id:string){

const task = await prisma.task.findMany({

where:{

class\_id:class\_id,

theme\_id:null

},

orderBy:{

date:'asc'

},

include:{

member:{

include:{

user:{

select:{

email:true,

surname:true,

name:true,

lastname:true,

file:true,

colorProfile:true,

id:true

}

}

}

},

class:true

}

})

return task

}

async allTaskWithTheme(class\_id:string){

const task = await prisma.theme.findMany({

where:{

class\_id:class\_id

},

include:{

task:{

orderBy: {

date:'asc'

},

include:{

class:true,

member:{

include:{

user:{

select:{

email:true,

surname:true,

name:true,

lastname:true,

file:true,

colorProfile:true,

id:true

}

}

}

}

}

},

class:true

}

})

return task

}

async viewTaskOne(id:string, user\_id:string){

const taskData = await prisma.task.findFirst({

where:{

id:id

}

})

if(!taskData)

throw ApiError.BadRequest('Задание не найдено')

const memberData = await prisma.member.findFirst({

where:{

class\_id:taskData.class\_id,

user\_id:user\_id

},

include:{

role:true,

class:{

select:{

title:true,

chapter:true,

subject:true,

audience:true,

cover:true,

decor:true,

isArchive:true,

user:{

select:{

email:true,

surname:true,

name:true,

lastname:true,

file:true,

colorProfile:true,

id:true

}

},

task:{

where: {

id: id

},

include:{

member:{

include:{

user:{

select:{

email:true,

surname:true,

name:true,

lastname:true,

file:true,

colorProfile:true,

id:true

}

}

}

}

}

}

}

},

user:{

select:{

email:true,

surname:true,

name:true,

lastname:true,

file:true,

colorProfile:true,

id:true

}

}

}

})

const files = await fileService.viewFiles(memberData?.class?.task[0].files)

if(memberData?.class)

memberData.class.task[0].files = files;

return memberData

}

}

export default new TaskService()

1.4.98. service\theme-service.ts

import { ApiError } from "../exception/api-error"

import prisma from "../middleware/prisma-middleware"

class ThemeService{

async create(title:string, class\_id:string){

const classData = await prisma.class.findFirst(({

where:{

id:class\_id

}

}))

if(!classData)

throw ApiError.BadRequest('Курс не найден')

const theme = await prisma.theme.create({

data:{

title:title,

class\_id:class\_id

}

}).catch(()=>{

throw ApiError.BadRequest('Ошибка создания')

})

return theme

}

async viewAll(class\_id:string){

const classData = await prisma.class.findFirst(({

where:{

id:class\_id

}

}))

if(!classData)

throw ApiError.BadRequest('Курс не найден')

const theme = await prisma.theme.findMany({

where:{

class\_id:class\_id

}

}).catch(()=>{

throw ApiError.BadRequest('Ошибка создания')

})

return theme

}

async update(id:string, title:string){

const themeData = await prisma.theme.findFirst({

where:{

id:id

}

})

if(!themeData)

throw ApiError.BadRequest('Тема не найдена')

const theme = await prisma.theme.update({

where:{

id:id

},

data:{

title:title

}

})

return theme

}

async delete(id:string){

const themeData = await prisma.theme.findFirst({

where:{

id:id

}

})

if(!themeData)

throw ApiError.BadRequest('Тема не найдена')

await prisma.task.updateMany({

where:{

theme\_id:id

},

data:{

theme\_id:null

}

})

const theme = await prisma.theme.delete({

where:{

id:id

}

})

return theme

}

}

export default new (ThemeService)

1.4.99. service\token-service.ts

import \* as jwt from 'jsonwebtoken'

import prisma from '../middleware/prisma-middleware'

import \* as dotenv from 'dotenv';

import { Secret } from 'jsonwebtoken';

dotenv.config();

class TokenService{

generateTokens(payload:any){

let accessToken: any

let refreshToken: any

if(process.env.JWT\_ACCESS\_SECRET&&process.env.JWT\_REFRESH\_SECRET){

accessToken= jwt.sign(payload,process.env.JWT\_ACCESS\_SECRET, {expiresIn:'30s'})

refreshToken = jwt.sign(payload, process.env.JWT\_REFRESH\_SECRET, {expiresIn:'30d'})

}

return {

accessToken,

refreshToken

}

}

validateAccessToken(token:string){

try {

if(process.env.JWT\_ACCESS\_SECRET){

const userData =jwt.verify(token,process.env.JWT\_ACCESS\_SECRET)

return userData

}

} catch (e) {

return null

}

}

validateRefreshToken(token:string){

try {

if(process.env.JWT\_REFRESH\_SECRET){

const userData =jwt.verify(token, process.env.JWT\_REFRESH\_SECRET)

return userData

}

} catch (e) {

return null

}

}

async saveToken(userId:any, refreshToken:string){

const tokenData = await prisma.token.findFirst({where:{

user\_id:userId

}})

if(tokenData){

await prisma.token.update({

where:{

id:tokenData.id

},

data: {

refreshToken:refreshToken,

user\_id:userId

},

})

}

else {

const token = await prisma.token.create({

data: {

refreshToken:refreshToken,

user\_id:userId

},

})

return token;

}

}

async removeToken(id:string){

const tokenData = await prisma.token.findFirst({where:{

user\_id:id

}})

await prisma.token.delete({

where:{

id:tokenData?.id

},

})

}

async findToken(refreshToken:string){

const tokenData = await prisma.token.findFirst({where:{

refreshToken:refreshToken

}})

return tokenData

}

}

export default new TokenService()

1.4.100. service\user-service.ts

import bcrypt from 'bcrypt'

import \* as uuid from 'uuid'

import mailService from './mail-service'

import tokenService from './token-service'

import {ApiError} from '../exception/api-error'

import prisma from '../middleware/prisma-middleware'

import fs from 'fs'

import { promisify } from 'util'

const unlinkAsync = promisify(fs.unlink)

class UserService{

async edit(id:string, email:string, surname:string, name:string, lastname:string){

const userCandiet = await prisma.user.findFirst({

where:{

id:id

},

select: {

id:true,

email: true

},

})

if(!userCandiet){

throw ApiError.BadRequest('Пользователь не найден')

}

if(userCandiet.email !== email){

const candiet = await prisma.user.findFirst({

where:{

email:email

},

select: {

id:true

},

})

if(candiet){

throw ApiError.BadRequest('Пользователь с таким email уже существует')

}

const activationLink = uuid.v4();

await mailService.sendActivationMail(email,`${process.env.API\_URL}/auth/activate/${activationLink}`)

await prisma.user.update({

where:{

id:id

},

data: {

activationLink:activationLink

},

})

}

await prisma.user.update({

where:{

id:id

},

data: {

email:email,

name:name,

surname:surname,

lastname:lastname

},

})

const user = await prisma.user.findFirst({

where:{

email:email

},

select: {

surname: true,

name: true,

lastname:true,

email:true,

passwordLink:true,

activationLink:true,

isActived:true,

file:true,

id:true

},

})

const tokens = tokenService.generateTokens({...user})

if(user)

await tokenService.saveToken(user.id, tokens.refreshToken)

return {

...tokens,

user

}

}

async reset(email:string){

const userCandiet = await prisma.user.findFirst({

where:{

email:email

}

})

if(!userCandiet){

throw ApiError.BadRequest('Пользователь не найден')

}

const passwordLink = uuid.v4();

await prisma.user.update({

where:{

id:userCandiet.id

},

data: {

passwordLink:passwordLink

},

})

await mailService.sendActivationPasswordMail(email,`${process.env.API\_URL}/user/activatePassword/${passwordLink}`)

const user = await prisma.user.findFirst({

where:{

email:email

},

select: {

surname: true,

name: true,

lastname:true,

email:true,

passwordLink:true,

activationLink:true,

isActived:true,

file:true,

id:true

},

})

return user

}

async activate\_password(passwordLink:string){

const user = await prisma.user.findFirst({

where:{

passwordLink:passwordLink

}

})

if(!user){

throw ApiError.BadRequest('Некорректная ссылка смены пароля')

}

}

async edit\_avatar(image:string, email:string){

const userCandiet = await prisma.user.findFirst({

where:{

email:email

}

})

if(!userCandiet){

throw ApiError.BadRequest('Пользователь не найден')

}

if(userCandiet.file){

fs.access('uploads\\images\\'+userCandiet.file, fs.constants.F\_OK, async(err) => {

if (!err) {

await unlinkAsync('uploads\\images\\'+userCandiet.file)

}

});

}

await prisma.user.update({

where:{

id:userCandiet.id

},

data: {

file:image

},

})

const user = await prisma.user.findFirst({

where:{

email:email

},

select: {

surname: true,

name: true,

lastname:true,

email:true,

passwordLink:true,

activationLink:true,

isActived:true,

file:true,

id:true

},

})

return user

}

async changePassword(password:string, passwordLink:string){

if(!password){

throw ApiError.BadRequest('Введите пароль')

}

const hashPassword = await bcrypt.hash(password, 4)

const userCandiet = await prisma.user.findFirst({

where:{

passwordLink:passwordLink

},

select: {

id:true,

email: true

},

})

if(!userCandiet){

throw ApiError.BadRequest('Пользователь не найден')

}

await prisma.user.update({

where:{

id:userCandiet.id

},

data: {

password:hashPassword

},

})

const user = await prisma.user.findFirst({

where:{

passwordLink:passwordLink

},

select: {

surname: true,

name: true,

lastname:true,

email:true,

passwordLink:true,

activationLink:true,

isActived:true,

file:true,

id:true

},

})

return user

}

async getUserInfo(id:string){

const userCandiet = await prisma.user.findFirst({

where:{

id:id

},

select: {

id:true,

email: true

},

})

if(!userCandiet){

throw ApiError.BadRequest('Пользователь не найден')

}

const user = await prisma.user.findFirst({

where:{

id:id

},

select: {

surname: true,

name: true,

lastname:true,

email:true,

passwordLink:true,

activationLink:true,

isActived:true,

file:true,

id:true

},

})

return user

}

}

export default new UserService()