Міністерство освіти і науки України Харківський національний Університет радіоелектроніки

Кафедра «Програмна інженерія»

# ЗВІТ

з лабораторної работи №4

з дисципліни «Архітектура програмного забезпечення»

на тему: «Розробка клієнтської/front-end частини програмної системи»

Виконав: Перевірив:

ст. гр. ПЗПІ-20-3 ст. викл. каф. ПІ

Кулешов Микита Побіженко І. О.

Харків 2023

# ВСТУП

Метою лабораторної роботи є розробка клієнтської/front-end частини проекту за темою: «Програмна система для автоматизації побудови розкладу та відстеження відвідування занять в навчальних закладах».

Хід лабораторної роботи:

1. опис компонентів системи;
2. опис прецедентів системи;
3. опис взаємодії клієнтської частини з серверною;
4. опис пакетів, з яких складається клієнтський додаток.

# 1 ДІАГРАМА КОМПОНЕНТІВ СИСТЕМИ

На рисунку 1.1 наведено діаграму компонентів, що описує архітектуру програми.

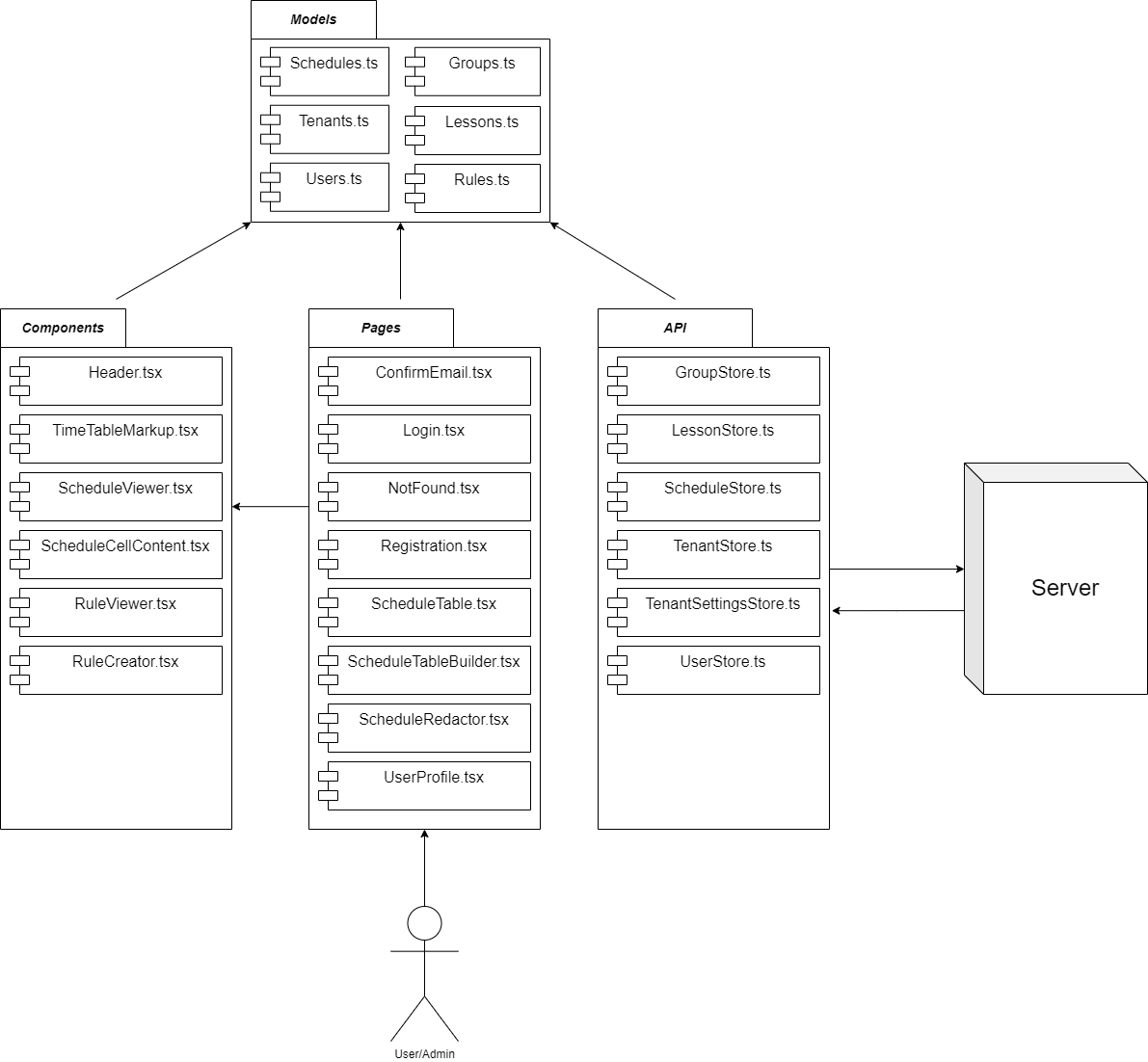


Рисунок 1.1 – Діаграма компонентів

Для створення програмного забезпечення було обрано наступні інструменти розробки:

− клієнтська частина написана на мові TypeScript з використанням технологій React та MobX для виконання запитів до сервера з використанням стилю REST;

− за основу для графічних компонентів була взята стороння бібліотека MUI, що надає обширну кількість готових React елементів для побудови сторінок.

Як приклад, код файлу, що відтворює та надає управління над сторінкою користувача наведено у додатку A. Відповідний клієнт-сервіс для цієї сторінки, що здійснює запити до сервера наведено у додатку Б. Повний проект завантажено до хмарного сховища Google Disk [1].

# 2 ДІАГРАМА ПРЕЦЕНДЕНТІВ

В програмі наявні дві ролі: диспетчер та вчитель. Щоб наглядно показати наявний функціонал та його доступність в залежності від ролі, було створено діаграму прецедентів (див. рис. 2.1).

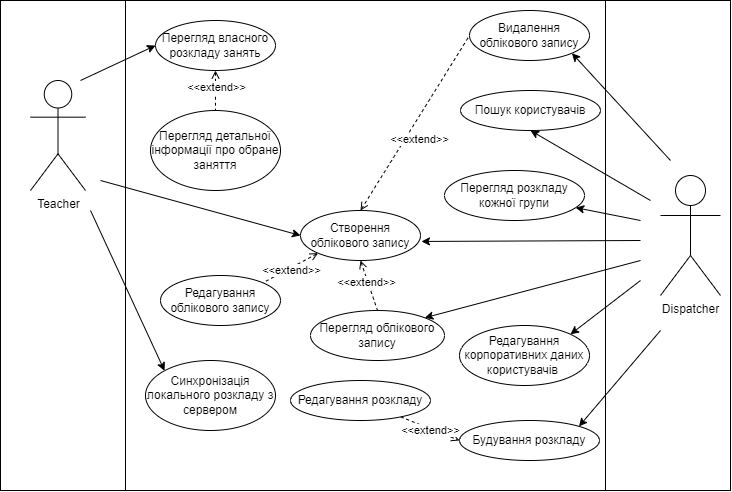


Рисунок 2.1 – Діаграма прецедентів

Виявлення доступного функціоналу користувача здійснюється шляхом отримання списку його ролей з JWT токена після успішної авторизації.

# 3 ДІАГРАМА ВЗАЄМОДІЇ

На рисунку 3.1 представлена діаграма взаємодії, що ілюструє яким чином клієнтська частина спілкується із серверною.

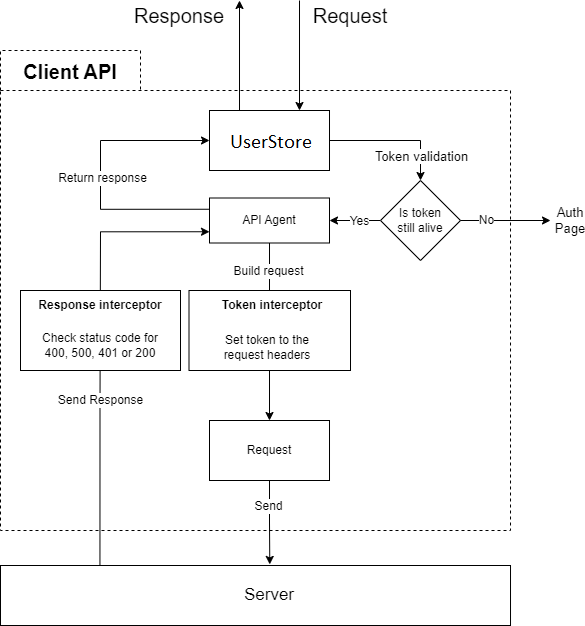


Рисунок 3.1 – діаграма взаємодії клієнтської частини із серверною

В цій діаграмі, як приклад, наведено запит до API-модуля UserStore, що відповідає за запити до сервера, які пов’язані з даними користувача. Усі інші API-модулі, наприклад, ScheduleStore працюють таким самим чином.

При виконанні будь якого запиту, система першим чином перевіряє, чи досі дійсний токен, який має користувач. Якщо ні, то його автоматично пересилає на сторінку авторизації для отримання нового токена. Якщо токен дійсний, то перед виконанням запиту, він проходить через Token Interceptor, завдання якого перехватити будь який запит та додати до його заголовку токен авторизації. Після встановлення токена Token Interceptor передає запит в місце, куди його було направлено спочатку.

При отримані будь якої відповіді від сервера працює інший буфер – Response Interceptor, його задача перевірити статус код відповіді на певні винятки такі як поганий запит (Bad Request), внутрішня помилка сервера (Internal Server Error), або неавторизований запит (Unauthorized).

# 4 ДІАГРАМА ПАКЕТІВ

На рисунку 4.1 представлена діаграма пакетів клієнтської частини. Та зв’язки між ними.

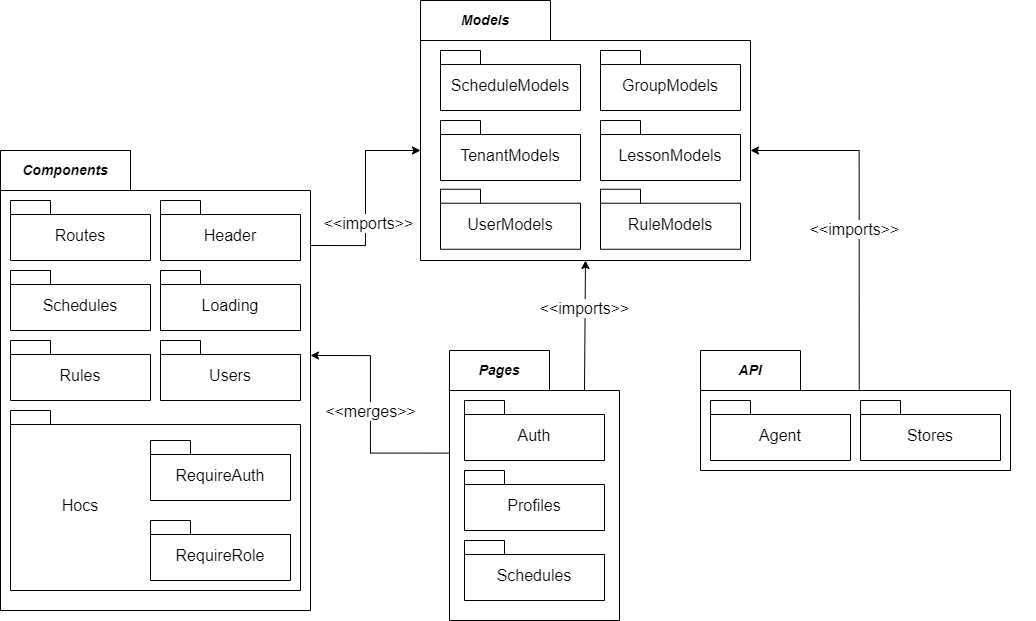


Рисунок 4.1 – діаграма пакетів клієнтської частини

Пакет Pages «зливається» з пакетом Components оскільки майже всі об’єкти з обох пакетів використовують хоча б один об’єкт з іншого пакету.

Об’єкти, що знаходяться у пакеті Pages насправді є звичайними компонентами, як ті, що зберігаються в пакеті Components, але вони відокремлені в окремий пакет оскільки мають принципово інше призначення. Пакет Components має підпакет Hocs, що означає High Order Components – це ті ж самі компоненти, але як правило, вони не описують ніяку сутність, а лише є обгортками над іншими компонентами та впроваджують певний додатковий функціонал, наприклад HOC під назвою RequiresAuth є обгорткою для компонентів-сторінок і все, що він робить, це перед тим як відтворити вікно, перевіряє чи авторизований користувач, якщо ні, то він автоматично перекидається на вікно авторизації.

# ВИСНОВКИ

В ході виконання лабораторної роботи була розроблена клієнтська/front-end частина до проекту на тему «Програмна система для автоматизації побудови розкладу та відстеження відвідування занять в навчальних закладах».

Представлені діаграми прецедентів, компонентів, пакетів та взаємодії.

Для кожної з них надано тексте пояснення окремих рішень.

Запис з демонстрацією роботи серверної частини можна подивитись за посиланням до хмарного сховища Google Disk [2].

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Код проекту. Google Disk. URL: (дата звернення: 05.05.2023) <https://drive.google.com/drive/folders/1nZhw6NsuYEzP3nLBFBzoYZ4eFoYMCdO8?usp=sharing>
2. Відео демонстрація роботи. Google Disk. URL: (дата звернення: 05.05.2023) <https://drive.google.com/drive/folders/1nZhw6NsuYEzP3nLBFBzoYZ4eFoYMCdO8?usp=sharing>

# ДОДАТОК А

Код файлу UserProfile.tsx, який відображає сторінку з даними певної тварини

1. import { Avatar, Button, TextField, Typography } from "@mui/material";
2. import { Box } from "@mui/system";
3. import { useEffect, useState, useRef } from "react";
4. import { useStore } from "../../api/stores/StoresManager";
5. import EditIcon from '@mui/icons-material/Edit';
6. import SaveIcon from '@mui/icons-material/Save';
7. import { mainBoxStyle,
8. profileBoxStyle,
9. avatarStyle,
10. userInfoBlocks,
11. userInfoSubSetBlock} from "./UserProfileStyles";
12. import { buttonImageIconStyle,
13. buttonHoverStyles,
14. buttonBoxStyles } from "../../styles/ButtonStyles";
15. import { toast } from "react-toastify";
16. import React from "react";
17. import { Role, UserUpdateModel } from "../../models/Users";
18. import { getEnumKey } from "../../utils/Utils";
19. import { useCult } from "../../hooks/Translator";
20. const EMAIL\_REGEX = /^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$/
21. type Focus = React.FocusEvent<HTMLInputElement | HTMLTextAreaElement, Element>;
22. export default function UserPage() {
23. const passwordSecret = "\*\*\*\*\*\*\*\*\*\*";
24. const { userStore } = useStore();
25. const { translator } = useCult();
26. const currentUser = userStore.user;
27. const [name, setName] = useState(currentUser!.name);
28. const [firstNameErrors, setNameErrors] = useState('');
29. const [lastName, setLastName] = useState(currentUser!.lastName);
30. const [lastNameErrors, setLastNameErrors] = useState('');
31. const [fatherName, setFatherName] = useState(currentUser!.fatherName);
32. const [fatherNameErrors, setFatherNameErrors] = useState('');
33. const [age, setAge] = useState(currentUser!.age);
34. const [ageErrors, setAgeErrors] = useState('');
35. const [email, setEmail] = useState(currentUser!.login);
36. const [emailErrors, setEmailErrors] = useState('');
37. const [password, setPassword] = useState(passwordSecret);
38. const [passwordErrors, setPasswordErrors] = useState('');
39. const [changeMode, setChangeMode] = useState(true);
41. useEffect(() => {
42. userStore.getAutenticatedUserInfo();
43. }, [userStore]);
45. const setProfileChanging = () => {
46. setChangeMode(!changeMode);
47. }
48. const firstNameRef = useRef<HTMLInputElement>();
49. const lastNameRef = useRef<HTMLInputElement>();
50. const fatherNameRef = useRef<HTMLInputElement>();
51. const ageRef = useRef<HTMLInputElement>();
52. const loginRef = useRef<HTMLInputElement>();
53. const passwordRef = useRef<HTMLInputElement>();
54. const handleFirstNameChange = (e: Focus) => {
55. const firstName = e.target.value;
56. if (firstName.length === 0) {
57. setNameErrors(translator('input-helpers.first-name-required'));
58. } else {
59. setNameErrors('');
60. }
61. setName(firstName);
62. }
63. const handleLastNameChange = (e: Focus) => {
64. const lastName = e.target.value;
65. if (lastName.length === 0) {
66. setLastNameErrors(translator('input-helpers.last-name-required'));
67. } else {
68. setLastNameErrors('');
69. }
70. setLastName(lastName);
71. }
72. const handleFatherNameChange = (e: Focus) => {
73. const fatherName = e.target.value;
74. if (fatherName.length === 0) {
75. setFatherNameErrors(translator('input-helpers.father-name-required'));
76. } else {
77. setFatherNameErrors('');
78. }
79. setFatherName(fatherName);
80. }
81. const handleAgeChange = (e: Focus) => {
82. const age = Number(e.target.value);
83. if(!age) {
84. return;
85. } else if (age < 5) {
86. setAgeErrors(translator('input-helpers.minimal-age-5'));
87. } else if (age > 99) {
88. setAgeErrors(translator('input-helpers.maximal-age-99'));
89. } else {
90. setAgeErrors('');
91. }
92. setAge(age);
93. }
94. const handleEmailChange = (e: Focus) => {
95. const email = e.target.value;
96. if (email.length === 0) {
97. setEmailErrors(translator('input-helpers.email-required'));
98. } else if (!email.match(EMAIL\_REGEX)) {
99. setEmailErrors(translator('input-helpers.email-should-be-correct'));
100. } else {
101. setEmailErrors('');
102. }
103. setEmail(email);
104. }
105. const handlePasswordChange = (e: Focus) => {
106. const password = e.target.value;
107. if (password.length === 0) {
108. setPasswordErrors(translator('input-helpers.please-enter-password'));
109. } else {
110. setPasswordErrors('');
111. }
112. setPassword(password);
113. }
114. const hasErrors = () => {
115. const hasAnyErrors = firstNameErrors.length
116. || lastNameErrors.length
117. || fatherNameErrors.length
118. || emailErrors.length
119. || passwordErrors.length;
120. return hasAnyErrors;
121. }
122. const submit = async () => {
123. if (hasErrors()) {
124. return;
125. }
126. setProfileChanging();
127. const user: UserUpdateModel = {
128. id: currentUser?.id!,
129. login: email,
130. age: age,
131. role: null,
132. groupId: null,
133. tenantId: null,
134. name: name,
135. lastName: lastName,
136. fatherName: fatherName,
137. password: password === passwordSecret || !password ? null : password
138. };
139. const result = await userStore.updateUserInfo(user);
141. if(result.isSuccessful) {
142. toast.success(translator('toast.profile-updated'));
143. }
144. }
145. function getGroupLabelName(): React.ReactNode {
146. return userStore.user?.role === Role.Teacher
147. ? translator('labels.underlying-group')
148. : translator('labels.group');
149. }
150. function getGroupFieldValue(): unknown {
151. return userStore.user?.group?.title
152. ?? translator('words.none');
153. }
154. return(
155. <Box sx={mainBoxStyle}>
156. <Box sx={profileBoxStyle}>
157. <Avatar sx={avatarStyle}>
158. {currentUser?.name[0].toUpperCase()}
159. {currentUser?.lastName[0].toUpperCase()}
160. </Avatar>
161. <Box sx={buttonBoxStyles}>
162. <Button
163. sx={buttonHoverStyles}
164. variant="contained"
165. onClick={setProfileChanging}
166. disabled={!changeMode}
167. >
168. {translator('buttons.change')}
169. <EditIcon sx={buttonImageIconStyle}/>
170. </Button>
171. <Button
172. sx={buttonHoverStyles}
173. variant="contained"
174. onClick={submit}
175. disabled={changeMode}
176. >
177. {translator('buttons.save')}
178. <SaveIcon sx={buttonImageIconStyle}/>
179. </Button>
180. </Box>
181. </Box>
182. <Box sx={userInfoBlocks}>
183. <Box sx={userInfoSubSetBlock}>
184. <Typography variant='h5'>
185. {translator('headers.personal-info')}
186. </Typography>
187. <TextField
188. label={translator('labels.first-name')}
189. variant="filled"
190. size="small"
191. helperText={firstNameErrors}
192. value={name}
193. required={true}
194. disabled={changeMode}
195. inputRef={firstNameRef}
196. error={firstNameErrors.length !== 0}
197. margin="dense"
198. onFocus={(e) => handleFirstNameChange(e)}
199. onChange={handleFirstNameChange}
200. />
201. <TextField
202. label={translator('labels.last-name')}
203. variant="filled"
204. size="small"
205. helperText={lastNameErrors}
206. value={lastName}
207. required={true}
208. disabled={changeMode}
209. inputRef={lastNameRef}
210. error={lastNameErrors.length !== 0}
211. margin="dense"
212. onFocus={(e) => handleLastNameChange(e)}
213. onChange={handleLastNameChange}
214. />
215. <TextField
216. label={translator('labels.father-name')}
217. variant="filled"
218. size="small"
219. helperText={fatherNameErrors}
220. value={fatherName}
221. required={true}
222. disabled={changeMode}
223. inputRef={fatherNameRef}
224. error={fatherNameErrors.length !== 0}
225. margin="dense"
226. onFocus={(e) => handleFatherNameChange(e)}
227. onChange={handleFatherNameChange}
228. />
229. <TextField
230. label={translator('labels.age')}
231. variant="filled"
232. size="small"
233. helperText={ageErrors}
234. value={age}
235. required={false}
236. disabled={changeMode}
237. inputRef={ageRef}
238. error={ageErrors.length !== 0}
239. margin="dense"
240. onFocus={(e) => handleAgeChange(e)}
241. onChange={handleAgeChange}
242. />
243. <hr/>
244. </Box>
245. <Box sx={userInfoSubSetBlock}>
246. <Typography variant='h5'>
247. {translator('headers.credentials')}
248. </Typography>
249. <TextField
250. label={translator('labels.email')}
251. variant="filled"
252. size="small"
253. helperText={emailErrors}
254. value={email}
255. required={true}
256. disabled={changeMode}
257. inputRef={loginRef}
258. error={emailErrors.length !== 0}
259. margin="dense"
260. onFocus={(e) => handleEmailChange(e)}
261. onChange={handleEmailChange}
262. />
263. <TextField
264. label={translator('labels.password')}
265. variant="filled"
266. size="small"
267. helperText={passwordErrors}
268. type="password"
269. value={password}
270. required={true}
271. disabled={changeMode}
272. inputRef={passwordRef}
273. error={passwordErrors.length !== 0}
274. margin="dense"
275. onFocus={(e) => handlePasswordChange(e)}
276. onChange={handlePasswordChange}
277. />
278. <hr/>
279. </Box>
280. <Box sx={userInfoSubSetBlock}>
281. <Typography variant='h5'>
282. {translator('headers.tenant-info')}
283. </Typography>
284. <TextField
285. label={translator('labels.role')}
286. variant="filled"
287. size="small"
288. value={translator(getEnumKey(Role, userStore.user?.role))}
289. required={false}
290. disabled
291. margin="dense"
292. />
293. <TextField
294. label={translator('labels.tenant')}
295. variant="filled"
296. size="small"
297. value={userStore.user?.tenant.tenantName}
298. required={false}
299. disabled
300. margin="dense"
301. />
302. <TextField
303. label={getGroupLabelName()}
304. variant="filled"
305. size="small"
306. value={getGroupFieldValue()}
307. required={false}
308. disabled
309. margin="dense"
310. />
311. <hr/>
312. </Box>
313. </Box>
314. </Box>
315. );
316. }

# ДОДАТОК Б

Код файлу UserStore.ts, який відповідає за здійснення запитів до сервера

1. import { makeAutoObservable } from "mobx";
2. import { agent } from "../agent";
3. import { UserModel, UserLoginModel, UserCreateModel, UserUpdateModel, Role } from "../../models/Users";
4. import { store } from "./StoresManager";
5. import BaseStore from "./BaseStore";
6. import { toast } from "react-toastify";
7. export default class UserStore {
8. base: BaseStore = new BaseStore();
9. client: any = null;
10. user: UserModel | null = null;
11. otherUsers: UserModel[] = [];
12. constructor() {
13. makeAutoObservable(this);
14. }
15. get isLoggedIn() {
16. return !!this.user;
17. }
18. login = async (creds: UserLoginModel) => {
19. const response = await agent.Auth.login(creds);
20. this.base.handleErrors(response);
21. console.debug("login successful, token - " + response.value);
22. store.commonStore.setToken(response.value);
23. const userInfo = await this.getAutenticatedUserInfo();
24. if(userInfo?.role === Role.Pupil) {
25. this.logout();
26. toast.error("Pupils are not allowed to access the web system");
27. return false;
28. }
29. this.user = userInfo;
30. return response.isSuccessful;
31. };
32. logout = () => {
33. store.commonStore.setToken(null);
34. window.localStorage.removeItem("jwt");
35. this.user = null;
36. };
37. updateUserInfo = async (user: UserUpdateModel) =>
38. await this.base.simpleRequest(async () => await agent.User.updateUser(user));
39. confirmEmail = async (key: string) => {
40. const response = await agent.Auth.confirmEmail(key);
41. this.base.handleErrors(response);
42. return response.isSuccessful;
43. }
44. register = async (creds: UserCreateModel) =>
45. await agent.Auth.register(creds);
46. getAutenticatedUserInfo = async () => {
47. if(this.isLoggedIn) {
48. return this.user;
49. } else {
50. const response = await agent.Auth.getAuthenticatedUserInfo();
51. this.base.handleErrors(response);
53. if(response.value !== null) {
54. this.user = response.value;
55. }
56. return response.value;
57. }
58. }
59. getUserInfo = async (userId: string) => {
60. const response = await agent.User.getUser(userId);
61. this.base.handleErrors(response);
62. if(response.isSuccessful && !this.otherUsers.includes(response.value)) {
63. this.otherUsers.push(response.value);
64. }
65. return response.value;
66. }
67. }