САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Фронт-энд разработка

Отчет

Лабораторная работа 2: взаимодействие с внешним API

Выполнил:

Малышенко Александр

Группа: К33392

Проверил:
Добряков Д. И.

Санкт-Петербург
2023 г.

1. Задача (Вариант 10):

Необходимо привязать то, что было сделано в первой лабораторной к внешнему API.

2. Ход выполнения работы.

Скрипт для обработки регистрации и логина пользователя (usersManage.js):

```
//  classes

class RegisterUser{
    constructor() {
        this.regData = {}
    }

    async registerHandler(event){
        await this._getInputs(event)
        await this._regRequestHandler()
    }

    async _getInputs(event){
        let regInputs = Array.from(event.target.querySelectorAll('input'))
        for(const regInput of regInputs){
            this.regData[regInput.name] = regInput.value
        }

        //add default data to reg data:
        this.regData['accountCreationDate'] = getCurrentDate();
        this.regData['friendsList'] = []
        this.regData['ownCapsulesList'] = []
        this.regData["fullUsername"] = ""
        this.regData["phoneNumber"] = ""
        this.regData["address"] = ""
        this.regData["extraProfileInfo1"] = ""
        this.regData["extraProfileInfo2"] = ""
    }

    async _regRequestHandler(){
        const response = await fetch('http://localhost:3000/users', {
            method: "POST",
            body: JSON.stringify(this.regData),
            headers: {'Content-Type': 'application/json'}
        })

        //add local storage data if there is access token
        const responseJson = await response.json()
        const {accessToken, user} = responseJson
        if(accessToken){
            localStorage.accessToken = accessToken
            localStorage.user = JSON.stringify(user)
        }
    }
```

```
}

class LoginUser{
    constructor() {
        this.loginData = {}
    }

    async loginHandler(event){
        await this._getInputs(event)
        await this._loginRequestHandler()
    }

    async _getInputs(event){
        const loginInputs =
Array.from(event.target.querySelectorAll('input'))
        for (const loginInput of loginInputs) {
            this.loginData[loginInput.name] = loginInput.value
        }
    }

    async _loginRequestHandler(){
        const response = await fetch('http://localhost:3000/login', {
            method: "POST",
            body: JSON.stringify(this.loginData),
            headers: {'Content-Type': 'application/json'}
        })

        //add local storage data if there is access token
        const responseJson = await response.json()
        const {accessToken, user} = responseJson
        if(accessToken){
            localStorage.accessToken = accessToken
            localStorage.user = JSON.stringify(user)
        }
    }
}

//  main functions for handling

async function registration(event) {
    event.preventDefault()
    let registrationHandler = new RegisterUser()
    await registrationHandler.registerHandler(event)
    window.location.reload()
}

async function logIn(event) {
    event.preventDefault()
    let loginHandler = new LoginUser()
    await loginHandler.loginHandler(event)
    window.location.reload()
}

function getCurrentDate(){
    let today = new Date();
    const dd = String(today.getDate()).padStart(2, '0');
    const mm = String(today.getMonth() + 1).padStart(2, '0'); //January is 0!
    const yyyy = today.getFullYear();

    today = dd + '.' + mm + '.' + yyyy;
    return today
}
```

Класс RegisterUser отвечает за регистрацию пользователей, а именно получение и обработку данных, полученных с модального окна (метод *_getInputs*) и запись их в базу данных json-сервера (метод *_regRequestHandler*). Метод *registerHandler* запускает остальные методы в нужном порядке для чтения и записи данных.

Класс LoginUser аналогичен классу RegisterUser, только выполняет не регистрацию, а вход пользователя.

Функции *registration* и *logIn* создают экземпляры класса и реагируют на события в html коде

Скрипт для создания новых капсул и генерации модального окна капсулы (capsulesManage.js):

```
//  classes

class CapsuleCreator{
    constructor() {
        this.capsuleData = {}
    }

    async createCapsule(event){
        await this._getInputs(event)
        await this._createCapsuleRequestHandler()
    }

    async _getInputs(event){
        const createCapsuleInputs =
Array.from(event.target.querySelectorAll('input'))
        for (const capsuleInput of createCapsuleInputs) {
            if (capsuleInput.name === 'radioPublic') {
                continue
            }

            this.capsuleData[capsuleInput.name] = capsuleInput.value
        }

        //add default data to capsule data &  get the value of the radio
button:
        this.capsuleData['text'] =
event.target.querySelector('textarea').value
        this.capsuleData['capsuleAvailability'] =
event.target.querySelector('input[name="radioPublic"]:checked').value
        this.capsuleData['creationDate'] = getCurrentDate()
        if (localStorage.user) {
            const localStorageParse =
JSON.parse(localStorage.getItem("user"))
            this.capsuleData['userId'] = localStorageParse['id']
            this.capsuleData['creatorName'] = localStorageParse['username']
        } else {
            this.capsuleData['userId'] = 0
            this.capsuleData['creatorName'] = null
        }
    }
```

```
    async _createCapsuleRequestHandler(){
        const response = await fetch('http://localhost:3000/capsules', {
            method: "POST",
            body: JSON.stringify(this.capsuleData),
            headers: {'Content-Type': 'application/json'}
        })
    }
}

class ModalForCapsuleGenerator{
    constructor(capsuleId, userId) {
        this.serverCapsulesUrl =
`http://localhost:3000/capsules?id=${capsuleId}`
        this.capsuleData = {}
    }

    async createModalForCapsule(){
        await this._dataRequestHandler()
        await this._modalHandler()
    }

    async _dataRequestHandler(){
        const responseForCapsules = await fetch(this.serverCapsulesUrl,
{method: "GET"})
        this.capsuleData = (await responseForCapsules.json())[0]
    }

    async _modalHandler(){
        document.querySelector("#openCapsuleModal").innerHTML =
            this._getModalCapsuleHtml(this.capsuleData['title'],
this.capsuleData['text'], this.capsuleData['file'])
        const modal = new
bootstrap.Modal(document.querySelector("#openCapsuleModal"));
        modal.show()
    }

    _getModalCapsuleHtml(capsuleTitle, mainText, file){
        return `
          <div class="modal-dialog modal-dialog-scrollable">
             <div class="modal-content">
                 <div class="modal-header change-bg-verylightgreen">
                     <h1 class="modal-title fs-5"
id="openCapsuleLabel">${capsuleTitle}</h1>
                     <button type="button" class="btn-close" data-bs-
dismiss="modal" aria-label="Close"></button>
                 </div>
                 <div class="modal-body">
                     <div class="input-group mb-3">
                         <div class="form-floating">${mainText}</div>
                     </div>
                 <hr>
                    <div class="mb-4">
                        <p>Files:</p>
                        <div>${file}</div>
                    </div>
                 </div>
                 <div class="modal-footer">
                     <button type="button" class="btn btn-secondary" data-bs-
dismiss="modal">Close</button>
                 </div>
             </div>
         </div>`
    }
```

```
}


//  main functions for handling

async function createCapsule(event) {
    event.preventDefault()
    let capsuleCreator = new CapsuleCreator()
    await capsuleCreator.createCapsule(event)
    window.location.reload()
}

async function getModalCapsule(event, capsuleId, userId){
    event.preventDefault()
    let capsuleModalCreator = new ModalForCapsuleGenerator(capsuleId, userId)
    await capsuleModalCreator.createModalForCapsule()
}

function getCurrentDate(){
    let today = new Date();
    const dd = String(today.getDate()).padStart(2, '0');
    const mm = String(today.getMonth() + 1).padStart(2, '0'); //January is 0!
    const yyyy = today.getFullYear();

    today = yyyy + '-' + mm + '-' + dd;
    return today
}
```

Класс CapsuleCreator отвечает за получение и обработку данных с модального окна для создания новой капсулы (метод *_getInputs*), а также за запись этих данных в базу данных капсул json-сервера (*_createCapsuleRequestHandler*).

Класс ModalForCapsuleGenerator считывает данные с json-сервера по определенной капсуле (метод *_dataRequestHandler*) и создает по этим данным код правильного модального окна, после чего можно открыть это окно для пользователя (метод *_modalHandler*).

Функции *createCapsule* и *getModalCapsule* создают экземпляры классов и реагируют на события в html коде для создания новой капсулы и вывода модального окна существующей капсулы соответственно.

Скрипт для генерации данных на странице (pagesLoader.js):

```
// classes

class User{
    constructor(id, username, email, accountCreationDate, friendsList, ownCapsulesList)
{
        this.id = id
        this.username = username
        this.email = email
        this.accountCreationDate =accountCreationDate
        this.friendsList = friendsList
        this.ownCapsulesList = ownCapsulesList
    }
}

class LoaderCapsulesList{
    constructor() {
        this.capsulesHtmlList = []
        this.constCapsuleUrl = 'http://localhost:3000/capsules'
        this.capsulesJson = {}
        this.usersJson = {}
        this.typeOfFirstSort = 'default'
        this.loginUserId = 0
        this.usersList = []
    }

    async loadCapsules(type, searchString = ''){
        document.querySelector("#capsules").innerHTML = ``
        this.capsulesHtmlList = []
        await this._setTypeOfFirstSort(type)
        await this._getCapsulesHandler(searchString)
        if(!this._loadUndefUserCapsules()){
            await this._createUsersListAndLoginUser()
            await this._loadLoginUserCapsules()
        }

        await this._uploadHtml()
    }

    _setTypeOfFirstSort(type){
        this.typeOfFirstSort = type
    }

    async _getCapsulesHandler(searchString){
        const response = await fetch(this._getCapsulesUrl(searchString), {method:
"GET"})
        this.capsulesJson = await response.json()
        const responseUsers = await fetch('http://localhost:3000/users', {method:
"GET"})
        this.usersJson = await responseUsers.json()
    }

    _getCapsulesUrl(searchString){
        if(searchString){
            const searchParams = new URLSearchParams()
            searchParams.set("title", searchString)
            const searchParamsString = searchParams.toString()
            return `${this.constCapsuleUrl}?${searchParamsString}`
        }

        return this.constCapsuleUrl
    }
```

```
    _createUsersListAndLoginUser(){
        this.loginUserId = JSON.parse(localStorage.getItem("user"))['id']
        this.usersList = []
        for(const user of this.usersJson){
            let userObj = new User(user['id'],
                user['username'],
                user['email'],
                user['accountCreationDate'],
                user['friendsList'],
                user['ownCapsulesList'])

            this.usersList.push(userObj)
        }
    }

    _uploadHtml(){
        for(const html of this.capsulesHtmlList){
            document.querySelector("#capsules").innerHTML += html
        }
    }

    _loadUndefUserCapsules(){
        if(!localStorage.accessToken || !localStorage.user){
            for(const capsule of this.capsulesJson){
                if(capsule['capsuleAvailability'] !== 'public'){
                    continue
                }
                this.capsulesHtmlList.push(getCapsuleHtml(capsule['id'],
capsule['title'], capsule['text'], capsule['creatorName'],
                    capsule['creationDate'], capsule['openingDate'], capsule['userId']))
            }
            return true
        }

        return false
    }

    _loadLoginUserCapsules(){
        for(const capsule of this.capsulesJson){
            //don't show private not owner capsules
            if(capsule['capsuleAvailability'] === 'private' && capsule['userId'] !==
this.loginUserId){
                continue
            }

            if ((this.typeOfFirstSort === 'myCapsules' || this.typeOfFirstSort ===
'default') && capsule['userId'] === this.loginUserId){
                this.capsulesHtmlList.push(getCapsuleHtml(capsule['id'],
capsule['title'], capsule['text'], capsule['creatorName'],
                    capsule['creationDate'], capsule['openingDate'], capsule['userId']))
                continue
            }

            if (this.typeOfFirstSort === 'openedCapsules'){
                if(capsule['capsuleAvailability'] === 'public' && capsule['openingDate']
<= getCurrentDate()){
                    this.capsulesHtmlList.push(getCapsuleHtml(capsule['id'],
capsule['title'], capsule['text'], capsule['creatorName'],
                        capsule['creationDate'], capsule['openingDate'],
capsule['userId']))
                    continue
                }
```

```
                if(capsule['userId'] === this.loginUserId && capsule['openingDate'] <=
getCurrentDate()){
                    this.capsulesHtmlList.push(getCapsuleHtml(capsule['id'],
capsule['title'], capsule['text'], capsule['creatorName'],
                        capsule['creationDate'], capsule['openingDate'],
capsule['userId']))
                    continue
                }

                if(capsule['capsuleAvailability'] === 'friendsOnly' &&
capsule['openingDate'] <= getCurrentDate()){
                    for(const user of this.usersList){
                        if(capsule['userId'] === user.id &&
user.friendsList.includes(this.loginUserId)){
                            this.capsulesHtmlList.push(getCapsuleHtml(capsule['id'],
capsule['title'], capsule['text'], capsule['creatorName'],
                                capsule['creationDate'], capsule['openingDate'],
capsule['userId']))
                            break
                        }
                    }
                }
            }

            if(this.typeOfFirstSort === 'default') {
                if (capsule['capsuleAvailability'] === 'public') {
                    this.capsulesHtmlList.push(getCapsuleHtml(capsule['id'],
capsule['title'], capsule['text'], capsule['creatorName'],
                        capsule['creationDate'], capsule['openingDate'],
capsule['userId']))
                    continue
                }

                if (capsule['capsuleAvailability'] === 'friendsOnly') {
                    for (const user of this.usersList) {
                        if (capsule['userId'] === user.id &&
user.friendsList.includes(this.loginUserId)) {
                            this.capsulesHtmlList.push(getCapsuleHtml(capsule['id'],
capsule['title'], capsule['text'], capsule['creatorName'],
                                capsule['creationDate'], capsule['openingDate'],
capsule['userId']))
                            break
                        }
                    }
                }
            }
        }
    }
}
```

Класс User – вспомогательный и лишь нужен для удобного хранения данных о пользователе в классе LoaderCapsulesList.

Класс LoaderCapsulesList необходим для загрузки всех капсул в виде таблицы на главную страницу:

Метод *_getCapsulesHandler* получает данные о капсулах и пользователях с json-сервера.

Метод *_getCapsulesUrl* создает правильную ссылку на базу данных капсул, так же, если пользователь использует поисковую строку, в ссылку добавляется поиск элемента.

Метод *_createUsersListAndLoginUser* записывает пользователей в список, состоящий из вышеописанного класса User

Метод *_loadUndefUserCapsules* добавляет в список всех капсул, которые необходимо отразить на странице, капсулы доступные даже незарегистрированным пользователям.

Метод *_loadLoginUserCapsules* добавляет в список всех капсул, которые необходимо отразить на странице, капсулы доступные авторизованному пользователю, так же список изменяется в зависимости от сортировки, выбранной пользователем.

Метод *loadCapsules* выполняет все вспомогательные методы необходимые для полного и правильного отображения капсул на странице с учетом поиска и сортировки.

Скрипт для генерации страницы с профилем пользователя (profileLoader.js):

```
// classes

class UserProfileGenerator{
    constructor(userProfileId) {
        this.userId = userProfileId
        this.serverUserUrl = `http://localhost:3000/users?id=${userProfileId}`
        this.userData = {}
    }

    async generateProfile(){
        await this._userRequestHandler()
        document.querySelector("#mainProfileInfo").innerHTML =
this._getMainInfoHtml(this.userData)
        document.querySelector("#profileNavbar").innerHTML =
this._getProfileNavbarHtml(this.userData)
    }

    _checkIsProfileOfUser(){
        return !(!this._getLoginUserId() || this.userId !== this._getLoginUserId());
    }

    _getLoginUserId(){
        if(localStorage.user){
            return JSON.parse(localStorage.getItem("user"))['id']
        }else{
            return false
        }
```

```
}

async _userRequestHandler(){
    const user = await fetch(this.serverUserUrl, {method: "GET"})
    this.userData = (await user.json())[0]
}

_getMainInfoHtml(user){
    if(!user){
        return false
    }

    return `
        <div class="col-lg-4">
            <div class="card mb-4 change-border-lightgreen">
                <div class="card-body text-center">
                    <img src="https://placekitten.com/528/560" alt="avatar"
                         class="rounded-circle img-fluid" style="width: 150px;">
                    <h5 class="my-3">${user['username']}</h5>
                    <p class="text-muted mb-1">${user['extraProfileInfo1']}</p>
                    <p class="text-muted mb-4">${user['extraProfileInfo2']}</p>
                </div>
            </div>
        </div>
        <div class="col-lg-8">
            <div class="card mb-4 change-border-lightgreen">
                <div class="card-body">
                    <div class="row">
                        <div class="col-sm-3">
                            <p class="mb-0">Full name</p>
                        </div>
                        <div class="col-sm-9">
                            <p class="text-muted mb-0">${user['fullUsername']}</p>
                        </div>
                    </div>
                    <hr>
                    <div class="row">
                        <div class="col-sm-3">
                            <p class="mb-0">Email</p>
                        </div>
                        <div class="col-sm-9">
                            <p class="text-muted mb-0">${user['email']}</p>
                        </div>
                    </div>
                    <hr>
                    <div class="row">
                        <div class="col-sm-3">
                            <p class="mb-0">Phone</p>
                        </div>
                        <div class="col-sm-9">
                            <p class="text-muted mb-0">${user['phoneNumber']}</p>
                        </div>
                    </div>
                    <hr>
                    <div class="row">
                        <div class="col-sm-3">
                            <p class="mb-0">Address</p>
                        </div>
                        <div class="col-sm-9">
                            <p class="text-muted mb-0">${user['address']}</p>
                        </div>
                    </div>
                    <hr>
                    <div class="row">
```

```
                            <div class="col-sm-3">
                                <p class="mb-0">Number of capsules</p>
                            </div>
                            <div class="col-sm-9">
                                <p class="text-muted mb-
0">${user['ownCapsulesList'].length}</p>
                            </div>
                        </div>
                    </div>
                </div>
            </div>`
    }

    _getProfileNavbarHtml(user){
        if(!user){
            return false
        }

        return `
        <a class="navbar-brand" href="index.html">Time capsules</a>
            <ul class="navbar-nav me-auto mb-2 mb-lg-0">
                <li class="nav-item">
                    <a class="nav-link active" aria-current="page"
href="index.html">Home</a>
                </li>
            </ul>
            <button class="navbar-toggler" type="button" data-bs-toggle="offcanvas" data-bs-
target="#offcanvasNavbar" aria-controls="offcanvasNavbar" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="offcanvas offcanvas-end" tabindex="-1" id="offcanvasNavbar" aria-
labelledby="offcanvasNavbarLabel">
                <div class="offcanvas-header change-bg-verylightgreen">
                    <h5 class="offcanvas-title"
id="offcanvasNavbarLabel">${user['username']}</h5>
                    <button type="button" class="btn-close" data-bs-dismiss="offcanvas"
aria-label="Close"></button>
                </div>
                <div class="offcanvas-body">
                    <ul class="navbar-nav justify-content-end flex-grow-1 pe-3">
                        <li class="nav-item">
                            <a class="nav-link active" aria-current="page"
href="userProfile.html">Profile</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href="userProfile.html">Settings</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href="index.html">My capsules</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" onclick="logout()" href="index.html">Log
out</a>
                        </li>
                    </ul>
                </div>
            </div>`
    }
}


//   main functions for handling

async function loadMainInfo(event){
```

```
    event.preventDefault()
    let userId = 1
    if(localStorage.user){
        userId = JSON.parse(localStorage.getItem("user"))['id']
    }

    let profileCreator = new UserProfileGenerator(userId)
    await profileCreator.generateProfile()
}

function logout(){
    localStorage.clear()
}
```

Класс UserProfileGenerator необходим для загрузки данных пользователя на страницу профиля. Метод _userRequestHandler запрашивает данные конкретного пользователя у json-сервера. Методы _getMainInfoHtml и _getProfileNavbarHtml выдают html коды основной информации и навигационной панели соответственно, после чего с помощью метода generateProfile все добавляется в html код userProfile.html

3. Вывод.

В ходе выполнения данной лабораторной работы я написал скрипты, благодаря которым можно практически полностью функционально пользоваться сайтом. Так как для моей задачи не существует внешнего API, был использован метод имитации внешнего API.