САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Фронд-энд разработка

Отчет

Лабораторная Работа №2

Выполнил:

Кобелев Л.К.

K33401

Проверил:
Добряков Д. И.

Санкт-Петербург

2022 г.

Задача

Сделать взаимодействие с внешним API:

- Аутентификация пользователей;
- Фильтрация и сортировка данных (сценариев);
- Взаимодействие со сценариями (лайки).

Ход работы

json server

Установлен npm-пакет `json-server`, для мока REST API.

`db.json` содержит:

- Пользователь `(user)`:
  - `email` — почта пользователя;
  - `password` — пароль пользователя;
  - `username` — имя пользователя;
  - `likes` — массив идентификаторов сценариев, которые нравятся пользователю;
  - `id` — идентификатор пользователя.
- Сценарий `(scenario)`:
  - `id` — идентификатор сценария;
  - `name` — название сценария;
  - `description` — описание сценария;
  - `tags` — массив идентификаторов тэгов сценария;
  - `system` — идентификатор системы сценария;
  - `finished` — закончен сценарий или нет;
  - `adult` — 18+ сценарий или нет;
  - `author` — идентификатор пользователя, который опубликовал сценарий;
  - `release` — дата публикации сценария;
  - `image` — ссылка на картинку сценария.
- Тэг `(tag)`:
  - `id` — идентификатор тэга;
  - `name` — название тэга.
- System `(system)`:

- `id` — идентификатор системы;
- `name` — название системы.

Для аутентификации пользователей дополнительно установлен npm-пакет `json-server json-server-auth.`

`auth.js` — скрипт с функциями авторизации:

- `getAuthToken` — возвращает токен пользователя из локального хранилища;
- `getUser` — возвращает пользователя из локального хранилища, если он есть, иначе `null`;
- `checkAuth` — проверяет залогинен ли пользователь, если нет, то отключает все элементы с классом `visible-login-only`, иначе все `invisible-login`;
- `login` — асинхронная функция для входа;
- `signup` — асинхронная функция для регистрации;
- `logout` — функция для выхода.

`login():`

```
async function login(event) {
    event.preventDefault()
    const inputs = Array.from(event.target.querySelectorAll( selectors: 'input'))
    const loginData = {}

    for (const input of inputs) {
        if (input.name !== "") {
            loginData[input.name] = input.value
        }
    }

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

    if (responseJson === "Incorrect password" || responseJson === "Cannot find user") {
        console.log("error!")
        return
    }

    const {accessToken, user} = responseJson
    localStorage.accessToken = accessToken
    localStorage.user = JSON.stringify(user)

    location.reload()
}
```

`signup():`

```
async function signup(event) {
    event.preventDefault()

    const signUpData = {}

    signUpData['username'] = document.getElementById( elementId: "usernameSignUo").value
    signUpData['email'] = document.getElementById( elementId: "emailSignUp").value
    signUpData['password'] = document.getElementById( elementId: "passwordSignUp").value


    const response = await fetch( input: 'http://localhost:3000/register', init: {
        method: "POST", body: JSON.stringify(signUpData), headers: {
            'Content-Type': 'application/json'
        }
    })

    const responseJson = await response.json()

    if (responseJson === "Email already exists" || responseJson === "Password is too short") {
        console.log("error")
        return
    }

    const {accessToken, user} = responseJson

    localStorage.accessToken = accessToken
    localStorage.user = JSON.stringify(user)

    location.reload()
}
```

`general.js` — скрипт с “общими” функциями:

- `changeSorting` — изменяет иконку у кнопки сортировки при нажатии;
- `selectTag` — выбирает тэг в фильтрации, тэг не может быть выбран, если выбрано уже 3, обновляет стили кнопки и локальную переменную количества выбранных тэгов;
- `likeScenario` — лайк на сценарий, добавляет / убирает лайк со сценария, а также из массива любимых сценариев пользователя.

`likeScenario():`

```
async function likeScenario(id, button) {
    const likeData = {}
    const userLikeData = {}

    let scenarios_url = "http://localhost:3000/scenarios"
    let response = await fetch( input: `${scenarios_url}?id=${id}`, init: {
        headers: {
            "Authorization": `Bearer ${getAuthToken()}`
        }
    })

    let responseJson = await response.json()

    let currentLikes = responseJson[0].likes

    let users_url = "http://localhost:3000/users"
    let user_id = JSON.parse(getUser()).id

    response = await fetch( input: `${users_url}?id=${user_id}`, init: {
        headers: {
            "Authorization": `Bearer ${getAuthToken()}`
        }
    })

    responseJson = await response.json()

    let likes = responseJson[0].likes
```

```
    if (button.classList.contains('active')) {
        currentLikes += 1
        likes.push(id)
    }
    else {
        currentLikes -= 1
        let index = likes.indexOf(id);
        likes.splice(index, 1);
    }

    userLikeData['likes'] = likes
    likeData['likes'] = currentLikes

    scenarios_url = `${scenarios_url}/${id}`
    await fetch(scenarios_url, init: {
        method: "PATCH", body: JSON.stringify(likeData), headers: {
            'Content-Type': 'application/json'
        }
    })

    users_url = `${users_url}/${user_id}`
    await fetch(users_url, init: {
        method: "PATCH", body: JSON.stringify(userLikeData), headers: {
            'Content-Type': 'application/json'
        }
    })

    localStorage.user.likes = likes

    button.textContent = currentLikes
}
```

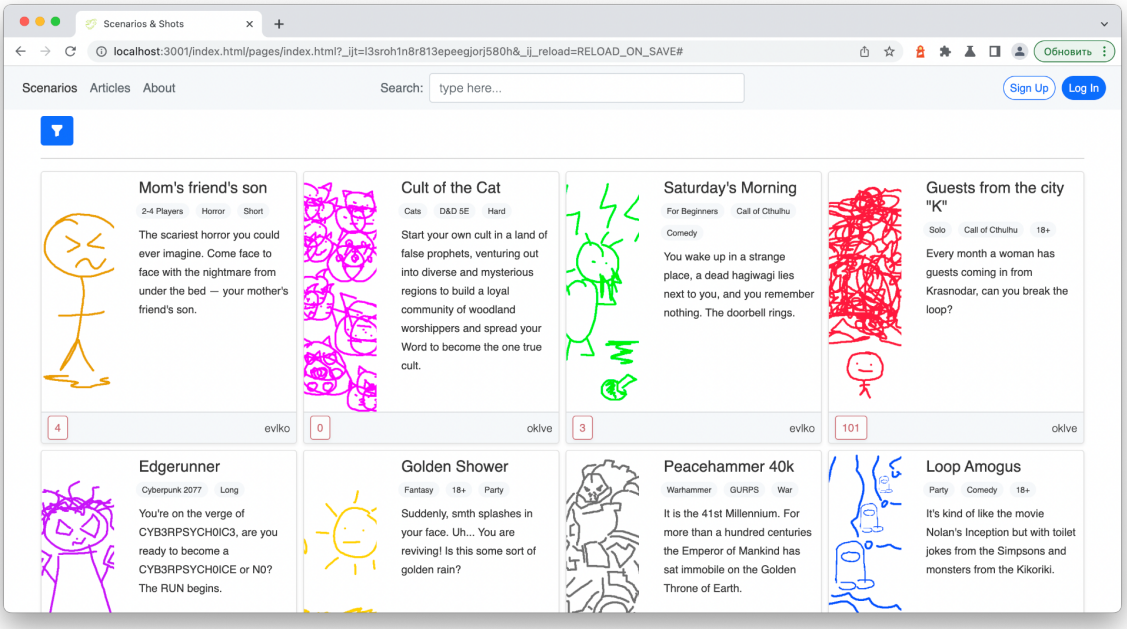
`api.js` — скрипт с функциями взаимодействия с API:

- `getCardHtml` — возвращает разметку карточки сценария в зависимости от сценария;
- `getScenarioTagHtml` — возвращает разметку бейджа тэга карточки сценария в зависимости от сценария;
- `getTagButtonHtml` — возвращает разметку кнопки тэга;
- `getSystemCheckboxHtml` — возвращает разметку чекбокса системы;
- `loadScenarios` — загружает сценарии;
- `loadTags` — загружает кнопки тэгов;
- `loadSystems` — загружает кнопки тэгов;
- `search` — фильтрует и сортирует, то есть ищет, сценарии.

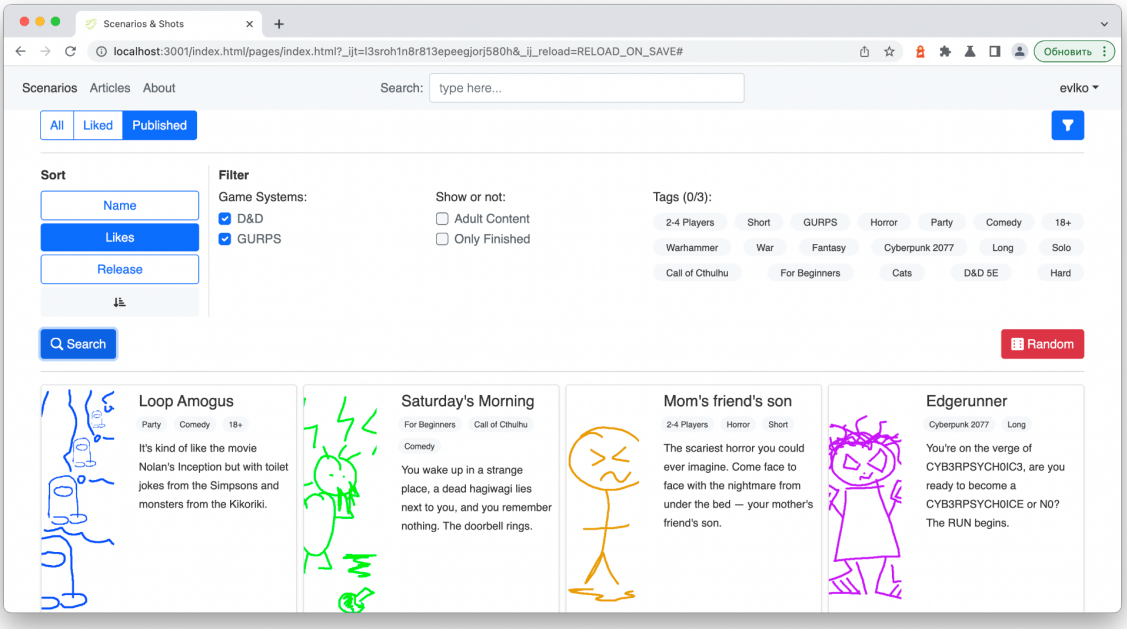
`search():`

```
function search() {
    const query = new URLSearchParams()

    const checkBoxesSelected = document.querySelector( selectors: "#gameSystems").querySelectorAll( selectors: 'input[type="checkbox"]:checked');
    for (const checkBoxSelected of checkBoxesSelected) {
        query.append( name: "system", checkBoxSelected.getAttribute( qualifiedName: "value"));
    }

    const tabSelected = document.querySelector( selectors: "#scenariosTabs").querySelector( selectors: 'input[type="radio"]:checked')
    query.append( name: "tab", tabSelected.value);

    const tagsSelected = document.querySelector( selectors: "#tagsButtons").querySelectorAll( selectors: ".active")
    for (const tagSelected of tagsSelected) {
        query.append( name: "tag", tagSelected.value);
    }

    query.append( name: "adult", value: "0");
    if (document.getElementById( elementId: "checkerAdultContent").checked) {
        query.append( name: "adult", value: "1");
    }

    if (document.getElementById( elementId: "checkerOnlyFinished").checked) {
        query.set("finished", "1");
    }

    query.set("search", searchBar.value);

    query.set("_sort", document.querySelector( selectors: 'input[name="sortingOption"]:checked').value);
    query.set("_order", document.getElementById( elementId: "sortingOrder").value);

    loadScenarios(query)
}
```

“Анонимный” пользователь не может ставить лайки и смотреть свои сценарии:



Авторизованный может:



Вывод

Получены базовые навыки работы с:

- js;
- API и json server, который оказался очень удобным модулем для мокания;
- POST, GET, PATCH запросами и параметрами поиска.