МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»
(БГТУ им. В.Г.Шухова)

Лабораторная работа №6
дисциплина «Технологии web-программирования»
по теме «Работа с HTTP запросами»

Выполнил: студент группы ВТ-41 Макаров Д.С.
Проверил: Картамышев С.В.

Белгород 2020

Лабораторная работа №6

«Работа с HTTP запросами»

Цель работы:изучить принципы работы отправки Ajax запросов. Изучить работу отправки синхронных и асинхронных запросов в Vue JS. Получить навыки работы с промисами.

Задание к лабораторной работе:

1. Изучить возможности Vue js для отправки http запросов.
2. Выбрать подходящую библиотеку для работы с запросами.
3. Реализовать взаимодействие фронтенда с REST API, спроектированном в прошлой лабораторной работе.
4. Продемонстрировать работу взаимодействия фронтенд приложения с REST API.

Ход работы

Для работы с HTTP запросами была выбрана библиотека axios.

Для авторизации приложения используется JWT, полученные с сервера токены хранятся в localStorage, сброс авторизации происходит по истечении refresh токена.

Получение access токена обернуто в функцию, хранящуюся в контексте всего приложения, для автоматического обновления токена по его истечению.

Каждый HTTP запрос, кроме запроса на получение токенов содержит заголовок Bearer, содержащий access токен.

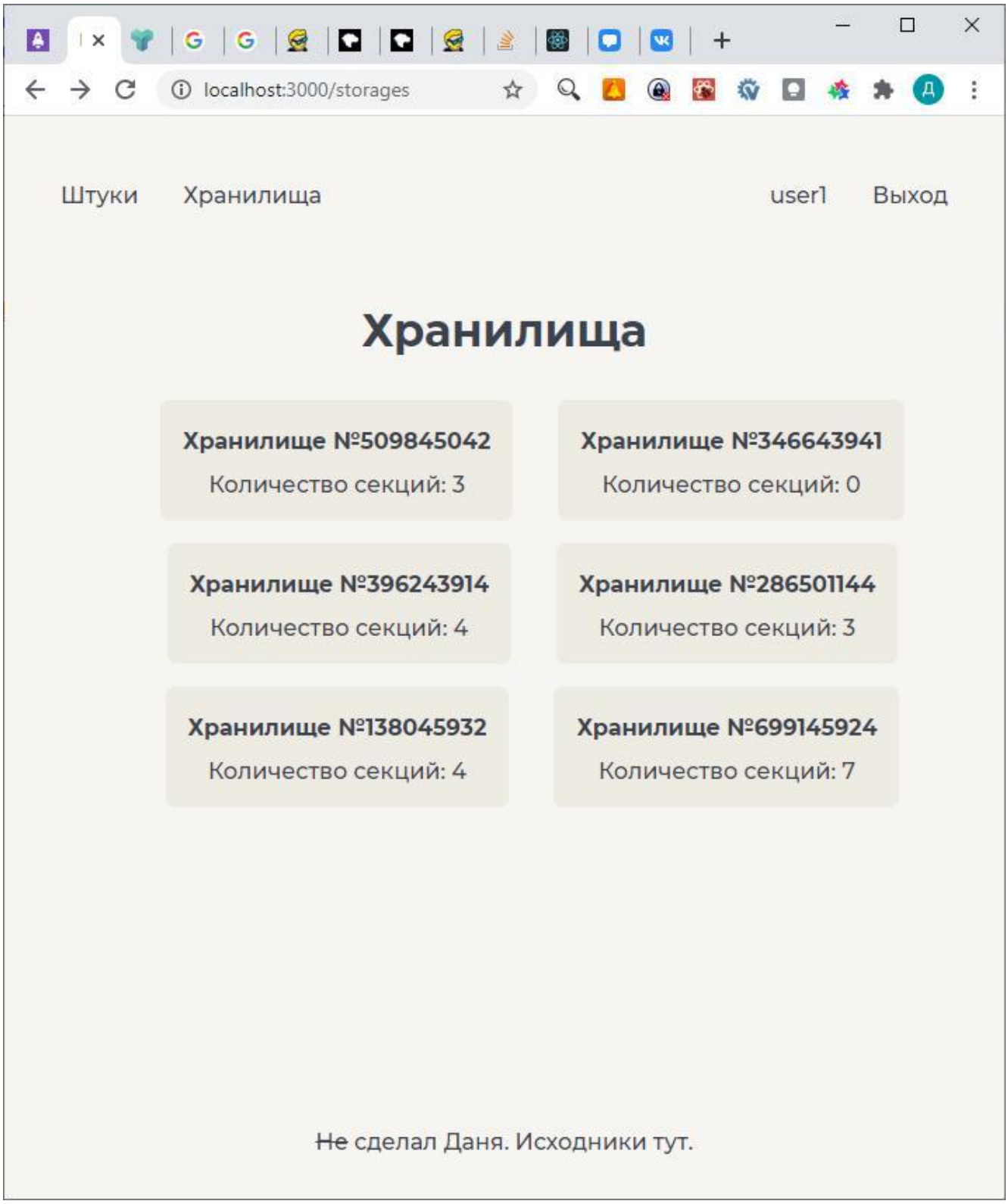

Рис. 1: Пример работы получения данных с API

Приложение

Содержимое файла LoginForm.js

```
import React from 'react';
import {Redirect} from 'react-router-dom'
import {useForm} from "react-hook-form";
import {useAuth} from "../../useAuth";
import {Spinner} from "../Spinner/Spinner";

export default function LoginForm() {
    const {register, errors, handleSubmit} = useForm();
    const auth = useAuth();

    const onSubmit = data => {
        auth.setLoginStatus('loginNone');
        auth.login(data.username, data.password);
    }

    const formBody = <form onSubmit={handleSubmit(onSubmit)}>
        {errors.username && <p className='form-error'>{errors.username.message}</p>}
        {errors.password && <p className='form-error'>{errors.password.message}</p>}
        <div className="form-input">
            <p>Имя пользователя</p>
            <input
                name='username'
                autoComplete='username'
                ref={register({
                    required: "Без имени не получится."
                })}
            />
            <p>Пароль</p>
            <input
                name='password'
                type="password"
                autoComplete='current-password'
                ref={register({
                    required: "Пароль нужно ввести."
                })}
            />
        </div>
        <button type="submit">Войти</button>
    </form>;

    switch (auth.loginStatus) {
        case('loginWait'): {
            return (
                <Spinner spinnerSize='medium'/>
            );
        }
        case('loginError'): {
            return (
                <>
                    <p className='form-error'>Неверный логин или пароль</p>
                    {formBody}
                </>
            );
        }
        case('loginNone'): {
```

```
            return <>{formBody}</>;
        }
        case('loginSuccess'): {
            return <Redirect to='/'/>
        }
    }
}
```

Содержимое файла useApi.js

```
import React, {createContext, useContext} from "react";
import baseUrl from './constants'
import {useAuth} from "./useAuth";
import axios from 'axios'

const apiContext = createContext();

export function ProvideApi({children}) {
    const auth = useProvideApi();
    return <apiContext.Provider value={auth}>{children}</apiContext.Provider>;
}

export const useApi = () => {
    return useContext(apiContext);
};

function useProvideApi() {
    const auth = useAuth();

    const getThing = async (uuid) => {
        let accessToken = await auth.getAccessToken();
        try {
            const response = await axios({
                method: 'get',
                url: baseUrl + `/api/things/${uuid.toString()}`,
                headers: {
                    Authorization: `Bearer ${accessToken}`
                }
            });
            const data = await response.data;
            return data;
        } catch (e) {
            return null
        }
    }

    const getStorage = async (uuid) => {
        let accessToken = await auth.getAccessToken();
        try {
            const response = await axios({
                method: 'get',
                url: baseUrl + `/api/storages/${uuid.toString()}`,
                headers: {
                    Authorization: `Bearer ${accessToken}`
                }
            });
            const data = await response.data;
            return data;
        } catch (e) {
            return null
```

```
    }
}

const postStorage = async (name) => {
    let accessToken = await auth.getAccessToken();
    const username = auth.user.username
    try {
        const response = await axios({
            method: 'post',
            url: baseUrl + `/api/storages/`,
            headers: {
                Authorization: `Bearer ${accessToken}`
            },
            data: {
               name: name,
               owner: username,
               sections: [],
               tags: []
            }
        });
    }catch (e){}
}

const patchStorageName = async (uuid,name) => {
    let accessToken = await auth.getAccessToken();
    try {
        const response = await axios({
            method: 'patch',
            url: baseUrl + `/api/storages/${uuid}/`,
            headers: {
                Authorization: `Bearer ${accessToken}`
            },
            data: {
                name: name
            }
        });
    }catch (e){}
}

const deleteStorage = async (uuid) => {
    let accessToken = await auth.getAccessToken();
    try {
        const response = await axios({
            method: 'delete',
            url: baseUrl + `/api/storages/${uuid}/`,
            headers: {
                Authorization: `Bearer ${accessToken}`
            },
        }).then(
            async ()=>{
                await auth.updateUser()
            }
        );
    }catch (e){}
}

const getSection = async (uuid) => {
    let accessToken = await auth.getAccessToken();
    try {
        const response = await axios({
```

```
            method: 'get',
            url: baseUrl + `/api/sections/${uuid.toString()}`,
            headers: {
                Authorization: `Bearer ${accessToken}`
            }
        });
        const data = await response.data;
        return data;
    } catch (e) {
        return null
    }
}

const getInstance = async (uuid) => {
    let accessToken = await auth.getAccessToken();
    try {
        const response = await axios({
            method: 'get',
            url: baseUrl + `/api/instances/${uuid.toString()}`,
            headers: {
                Authorization: `Bearer ${accessToken}`
            }
        });
        const data = await response.data;
        return data;
    } catch (e) {
        return null
    }
}

const getCategory = async (uuid) => {
    try {
        const response = await axios({
            method: 'get',
            url: baseUrl + `/api/category/${uuid.toString()}`,
            headers: {
                Authorization: `Bearer ${auth.accessToken.toString()}`
            }
        });
        return response.data;
    } catch (e) {
        return null
    }
}

const postThing = async (name,description) => {
    let accessToken = await auth.getAccessToken();
    try {
        const response = await axios({
            method: 'post',
            url: baseUrl + `/api/things/`,
            headers: {
                Authorization: `Bearer ${accessToken}`
            },
            data: {
                category: '',
                name: name,
                description: description,
                tags: [],
                things_instances: []
```

```
                }
            });
        } catch (e) {
            return null
        }
    }

    const postInstance = async (count,thing,section) => {
        let accessToken = await auth.getAccessToken();
        try {
            const response = await axios({
                method: 'post',
                url: baseUrl + `/api/instance/`,
                headers: {
                    Authorization: `Bearer ${accessToken}`
                },
                data: {
                    count: count,
                    type: thing,
                    section: section
                }
            });
        } catch (e) {
            return null
        }
    }

    const getThings = async () => {
        let accessToken = await auth.getAccessToken();
        try {
            const response = await axios({
                method: 'get',
                url: baseUrl + `/api/things/`,
                headers: {
                    Authorization: `Bearer ${accessToken}`
                }
            });
            const data = await response.data
            return data
        } catch (e) {
            return null
        }
    }


    return {
        getStorage,
        postStorage,
        patchStorageName,
        deleteStorage,
        getSection,
        getInstance,
        getThing,
        postThing,
        postInstance,
        getThings
    };
}
```

Содержимое файла useAuth.js

```
import React, {createContext, useContext, useState} from "react";
import baseUrl from './constants'
import axios from 'axios'

const authContext = createContext();

export function ProvideAuth({children}) {
    const auth = useProvideAuth();
    return <authContext.Provider value={auth}>{children}</authContext.Provider>;
}

export const useAuth = () => {
    return useContext(authContext);
};

function useProvideAuth() {
    const [user, setUser] = useState(null)

    const getAccessToken = async () => {
        const accessToken = window.localStorage.getItem('accessToken');
        if (verifyToken(accessToken)) return accessToken;
        else {
            const refreshToken = getRefreshToken();
            try {
                const response = await axios({
                    method: 'post',
                    url: baseUrl + '/auth/jwt/refresh',
                    data: {
                        refresh: refreshToken.toString()
                    }
                })
                const accessToken = await response.data.access;
                window.localStorage.setItem('accessToken', accessToken);
                return accessToken;
            } catch (e) {
                window.localStorage.removeItem('accessToken');
                setLoginStatus('loginNone');
                return null
            }
        }
    }

    const getRefreshToken = () => {
        let refreshToken = window.localStorage.getItem('refreshToken');
        if (verifyToken(refreshToken)) {
            return refreshToken;
        } else {
            window.localStorage.removeItem('refreshToken');
            return null
        }
    }

    const verifyToken = (token) => {
        try {
            const jwt_token = JSON.parse(atob(token.split('.')[1]))
            return ((Date.now() / 1000 | 0) < jwt_token.exp)
        } catch (e) {
            return false
        }
    }
```

```
const login = async (username, password) => {
    try {
        setLoginStatus('loginWait');
        const response = await axios({
            method: 'post',
            url: baseUrl + '/auth/jwt/create',
            data: {
                username: username,
                password: password
            }
        })
        const data = await response.data;
        window.localStorage.setItem('accessToken', data.access);
        window.localStorage.setItem('refreshToken', data.refresh);
        setLoginStatus('loginSuccess');
        const result = await getUser();
        setUser(result);
    } catch (e) {
        console.log('loginError');
        setLoginStatus('loginError');

    }
};

const logout = () => {
    window.localStorage.removeItem('accessToken');
    window.localStorage.removeItem('refreshToken');
    setLoginStatus('loginNone');
};

const [loginStatus, setLoginStatus] = useState(
    verifyToken(getRefreshToken()) ? 'loginSuccess' : 'loginNone'
)


const getUserInfo = async (userId) => {
    const accessToken = await getAccessToken();
    try {
        const response = await axios({
            method: 'get',
            url: baseUrl + `/api/users/${userId.toString()}`,
            headers: {
                Authorization: `Bearer ${accessToken}`
            }
        });
        return response.data;
    } catch (e) {
        return {}
    }
}

const getUser = async () => {
    let accessToken = await getAccessToken();
    try {
        const response = await axios({
            method: 'get',
            url: baseUrl + `/auth/users/me`,
            headers: {
                Authorization: `Bearer ${accessToken}`
```

```
                }
            })
            const userId = await response.data.id;
            const userInfo = await getUserInfo(userId);
            return userInfo
        } catch (e) {
            return null
        }
        ;
    };

    const updateUser = async () => {
        let accessToken = await getAccessToken();
        try {
            const response = await axios({
                method: 'get',
                url: baseUrl + `/auth/users/me`,
                headers: {
                    Authorization: `Bearer ${accessToken}`
                }
            })
            const userId = await response.data.id;
            const userInfo = await getUserInfo(userId);
            setUser(userInfo)
        } catch (e) {
            return null
        }
        ;
    };


    if (loginStatus === 'loginSuccess' && user == null) {
        getUser().then(
            (result) => {
                setUser(result)
            }
        )
    }
    ;

    return {
        loginStatus,
        getAccessToken,
        setLoginStatus,
        updateUser,
        user,
        login,
        logout
    };
}
```