БЫСТРОЕ РАЗВЕРТЫВАНИЕ АУТЕНТИФИКАЦИИ В ДВА ФАЙЛА ДЛЯ ВЕБ-ПРИЛОЖЕНИЯ НА REACT.JS

Воробей И.О.

Институт информационных технологий Белорусского государственного университета информатики и радиоэлектроники
г. Минск, Республика Беларусь

Савенко А.Г. – старший преподаватель, м.т.н.

Аннотация. В работе предложено решение проблемы, когда при разработке программного средства появляются приватные API ресурсы, а функционал для работы с доступом к ним нужно реализовать в кратчайшие сроки, при этом, возможность работы с токеном (token) должны разделять две отличающиеся друг от друга кодовые базы.

Для начала подготовим объект, который будет отвечать за взаимодействие с аутентификацией, и который будет относиться к глобальному состоянию проекта (global state). Назовем этот файл *authSlice.js*. В качестве библиотеки для управления глобальным состоянием выбран Redux Toolkit [1].

При перезагрузке страницы глобальное состояние будет переходить в значения, указанные при инициализации. Поэтому начальное состояние объекта, отвечающего за аутентификацию, будет инициализироваться с помощью функции, в которой будет бизнес логика, отвечающая за получение токена из Local Storage, при наличии этого токена в нем, иначе объект аутентификации будет сообщать программе, что пользователь не вошел в аккаунт. Листинг кода представлен ниже:

```
const composeInitialState = () => {
  const initialState = {
    authenticated: false,
  };
  const token = localStorage.getItem(COGNITO_ACCESS_TOKEN);
  const userId = localStorage.getItem(COGNITO_USER_ID);
  if (token && userId) initialState.authenticated = true;

  return initialState;
};
```

Далее в этом же объекте задаются функции-редьюсеры (reducer functions) [2], которые будут содержать логику, для изменения полей, хранящимся в глобальном объекте. В дальнейшем эти функции можно вызывать из любого модуля проекта.

Листинг кода функции-редьюсера, которая при вызове сообщит программе, что пользователь вошел в свой аккаунт представлена ниже:

```
setAuthenticated: (state, action) => {
    const {
      payload: { token, userId },
    } = action;
    localStorage.setItem(COGNITO_ACCESS_TOKEN, token);;
    localStorage.setItem(COGNITO_USER_ID, userId);;
    state.authenticated = true;
  }
```

И наконец, функция-редьюсер, которая сообщит программе, что пользователь вышел из своего аккаунта:

```
logout: (state) => {
    localStorage.removeItem(COGNITO_ACCESS_TOKEN)
    localStorage.removeItem(COGNITO_USER_ID);
    state.authenticated = false;
  },
```

Теперь необходимо пробросить токен в запрос к приватному ресурсу (REST API Resource). Для этого необходимо создать второй и последний файл *axiosApp.js*. В этом файле будет конфигурация запроса, которую можно использовать из любой точки программы. А для того, чтобы в этой конфигурации использовать токен, необходимо воспользоваться Axios [3] Interceptors [4]. Листинг кода представлен ниже:

```
function authRequestInterceptor(config) {
  const token = localStorage.getItem(COGNITO_ACCESS_TOKEN);
  config.headers.Accept = "application/json";

  if (token) {
    config.headers.authorization = token;
  }

  return config;
}

axiosApp.interceptors.request.use(authRequestInterceptor);
```

Так же с помощью Interceptor'a, в зависимости от кода статуса (code status), который вернет сервер после запроса к нему, реализуем логику выхода из аккаунта, при истекшем сроке действия токена. Листинг кода представлен ниже:

```
axiosApp.interceptors.response.use(
  (response) => {
    return response;
  },
  (error) => {
    if (error.response.status === 401) {
      store.dispatch(logout());
    }
    const message = error.response?.data?.message || error.message;

    return Promise.reject(error);
  }
);
```

Теперь, эти файлы можно интегрировать в проект.
Пример кода для компонента, который отвечает за вход в аккаунт представлен ниже:

```
dispatch(setAuthenticated({ token, userId }));
```

Пример кода, отвечающего за переадресацию пользователя на страницу входа в аккаунт, например, если пользователь открыл приложение впервые либо вышел из аккаунта представлен ниже:

```
useEffect(() => {
    if (!authenticated) {
      navigate("/log-in");
    } else {
      if (location.pathname === "/ log-in ") {
        navigate("/");
      }
    }
  }, [authenticated, location.pathname, navigate]);
```

Таким образом, было реализовано быстрое развертывание аутентификации для веб-приложений на фреймворке ReactJS.

Список использованных источников:

1. Redux Toolkit Core Concepts [Электронный ресурс]. Режим доступа: https://redux.js.org/introduction/core-concepts. Дата доступа: 05.04.2022.

2. Writing Reducers [Электронный ресурс]. Режим доступа: https://redux.js.org/tutorials/fundamentals/part-3-state-actions-reducers#writing-reducers. Дата доступа: 05.04.2022.

3. Axios Getting Started [Электронный ресурс]. Режим доступа: https://axios-http.com/docs/intro. Дата доступа: 05.04.2022.

4. Interceptors [Электронный ресурс]. Режим доступа: https://axios-http.com/docs/interceptors. Дата доступа: 05.04.2022.