Практическая работа №6. Аутентификация и автоизация в React-приложениях

Цель практической работы по аутентификации и авторизации в React-приложениях заключается в создании безопасного и защищенного способа идентификации пользователей и предоставления доступа к определенным частям приложения на основе их ролей и разрешений.

Задачи:

- Создать или дополнить существующие React-приложение формой аутоидентификации пользователей с функционалом авторизации.
- Использовать поток кода авторизации PKCE.

Сдача работы:

- Показать выполненное задание во время проведения занятия.
- Подготовить и сдать отчет по практической работе.

Теоретические сведения

Большинству приложений требуются механизмы аутентификации и механизмы авторизации. Аутентификация и авторизация являются ключевыми компонентами безопасности веб-приложений. Когда механизм аутентификации подтверждает, что определенные объекты являются законными пользователями, механизм авторизации определяет, разрешено ли пользователю выполнять эти операции на основе роли и разрешений пользователя. Иными словами, ***аутентификация*** - это процесс проверки учетных данных пользователя для подтверждения его личности . ***Авторизация*** же - это процесс определения прав доступа пользователя к определенным ресурсам и функциональности веб-приложения. Не стоит путать эти два термина между собой.

В большинстве случаев нам обычно не нужны специальные модули или библиотеки для обработки авторизации, достаточно большинства служебных функций, которые обычно присутствуют в стандартных библиотеках. Предлагаемое решение для проверки или авторизации в приложении может отличаться в зависимости от особенностей проекта. Вы можете решить поместить статус пользователя в Redux для управления, вы также можете создать специальный модуль и т.д.

Простой механизм авторизации

Предположим, у нас есть пользовательский объект, этот объект обычно получается путем вызова терминала после входа в систему. Он имеет следующую структуру, представленную в листинге 6.1.

```
const user = {
  name: 'Paimon',
  // ...
  roles: ['user'],
  rights: ['can_view_articles']
    };
```

У пользователей есть несколько разрешений, которые можно сгруппировать по ролям. Для вашего приложения вам могут потребоваться только роли, или только разрешения, или и то, и другое, это не проблема. REST API может предоставлять вам разрешения, вложенные в роли, это не проблема. Помните, что вам необходимо настроить решение в соответствии с вашими потребностями. Самое главное, что у нас есть пользовательский объект.

Затем мы создаем *auth.js* файл. Существуют некоторые инструменты и методы, которые могут помочь нам проверить авторизацию пользователя.

Листинг 6.2 - auth.js

```
export const isAuthenticated = user => !!user;

export const isAllowed = (user, rights) =>
  rights.some(right => user.rights.includes(right));

export const hasRole = (user, roles) =>
  roles.some(role => user.roles.includes(role));
```

Используя прототип массива *somecincludesMethod*, мы проверили, есть ли у клиента хотя бы одно разрешение или роль для UF, которых должно быть достаточно для выполнения некоторых базовых проверок разрешений.

Поскольку пользовательский объект может храниться где угодно, взяв в качестве примера Redux, мы позволяем передавать его функции в качестве параметра.

В итоге мы создали простой компонент React, который использует *auth.js* чтобы иметь возможность отображать различные части интерфейса в соответствии с условиями.

Условная маршрутизация

При использовании React Router, можно использовать тот же метод для визуализации маршрутов.

Листинг 6.3 - Пример маршрутизации и авторизации

```
import React from 'react';
import { BrowserRouter, Route } from 'react-router-dom';

const App = ({ user }) => (
  <BrowserRouter>
  <Route>{hasRole(user, ['user']) && <Route path='/user'
component={User} />}
  {hasRole(user, ['admin']) && <Route path='/admin'
component={Admin} />}
      <Route exact path='/' component={Home} />
    </Route>
  </BrowserRouter>
);
```

Дополнительный материал

Списки и ключи - https://ru.reactjs.org/docs/lists-and-keys.html (дата обращения: 01.02.2022).

Вызов API Microsoft Graph из приложения React - https://docs.microsoft.com/ru-ru/azure/active-directory/develop/tutorial-v2-react (дата обращения: 01.02.2022).