Столыпинский вестник

Научная статья

Original article

УДК 004

БЕСПАРОЛЬНАЯ АУТЕНТИФИКАЦИЯ В REACT + MOJOAUTH

PASSWORD-FREE AUTHENTICATION IN REACT + MOJOAUTH

Берьянов Максим Сергеевич, Магистрант, 2 курс, Факультет ПИиКТ

Университет ИТМО Россия, г. Санкт-Петербург

Монченко Артем Сергеевич, Магистрант, 2 курс, Факультет ПИиКТ

Университет ИТМО Россия, г. Санкт-Петербур

Богданов Илья Александрович, Магистрант, 2 курс, Факультет ПИиКТ

Университет ИТМО Россия, г. Санкт-Петербург

Buryanov Maxim Sergeevich, berjnov.ru@mail.ru

Monchenko Artem Sergeevich, berjnov.ru@mail.ru

Bogdanov Ilya Alexandrovich, berjnov.ru@mail.ru

Аннотация. Аутентификация без пароля – это новая улучшенная альтернатива традиционному способу аутентификации с использованием имени пользователя и пароля. Она повышает общую безопасность и снижает затраты для компании. Создать приложение с подобной аутентификацией с нуля может быть сложно, но MojoAuth берет на себя многие нетривиальные аспекты рассматриваемой технологии, предоставляя готовое решение и обеспечивая

гибкость выбора режима аутентификации. В данной статье будет произведено исследование создания React-приложения с использованием данного сервиса.

Abstract. Passwordless authentication is a new improved alternative to the traditional username and password authentication method. It improves overall security and reduces costs for the company. It can be difficult to create an application with this kind of authentication from scratch, but MojoAuth takes care of many non-trivial aspects of the technology in question, providing a turnkey solution and flexibility to choose an authentication mode. This article will explore the creation of a React application using this service.

Ключевые слова: веб-разработка, React.JS, аутентификация, MojoAuth.

Keywords: web development, React.JS, authentication, MojoAuth.

Понятие беспарольной аутентификации

Аутентификация без пароля [1], как следует из названия — это способ аутентификации пользователей без необходимости использования паролей. Вместо этого для аутентификации используется какой-либо однозначный идентификатор, например адрес электронной почты или номер телефона.

Затем на данный идентификатор отправляется «магическая» ссылка [2] или одноразовый пароль [3], который используется для аутентификации пользователя. Каждый запрос генерирует новую ссылку или OTP и делает недействительными все предыдущие успешные аутентификации для обеспечения безопасности.

Аутентификация без пароля работает практически как двухфакторная аутентификация, поскольку для этого требуется, чтобы вы имели при себе свой адрес электронной почты или номер телефона.

Проверка подлинности без пароля — лучший выбор для современного приложения, поскольку она устраняет необходимость в паролях. Поскольку пользователь создает учетную запись на нескольких сайтах, сложность паролей в

общем случае становится все ниже. Пользователи предпочитают удобство, нежели максимальную безопасность, поэтому они часто используют слабые или одинаковые пароли для своих учетных записей, тем самым повышая уязвимость своего аккаунта на определенном сервисе.

В этой статье мы получим базовое представление об аутентификации без пароля и научимся включать ее в приложение React [4]. Мы создадим интуитивно понятное приложение, управляющее пользовательскими маршрутами для лучшего пользовательского взаимодействия.

Приложение на React поддерживает принципы, которых придерживается сама библиотека. Безусловно, в качестве одного и принципов можно выделить декларативность – при разработке довольно просто создавать интерфейсы, части которых будут автоматически обновляться при изменении данных. Более того, в силу определенно вступает упомянутая концепция компонентов – возможно создавать инкапсулированные компоненты с собственным состоянием, а затем объединять их в сложные пользовательские интерфейсы.

В первую очередь, необходимо проверить, установлен ли на компьютере инструмент Node.JS с помощью команд *node -v* и *npm -v* в терминале. В случае отображения установленной версии, можно продолжать дальнейшую настройку проекта, иначе Node.JS нужно скачать с официального сайта (Node Package Manager установится автоматически).

Исследование процесса интеграции

Для демонстрации мы будем использовать библиотеку MojoAuth [5] в качестве службы аутентификации без пароля. Для начала нужно создать учетную запись на MojoAuth. После этого нужно инициализировать React проект с помощью приложения create-react-app: *npx create-react-app mojoauth-demo*. Данная технология облегчает нам задачу по настройке множества аспектов разработки, к примеру, таких как сборщик проекта, его окружение и другие. Иными словами,

запустив данную команду мы получаем уже готовый проект «из коробки». Далее нужно установить обсуждаемую нам библиотеку: *npm install mojoauth-web-sdk*. В качестве UI, мы будем использовать компоненты Semantic UI [6]: *npm install semantic-ui-react semantic-ui-css*. Чтобы добавить маршрутизацию в проект, необходимо установить библиотеку React Router [7]: *npm install react-router-dom*.

В панели инструментов MojoAuth необходимо получить свой ApiKey и добавить URL-адрес проекта в список разрешенных в разделе настроек. Также, предыдущие данные нужно указать в файле *config.js*, который нужно создать в папке *src* (рисунок 1).

Рисунок 1 – Конфигурационный файл

Теперь приступим к созданию интерфейса *Login* пользователя. В папке *src* необходимо создать еще одну папку с именем *login*, где будет находиться файл с именем *index.js*. Этот файл будет содержать необходимый код.

В хуке *useEffect* инициализируем объект *MojoAuth* и вызовем метод *signIn*. Сохраним токен доступа и идентификатор в локальном хранилище для дальнейшего использования (рисунок 2).

В компоненте панели мониторинга мы хотим войти в систему, когда пользователь перенаправляется с идентификатором состояния на URL-адрес перенаправления, указанный в компоненте входа.

Во-первых, импортируем необходимые компоненты. Во-вторых, в

```
const Dashboard = () => {
  const search = useLocation();
  const params = search ? QueryString.parse(search.search) : {};

  useEffect( effect: () => {
    if (params.state_id) {
      const mojoauth = new MojoAuth(config.api_key);
      mojoauth.signInWithStateID(params.state_id).then(payload => {
        localStorage.setItem('React-AccessToken', payload.oauth.access_token);
useEffect( effect: () => {
  let mojoconfig = {
    language: "en",
    source: [{type: 'phone', feature: 'otp'}, {type: 'email', feature: 'magiclink'}],
    redirect_url: config.redirect_url
  }

  const mojoauth = new MojoAuth(config.api_key, mojoconfig);
  localStorage.getItem( key: 'React-AccessToken') ? navigate('/dashboard') :
    mojoauth.signIn().then(payload => {
      localStorage.setItem('React-AccessToken', payload.oauth.access_token)
      localStorage.setItem('React-Identifier', payload.user.identifier)
    });
}, deps: []);
```

Рисунок 2 – Объект входа MojoAuth

компоненте *Dashboard* получаем *state_id* из URL-адреса. Далее, в хуке *useEffect*

```
return (<>
  <div className='container'>
    <h1 className='main-title'>React + MojoAuth Research Article </h1>
    <div className='login-container'>
      <h1 className='title'>Добро пожаловать!</h1>
      <h4 className='subtitle'>Войдите, используя MojoAuth</h4>
      <Modal
        onClose={() => setOpen( value: false)}
        onOpen={() => setOpen( value: true)}
        open={open}
        dimmer={true}
        basic={true}
        size='tiny'
        trigger={<Button className='login' primary>Войти</Button>}
      >
        <div id='mojoauth-passwordless-form'></div>
      </Modal>
      <p className='footer'>Используя MojoAuth</p>
    </div>
  </div>
</>);
```

Рисунок 4 – DOM-объект входа

входим в систему, используя идентификатор состояния и вызывая функцию *signInWithStateID* (рисунок 3).

Создание интерфейса

Теперь мы хотим показать форму входа во всплывающем окне. Для этого мы будем использовать сущность *Modal* из Semantic UI. Для этого нужно определить состояние, которое будет содержать параметры открытия и закрытия всплывающего (модального) окна.

В качестве возвращаемого значения компонента *Login* обозначим модальное окно, чтобы отобразить его при нажатии на кнопку входа (логина), как это можно заметить на рисунке 4. Для того, чтобы экземпляр *MojoAuth* запускался только при

открытии всплывающего окна, устанавливаем условие *if (open) {}* в функции *useEffect*.

В компоненте *Dashboard* покажем имя пользователя (идентификатор), если пользователь вошел в систему. Кроме того, также отобразим кнопку выхода, которая выведет пользователя из системы и удалит токен доступа из локального хранилища (рисунок 5).

После того, как наши компоненты логина и панели управления (*Login, Dashboard*) готовы, наступает очередь маршрутизации. В файле *app.js* предлагается использовать маршруты из загруженной в начале данной статьи библиотеки *react-*

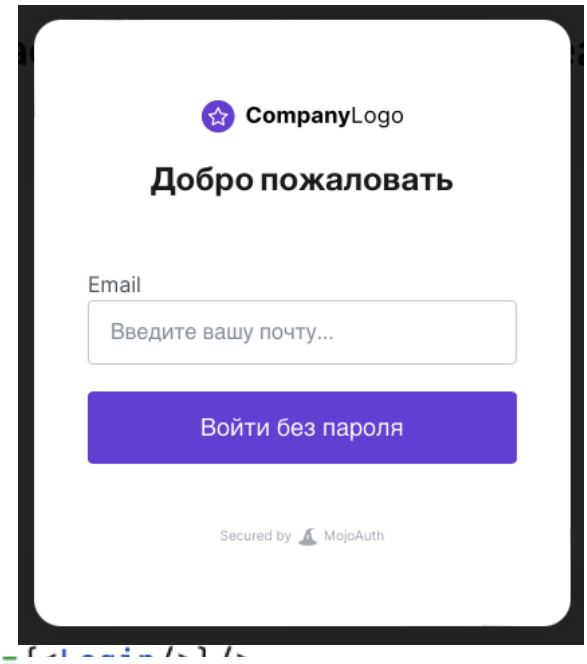

Рисунок 7 – Ввод почтового ящика

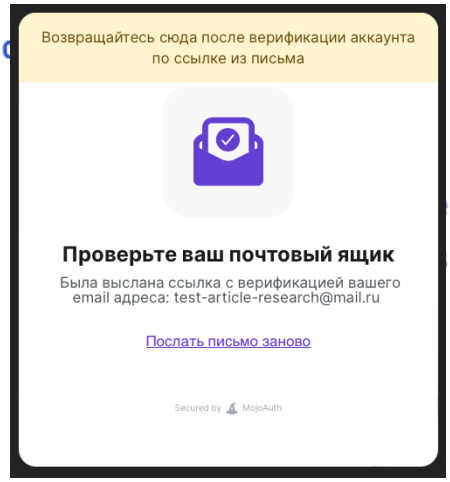

Рисунок 8 – Ожидание

Маршрутизация в приложении

router-dom. Данные маршруты будут выводить созданные ранее компоненты в зависимости от названия пути. Подробнее это можно увидеть на рисунке 6.

```
return (
  <>
    <div className='container'>
      <h1 className='main-title'>React + Mo
      <h3 className="main-subtitle">Вы успеш
      <div className='login-container'>
        <h1 className='title'>Добро пожалова
        <h4 className='subtitle'>{localStora
        <Button className='login' primary on
          localStorage.removeItem( key: 'Reac
        }}>Logout</Button>
        <p className='footer'>Используя Moj
      </div>
    </div>
  </>
```

Рисунок 6 – DOM-объект успешного входа

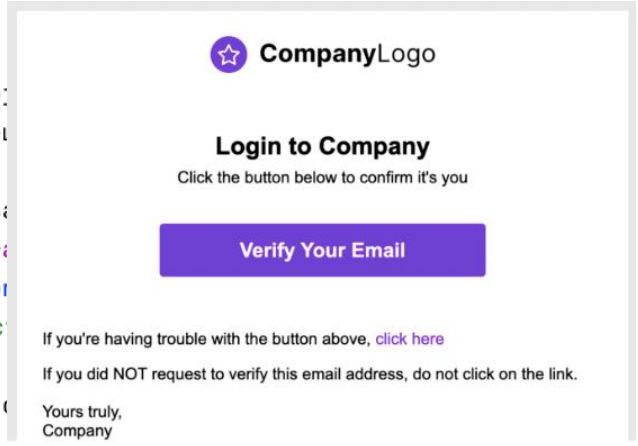

Рисунок 9 – Верификационное письмо

Таким образом, можно запустить приложение, используя команду *npm start*. Главный результат данного исследования заключается в полностью работающем приложении на библиотеке React, предоставляющем важную функцию беспарольной аутентификации с помощью почтового ящика. Примеры работы представлены на рисунках 7 и 8. Письмо, высланное на почтовый ящик, можно увидеть на рисунке 9. После того, как произойдет переход по ссылке, предложенный почтовый ящик будет подтвержден, и мы получим новый идентификатор состояния, тем самым достигая необходимого функционала от приложения в виде беспарольной аутентификации по email.

Таким образом, было продемонстрировано использование современных технологий в области веб-разработки интерфейсов и, в частности, контексте аутентификации. Удобство беспарольной аутентификации, несомненно, является полновесным доказательством того, что данный принцип аутентификации будет распространяться на многие приложения и веб-сервисы в будущем. Подобные

сервисы, как MojoAuth помогают реализовывать данный принцип, и, как мы убедились ранее, достаточно тривиальным и проверенным способом.

Библиографический список

1. Беспарольная аутентификация [Электронный ресурс] URL: https://habr.com/ru/company/vk/blog/343288/ (дата обращения: 08.11.2022).
2. Отправляем магические ссылки с помощью Node.js [Электронный ресурс] URL: https://habr.com/ru/company/vk/blog/343288/ (дата обращения: 08.11.2022).
3. TOTP (Алгоритм Time-based One-Time Password) [Электронный ресурс] URL: https://habr.com/ru/post/534064/ (дата обращения: 09.11.2022).
4. React.js [Электронный ресурс] URL: https://reactjs.com (дата обращения: 09.11.2022).
5. Mojo.Auth [Электронный ресурс] URL: https://mojoauth.com/docs/ (дата обращения: 09.11.2022).
6. Semantic UI [Электронный ресурс] URL: https://semantic-ui.com/ (дата обращения: 09.11.2022).
7. React Router [Электронный ресурс] URL: https://reactrouter.com/en/main (дата обращения: 09.11.2022).

Bibliographic list

1. Password-free authentication [Electronic resource] URL: https://habr.com/ru/company/vk/blog/343288 / (accessed: 08.11.2022).
2. We send magic links using Node.js [Electronic resource] URL: https://habr.com/ru/company/vk/blog/343288 / (accessed: 08.11.2022).
3. TOTP (Time-based One-Time Password Algorithm) [Electronic resource] URL: https://habr.com/ru/post/534064 / (accessed: 09.11.2022).
4. React.js [Electronic resource] URL: https://reactjs.com (accessed: 09.11.2022).

5. Mojo.Auth [Electronic resource] URL: https://mojoauth.com/docs / (accessed: 09.11.2022).
6. Semantic UI [Electronic resource] URL: https://semantic-ui.com / (accessed: 09.11.2022).
7. React Router [Electronic resource] URL: https://reactrouter.com/en/main (accessed: 09.11.2022).

Для цитирования: Берьянов М.С., Монченко А.С., Богданов И.А. БЕСПАРОЛЬНАЯ АУТЕНТИФИКАЦИЯ В REACT + MOJOAUTH // Научный сетевой журнал «Столыпинский вестник» №9/2022.