Как создать безголовый WordPress сайт с помощью React.js

Описание

WordPress – одна из самых популярных систем управления контентом (CMS), которой пользуются 810 миллионов человек – это 41% всего интернета! Это лучший выбор для тех, кто хочет быстро создать сайт, потому что он прост, удобен в использовании, предлагает широкий спектр возможностей настройки, а также имеет сильную экосистему плагинов и других ресурсов. Один из способов максимально использовать возможности WordPress – это “безголовый” подход.

Безголовая CMS, также известная как headless system, – это тип бэкенд CMS, который используется исключительно для управления контентом. Это помогает интегрировать контент в любую систему или сайт, просто обращаясь к API безголовой CMS. Однако при этом фронтенд остается для отдельного управления. Именно здесь на помощь приходит API. API позволяют двум или более различным приложениям обмениваться данными. В данном случае API используется для передачи данных из CMS на внешний сайт, что обеспечивает большую гибкость и более высокую производительность. В этой статье мы рассмотрим, что такое безголовая CMS, почему вы можете захотеть ее создать и как настроить ее для WordPress.

Что такое безголовый WordPress?

Безголовый сайт WordPress – это тип сайта, который в основном использует WordPress в качестве CMS, или системы управления контентом, и использует другие технологии фронтенда, такие как React или Vue для фронтенда. Библиотеки и

фреймворки JavaScript используются для отображения содержимого сайта. Таким образом, безголовый WordPress имеет отдельный фронтенд и бэкенд, а для связи используется API. Проще говоря, безголовая архитектура означает, что CMS используется только для хранения и управления контентом, а фронтенд сайта ее не волнует.

С другой стороны, основной задачей фронтенда является отображение контента, а ему, в свою очередь, все равно, где хранится или управляется контент, лишь бы он мог до него добраться. Безголовый WordPress имеет более высокую производительность, поскольку запросы фронтенда обрабатываются более быстрыми технологиями, такими как React, и только бэкенд управляется WordPress. Разделение фронтенда и бэкенда позволяет масштабировать компоненты независимо друг от друга.

Плюсы и минусы безголового WordPress

Как и во всех других вариантах разработки, у безголового решения WordPress есть как преимущества, так и недостатки. Важно понять и то, и другое, прежде чем принимать решение.

Плюсы безголового WordPress

Среди основных преимуществ внедрения безголового WordPress можно выделить следующие:

- **Гибкость:** Безголовый WordPress позволяет разработчикам создавать пользовательские фронтенды, не будучи ограниченными традиционной системой тем WordPress. Это означает, что вы можете создавать уникальные пользовательские интерфейсы и опыт для конкретных потребностей.
- **Производительность:** Отделение фронтенда от бэкенда сайта WordPress может ускорить загрузку сайта и повысить его производительность, поскольку сервер предоставляет данные только через API, а не отрисовывает HTML для каждого запроса.
- **Безопасность:** Разделяя фронтенд и бэкенд, безголовый WordPress может обеспечить дополнительный уровень безопасности, ограничивая доступ к основной кодовой базе WordPress и базе данных.

Минусы безголового WordPress

К недостаткам безголового WordPress можно отнести:

- **Отсутствие тем:** Поскольку безголовый WordPress не полагается на предварительно созданные темы, вам придется создавать свои собственные темы. Это может занять много времени и потребовать дополнительных ресурсов.
- **Стоимость:** Создание безголового сайта WordPress может быть дороже, чем традиционного сайта WordPress, поскольку для его создания и поддержки требуется больше технических знаний и ресурсов.
- **Ограничения плагинов:** Некоторые плагины WordPress могут не работать с безголовым WordPress, поскольку их правильное функционирование зависит от тем WordPress.

Что такое WordPress REST API?

WordPress REST API используется в качестве интерфейса между бэкендом и фронтендом. С помощью API можно легко отправлять и извлекать данные с сайта, поскольку API имеет контрольный доступ к данным сайта. Он также гарантирует, что только авторизованные пользователи могут взаимодействовать с ним. API может поддерживать широкий спектр форматов данных, включая JSON, что упрощает взаимодействие с системой.

WordPress REST API – это мощный инструмент для разработчиков, позволяющий создавать, обновлять или удалять данные в дополнение к созданию пользовательской функциональности для сайтов или интеграции с другим сервисом. Более того, существуют плагины, которые расширяют функциональность API, например, интегрируют дополнительные методы аутентификации.

Что такое React?

React – это библиотека JavaScript для создания пользовательских интерфейсов. Это многократно используемая библиотека фронтенда с открытым исходным кодом, которая позволяет разработчикам создавать компоненты пользовательского интерфейса (UI) с помощью декларативного синтаксиса. React создает интерактивный пользовательский интерфейс и отображает компоненты при

изменении данных.

Эта библиотека пользуется большой популярностью среди разработчиков, желающих создавать сложные, многократно используемые компоненты, эффективно управлять состоянием и легко обновлять пользовательский интерфейс в режиме реального времени. Сильное сообщество разработчиков React создало набор инструментов, библиотек и ресурсов для расширения функциональности библиотеки. Многие организации и предприятия используют React в качестве технологии для создания сложных, динамичных и быстродействующих веб-приложений.

Почему стоит использовать React?

React обладает множеством преимуществ, которые делают его отличным выбором для разработки сложных и динамичных веб-приложений.

Вот некоторые из ключевых преимуществ использования React:

- **Декларативный синтаксис:** React использует декларативный подход к созданию компонентов пользовательского интерфейса, что позволяет легко создавать многократно используемые и высокоэффективные компоненты.
- **Большое сообщество и экосистема:** React имеет большое и активное сообщество разработчиков, что привело к созданию широкого спектра инструментов и библиотек для расширения его функциональности.
- **Виртуальный DOM:** React использует виртуальный DOM для обновления пользовательского интерфейса в режиме реального времени. Это означает, что при изменении состояния он обновляет только те компоненты, которые должны быть изменены, а не перерисовывает всю страницу.
- **Возможность повторного использования:** React.js предлагает многократно используемые компоненты, которые можно использовать в различных приложениях, что значительно сокращает время и усилия на разработку.

Как создать безголовый сайт WordPress с помощью React

Теперь пришло время испачкать руки и узнать, как создать и развернуть безголовый сайт WordPress с помощью React. Продолжайте читать, чтобы погрузиться в процесс.

Предварительные условия

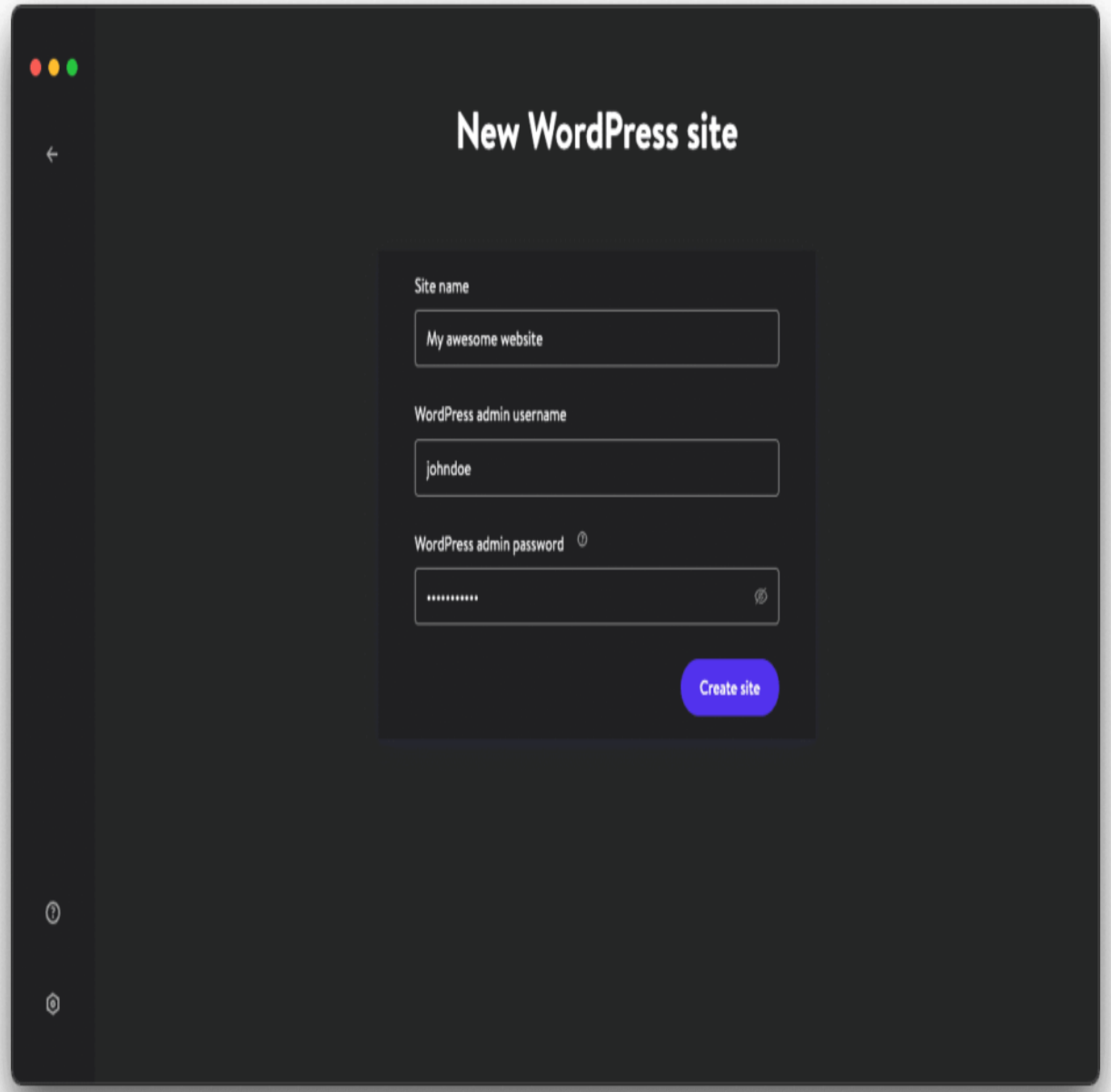
Прежде чем приступить к этому руководству, убедитесь, что у вас есть:

- Хорошее понимание React
- Node.js v14 или выше установлен на вашей машине

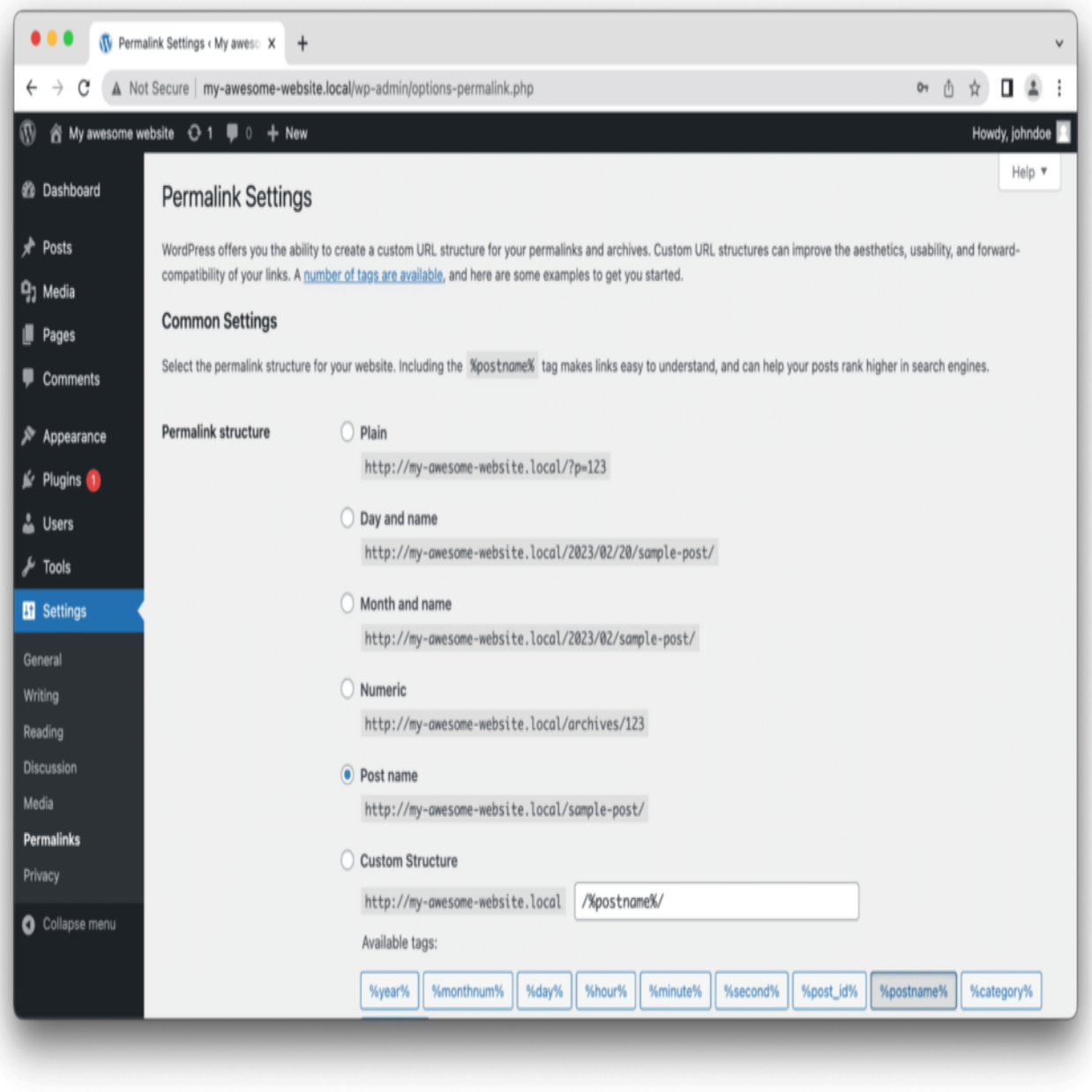
Шаг 1. Настройка веб-сайта WordPress

Давайте начнем с настройки сайта WordPress, так как он будет служить источником

данных для приложения React. Если у вас уже настроен сайт WordPress, вы можете пропустить этот раздел, в противном случае просто следуйте дальше. В этом руководстве мы будем использовать DevKinsta, широко используемую быструю и надежную локальную среду разработки для создания, развития и развертывания сайтов WordPress. Она совершенно бесплатна в использовании – просто скачайте ее с официального сайта и установите на свою систему. После завершения установки откройте приборную панель DevKinsta и нажмите New WordPress site, чтобы создать новый сайт WordPress. Заполните все необходимые данные и нажмите кнопку Создать сайт, чтобы продолжить.



Это может занять несколько минут, но как только ваш сайт будет создан, вы сможете получить доступ к нему и его админ-панели, нажав кнопку Открыть сайт” или опции WP Admin соответственно. Далее, чтобы включить JSON API, вам нужно будет обновить пермалинки вашего сайта. В панели администратора WordPress нажмите на Настройки, а затем на Permalinks. Выберите опцию Post name и нажмите Save Changes.



Вы также можете использовать такие инструменты, как Postman, чтобы легко тестировать и отправлять запросы к REST API WordPress.

Шаг 2: Настройка приложения React

Теперь, когда мы настроили наш сайт WordPress, мы можем начать работу над фронтендом. Как упоминалось выше, в этом учебнике мы будем использовать React для фронтенда нашего приложения. Чтобы начать работу, запустите приведенный ниже код в терминале, чтобы создать приложение React.

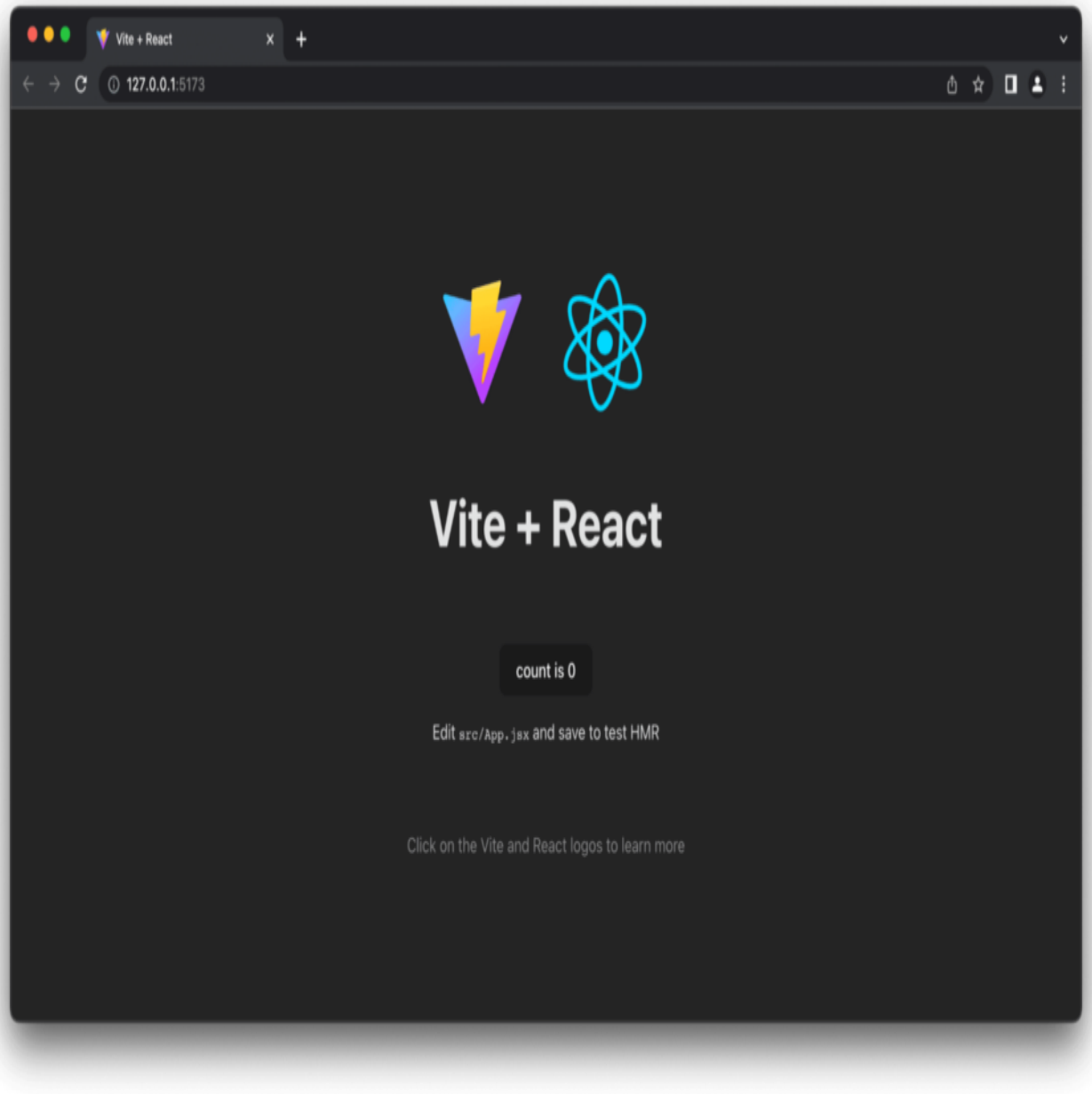
```
npm create vite@latest my-blog-app
cd my-blog-app
```

```
npm install
```

Приведенные выше команды создадут приложение React и установят необходимые зависимости. Нам также необходимо установить Axios, библиотеку JavaScript для выполнения HTTP-запросов. Выполните приведенную ниже команду для ее установки.

```
npm install axios
```

Теперь, чтобы запустить сервер разработки, вы можете запустить npm run dev в терминале. После этого сервер должен инициализировать ваше приложение по адресу http://127.0.0.1:5173.



Далее откройте проект в любимом редакторе кода и удалите все ненужные файлы, такие как папка assets, index.css и app.css. Вы также можете заменить код внутри main.jsx и App.jsx следующим кодом:

```
// main.jsx
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App'

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
```

```
)

// App.jsx

import React from 'react'

export default function App() {
  return (
    <App />
  )

}
```

Шаг 3: Получение сообщений из WordPress

Теперь пришло время получить сообщения с нашего сайта WordPress.

Внутри App.jsx добавьте указанное ниже состояние и импортируйте <code>useState</code> из React:

```
const [posts, setPosts] = useState([])
```

useState – это встроенный хук в React, который используется для добавления состояний в компонент, причем состояние относится к данным или свойству. posts используется для получения данных из состояния, а setPosts – для добавления новых данных в post. Мы также передали пустой массив в state по умолчанию. Далее добавьте следующий код после state, чтобы получить посты из WordPress REST API:

```
const fetchPosts = () => {
    // Using axios to fetch the posts
    axios
      .get("http://headless-wordpress-website.local/wp-json/wp/v2/posts")
      .then((res) => {
        // Saving the data to state
        setPosts(res.data);
      });
  }


  // Calling the function on page load
  useEffect(() => {
    fetchPosts()
  },
[])
```

Приведенный выше код запускает функцию fetchPosts() при загрузке страницы. Внутри функции fetchPosts() мы отправляем GET-запрос к API WordPress, используя

Axios, чтобы получить посты, а затем сохранить их в состоянии, которое мы объявили ранее.

Шаг 4: Создание компонента блога

В корневом каталоге создайте новую папку с именем components, а в ней – два новых файла: blog.jsx и blog.css.

Сначала добавьте следующий код в файл blog.jsx:

```
import axios from "axios";
import React, { useEffect, useState } from "react";
import "./blog.css";

export default function Blog({ post }) {
  const [featuredImage, setFeaturedimage] = useState();

  const getImage = () => {
    axios
     .get(post?._links["wp:featuredmedia"][0]?.href)
     .then((response) => {
      setFeaturedimage(response.data.source_url);
    });
  };

  useEffect(() => {
    getImage();
  }, []);

  return (
    <div class="container">
      <div class="blog-container">
        <p className="blog-date">
          {new Date(Date.now()).toLocaleDateString("en-US", {
            day: "numeric",
            month: "long",
            year: "numeric",
          })}
        </p>
        <h2 className="blog-title">{post.title.rendered}</h2>
        <p
          className="blog-excerpt"
          dangerouslySetInnerHTML={{ __html: post.excerpt.rendered }}
        />
        <img src={featuredImage} class="mask" />
      </div>
    </div>
  );

}
```

В приведенном выше коде мы создали компонент карточки, который принимает свойство posts, содержащее информацию о записи блога из WordPress API. В функции getImage() мы используем Axios для получения URL-адреса главного изображения, а затем сохраняем эту информацию в state. Затем мы добавили хук useEffect для вызова функции getImage() после установки компонента. Мы также добавили оператор return, в котором мы выводим данные поста, заголовок, отрывок и изображение.

Далее добавьте приведенные ниже стили в файл blog.css:

```
@import url("https://fonts.googleapis.com/css?family=Poppins");

.blog-container {
  width: 400px;
  height: 322px;
  background: white;
  border-radius: 10px;
  box-shadow: 0px 20px 50px #d9dbdf;
  -webkit-transition: all 0.3s ease;
  -o-transition: all 0.3s ease;
  transition: all 0.3s ease;
}

img {
  width: 400px;
  height: 210px;
  object-fit: cover;
  overflow: hidden;
  z-index: 999;
  border-bottom-left-radius: 10px;
  border-bottom-right-radius: 10px;
}

.blog-title {
  margin: auto;
  text-align: left;
  padding-left: 22px;
  font-family: "Poppins";
  font-size: 22px;
}

.blog-date {
  text-align: justify;
  padding-left: 22px;
  padding-right: 22px;
  font-family: "Poppins";
  font-size: 12px;
  color: #c8c8c8;
  line-height: 18px;
  padding-top: 10px;
}
```

```
.blog-excerpt {
  text-align: justify;
  padding-left: 22px;
  padding-right: 22px;
  font-family: "Poppins";
  font-size: 12px;
  color: #8a8a8a;
  line-height: 18px;
  margin-bottom: 13px;

}
```

Затем, в файле App.jsx, добавьте следующий код в оператор возврата для рендеринга компонента блога:

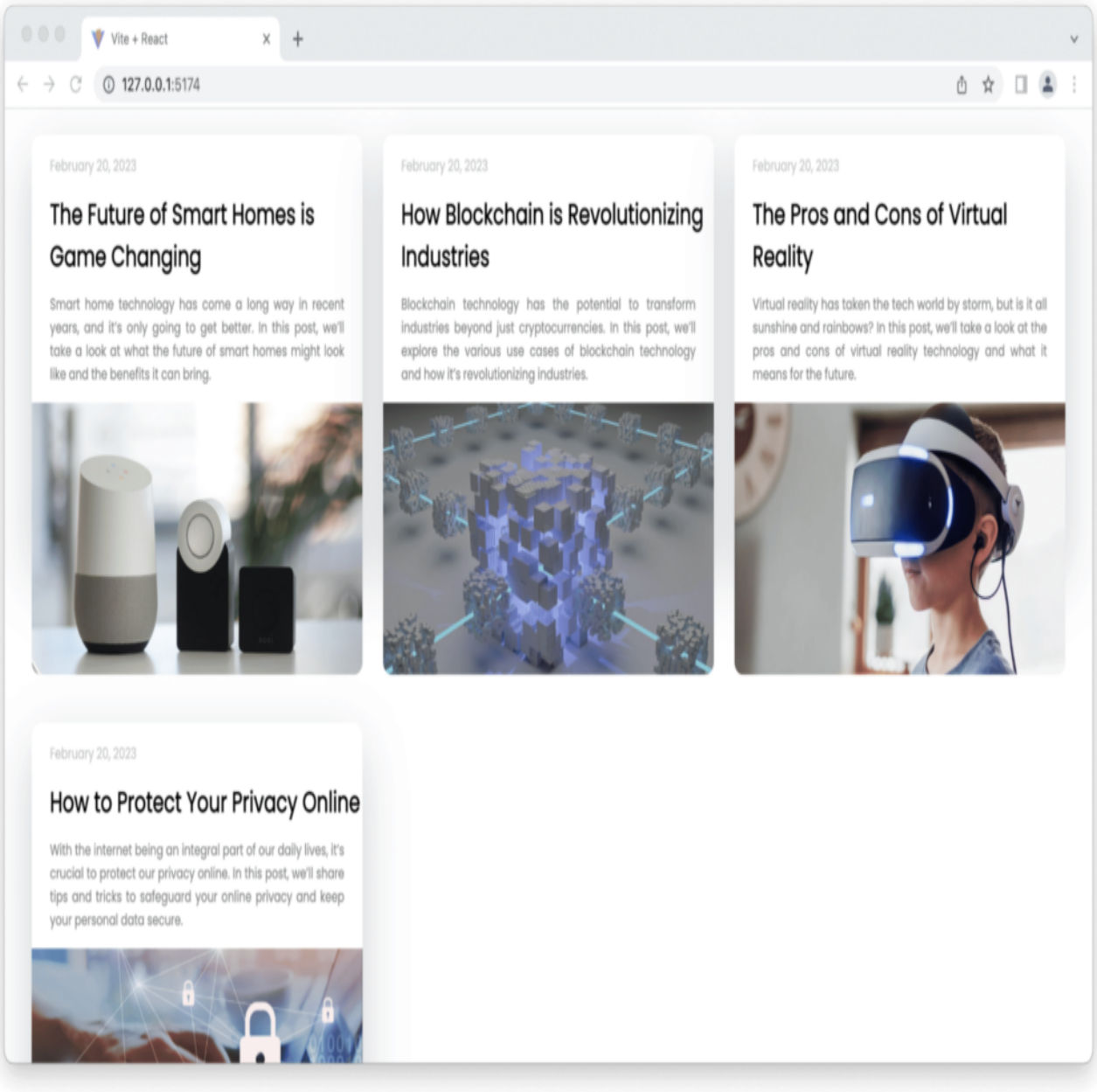
```
<div>
 {posts.map((item) => (
  <Blog post={item} />
 ))}

</div>;
```

Наконец, вот как должен выглядеть ваш App.jsx:

```
import React, { useEffect, useState } from 'react'
import axios from "axios"
import Blog from './components/Blog';

export default function App() {
  const [posts, setPosts] = useState([]);

  const fetchPosts = () => {
    axios
      .get("http://my-awesome-website.local/wp-json/wp/v2/posts")
      .then((res) => {
        setPosts(res.data);
      });
  }

  useEffect(() => {
    fetchPosts()
  }, [])

  return (
    <div>
      {posts.map((item) => (
        <Blog
          post={item}
        />
      ))}
    </div>
  )
```

```
}
```

Сохраните файл и обновите вкладку браузера. Теперь вы должны увидеть, как записи вашего блога отображаются на странице.



Заключение

Безголовый WordPress предлагает отличный способ создания быстродействующих веб-сайтов с масштабируемой архитектурой. Благодаря использованию React и WordPress REST API стало как никогда просто создавать динамичные и интерактивные веб-сайты с WordPress в качестве системы управления контентом.

Дата Создания
29.05.2023