

Интеграция OpenWeatherMap API для отображения погоды на Node.js

Языков Кирилл Александрович

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В данной статье рассматривается интеграция OpenWeatherMap API для отображения погоды в приложениях на Node.js. Основное внимание уделяется процессу регистрации на платформе OpenWeatherMap, получению API ключа и использованию его для получения данных о текущей погоде и прогнозах. Статья включает примеры кода, демонстрирующие, как отправлять запросы к API и обрабатывать полученные данные. В заключение подчеркивается важность использования API для создания динамичных и интерактивных приложений, способных предоставлять актуальную информацию о погоде.

Ключевые слова: Интеграция OpenWeatherMap API, Node.js, API ключ.

Integration of the OpenWeatherMap API for displaying weather on Node.js

Yazykov Kirill Alexandrovich

Sholom-Aleichem Priamursky State University

Student

Abstract

This article discusses the integration of the OpenWeatherMap API for displaying weather in applications on Node.js. The main focus is on the registration process on the OpenWeatherMap platform, obtaining an API key and using it to obtain data on the current weather and forecasts. The article includes code examples demonstrating how to send API requests and process the received data. In conclusion, the importance of using the API to create dynamic and interactive applications capable of providing up-to-date weather information is emphasized.

Keywords: Integration of the OpenWeatherMap API, Node.js, API key.

1 Введение

1.1 Актуальность

Погода является одним из самых запрашиваемых типов информации. Использование API позволяет разработчикам легко и быстро получать актуальные данные о погоде в реальном времени. Для начинающих разработчиков интеграция API предоставляет возможность изучить основы работы с внешними сервисами, HTTP-запросами и обработкой данных, что является важным шагом в карьерном росте. Бизнесы могут использовать данные о погоде для анализа рынка, оптимизации логистики и повышения

эффективности рекламных стратегий, что делает эту тему особенно актуальной для предпринимателей.

1.2 Обзор исследований

А. И. Ямром, С. В. Киргинцева в своей статье продемонстрировали подключение API к приложению, описав сложные этапы работы с Node.js. Для того чтобы ее решить были разработаны документации, которые были сделаны при помощи swagger [1]. А. А. Абсалямова, Ж. Ф. Сафиканова для разработки и использования API применили различные технологии и инструменты. Подробно описали классификации, основные типы и области применения различных WEB-API [2]. С. В. Черномордов важным аспектом выделяет подробное описание этапов для разработки различных web-проектов, описано множество инструментов, библиотек, которые ускоряют разработку на front и backend сторон. Наиболее полезным является оптимизация различных процессов, а также кода, что является главным аспектом любого программиста [3]. А. О. Осипов создал приложение, которое прогнозирует погоду активных дней при помощи новых технологий, которые позволяют с предельной точностью рассчитать дни отмены занятий по климатическим показателям [4].

1.3 Цель исследования

Целью исследования является анализ возможностей интеграции OpenWeatherMap API в приложения на Node.js для получения и отображения актуальных данных о погоде. Также предполагается изучение методов обработки и визуализации полученной информации для повышения пользовательского опыта.

2 Материалы и методы

Для реализации данного исследования использовались Visual Studio (IDE), OpenWeatherMap API, библиотеки express, axios и fs.

3 Результаты

В данной работе описан процесс создания небольшого сайта для отображения погоды в разных городах. При запросе погоды для конкретного города приложение отправляет запрос к OpenWeatherMap API, получая детальную информацию о текущих погодных условиях. Результатом работы стал простой, но функциональный сайт, демонстрирующий основные принципы работы API. Этот опыт позволяет студентам понять, как лучше всего использовать API ключи и реализовывать сайты с запросами, а также развивает навыки работы с библиотеками Node.js.

Для работы с сервисом, который показывать пользователю города и погоды, необходимо получить API ключ.

Для начала работы с API OpenWeatherMap требуется регистрация на официальном сайте, расположенном по адресу <https://openweathermap.org> (Рис. 1).

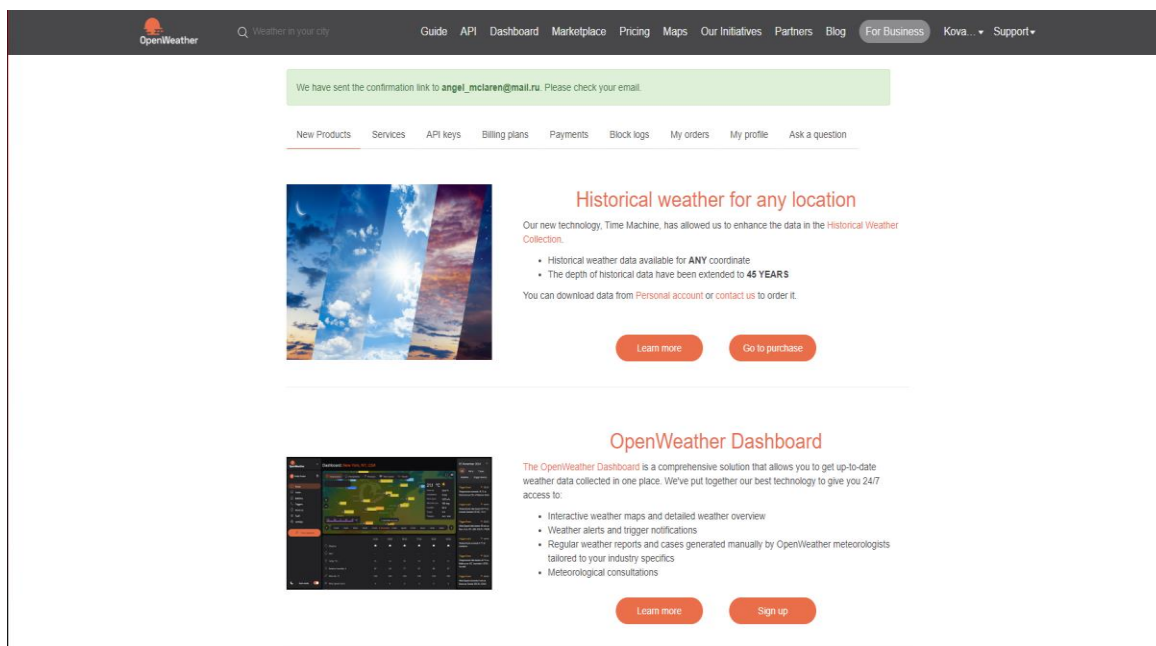


Рисунок 1 – Главная страница сайта OpenWhetherMap

После успешной регистрации необходимо перейти ко вкладке «API keys» и посмотреть «ключ» уже готовой API.

Теперь переходим в командную строку, предварительно установив Node.js на компьютер.

В командной строке необходимо переместиться в папку, где будет находится проект. Для того чтобы в командной строке перейти в папку, пропишите `cd <Ваш путь до папки>` (без `<>`). Если же не создали папку для проекта, то сделать это можно через командную строку, при помощи `mkdir <Название папки>`. После инициализируем проект в папке: `npm init -y` (Рис. 2).

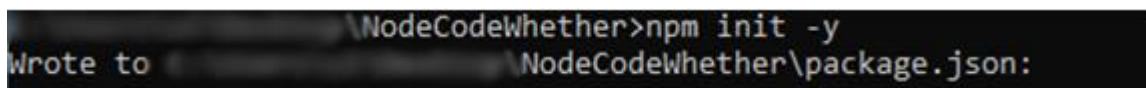


Рисунок 2 – Инициализация проекта

Также необходимо установить две библиотеки: `express` и `axios`. Express: для создания веб-сервера. Axios для выполнения запросов к API. Чтобы установить, необходимо прописать в командной строке: `npm install express axios`.

После установки необходимых библиотек нужно создать папку `public`, в которой поместим два файла: `index.html` и `styles.css`. В папке проекта, где скачивались библиотеки необходимо создать файл: `index.js`. Директория файлов должна быть построена следующим образом (Рис. 3).

Имя	Дата изменения	Тип	Размер
node_modules	22.01.2025 15:23	Папка с файлами	
public	22.01.2025 15:17	Папка с файлами	
index.js	22.01.2025 15:17	файл JavaScript	0 КБ
package.json	22.01.2025 15:23	JSON File	1 КБ
package-lock.json	22.01.2025 15:23	JSON File	32 КБ

Рисунок 3 – Структура папок

Код в файле index.js начинается с указания локального подключения с определенным портом. Импортируются библиотеки, необходимые для создания веб-приложения, такие как express, для выполнения запросов используется axios, а для работы с файловой системой применяется fs.

```
const express = require('express');
const axios = require('axios');
const fs = require('fs');
const app = express();
const PORT = process.env.PORT || 3000;

app.use(express.static('public'));

let cities = [];
fs.readFile('cities.txt', 'utf8', (err, data) => {
  if (err) {
    console.error('Ошибка при чтении файла cities.txt:', err);
    return;
  }
  cities = data.split('\n').map(city => city.trim());
});

app.get('/cities', (req, res) => {
  const query = req.query.q.toLowerCase();
  const filteredCities = cities.filter(city => city.toLowerCase().startsWith(query));
  res.json(filteredCities);
});

function getWeatherDescription(weather) {
  const mainWeather = weather[0].main.toLowerCase();
  switch (mainWeather) {
    case 'clear':
      return 'ясно';
    case 'clouds':
      return 'облачно';
    case 'rain':
```

```
        return 'дождливо';
      case 'snow':
        return 'снежно';
      case 'mist':
      case 'fog':
        return 'туманно';
      default:
        return 'неизвестно';
    }
  }

app.get('/weather', async (req, res) => {
  const city = req.query.city;
  const apiKey = ""; // ваш API ключ
  const url = `https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${apiKey}&units=metric`;

  try {
    const response = await axios.get(url);
    const data = response.data;

    if (!data || !data.main) {
      return res.status(404).json({ error: 'Город не найден' });
    }

    const weatherInfo = {
      name: data.name,
      temperature: data.main.temp,
      description: getWeatherDescription(data.weather),
      pressure: data.main.pressure,
      precipitation: data.rain ? data.rain['1h'] : 0,
      humidity: data.main.humidity,
      windSpeed: data.wind.speed,
      clouds: data.clouds.all
    };

    res.json(weatherInfo);
  } catch (error) {
    console.error('Ошибка при получении данных о погоде:',
error.message);
    if (error.response) {
      return res.status(error.response.status).json({
error: error.response.data.message });
    }
  }
}
```

```
    res.status(500).json({ error: 'Ошибка при получении данных о погоде'
});
  }
});

app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});
```

Код отвечает за серверную часть, включая обработку API ключа, получение и отправку ответов, а также возвращение значений.

Перейдем к коду для файла index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="styles.css">
  <title>Weather App</title>
</head>
<body>
  <h1>Погода в городе</h1>
  <input type="text" id="city" placeholder="Введите город"
oninput="suggestCities()">
  <ul id="suggestions" style="list-style-type: none; padding: 0;"></ul>
  <button onclick="getWeather()">Получить погоду</button>
  <div id="weather"></div>

  <script>
    async function suggestCities() {
      const input = document.getElementById('city').value;
      const response = await fetch(`/cities?q=${input}`);
      const cities = await response.json();
      const suggestions = document.getElementById('suggestions');
      suggestions.innerHTML = "";

      cities.forEach(city => {
        const li = document.createElement('li');
        li.textContent = city;
        li.onclick = () => {
          document.getElementById('city').value = city;
          suggestions.innerHTML = "";
        };
        suggestions.appendChild(li);
      });
    };
  </script>
```

```

    }

    async function getWeather() {
        const city = document.getElementById('city').value;
        const response = await fetch(`/weather?city=${city}`);

        if (!response.ok) {
            const errorData = await response.json();
            document.getElementById('weather').innerText = `Ошибка:
${errorData.error}`;
            return;
        }

        const data = await response.json();
        document.getElementById('weather').innerHTML = `
            <strong>Город:</strong> ${data.name}<br>
            <strong>Температура:</strong> ${data.temperature}°C<br>
            <strong>Описание:</strong> ${data.description}<br>
            <strong>Давление:</strong> ${data.pressure} мм рт. ст.<br>
            <strong>Осадки:</strong> ${data.precipitation > 0 ?
data.precipitation + ' мм' : 'Без осадков'}<br>
            <strong>Влажность:</strong> ${data.humidity}%<br>
            <strong>Скорость ветра:</strong> ${data.windSpeed} м/с<br>
            <strong>Облачность:</strong> ${data.clouds}%<br>
        `;
    }
</script>
</body>
</html>

```

В данном коде создается внешний вид сайта, включающий название, строку ввода и кнопку. Также происходит визуализация ответов, полученных от сервера.

Стили для сайта в файле styles.css выглядят следующим образом:

```

body {
    font-family: Arial, sans-serif;
    text-align: center;
    margin: 50px;
}

h1 {
    color: #333;
}

input {

```

```
padding: 10px;
width: 200px;
margin-right: 10px;
}

button {
padding: 10px;
}

#weather {
margin-top: 20px;
font-size: 1.2em;
color: #555;
}

#suggestions {
border: 1px solid #ccc;
max-height: 150px;
overflow-y: auto;
position: absolute;
background: white;
z-index: 1000;
}

#suggestions li {
padding: 8px;
cursor: pointer;
}

#suggestions li:hover {
background-color: #f0f0f0;
}
```

В файле cities.txt стоит написать несколько городов в столбик:

London

Tokyo

Moscow

Для проверки работы сайта необходимо в командной строке с помощью команды `cd <директория>` перейти в папку, где находится файл `index.js`. Затем в командной строке выполняется команда `node index.js` (Рис. 4).

```
Microsoft Windows [Version 10.0.19045.5371]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\...> cd C:\Users\...
C:\Users\...> node index.js
Server is running on http://localhost:3000
```

Рисунок 4 – Запуск проекта

Необходимо открыть браузер по адресу: <http://localhost:3000>. Протестировать, написав, «Moscow» в поле ввода и проверить (Рис. 5).

Погода в городе

Moscow	Получить погоду
-------------------------------------	--

Город: Moscow
Температура: -3.32°C
Описание: облачно
Давление: 1022 мм рт. ст.
Осадки: Без осадков
Влажность: 52%
Скорость ветра: 4.98 м/с
Облачность: 100%

Рисунок 5 – Проверка работы сайта

Для получения информации о погоде в Москве используется сервис прогнозирования погоды, доступный по адресу <https://yandex.ru/weather/moscow>. Результаты, полученные с написанного сайта, сравниваются с данными популярного сервиса погоды (Рис. 6).

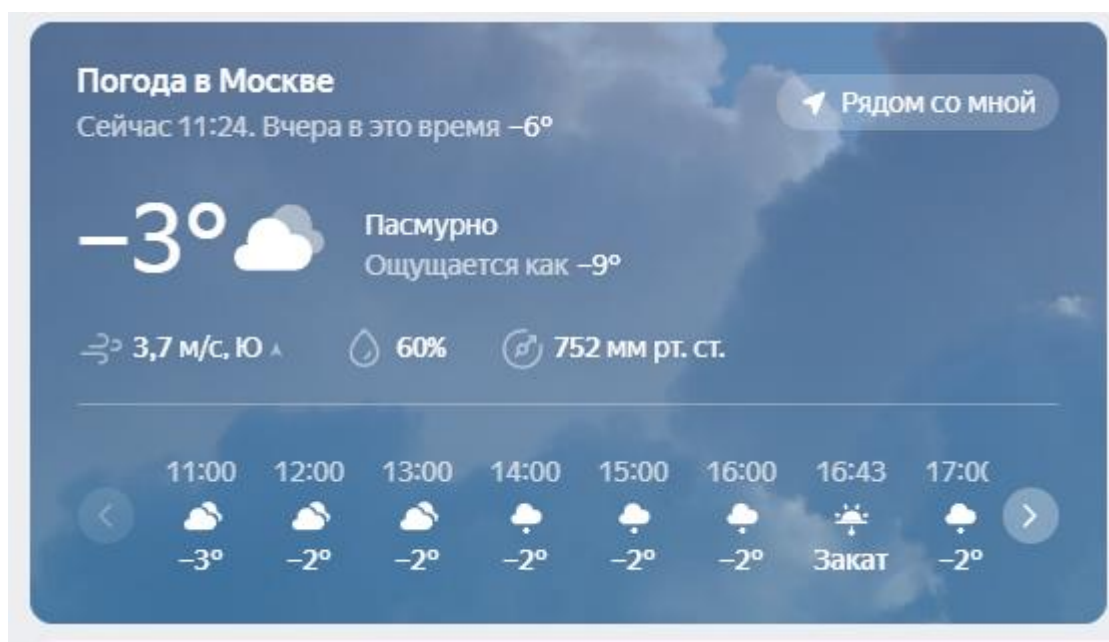


Рисунок 6 – Погода от сервиса прогнозирования погоды

4 Выводы

Интеграция OpenWeatherMap API в приложение на Node.js позволяет легко получать актуальные данные о погоде, однако стоит отметить, что температура, как правило, отображается точно, в то время как другие параметры, такие как влажность или скорость ветра, могут не совпадать с реальными значениями. Это может быть связано с тем, что используется бесплатная версия API, которая имеет ограничения по точности и частоте обновления данных.

Ограниченная точность бесплатной версии может влиять на отображаемые параметры, а также локальные условия могут варьироваться в зависимости от конкретного местоположения, что приводит к усреднению данных. Важно также учитывать, что время обновления данных может не совпадать с фактическими изменениями погоды, что создает дополнительные расхождения. Чтобы улучшить приложение, можно добавить интерактивную карту, на которой пользователи смогут видеть погоду в разных регионах, а также внедрить уведомления о резких изменениях погоды, таких как сильный дождь или снегопад. Возможность просмотра исторических данных о погоде поможет пользователям анализировать климатические изменения, а персонализированные настройки позволят настраивать параметры отображения погоды.

Интеграция с другими API для получения информации о качестве воздуха или уровне загрязнения значительно расширит функционал приложения. Социальные функции, позволяющие делиться погодными

условиями с друзьями через социальные сети, сделают приложение более привлекательным, а разработка мобильной версии обеспечит удобный доступ к погодным данным на ходу. Эти улучшения не только сделают приложение более функциональным, но и повысят его привлекательность для пользователей, что может привести к увеличению аудитории и улучшению пользовательского опыта.

Библиографический список

1. Ямром А. И., Киргинцева С. В. Разработка backend-части веб-приложения «Meetup-API» с использованием NODE. JS и EXPRESS // Новые математические методы и компьютерные технологии в проектировании, производстве и научных исследованиях: материалы XXVI Республиканской научной конференции студентов и аспирантов (Гомель, 20–22 марта 2023 года). 2023. С. 169-170.
2. Абсалямова А. А., Сафиканова Ж. Ф. Разработка и использование веб-арі: обзор технологий и инструментов // Современная российская наука: актуальные вопросы. 2023. С. 19.
3. Черномордов С. В. Разработка web-проекта на платформе NODE. JS // Школа молодых учёных по проблемам естественных наук. 2018. С. 120-123.
4. Осипов А. О. Android-приложение прогнозирования активированных дней в г. Сургуте // Успехи современной науки. 2016. Т. 2. №. 10. С. 61-62.