УДК 004

Создание Rest API с использованием NodeJS и Express

Вавилов Егор Дмитриевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье метод создания REST API и осуществлен метод GET запроса. В ходе работы используются средства разработки и тестирования: NodeJS, Express, Postman. Практическим результатом является серверный код обрабатывающий поступивший запрос и возвращающий его на web-страницу.
Ключевые слова: Express, NodeJS, Разработка

Building Rest API using NodeJS and Express

Vavilov Yegor Dmitrievich
Sholom-Aleichem Priamursky State University
student

Abstract
In this article, a method for creating a REST API and a GET request method is implemented. During the work, development and testing tools are used: NodeJS, Express, Postman. The practical result is server-side code that processes the incoming request and returns it to the web page.
Keywords: Express, NodeJS, Development

Express - это среда разработки Node, призванная помочь разработчикам JavaScript создавать серверы очень быстро. NodeJS может быть серверной частью, но он может делать гораздо больше, чем просто обслуживать страницы и данные. NodeJS - это мощная платформа, которая позволяет запускать JavaScript.

Цель исследования – написать вэб-приложение, которое будет получать сторонний GET запрос и выводить данные в окне.

Исследованиями в данной теме занимались следующие авторы. Л.Н.Абдурайимов, З.Э.Халилова привели описание различных типов баз данных: реляционных и не реляционных. Указали, что программная платформа Node.js позволяет использовать оба типа баз данных [1]. H.Kahn, N.Filer, A.Williams, N.Whitaker рассмотрели возможность сохранить концепцию STEP о прямом отображении информационной модели на реализацию, но сделать это таким образом, чтобы можно было принять альтернативные стратегии реализации [2]. П.А.Васильев предложил сведения о библиотеке Passport JS. О ее достоинствах и недостатках в использовании

на web-приложениях [3]. А.А.Поляков разработал web-приложение для сетей общественного питания с использованием NodeJS [4].

Для начала перейдем на официальный сайт и загрузим наиболее стабильную версию NodeJS.

Далее необходимо инициализировать приложение. Для того нужно открыть cmd в папке, в которой необходимо создать приложение, и ввести «npm init» для интерактивного создания файла «package.json».

Следующим шагом установим Express, но установим его только для каталога приложения, для этого используем команду «npm install express --save» Затем создадим файл «app.js» и добавим следующий код(рис.1).

```
var express = require("express");
var app = express();


app.listen(3000, () => {
 console.log("Server running on port 3000");
});
```

Рисунок 1 – app.js

Первая строка требует выражения «Express» и использует переменную «express» для его представления. Вторая строка инициализируется с использованием скобок, что инициализирует экспресс-сервер и помещает инициализированный сервер в переменную «app». Затем настраиваем приложение на прослушивание порта 3000 и создаем функцию обратного вызова, которая сообщает, что сервер работает на порту 3000.

Далее настроим обработчик запросов. Сервер получает запросы, обрабатывает их и возвращает ответ. Поэтому нужно использовать маршруты для обработки этих запросов. У запросов есть четыре основных типа: запрос GET, который получает данные, запрос POST, который отправляет данные безопасно, запрос PUT, который обновляет данные, и запрос DELETE, который удаляет данные. Создадим простой запрос GET, который возвращает список пользователей. В разделе «var app = express ()» запишем следующий код (рис.2).

```
app.get ("/ url", (req, res, next) => {
res.json (["Овощи", "Фрукты", "Колбаса", "Хлеб", "Пельмени"]);
});
```

Рисунок 2 – get-запрос

Эта простая функция заставляет экспресс-приложение использовать дескриптор URL-адреса «/url» для запуска последующего обратного вызова. Обратный вызов принимает три параметра, req - это тело запроса и несет информацию о запросе. Res является телом ответа и используется для

обработки таких функций ответа, как «.render ()» для отображения шаблонов и «.json ()» для возврата данных json.

Чтобы запустить приложение, используем команду «node app.js». «cmd» после выполнения этой команды должен выглядить следующим образом (рис.3).

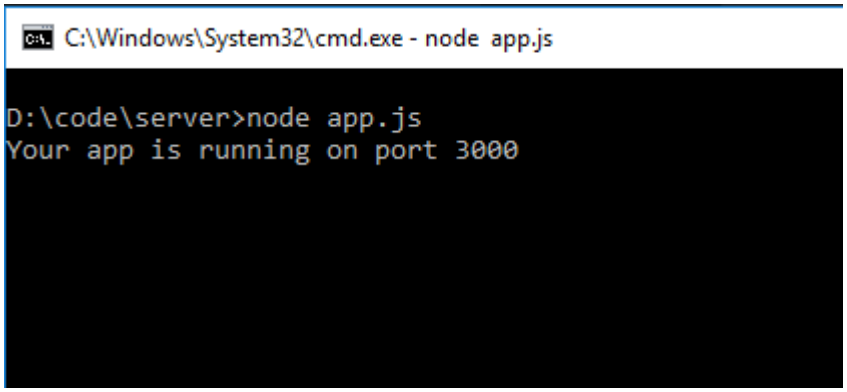

Рисунок 3 - cmd

Это означает, что приложение теперь успешно работает на порту 3000. Чтобы просмотреть данные, откроем браузер и введем «http://localhost:3000/url» (Рис.4).

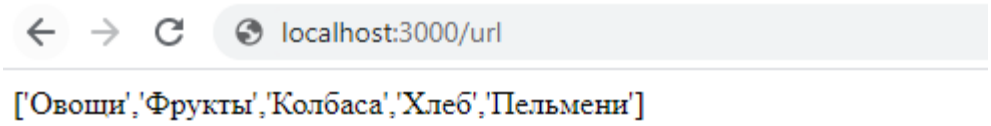

Рисунок 4 – Вид в браузере

REST означает «REpresentational State Transfer». Это означает, что между клиентом и сервером нет состояния. Нет web-страниц для анализа, только данные. Все, что нужно сделать, это написать некоторую логику для определенного URL-адреса, который подключается к базе данных, использует эту логику для обработки данных и возврата их в формате JSON. Теперь клиентом может быть приложение Android, созданное на Java, или настольное приложение Windows, созданное на C #, или проект Arduino. В этом весь смысл использования REST, он делает соединение без состояния, поэтому любой клиент, использующий протокол HTTP, может получить доступ к этим данным. Теперь можно перебирать данные и отображать их в любом месте (Рис.5).

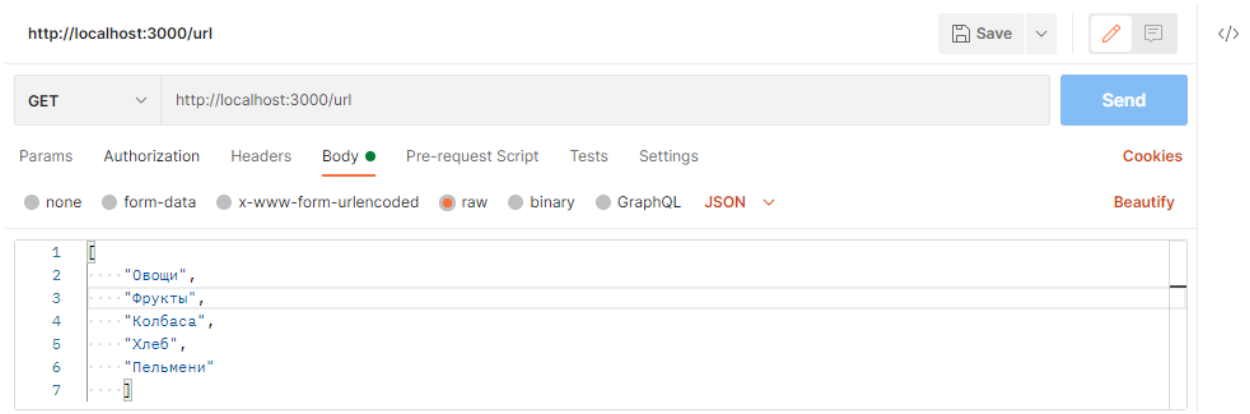

Рисунок 5 – Отправка через Postman

Сервер не ограничивается только браузерами. Также можно использовать собственные приложения и устройства «IoT» для получения данных, если они реализуют протокол HTTP.

В данной статье был рассмотрен процесс создания сервера принимающего запрос и отвечающего на него данными. В ходе работы использовались инструменты: Postman, NodeJS, Express.

Библиографический список

1. Абдурайимов Л.Н., Халилова З.Э. Интеграция базы данных в веб-приложение на node.js и фреймворке express // Теория и практика модернизации научной деятельности в условиях цифровизации. 2020. №4(18). С. 33-37.
2. Kahn H., Filer N., Williams A., Whitaker N. A generic framework for transforming express information models // Computer-aided design 2001. №7. С. 501-510.
3. Васильев П.А. JS языка программирования библиотеки node js // Информационные системы и технологии в образовании, науке и бизнесе. 2020. №1(27). С. 126-129.
4. Поляков А.А. Разработка web-приложения сети общественного питания с использованием платформы node js и библиотеки react js // Молодой Исследователь Дона. 2018. №1(10). С. 85-97.