Лабораторная работа №4

**Тема**: «Http/Https»

**Цель**: освоить возможности NodeJs по созданию локальных серверов с выводом статической информации, ознакомиться с фреймворком Express.

**Теоретический материал**:

Современные браузеры способны отличать настоящие URL-адреса, вводимые в их адресную строку, от поисковых запросов, для обработки которых обычно используется заданная по умолчанию поисковая система. Мы будем говорить именно об URL-адресах. Если вы введёте в строку браузера адрес сайта, вроде flaviocopes.com, браузер преобразует этот адрес к виду http://flaviocopes.com, исходя из предположения о том, что для обмена данными с указанным ресурсом будет использоваться протокол HTTP. Обратите внимание на то, что в Windows то, о чём мы будем тут говорить, может выглядеть немного иначе, чем в macOS и Linux.

Итак, браузер, начиная работу по загрузке данных с запрошенного пользователям адреса, выполняет операцию DNS-поиска (DNS Lookup) для того, чтобы выяснить IP-адрес соответствующего сервера. Символьные имена ресурсов, вводимые в адресную строку, удобны для людей, но устройство интернета подразумевает возможность обмена данными между компьютерами с использованием IP-адресов, которые представляют собой наборы чисел наподобие 222.324.3.1 (для протокола IPv4).  
Сначала, выясняя IP-адрес сервера, браузер заглядывает в локальный DNS-кэш для того, чтобы узнать, не выполнялась ли недавно подобная процедура. В браузере Chrome, например, есть удобный способ посмотреть DNS-кэш, введя в адресной строке следующий адрес: chrome://net-internals/#dns.  
Если в кэше ничего найти не удаётся, браузер использует системный вызов POSIX gethostbyname для того, чтобы узнать IP-адрес сервера.

Узнав IP-адрес сервера, клиент может инициировать процедуру TCP-подключения к нему. В процессе установления TCP-соединения клиент и сервер передают друг другу некоторые служебные данные, после чего они смогут обмениваться информацией. Это означает, что, после установления соединения, клиент сможет отправить серверу запрос.

Запрос представляет собой структурированный в соответствии с правилами используемого протокола фрагмент текста. Он состоит из трёх частей:

* Строка запроса.
* Заголовок запроса.
* Тело запроса.

Строка запроса

Строка запроса представляет собой одну текстовую строку, в которой содержатся следующие сведения:

* Метод HTTP.
* Адрес ресурса.
* Версия протокола.

Заголовок запроса представлен набором пар вида поле: значение. Существуют 2 обязательных поля заголовка, одно из которых — Host, а второе — Connection. Остальные поля необязательны.

Тело запроса необязательно, в GET-запросах оно не используется. Тело запроса используется в POST-запросах, а также в других запросах. Оно может содержать, например, данные в формате JSON.

После того, как сервер получает отправленный клиентом запрос, он его обрабатывает и отправляем клиенту ответ.

После того, как браузер получает ответ сервера, в теле которого содержится HTML-код, он начинает его разбирать, повторяя вышеописанный процесс для каждого ресурса, который нужен для формирования страницы. К таким ресурсам относятся, например, следующие:

CSS-файлы.

Изображения.

Значок веб-страницы (favicon).

JavaScript-файлы.

То, как именно браузер выводит страницу, к нашему разговору не относится. Главное, что нас тут интересует, заключается в том, что вышеописанный процесс запроса и получения данных используется не только для HTML-кода, но и для любых других объектов, передаваемых с сервера в браузер с использованием протокола HTTP

***Создание простого сервера на NodeJS:***



Рисунок 1 – Листинг сервера

Для проверки сервера откройте какой-нибудь браузер и введите в адресной строке http://127.0.0.1:3000, то есть — тот адрес сервера, который будет выведен в консоли после его успешного запуска. Если всё работает как надо — на странице будет выведено «Hello World».

Для выполнения HTTP-запросов средствами Node.js используется соответствующий модуль http. В приведённых ниже примерах применяется модуль https. Дело в том, что в современных условиях всегда, когда это возможно, нужно применять именно протокол HTTPS.

***Выполнение Get – запрос:***



Рисунок 2 – Пример Get запроса

***Выполнение Post– запрос:***



Рисунок 2 – Пример Post запроса

**Практическая часть:**

**Задание 1** Создать сервер на NodeJs без использования фреймворка express. На базе сервера разработать статический web-сайт, состоящий из нескольких страниц. Тема выбирается студентом самостоятельно. К страницам сайта применить стилизацию. Предусмотреть переходы между станицами. Страницы храняться как статические файлы.

**Задание 2** Выполнить задание1 с использованием фреймворка Express. Дополнительно: все переходы логгировать в отдельный файл server.log.

**Задание 3** Используя фреймворк Express, разработать REST API. В качестве предметной области взять задание на курсовую работу. В качестве источника данных создать файл в формате JSON. Установить Postman и протестировать все разработанный REST API.

**Ссылки для детального просмотра**:

Модуль http: <https://nodejs.org/api/http.html>

Реализация RestApi без фремворков: <https://www.section.io/engineering-education/a-raw-nodejs-rest-api-without-frameworks-such-as-express/>