Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии (ИУ-7)

ОТЧЕТ

по лабораторной работе №1

по курсу Web-технологии

Выполнил — Студент: Иванников А.В.
Группа: ИУ7-78

Принял — Преподаватель: Бекасов Д.Е.

Москва, 2020

Задание

Данная работа является вводной в курсе и призвана ознакомить студента с базовым протоколом web – протоколом HTTP.

Результатом работы является отчет в формате MD и код сервера, выложенные в репозиторий.

1. Базовая часть работы

1.1. Цель данной работы – ознакомится с применением протокола HTTP на практике, в реальных системах. Каждый из рассмотренных типов запросов предлагается отправить на несколько известных интернет-сервисов. Впрочем, сервисы указаны лишь как примеры и при желании вы можете выбрать другие (социальные сети, почта, облака, новостные сайты и т.д.).

1.2. С помощью специального ПО (Postman, либо многочисленные аналоги, например, Restlet Clent - расширение для Chrome) вручную отправить следующие запросы и ответить на предлагаемые вопросы.

1.2.1. Запрос OPTIONS. Отправьте запрос на http://mail.ru, http://ya.ru, www.rambler.ru, https://www.google.ru, https://github.com/, www.apple.com/. Для чего используется запрос OPTIONS? Какие коды ответов приходят при этом запросе? Какие сайты правильно обработали запрос и вернули ожидаемые данные?

1.2.2. Запрос HEAD. Отправьте запрос на vk.com, www.apple.com, www.msn.com. Для чего нужен запрос HEAD? Какой сайт прислал ожидаемый ответ?

1.2.3. Запросы GET и POST. Отправьте запрос на yandex.ru, google.com и apple.com. Что они вернули? Что содержится в теле ответа?

1.3. Работа с api сайта. Многие крупные сервисы предоставляют открытое api. Как правило, оно реализовано на подходе REST, но это необязательно. Такое api используется сторонними сервисами и приложениями, которые хотят воспользоваться услугами предоставляющего такое api сервиса.

Рассмотрим такое api на примере сайта vk.com (при желании можно выбрать другой подходящий сервис).

1.3.1. Зайдите на https://vk.com/dev/api_requests и посмотрите структуру запросов к данному api.

1.3.2. Используя документацию (https://vk.com/dev/methods) выполните следующие задания (обратите внимание, запросы нужно отправлять не из предложенной на сайте формы, а как в предыдущем задании):

1.3.2.1. Получите список всех факультетов МГТУ им. Н.Э. Баумана.

1.3.2.2. Получите свою аватарку.

1.3.2.3. Ответьте на вопросы: какой код ответа присылается от api? Что содержит тело ответа? В каком формате и какой кодировке содержаться данные? Какой веб-сервер отвечает на запросы? Какая версия протокола HTTP используется?

1.3.3. POST запросы проще отправлять с формы, встроенной в документацию api. Чтобы посмотреть, как выглядит запрос, можно воспользоваться панелью разработчика браузера (F12 в Chrome -> вкладка Network).

1.3.3.1. Отправьте запись на стену любому пользователю/группе и убедитесь, что она пришла.

1.3.3.2. Ответьте на вопрос: каким образом передаются данные от пользователя к серверу в POST-запросах?

2. Реализуйте небольшое серверное приложение, с использованием любого фреймворка. Лучшего всего для этой цели подойдет NodeJS: решение получится очень компактным и простым.

Сервер должен содержать предоставлять некоторое API с поддержкой (GET, POST, DELETE, PUT, OPTION). Данные отправлять в формате json. Конкретное содержание запросов - на ваше усмотрение. Подключите фантазию. (Можно сделать простейший CRUD-сервис с хранением данных в RAM).

3. Доп. задание. Статика и маршрутизация.

3.1. Добавьте папку static (классическое название для статически раздаваемой папки).

3.2. В папке static создайте папки html и img.

3.3. В папке static/html создайте файл index.html со следующим содержанием (или любым другим):

<head></head>

<body>

<h1>Hello, world!</h1>

<img src=”/img/image.jpg”>

</body>

3.3. Настройте сервер так, чтобы при запросе из браузера отображалась эта страница.

3.4. Настройте routing (маршрутизацию) на вашем сервере. Например, чтобы путь /hack тоже отдавал файл index.html, а путь /, по умолчанию отдающий index, выдавал дополнительную страницу hack.html.

3.5. Переименуйте hack.html (содержащую теги html) в hack.txt. Что изменилось? Почему? Как сделать так, чтобы страница отображалась корректно?

1. Базовая часть работы

1) Запрос OPTIONS

Запрос OPTIONS используется для определения возможностей веб-сервера или параметров соединения для конкретного ресурса. В ответ серверу следует включить в заголовок Allow со списком поддерживаемых методов. Также в заголовке ответа может включаться информация о поддерживаемых расширениях.

Некоторые часто встречающиеся коды ответа HTTP:

- 200 OK - запрос успешно обработан. Если клиентом были запрошены какие-либо данные, то они находятся в заголовке и/или теле сообщения. Появился в HTTP/1.0;
- 403 Forbidden - сервер понял запрос, но он отказывается его выполнять из-за ограничений в доступе для клиента к указанному ресурсу. Иными словами, клиент не уполномочен совершать операции с запрошенным ресурсом. Если для доступа к ресурсу требуется аутентификация средствами HTTP, то сервер вернёт ответ 401, или 407 при использовании прокси. В противном случае ограничения были заданы администратором сервера или разработчиком веб-приложения и могут быть любыми в зависимости от возможностей используемого программного обеспечения. В любом случае клиенту следует сообщить причины отказа в обработке запроса;
- 405 Method Not Allowed - указанный клиентом метод нельзя применить к текущему ресурсу. В ответе сервер должен указать доступные методы в заголовке Allow, разделив их запятой. Эту ошибку сервер должен возвращать, если метод ему известен, но он не применим именно к указанному в запросе ресурсу, если же указанный метод не применим на всём сервере, то клиенту нужно вернуть код 501 (Not Implemented).

Коды ответов HTTP приведены в таблице 1

Таблица 1 – Коды ответов HTTP

1xx - Информационные коды		
100	Continue	"Продолжить". Этот промежуточный ответ указывает, что запрос успешно принят и клиент может продолжать присылать запросы либо проигнорировать этот ответ, если запрос был завершён.
101	Switching Protocol	"Переключение протокола". Этот код присылается в ответ на запрос клиента, содержащий заголовок Upgrade:, и указывает, что сервер переключился на протокол, который был указан в заголовке. Эта возможность позволяет перейти на несовместимую версию протокола и обычно не используется.
102	Processing	"В обработке". Этот код указывает, что сервер получил запрос и обрабатывает его, но обработка еще не завершена.
103	Early Hints	Ранние подсказки". В ответе сообщаются ресурсы, которые могут быть загружены заранее, пока сервер будет подготовливать основной ответ.
2xx - Успешные коды		
200	OK	"Успешно". Запрос успешно обработан. Что значит "успешно", зависит от метода HTTP, который был запрошен: GET: "ПОЛУЧИТЬ". Запрошенный ресурс был найден и передан в теле ответа. HEAD: "ЗАГОЛОВОК". Заголовки переданы в ответе. POST: "ПОСЫЛКА". Ресурс, описывающий результат действия сервера на запрос, передан в теле ответа. • TRACE: "ОТСЛЕЖИВАТЬ". Тело ответа содержит тело запроса полученного сервером.
201	Created	"Создано". Запрос успешно выполнен и в результате был создан ресурс. Этот код обычно присылается в ответ на запрос PUT "ПОМЕСТИТЬ".
202	Accepted	"Принято". Запрос принят, но ещё не обработан. Не поддерживаемо, т.е., нет способа с помощью HTTP отправить асинхронный ответ позже, который будет показывать итог обработки запроса. Это предназначено для случаев, когда запрос обрабатывается другим процессом или сервером, либо для пакетной обработки.
203	Non-Authoritative Information	"Информация не авторитетна". Этот код ответа означает, что информация, которая возвращена, была предоставлена не от исходного сервера, а из какого-нибудь другого источника. Во всех остальных ситуациях более предпочтителен код ответа 200 OK.
204	No Content	"Нет содержимого". Нет содержимого для ответа на запрос, но заголовки ответа, которые могут быть полезны, присылаются. Клиент может использовать их для обновления кешированных заголовков полученных ранее для этого ресурса.

205	Reset Content	"Сбросить содержимое". Этот код присылается, когда запрос обработан, чтобы сообщить клиенту, что необходимо сбросить отображение документа, который прислал этот запрос.
206	Partial Content	"Частичное содержимое". Этот код ответа используется, когда клиент присылает заголовок диапазона, чтобы выполнить загрузку отдельно, в несколько потоков.
3xx - Коды перенаправления		
300	Multiple Choice	"Множественный выбор". Этот код ответа присылается, когда запрос имеет более чем один из возможных ответов. И User-agent или пользователь должен выбрать один из ответов. Не существует стандартизированного способа выбора одного из полученных ответов.
301	Moved Permanently	"Перемещён на постоянной основе". Этот код ответа значит, что URI запрашиваемого ресурса был изменен. Возможно, новый URI будет предоставлен в ответе.
302	Found	"Найдено". Этот код ответа значит, что запрошенный ресурс *временно изменен*. Новые изменения в URI могут быть доступны в будущем. Таким образом, этот URI, должен быть использован клиентом в будущих запросах.
303	See Other	"Просмотр других ресурсов". Этот код ответа присылается, чтобы направлять клиента для получения запрашиваемого ресурса в другой URI с запросом GET.
304	Not Modified	"Не модифицировано". Используется для кэширования. Это код ответа значит, что запрошенный ресурс не был изменен. Таким образом, клиент может продолжать использовать кэшированную версию ответа.
305	Use Proxy	"Использовать прокси". Это означает, что запрошенный ресурс должен быть доступен через прокси. Этот код ответа в основном не поддерживается из соображений безопасности.
306	Switch Proxy	Больше не использовать. Изначально подразумевалось, что " последующие запросы должны использовать указанный прокси."
307	Temporary Redirect	"Временное перенаправление". Сервер отправил этот ответ, чтобы клиент получил запрошенный ресурс на другой URL-адрес с тем же методом, который использовал предыдущий запрос. Данный код имеет ту же семантику, что код ответа 302 Found, за исключением того, что агент пользователя не должен изменять используемый метод HTTP: если в первом запросе использовался POST, то во втором запросе также должен использоваться POST.

308	Permanent Redirect	"Перенаправление на постоянной основе". Это означает, что ресурс теперь постоянно находится в другом URI, указанном в заголовке Location: HTTP Response. Данный код ответа имеет ту же семантику, что и код ответа 301 Moved Permanently, за исключением того, что агент пользователя не должен изменять используемый метод HTTP: если POST использовался в первом запросе, POST должен использоваться и во втором запросе.
4xx - Коды ошибок клиента		
400	Bad Request	"Плохой запрос". Этот ответ означает, что сервер не понимает запрос из-за неверного синтаксиса.
401	Unauthorized	"Неавторизовано". Для получения запрашиваемого ответа нужна аутентификация. Статус похож на статус 403, но,в этом случае, аутентификация возможна.
402	Payment Required	"Необходима оплата". Этот код ответа зарезервирован для будущего использования. Первоначальная цель для создания этого когда была в использовании его для цифровых платежных систем(на данный момент не используется).
403	Forbidden	"Запрещено". У клиента нет прав доступа к содержимому, поэтому сервер отказывается дать надлежащий ответ.
404	Not Found	"Не найден". Сервер не может найти запрашиваемый ресурс. Код этого ответа, наверно, самый известный из-за частоты его появления в вебе.
405	Method Not Allowed	"Метод не разрешен". Сервер знает о запрашиваемом методе, но он был деактивирован и не может быть использован. Два обязательных метода, GET и HEAD, никогда не должны быть деактивированы и не должны возвращать этот код ошибки.
406	Not Acceptable	Этот ответ отсылается, когда веб сервер после выполнения server-driven content negotiation, не нашел контента, отвечающего критериям, полученным из user agent.
407	Proxy Authentication Required	Этот код ответа аналогичен коду 401, только аутентификация требуется для прокси сервера.
408	Request Timeout	Ответ с таким кодом может прийти, даже без предшествующего запроса. Он означает, что сервер хотел бы отключить это неиспользуемое соеднинение. Этот метод используется все чаще с тех пор, как некоторые браузеры, вроде Chrome и IE9, стали использовать HTTP механизмы предварительного соединения для ускорения серфинга (смотрите баг 881804, будущей реализации этого механизма в Firefox). Также учитывайте, что некоторые серверы прерывают соединения не отправляя подобных сообщений.

409	Conflict	Этот ответ отсылается, когда запрос конфликтует с текущим состоянием сервера.
410	Gone	Этот ответ отсылается, когда запрашиваемый контент удален с сервера.
411	Length Required	Запрос отклонен, потому что сервер требует указание заголовка Content-Length, но он не указан.
412	Precondition Failed	Клиент указал в своих заголовках условия, которые сервер не может выполнить
413	Request Entity Too Large	Размер запроса превышает лимит, объявленный сервером. Сервер может закрыть соединение, вернув заголовок Retry-After
414	Request-URI Too Long	URI запрашиваемый клиентом слишком длинный для того, чтобы сервер смог его обработать
415	Unsupported Media Type	Медиа формат запрашиваемых данных не поддерживается сервером, поэтому запрос отклонен
416	Requested Range Not Satisfiable	Диапазон указанный заголовком запроса Range не может быть выполнен; возможно, он выходит за пределы переданного URI
417	Expectation Failed	Этот код ответа означает, что ожидание, полученное из заголовка запроса Expect, не может быть выполнено сервером.
5xx - Коды ошибок сервера		
500	Internal Server Error	"Внутренняя ошибка сервера". Сервер столкнулся с ситуацией, которую он не знает как обработать.
501	Not Implemented	"Не выполнено". Метод запроса не поддерживается сервером и не может быть обработан. Единственные методы, которые сервера должны поддерживать (и, соответственно, не должны возвращать этот код) - GET и HEAD.
502	Bad Gateway	"Плохой шлюз". Эта ошибка означает что сервер, во время работы в качестве шлюза для получения ответа, нужного для обработки запроса, получил недействительный (недопустимый) ответ.
503	Service Unavailable	"Сервис недоступен". Сервер не готов обрабатывать запрос. Зачастую причинами являются отключение сервера или то, что он перегружен. Обратите внимание, что вместе с этим ответом удобная для пользователей(user-friendly) страница должна отправлять объяснение проблемы. Этот ответ должен использоваться для временных условий и Retry-After: HTTP-заголовок должен, если возможно, содержать предполагаемое время до восстановления сервиса. Веб-мастер также должен позаботиться о заголовках, связанных с кэшем, которые отправляются вместе с этим ответом, так как эти ответы, связанные с временными условиями, обычно не должны кэшироваться.

504	Gateway Timeout	Этот ответ об ошибке предоставляется, когда сервер действует как шлюз и не может получить ответ вовремя.
505	HTTP Version Not Supported	"HTTP-версия не поддерживается". HTTP-версия, используемая в запросе, не поддерживается сервером.

Результаты отправления запроса OPTIONS:

- http://mail.ru — status 200 OK
- http://ya.ru — status 403 Forbidden

Date →Fri, Mon, 24 Feb 2020

08:51:29 GMT

Content-Type →text/html;

charset=utf-8

ETag →W/"5c9251d5-3077"

Content-Encoding →gzip

X-Content-Type-Options →nosniff

Transfer-Encoding →chunked

- www.rambler.ru — status 200 OK
- www.google.ru — status 405 Method Not Allowed
- https://github.com/ — status 404 Not Found
- www.apple.com/ — status 200 OK

2). Запрос HEAD

Запрос HEAD запрашивает заголовки, идентичные тем, что возвращаются, если указанный ресурс будет запрошен с помощью HTTP-метода GET. Такой запрос может быть выполнен перед загрузкой большого ресурса, например, для экономии пропускной способности.

Ответ на метод HEAD не должен содержать тело. Если это не так, то его следует игнорировать. Но даже в этом случае заголовки сущности, описывающие содержимое тела, например Content-Length, должны быть включены в ответ. Они не относятся к телу ответа на запрос HEAD, которое

должно быть пустым, они относятся к телу ответа, полученный на аналогичный запрос с помощью метода GET.

Если результат запроса HEAD показывает, что кешированный после запроса GET ресурс устарел, то кеш становится недействительным, даже если запрос GET не был сделан.

HTTP код ошибки 418 I'm a teapot сообщает о том, что сервер не может приготовить кофе, потому что он чайник. Эта ошибка ссылается на Hyper Text Coffee Pot Control Protocol (гипертекстовый протокол кофейников) который был первоапрельской шуткой в 1998 году.

Тело ответа имелось в случае запроса HEAD к сайту www.apple.com.

3). Запросы GET и POST

Запрос GET используется для запроса содержимого указанного ресурса. С помощью метода GET можно также начать какой-либо процесс. В этом случае в тело ответного сообщения следует включить информацию о ходе выполнения процесса.

Запрос POST применяется для передачи пользовательских данных заданному ресурсу. Например, в блогах посетители обычно могут вводить свои комментарии к записям в HTML-форму, после чего они передаются серверу методом POST и он помещает их на страницу. При этом передаваемые данные (в примере с блогами — текст комментария) включаются в тело запроса. Аналогично с помощью метода POST обычно загружаются файлы на сервер.

Результаты отправления запросов GET и POST:

- yandex.ru:

- GET: status 200 OK

В теле ответа содержится ресурс.

- POST: status 403 Forbidden

Произошла ошибка на сервере Мы знаем о проблеме и работаем над её решением. Попробуйте обновить страницу через некоторое время.

- google.com:

- GET: status 200 OK

В теле ответа содержится ресурс.

- 405 - Method Not Allowed

405. That’s an error. The request method POST is inappropriate for the URL /. That’s all we know.

- apple.com:

- GET: status 200 OK В теле ответа содержится ресурс.

- POST: status 200 OK В теле ответа содержится ресурс.

4). Работа с API сайта

Сначала необходимо получить ключа доступа access_token. Для этого нужно:

- перейти по ссылке https://vk.com/editapp?act=create и создать приложение;

- в настройках найти ID созданного приложения путем отправления запроса GET https://oauth.vk.com/authorize?client_id=7333223&display=page&scope=friends&response_type=token&v=5.103&state=123456, вставки ссылки в браузер и авторизации. В результате будет получено сообщение: «Пожалуйста, не копируйте данные из адресной строки

для сторонних сайтов. Таким образом Вы можете потерять доступ к Вашему аккаунту.»

- скопировать из адресной строки acess_token=Наш_access_token.

Далее необходимо получить список стран и найти в списке стран Россию. Для этого нужно вставить ссылку https://api.vk.com/method/database.getCountries?need_all=1&offset=140&count=20& access_token=access_token&v=5.103 в браузер. В результате будет получен следующий ответ:

{"response":{"count":234,"items":[{"id":146,"title":"Оман"},{"id":147,"title":"Остров Мэн"},{"id":148,"title":"Остров Норфолк"},{"id":149,"title":"Острова Кайман"},{"id":150,"title":"Острова Кука"},{"id":151,"title":"Острова Теркс и Кайкос"},{"id":152,"title":"Пакистан"},{"id":153,"title":"Палау"},{"id":154,"title":"Палестинская автономия"},{"id":155,"title":"Панама"},{"id":156,"title":"Папуа — Новая Гвинея"},{"id":157,"title":"Парагвай"},{"id":158,"title":"Перу"},{"id":159,"title":"Питкерн"},{"id":160,"title":"Польша"},{"id":161,"title":"Португалия"},{"id":162,"title":"Пуэрто-Рико"},{"id":163,"title":"Реюньон"},{"id":1,"title":"Россия"},{"id":164,"title":"Руанда"}]}}

Интересующая строка: {"id":1,"title":"Россия"}

Далее необходимо получить список городов России и найти в списке городов Москву. Для этого нужно вставить ссылку https://api.vk.com/method/database.getCities?country_id=1& access_token=access_token&v=5.103 в браузер. В результате будет получен следующий ответ::

{"response":{"count":21,"items":[{"id":1,"title":"Москва","important":1},{"id":2,"title":"Санкт-Петербург","important":1},{"id":10,"title":"Волгоград"},{"id":37,"title":"Владивосток"},{"id":42,"title":"Воронеж"},{"id":49,"title":"Екатеринбург"},{"id":60,"tit

le":"Казань"},{"id":61,"title":"Калининград"},{"id":72,"title":"Краснодар"},{"id":73,"title":"Красноярск"},{"id":95,"title":"Нижний Новгород"},{"id":99,"title":"Новосибирск"},{"id":104,"title":"Омск"},{"id":110,"title":"Пермь"},{"id":119,"title":"Ростов-на-Дону"},{"id":123,"title":"Самара"},{"id":151,"title":"Уфа"},{"id":153,"title":"Хабаровск"},{"id":158,"title":"Челябинск"},{"id":185,"title":"Севастополь"},{"id":627,"title":"Симферополь"}]}}

Интересующая строка: {"id":1,"title":"Москва","important":1}

Далее необходимо получить список высших учебных заведений Москвы и найти в списке высших учебных заведений МГТУ. Для этого нужно вставить ссылку https://api.vk.com/method/database.getUniversities?q=МГТУ&country_id=1&city_id=1& access_token=access_token&v=5.103 в браузер. В результате будет получен следующий ответ:
{"response":{"count":5,"items":[{"id":248,"title":"МГТУ им. А. Н. Косыгина (бывш. МГТА им. А. Н. Косыгина, МТИ)"},{"id":249,"title":"МГТУ ГА"},{"id":250,"title":"МГТУ им. Н. Э. Баумана"},{"id":252,"title":"МГТУ «Станкин»"},{"id":169759,"title":"ИСОТ МГТУ им. Н. Э. Баумана (бывш. МИПК МГТУ им. Н. Э. Баумана)"}]}}

Интересующая строка: {"id":250,"title":"МГТУ им. Н. Э. Баумана"}

Далее необходимо получить список факультетов МГТУ им. Н.Э. Баумана. Для этого нужно вставить ссылку https://api.vk.com/method/database.getFaculties?university_id=250& access_token=access_token&v=5.103 в браузер. В результате будет получен следующий ответ:
{"response":{"count":20,"items":[{"id":1031,"title":"Аэрокосмический факультет"},{"id":1032,"title":"Факультет инженерного бизнеса и

менеджмента"},{"id":1033,"title":"Факультет информатики и систем управления"},{"id":1034,"title":"Факультет машиностроительных технологий"},{"id":1035,"title":"Факультет оптико-электронного приборостроения"},{"id":1036,"title":"Приборостроительный факультет"},{"id":1037,"title":"Радиотехнический факультет"},{"id":1038,"title":"Факультет радиоэлектроники и лазерной техники"},{"id":1039,"title":"Факультет ракетно-космической техники"},{"id":1040,"title":"Факультет робототехники и комплексной автоматизации"},{"id":1041,"title":"Факультет специального машиностроения"},{"id":1042,"title":"Факультет фундаментальных наук"},{"id":1043,"title":"Факультет энергомашиностроения"},{"id":1044,"title":"Кафедра юриспруденции, интеллектуальной собственности и судебной экспертизы"},{"id":1803,"title":"Факультет биомедицинской техники"},{"id":1804,"title":"Факультет социально-гуманитарных наук"},{"id":56430,"title":"Факультет лингвистики"},{"id":56431,"title":"Физкультурно-оздоровительный факультет"},{"id":2071503,"title":"Головной учебно-исследовательский и методический центр (ГУИМЦ)"},{"id":2183736,"title":"Факультет военного обучения (Военный институт)"}]}}

Получение аватарки

Для этого нужно вставить ссылку https://api.vk.com/method/users.get?user_id=44932940&fields=photo_50&access_token=access_token&v=5.103 в браузер. Будет получен следующий ответ:

{"response":[{"id":44932940,"first_name":"Алексей","last_name":"Иванников","is_closed":true,"can_access_closed":true,"photo_50":"https:\/\/sun9-25.userapi.com\/c847221\/v847221555\/f54ab\/VhMRNaS2PnM.jpg?ava=1"}]}

От API сайта присылается код ответа 200 OK. Тело ответа содержит запрошенную информацию. Данные содержатся в формате JSON в кодировка UTF-8. Веб-сервер Internet Information Services отвечает на запросы. Используется версия протокола HTTP 1.1

Приложение

Листинг кода серверного приложения

```
const express = require("express");
const fs = require("fs");

const app = express();
const jsonParser = express.json();

app.use(express.static(__dirname + "/public"));

app.get("/", function (req, res) {
    var content = fs.readFileSync("Books.txt", "utf8");
    var books = JSON.parse(content);

    res.send(books);
});

app.post("/", jsonParser, function (req, res) {
    var bookId = req.body.id;
    var bookTitle = req.body.title;
    var bookAuthor = req.body.author;
    var book = {id:bookId, title: bookTitle, author: bookAuthor };
    var content = fs.readFileSync("Books.txt", "utf8");
    var books = JSON.parse(content);
    var id = Math.max(...books.map((book) => book.id));

    if (Number.isFinite(id)) {
        book.id = id + 1;
    } else {
        book.id = 1;
    }

    book = { id: id + 1, title: "Fathers and Sons", author: "Ivan Turgenev" };

    books.push(book);

    content = JSON.stringify(books);

    fs.writeFileSync("Books.json", content);
    res.send(books);
});

app.delete("/", function (req, res) {
    var content = fs.readFileSync("Books.txt", "utf8");
    var books = JSON.parse(content);
    var index = 0;
    book = books.splice(index, 1);
    content = JSON.stringify(books);

    fs.writeFileSync("Books.json", content);
    res.send(books);
});
```

```
app.put("/", jsonParser, function (req, res) {
    var bookId = req.body.id;
    var bookTitle = req.body.title;
    var bookAuthor = req.body.author;
    var content = fs.readFileSync("Books.txt", "utf8");
    var books = JSON.parse(content);
    var book = { id: bookId, title: bookTitle, author: bookAuthor };
    var index = 1;

    book = books.splice(index, 1);
    book = {id: 2, title: "The Government Inspector", author: "Nikolai Gogol" }
    book = books.splice(index, 0, book);
    content = JSON.stringify(books);

    fs.writeFileSync("Books.json", content);
    res.send(books);
});

app.options("/", function (req, res) {
    var content = fs.readFileSync("Options.txt", "utf8");
    var options = JSON.parse(content);

    res.send(options);
});

var server = app.listen(8080, function () {
    var host = server.address().address
    var port = server.address().port
    console.log('Example app listening at http://%s:%s', host, port)
       });
```