ЕЛЕНА БЕНКЕН

# Экскурс в REST API и знакомство с Postman

# как инструментом для вызова REST API-методов

Тестирование REST API является важной частью процесса разработки программного продукта. В этой статье мы поговорим о программной архитектуре, рассмотрим что такое API вообще и REST API в частности, познакомимся с Postman — инструментом для тестирования и документирования API-методов.

## Что такое REST?

1. **Client-Server**. Система должна быть разделена на клиентов и серверов.
2. **Stateless**. Сервер не должен хранить какой-либо информации о клиентах. В запросе должна храниться вся необходимая информация для обработки запроса и, если необходимо, идентификации клиента.
3. **Cache**․ Клиенты и промежуточные узлы могут кешировать ответы сервера.
4. **Uniform Interface**. Единый интерфейс определяет взаимодействие между клиентами и серверами.
5. **Layered System**. Допускается разделить систему на иерархию слоев, но с условием, что каждый компонент может видеть компоненты только непосредственно следующего слоя.
6. **Code-On-Demand** (опционально). Возможно выполнение кода на стороне клиента.

Таким образом, REST — способ организации программной системы, построенный на вышеуказанных принципах. Приложения, которые полностью придерживаются этих принципов, называются RESTful. Для приложений, которые частично учитывают принципы REST, используют термин REST-like.

## Преимущества приложения,

## разработанного на основании REST

Рой Филдинг в своей [диссертации](https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm) писал, что применение указанных выше ограничений способствует созданию приложения, которое будет обладать важными преимуществами:

1. **Надёжность**. Обеспечивается за счёт отсутствия необходимости сохранять информацию о состоянии клиента, которая может быть утеряна.
2. **Производительность**. Достигается за счёт использования кеша.
3. **Масштабируемость**. Важна для обеспечения большого числа компонентов в системе и взаимодействий между ними.
4. **Прозрачность взаимодействия между системами** по сети.
5. **Простота интерфейсов**.
6. **Портативность компонентов**. Другими словами, переносимость компонентов системы путем перемещения программного кода вместе с данными.
7. **Легкость внесения изменений**. Можно легко вносить изменения в существующий компонент системы, если он слабо связан с другими компонентами.
8. **Способность эволюционировать**, приспосабливаясь к новым требованиям.

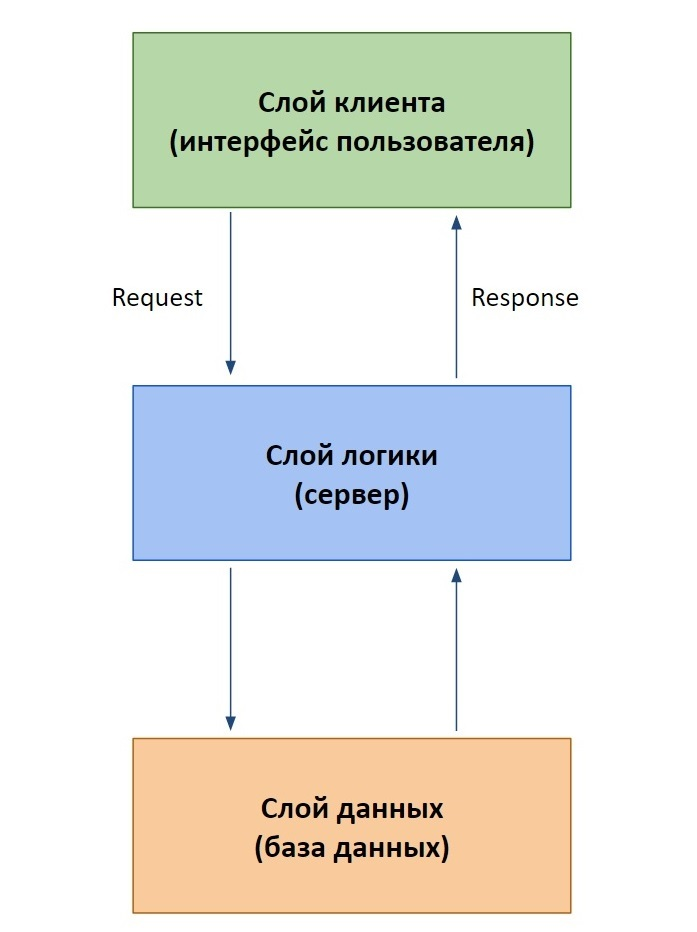
Важно понимать, что на сегодняшний день практически все приложения хотят обладать всеми этими характеристиками. Чтобы разобраться, благодаря чему приложения получают все эти преимущества, необходимо заглянуть чуть глубже и разобраться с принципом построения трёхуровневой архитектуры.

## Трёхуровневая архитектура

Трёхуровневая архитектура предполагает, что инициатором запроса является так называемый «клиент», то есть клиентское приложение. Уровни архитектуры мы далее будем называть «слои» — это общепринятый в ИТ-среде термин.

Итак, выделяют следующие слои трёхуровневой архитектуры:

1. **Слой клиента** (интерфейс пользователя) — необходим для того, чтобы разные клиенты могли единообразно обращаться к бизнес-логике, реализованной на сервере.
2. **Слой логики** (сервер) — обеспечивает выполнение бизнес-логики.
3. **Слой данных** (база данных) — обеспечивает хранение данных. Сервер получает данные и использует их для осуществления бизнес-логики.



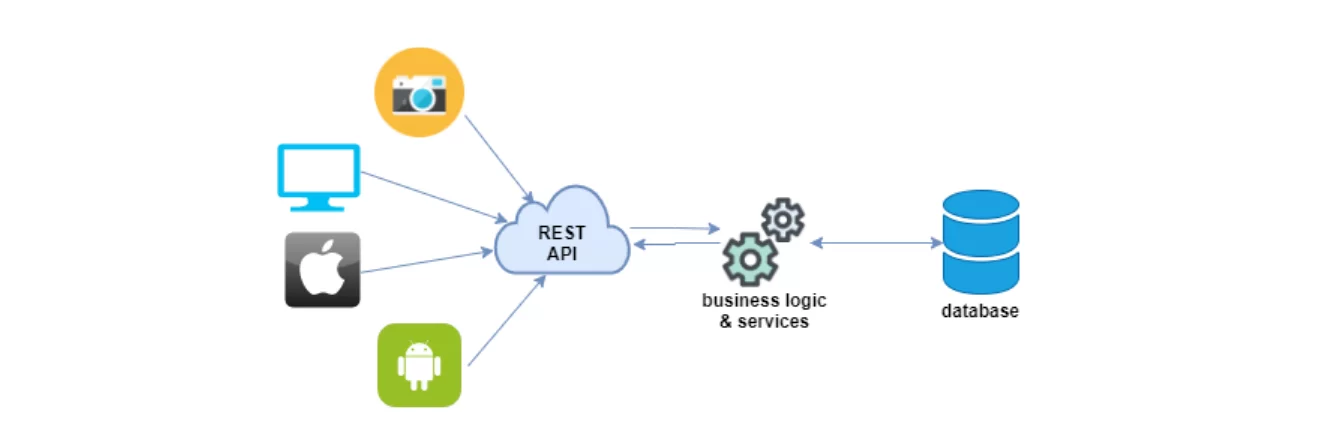
Трёхуровневая архитектура

## Application programming interface (API)

Каждый слой архитектуры взаимодействует только с соседним слоем, то есть интерфейс пользователя не обращается к базе данных напрямую — только через сервер приложений.

Поговорим теперь подробнее о том, как получить все преимущества приложения, разработанного на основании REST.

Для этого рассмотрим следующую схему:



Способы взаимодействия компьютерных программ

Справа на схеме мы видим базу данных, с которой взаимодействует бизнес-логика.

Слева — различные клиенты, которые хотят добраться до бизнес-логики, реализованной на сервере. Как сделать так, чтобы обращение к бизнес-логике было единообразным для всех клиентов? Для решения данной задачи и существует программный интерфейс (на схеме это REST API).

Таким образом, **API (Application programming interface)** — это описание способов, с помощью которых одна компьютерная программа может взаимодействовать с другой.

API отвечает на вопрос «как можно обратиться к системе?» и включает в себя:

* Операцию, которую мы можем выполнить;
* Данные, которые поступают на вход;
* Данные на выходе (данные или сообщение об ошибке).

REST API — это программный интерфейс, построенный по принципам архитектуры REST.

## Вызов операций REST API

Клиентские приложения обращаются к REST API по протоколу HTTP. Другими словами это называется отправляют «HTTP-request». В рамках данной статьи мы будем обращаться к REST API стороннего приложения с помощью программы Postman. Прежде, чем мы приступим непосредственно к практической части статьи, давайте проговорим термины, необходимые для понимания сути происходящего.

**HTTP** — широко распространённый протокол передачи данных, изначально предназначенный для передачи документов. Когда мы говорим «протокол», то имеем ввиду четко специфицированные правила передачи информации, которые клиент и сервер должны соблюдать и выполнять.

Давайте рассмотрим структуру HTTP-запроса, поскольку для обращения к стороннему приложению мы должны сделать как раз это — отправить http-request. Любой HTTP-запрос состоит из трёх частей:

1. **Стартовая строка**

Стартовая строка содержит метод, адрес (URI), а также используемую версию протокола HTTP

Приведу несколько примеров:

“

GET /wiki/index.html HTTP/1.1

GET /index.php?month=august&date=24 HTTP/2.0

Здесь

* «GET» — это метод (или другими словами команда)
* «/wiki/index.html» — это адрес на сервере, к которому мы обращаемся
* далее указываются версии протокола HTTP. Версии. 1.1 больше 20 лет и, в отличие от нее, версия протокола HTTP 2.0 используется исключительно с шифрованием, то есть только HTTPS (кроме того в этой версии протокола есть возможность сжимать данные и передавать их быстрее)
* «month=august&date=24», которые вы можете увидеть во втором примере после знака вопроса — это параметры запрос. Слева от знака «=» это параметр, справа — значение. Конкретно в данном случае в таком виде передается дата «24 августа».

2. **Заголовки** (Headers) прописаны в спецификации HTTP и состоят из имени и значения.

В качестве примера можно посмотреть вот такие заголовки:

“

Host: ru.wikipedia.org

Accept: text/html, application/xml

Accept-Encoding: gzip, deflate

Accept-language: en-GB,en-US;q=0.9,en;q=0.8,de;q=0.7

Первый заголовок представляет собой адрес, куда мы отправляем свой запрос.

Все заголовки прописаны в спецификации протокола HTTP. Заголовок имеет имя. В первом примере мы имеем имя заголовка «Host», после двоеточия указывается значение заголовка. В нашем примере это «ru.wikipedia.org».

Вторая роль заголовков — описание того, кто именно обращается к серверу (еще говорят «стучится»). Мы помним, что важным принципом REST является stateless, что значит сервер не хранит никакую информацию об обращающихся к нему клиентах. Обращаясь к серверу, мы должны объяснить кто мы такие, что мы умеем. Например Accept-language показывает, на каких языках работает клиент.

Надо сказать, что вариантов заголовков может быть множество и они, как правило, явно прописаны в документации.

2. **Тело сообщения**

Если сообщение подразумевает, что мы собираемся передавать какие-то данные, то после заголовка идет тело сообщения. Например, если вы хотите загрузить картинку в социальную сеть, то файл картинки можно передать в теле сообщения.

Однако, картинки не передают с помощью метода GET, он этого не позволяет. Что же делать? На самом деле HTTP-методов достаточно много, и сейчас мы рассмотрим некоторые из них.

## Стартовая строка. Методы HTTP

Для заполнения стартовой строки HTTP-запроса необходимо указать метод HTTP. Говоря по-простому, **метод HTTP** — это команда HTTP, указывающая на основную операцию над ресурсом, с которой начинается первая строка клиентского запроса.

Не все методы HTTP реально используются на практике. Давайте посмотрим наиболее часто используемые:

| Метод | Описание |
| --- | --- |
| GET | Используется для запроса содержимого указанного ресурса |
| HEAD | Применяется для извлечения метаданных, проверки наличия ресурса |
| PUT | Применяется для загрузки содержимого запроса на указанный в запросе URI. Используется, когда надо не создать ресурс, а поменять его значения (атрибуты) |
| DELETE | Удаляет указанный ресурс |
| POST | Применяется для передачи пользовательских данных заданному ресурсу. Например, как раз для передачи упомянутой нами выше картинки. |
| PATCH | Позволяет только создание или полную замену документа |

Часто возникает вопрос, в чём разница между GET и POST. Дело в том, что GET передает все параметры в текстовой строке, не имеет Body (тело сообщения), в отличие от метода POST. Ну, и как видно из таблицы, используются эти методы для разных целей.

## Тело сообщения. Формат JSON

В этой статье мы рассмотрим пример REST API, который использует формат данных JSON. Такой формат используется как для описания тела запроса, так и для ответа сервера.

**JSON** (JavaScript Object Notation) — текстовый формат представления данных в нотации объекта JavaScript.

В качестве значений в JSON могут быть использованы:

* **запись** — это неупорядоченное множество пар ключ: значение, заключённое в «{ }».
* **массив** — это упорядоченное множество значений. Массив заключается в «[ ]».
* **число** (целое или вещественное).
* **литералы** true («истина»), false («ложь») и null.
* **строка** — это упорядоченное множество символов юникода, заключённое в двойные кавычки.

Пример тела ответа в формате JSON:

{

"users":[{

"Id":1,

"name": "Sergey Brin",

"age": 43

}, {

"id":2,

"name": "Larry Page",

"age": 44

}]

}

Данный пример представляет из себя массив, состоящий из 2-х пользователей (записей), обладающих свойствами id, name, age.

Существует огромное количество бесплатных инструментов для просмотра и редактирования формата JSON. Вот один из них: [https://jsoneditoronline.org](https://jsoneditoronline.org/)

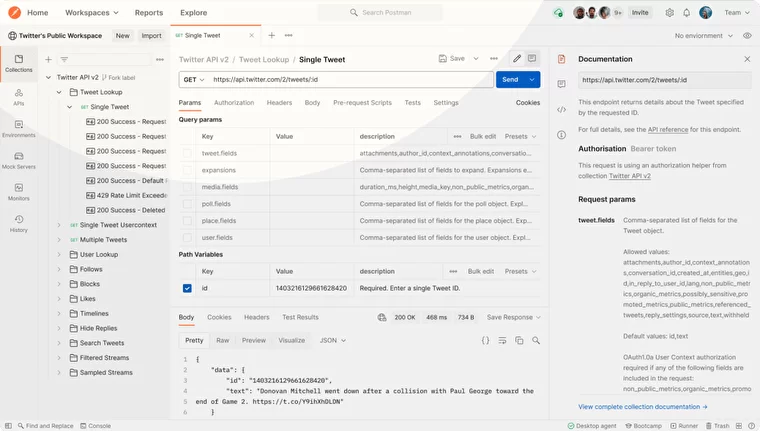
Надо понимать, что REST никак не ограничивает нас в формате передаваемых данных. Вы можете использовать xml или что угодно другое. Однако json, несомненно, можно назвать лидером по использованию в REST API.

## Вызов API с помощью Postman

Назначение Postman

Теперь, когда мы познакомились с REST API и основными составляющими HTTP-запросов, можем перейти непосредственно к вызову этих запросов. Для начала обращения к REST API необходимо установить [Postman](https://www.postman.com/), именно его часто используют как инструмент тестирования и документирования REST API.

**Postman** — бесплатный инструмент для тестирования API, который доступен для работы на Windows, Linux и macOS. Скачать его desktop-версию вы можете по ссылке.



Postman

Основные функции программы:

* отправка запросов и просмотр ответов (выполняет роль клиента API);
* документирование API — создание машиночитаемой документации;
* автоматическое тестирование приложений;
* проектирование и макетирование API;
* мониторинг приложений;
* организация командной работы.

Несомненно, существуют и другие инструменты, но наибольшую популярность на данный момент приобрел именно Postman.

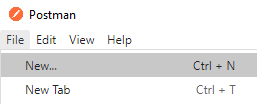
Создание запроса в Postman

Мы будем двигаться шаг за шагом, и я призываю вас повторять эти действия, чтобы наработать практику. Сейчас практически в любой компании от аналитика требуются минимальный опыт работы с REST API.

1. **Создать запрос**

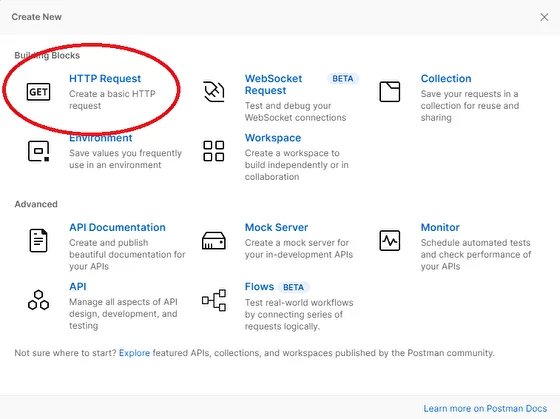
**﻿**

Для начала нажмите File -> New…



Создание запроса

Выберите HTTP Request

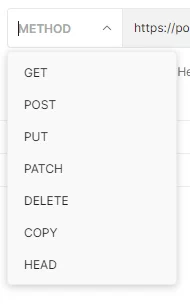


Создание запроса

2. **Выбрать метод**

**﻿**

Давайте укажем метод GET, обычно он проставляется постманом по умолчанию



Выбор метода

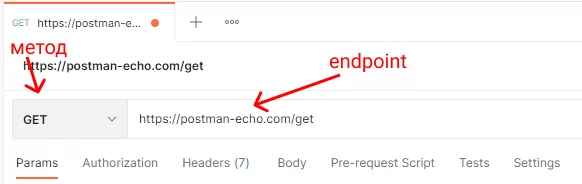
3. **Ввести адрес самого запроса (endpoint)**

**﻿**

Вставьте в строку запроса текст

Этот запрос нужен только для того, чтобы проверить работоспособность приложения и сделать свой первый шаг в работе с REST API.

При копировании обратите внимание, что иногда строка вставляется со знаком пробела или переноса строки в самом конце. Чтобы запрос отработал корректно, необходимо ее удалить.



Отправка запроса

4. **Отправить запрос**

**﻿**

Для отправки запроса нажмите синюю кнопку send.



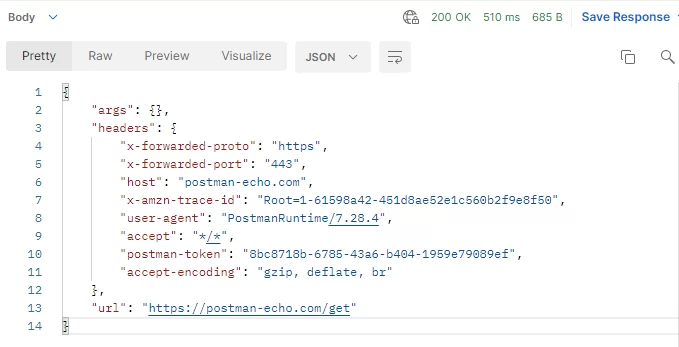
Кнопка отправки запроса

5. **Изучить ответ**

**﻿**

Давайте изучим json-документ, который мы получили. По элементу «headers» мы можем догадаться, что речь идет о каких-то заголовках.

Нам не слишком интересно рассматривать все заголовки, достаточно посмотреть общий принцип их анализа.



json-документ

«Accept» — это то, что мы принимаем.

\*/\* означает, что принимается любой формат.

Тут могло быть написано «application/json», что означало бы, что принимается только запрос с телом в формате json и никак иначе.

Дальше идут куки ([cookies](https://ru.wikipedia.org/wiki/Cookie)), которые мы получили и «[URL](https://ru.wikipedia.org/wiki/URL)» — адрес, на который мы отправили запрос.

Итак, давайте резюмируем. Для отправки запроса необходимо указать следующие данные:

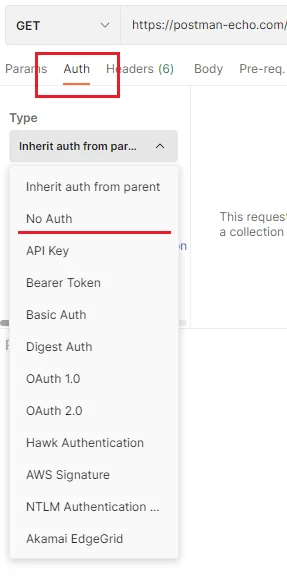
1. HTTP-метод
2. Адрес (URI, endpoint). Endpoint — точка приема запроса на стороне сервера.
3. Заголовки запроса.
4. Параметры запроса.
5. Аутентификационные данные (авторизация).
6. Тело запроса.

Авторизация

Для отправки запроса может потребоваться авторизация на сервере. **Авторизация** — это предоставление определённому лицу или группе лиц прав на выполнение определённых действий. Необходимость авторизации продиктована принципом REST — сервер не должен хранить какой-либо информации о клиентах.

Итак, **авторизация** — это предоставление определённому лицу или группе лиц прав на выполнение определённых действий.

Чтобы заполнить авторизационные данные в Postman, необходимо перейти во вкладку «Authorization» и выбрать способ авторизации.

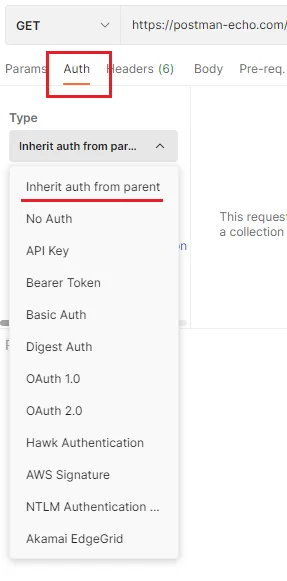


Авторизация

Давайте выберем тип авторизации «No Auth», что означает «без авторизации». Из наиболее распространенных стоит отметить также авторизацию по ключу, авторизацию по токену и OAuth.

В постмане вы можете делать целые коллекции запросов и прописывать правила авторизации как «Inherit auth from parent. Это позволит указать способ авторизации один раз, в коллекции, а дальше все запросы в коллекции будут этот вариант авторизации использовать.

В Postman также предусмотрена возможность создавать коллекции запросов и прописывать правила авторизации для всей коллекции:



Авторизация для коллекции

Ответ сервера

В ответ на наш HTTP-request сервер возвращает ответ. Он может быть как положительный (говорят, 200 ОК), так и вернуть ошибку.

Хотелось бы показать вам, какой может быть структура ответа сервера:

**1. Версия HTTP, код статуса и описание статуса.**

Например:

“

HTTP/1.1 200 OK

В качестве ответа могут прийти такие коды ответов:

| Код и описание статуса | Значение |
| --- | --- |
| 200 ОК | Клиентский запрос успешен, и в ответе сервера содержатся запрошенные данные |
| 403 Forbidden | Запрос отклонен (как правило этот ответ приходит, когда клиент не прошел аутентификацию) |
| 404 Not Found | Документ с указанным URL не существует |
| 500 Internal Server Error | Код указывает на возникновение аварийной ситуации на сервере |

2. **Заголовки ответа**

Например:

“

Date: Fri, 20 Sep 2019 08:17:58 GMT

Server: NCSA/1.5.2

Content-type: text/html

Content-length: 542

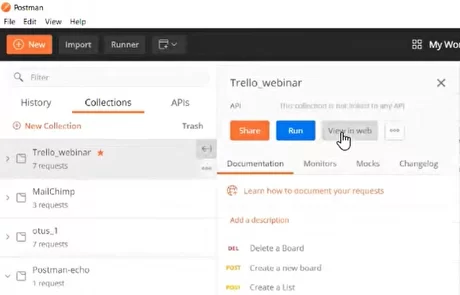
Аналитику как правило, не требуется анализировать эти заголовки.

3. **Тело ответа**

В случае, если клиентский запрос оказался успешным, запрошенные данные посылаются клиенту. Если же запрос выполнить не удалось, дополнительные данные могут содержать текст, объясняющий причину отказа сервера (например, некорректный запрос или недоступность сервера).

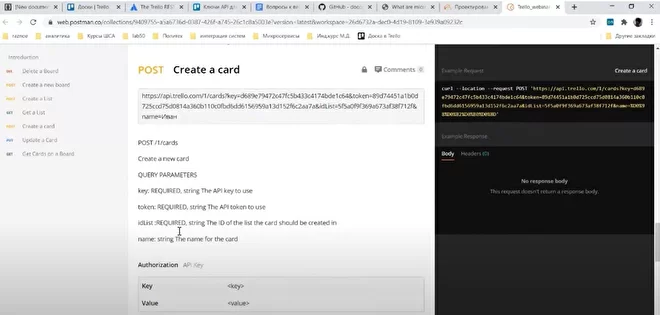
Документирование API методов

С помощью инструмента Postman можно документировать API методы. Для этого необходимо для коллекции выбрать функцию «View in web:



Документирование API методов

В браузере откроется новая вкладка с описанием методов, которые были созданы в коллекции:



Методы коллекции

Надо признать, что документация с помощью Postman получается не слишком подробная (как минимум, нет информации об ответов), но любим мы его, как говорится, не за это.