



# Шпаргалка: Postman, Swagger

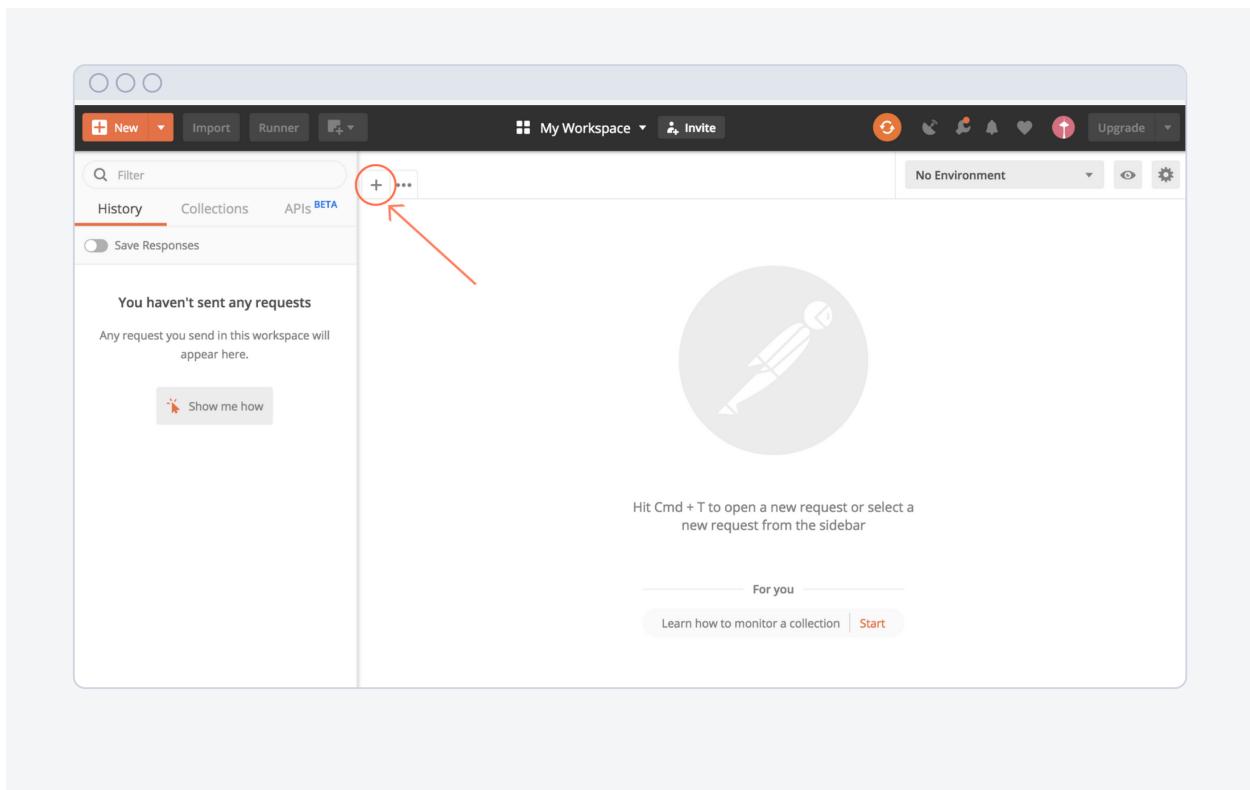
## Тестирование сервера: Postman

Чтобы протестировать API, нужно отправить запросы и проверить ответы сервера. Для этого есть специальные приложения. Самое популярное — Postman.

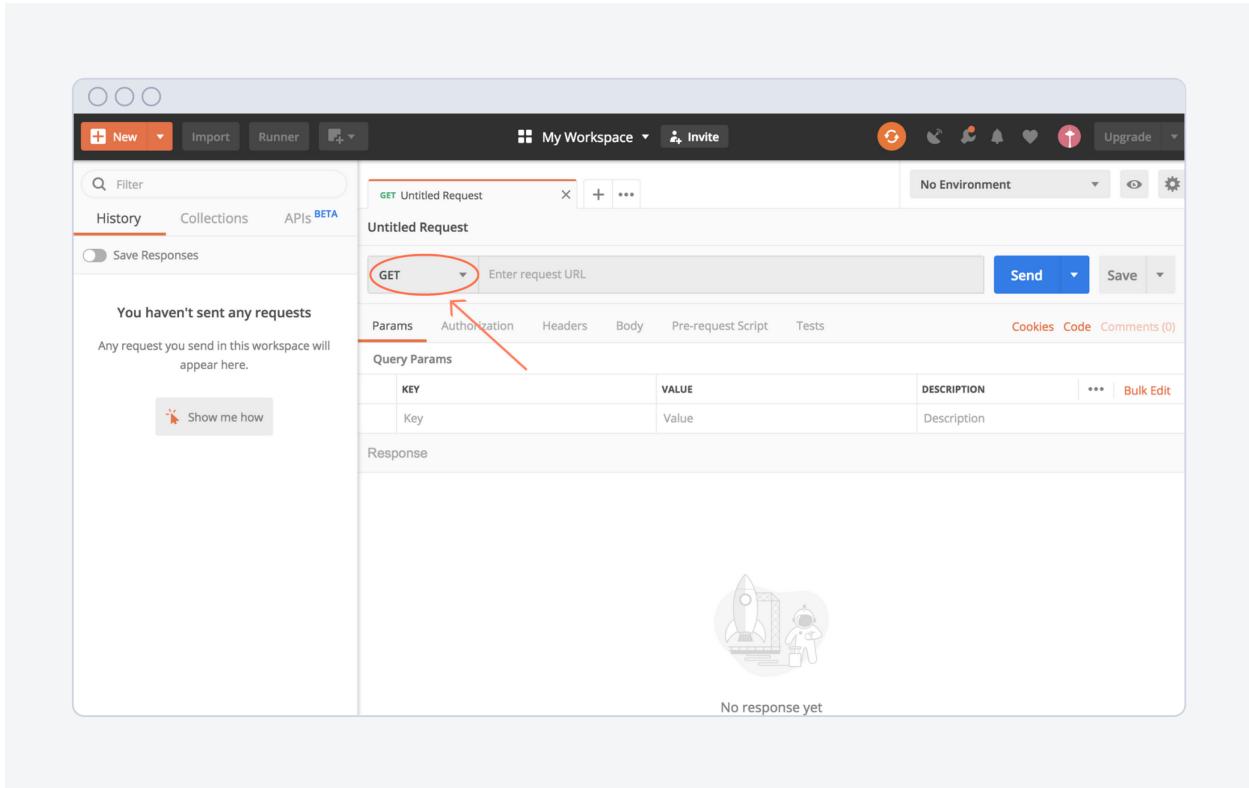
### Установка

Скачать Postman можно [с официального сайта](#).

Когда установишь приложение и зайдёшь в него, увидишь стартовый экран. Нажми на плюс в левом верхнем углу:

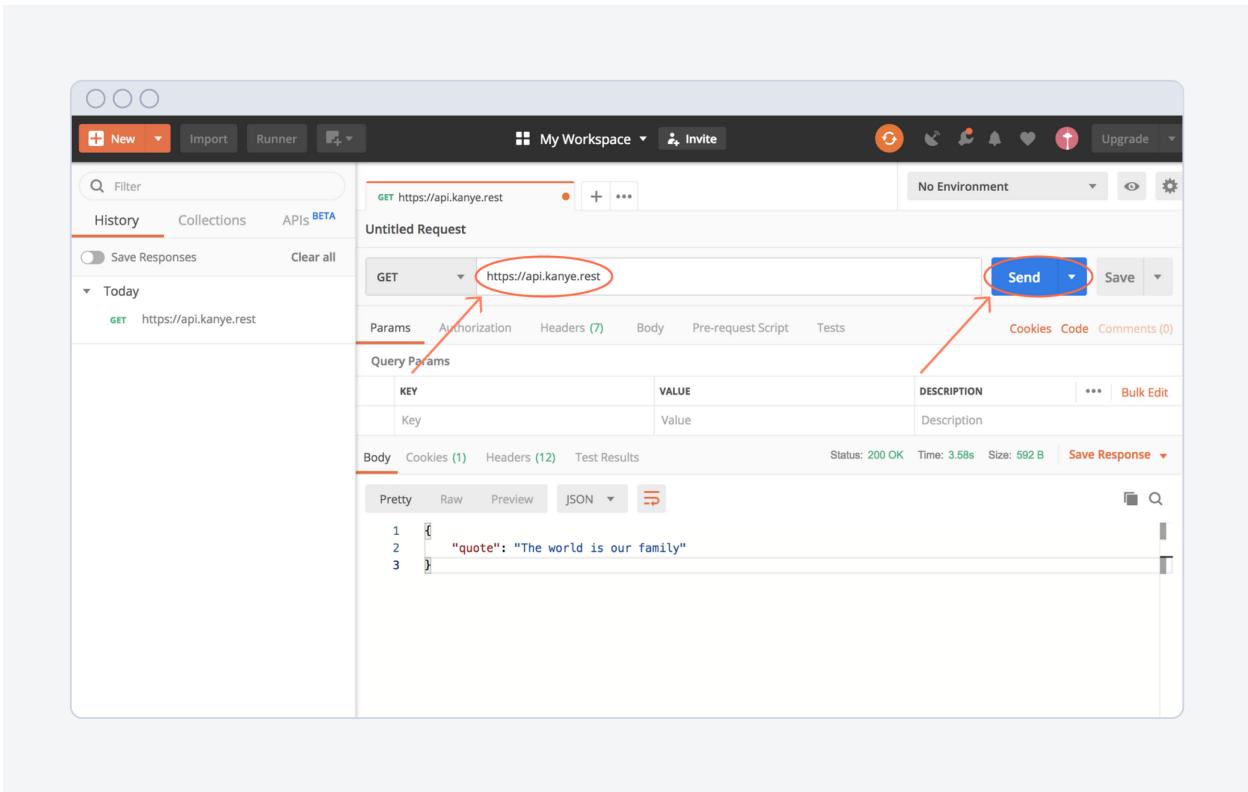


Появится окно, в котором можно создать запрос. В выпадающем списке слева от адресной строки укажи тип запроса. Пока оставь GET:



В адресной строке нужно указать URL, на который нужно отправить запрос.  
Потренируйся на <https://api.kanye.rest> — это открытое API. Оно выдаёт рандомные цитаты Канье Уэста.

Вбей адрес и нажми Send:



Kanye.rest вернёт ответ. В нижней панели отобразятся его статус, тело и заголовки:

## Редактирование запроса

В Postman можно редактировать тело и заголовки запроса. Создай такой запрос:

```
POST https://jsonplaceholder.typicode.com/posts

Content-Type: application/json

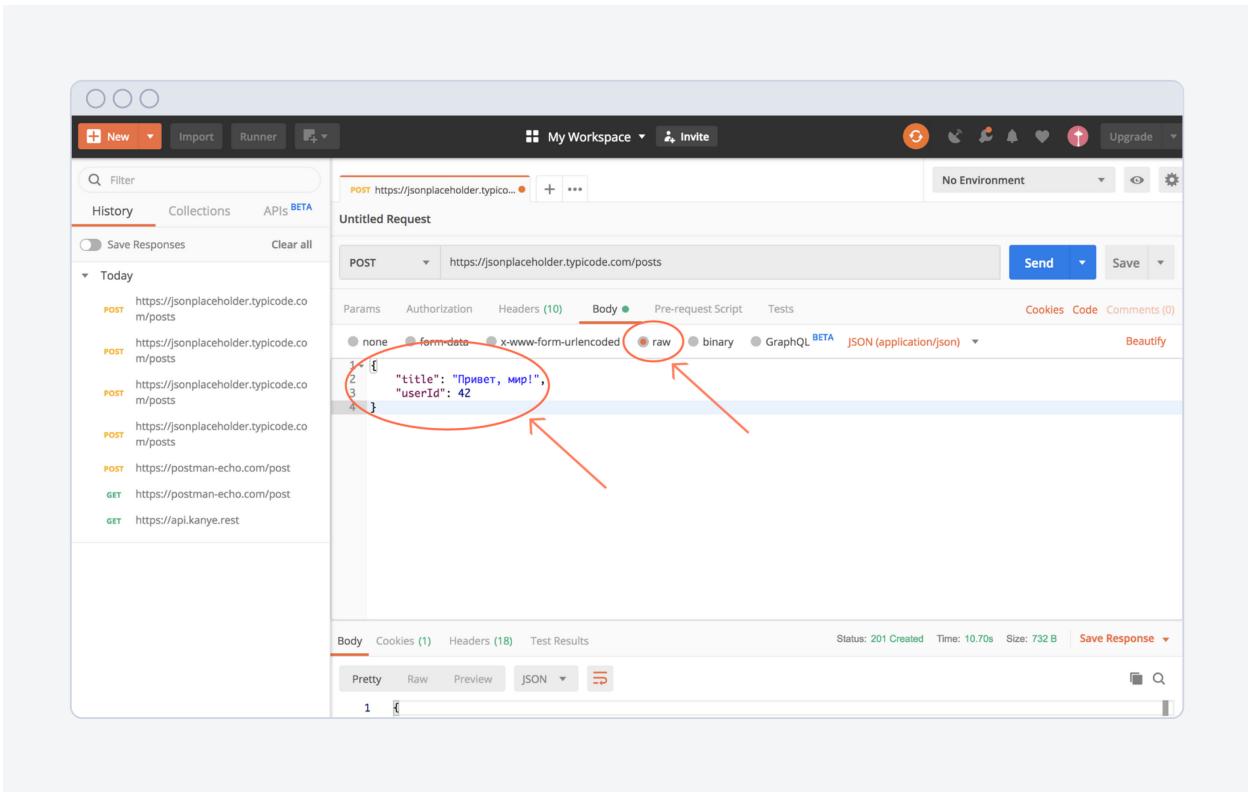
{
  "title": "Привет, мир!",
  "userId": 42
}
```

Сначала сделай POST-запрос. Обрати внимание на URL и заголовки:

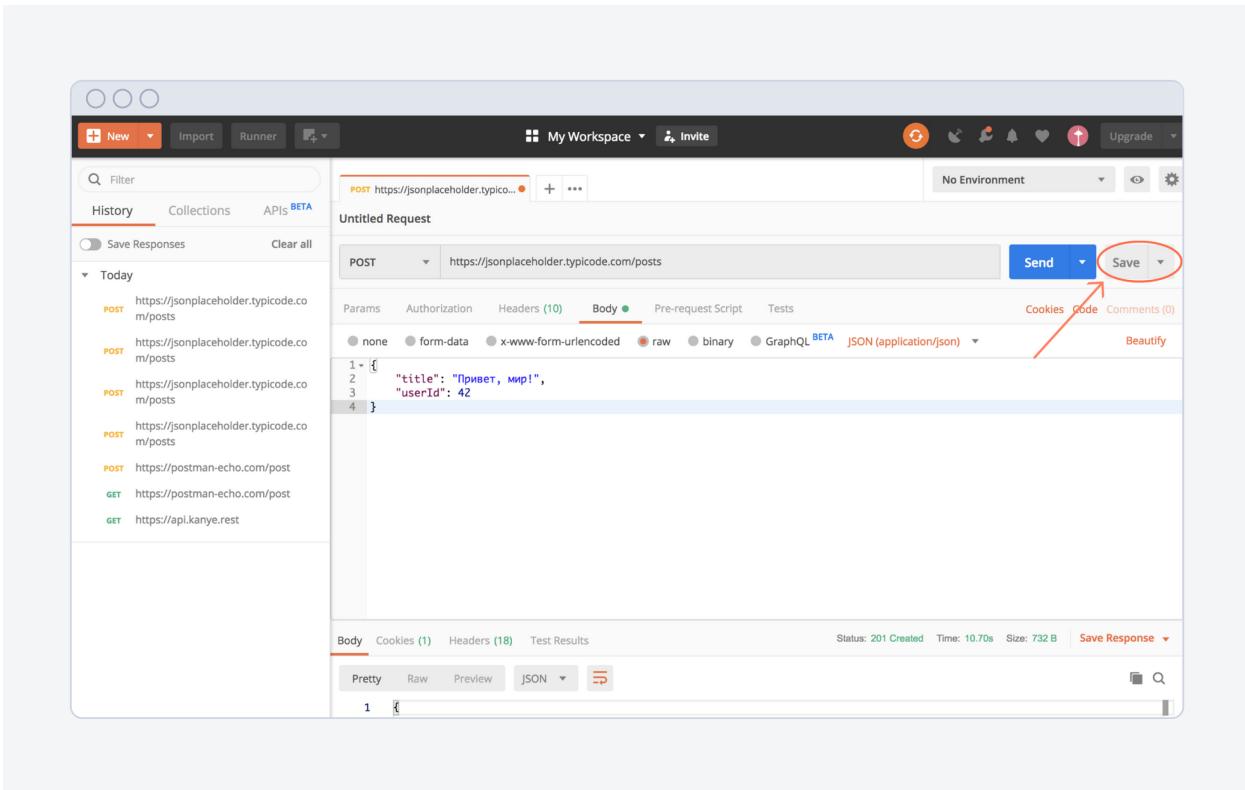
The screenshot shows the Postman application interface. On the left, there's a sidebar with 'History' (selected), 'Collections', and 'APIs BETA'. Below 'History' is a list of recent requests. The main area is titled 'Untitled Request' with a 'POST' method and the URL 'https://jsonplaceholder.typicode.com/posts'. The 'Headers' tab is active, showing a single header 'Content-Type: application/json'. The 'Body' tab is selected, displaying a JSON object:

```
1 {  
2   "title": "Привет, мир!",  
3   "userId": 42,  
4   "id": 101  
5 }
```

Настрой тело запроса. Перейди во вкладку Body, выбери радиокнопку raw и скопирай в текстовую область ниже JSON:



Чтобы сохранить запрос, нажми Save. Это пригодится, если собираешься много раз тестировать сервер с этим запросом:



Postman — простой и удобный инструмент для тестирования запросов. Он сэкономит много времени при разработке сервера или API.

## Swagger

Хранить документацию можно в текстовом виде, если API и документация к нему — простые и небольшие.

Если система сложная, документация разрастается. Тогда её сложнее поддерживать и ориентироваться внутри.

Чтобы упростить работу с документацией, пользуются инструментами, которые генерируют её автоматически. Например, Swagger.

## Как работает Swagger

Разработчик готовит специальный файл с описанием запросов для API; инструмент автоматически переводит её в структуру.

Посмотри на примере Яндекс.Прилавка — это учебное приложение, которое помогает пользователям заказывать продукты домой.

Пример документации к тестовому приложению Яндекс.Прилавок в Swagger:

The screenshot shows the Swagger UI for the Yandex.Prilavok API. The main title is "API Сервера Яндекс.Прилавок 1.0". There are three expandable sections: "API Kartochek", "API Курьерских служб", and "API содержимого Базы данных". Under "API Kartochek", there are four items: a green-outlined POST method for "/api/v1/cards", and three blue-outlined GET methods for "/api/v1/cards", "/api/v1/cards/{id}", and "/api/v1/cards/{id}/kits". Under "API Курьерских служб", there are two items: a blue-outlined GET method for "/api/v1/couriers" and a green-outlined POST method for "/api/v1/couriers/check". Under "API содержимого Базы данных", there is one item: a blue-outlined GET method for "/api/db/resources/\*.csv". A "Authorize" button is located at the top right.

Каждая строка — это тип запроса. Цвет — HTTP-метод, которым нужно делать запрос. Например, POST выделен зелёным, а GET — синим.

В каждой строке указан путь, который нужно добавить к URL, чтобы выполнить запрос. Например, приложение Яндекс.Прилавок расположено по адресу

<http://prilavok.yandex.ru>.

Например, нужно сделать запрос, который поможет получить все карточки в приложении. Запрос структуры `URL + путь` будет выглядеть так:

<http://prilavok.yandex.ru/api/v1/cards>

Запрос вызывает в API соответствующую функцию: клиент обращается к серверу.

## Эндпоинт

Часть сервиса, к которой отправляет запрос клиент, называют **эндпоинт** (endpoint, «конечный пункт»). Эндпоинт также называют «метод API», «URL ресурса» или просто «ручка».

Когда к нему обращаются по URL, эндпоинт выполняет нужное действие.

В документации можно кликнуть по строке с методом API, чтобы посмотреть подробную информацию о запросе.

Например, GET-запрос на скриншоте позволяет узнать, из каких наборов и продуктов в них состоит карточка. Идентификатор карточки `cardId` передаётся в параметре URL:

## Обязательные поля

В документации указывают обязательность заполнения полей:

Например, чтобы добавить продукты в набор, нужно ввести `id` набора и `body` (тело) запроса. В Swagger обязательные параметры подсвечиваются красным

\*required .

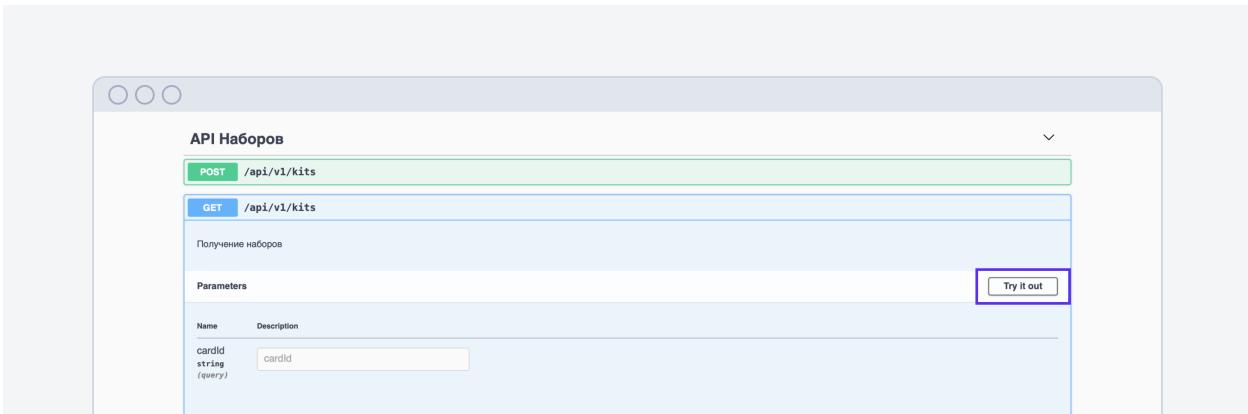
В блоке **Response** можно увидеть список ответов с кодом и описанием. Здесь же есть пример данных, которые возвращаются на запрос в теле сообщения:

The screenshot shows the Swagger UI for a `GET /api/v1/kits` endpoint. The top section is titled "Получение наборов". It has a "Parameters" table with one row: `cardid` (string, query). A "Responses" section follows, with a "Responses" table. The first row is for code `200`, which has a description "Успешное получение набора" and an example value of a JSON array representing a kit with products. The second row is for code `400`, which has a description "Не все необходимые параметры были переданы" and an example value of a JSON object with `code: 400` and `message: "Не все необходимые параметры были переданы"`.

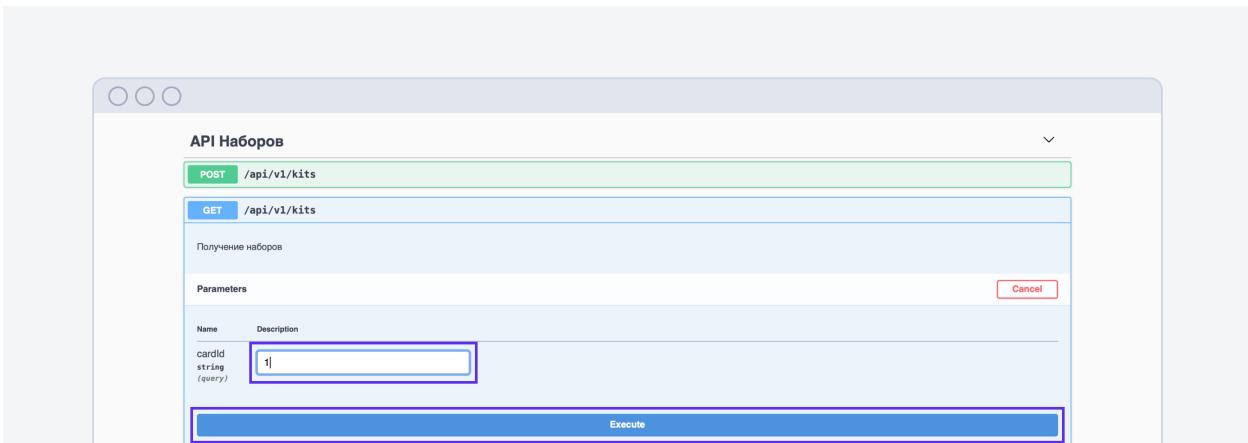
Например, можно отправить GET-запрос, чтобы узнать, из каких наборов и продуктов в них состоит карточка. Если запрос выполнен успешно, вернётся **Code: 200**. Информация по запросу должна появиться в теле ответа в чёрной рамке.

Если не передать обязательный параметр, вернётся **Code: 400**. В теле ответа в чёрной рамке появится сообщение об ошибке.

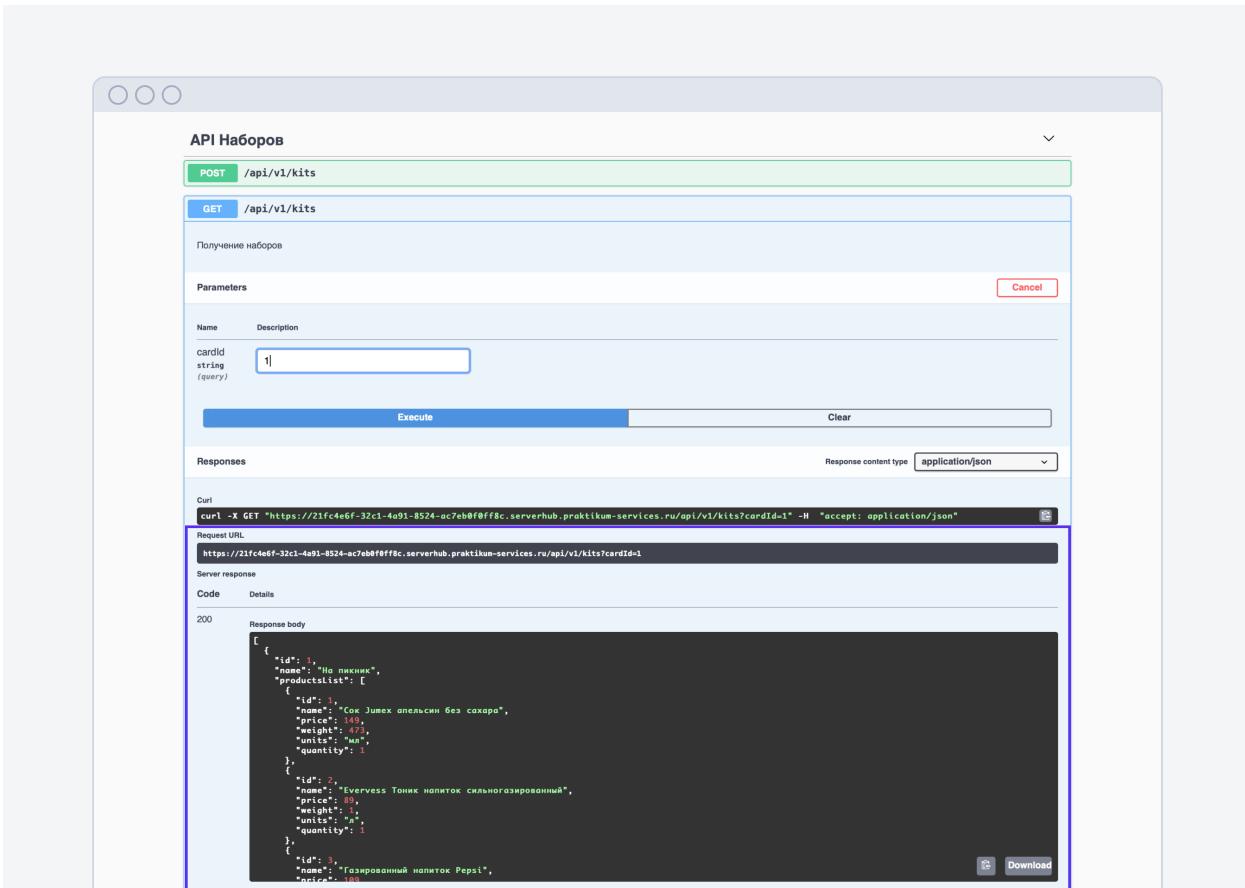
Через документацию в Swagger можно сразу отправлять запросы, как в Postman. Для этого нужно выбрать эндпоинт и нажать кнопку Try it out.



Если запрос содержит параметры, нужно заполнить их поля: например, идентификатор `cardId` карточки, чтобы получить список наборов и продуктов в них. Чтобы отправить запрос, нужно нажать кнопку Execute.



Будет выполнен запрос по указанному URL. Когда сервер обработает запрос, ниже появится информация: по какому URL был выполнен запрос и какой вернулся результат.



## Авторизация

Чтобы пользоваться функциями приложения в браузере, тебе приходилось проходить авторизацию: вводить логин и пароль.

Для API это работает так же: большинство ручек не будут работать с неавторизованным пользователем.

## Как авторизоваться в API

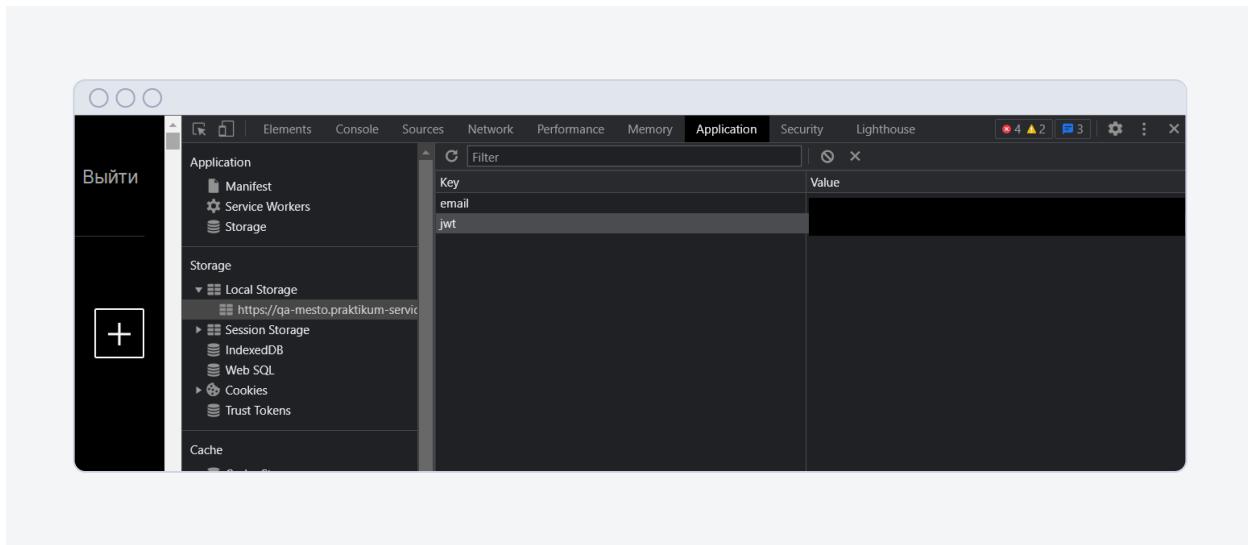
Есть разные **протоколы авторизации**. Приложением Mesto использует протокол [OAuth2](#).

Чтобы авторизоваться, в заголовках запроса нужно передать **токен доступа JWT** — набор символов, с помощью которого клиент сервиса подтверждает свою личность.

## Как получить токен

Чтобы получить токен:

1. Открой браузер Google Chrome и авторизуйся в веб-интерфейсе [приложения Mesto](#).
2. В этой же вкладке браузера открой DevTools.
3. Перейди на вкладку Application.
4. В меню слева разверни выпадающий список Local Storage раздела Storage.
5. Кликни по строке, соответствующей адресу приложения <https://qa-mesto.praktikum-services.ru/>.
6. В правой части окна DevTools отобразится список пар ключ-значение.  
Значение, которое соответствует ключу `jwt`, — это токен авторизации.

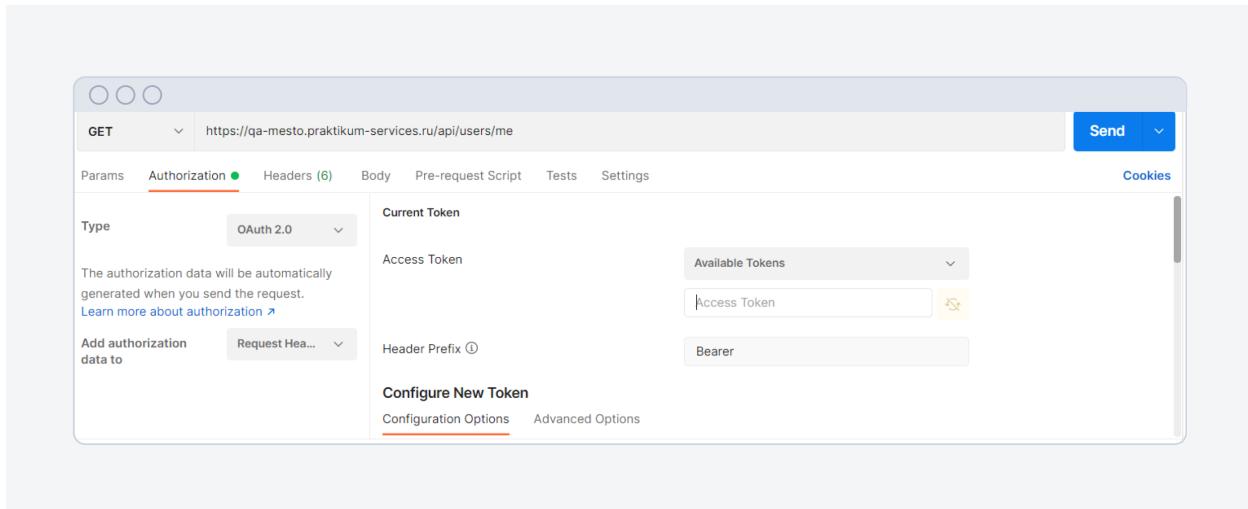


Обрати внимание: токен, как и свой пароль, не нужно передавать коллегам. Так никто не сможет работать в приложении от твоего имени.

## Как добавить токен к запросу

Чтобы добавить токен к запросу:

1. В Postman перейди на вкладку Authorization — в настройках запроса.
2. В выпадающем списке Type выбери протокол OAuth2.0.
3. Введи токен в поле ввода с плейсхолдером Access Token в правой части окна:



Время жизни токена, как правило, ограничено. В зависимости от настроек приложения это от нескольких секунд до нескольких дней.

Если на твои запросы приходят ответы со статус-кодом `HTTP 401 Unauthorized`, нужно получить новый токен и внести его в настройки запроса.