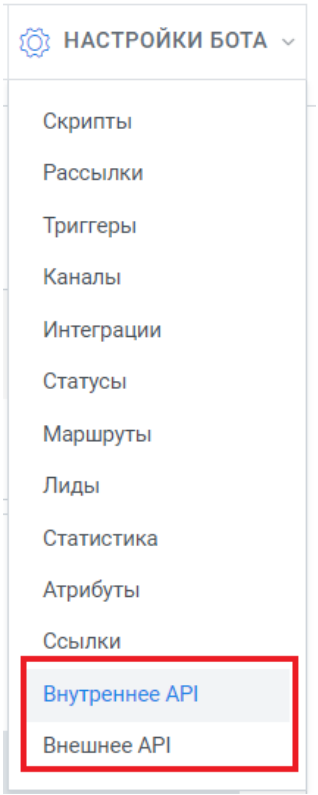
Конструктор API

«API — это продукт. API — это также то, как вы разрабатываете, продаете и доставляете цифровые и все чаще физические продукты своим клиентам»
— Кин Лейн, главный евангелист Postman

Коммуникационная low-code платформа Metabot обладает встроенным конструктором API для организации взаимодействия с внешними системами по REST в соответствии со стандартом OpenAPI 3.

У любого чат-бота, создаваемого на Metabot, есть внутреннее API — для обращений внешних систем к чат-боту, и внешнее API — для обращений чат-бота к внешним системам.

API состоит из **точек интеграции**, называемых **эндпоинтами (endpoints)** или **конечными точками** (точками доступа, точками связи и т.п.). Настройка точек интеграции происходит в разделах **Внутреннее** и **Внешнее API** доступное из меню **Настроек бота**.



Внутреннее API представляет из себя конструктор API для вашего чат-бота. Здесь вы можете создать и кастомизировать ваши точки доступа к чат-боту, задать **алиас (alias)** точки и возвращаемый ответ с помощью JavaScript. В JavaScript можно планировать другие задания в очереди работ чат-бота (асинхронные), а также вызывать внешнее API (колбэк/callback).

Внешнее API представляет из себя конструктор API для обращения к любым внешним ресурсам. Здесь, в едином списке вы можете описать конечные точки доступа к API внешних ресурсов, описать как отправлять запросы, с каким содержимым, какими заголовками и т.д. и далее вызывать эти внешние точки из любой точки платформы, где доступен вызов JavaScript, например, из диалогового сценария чат-бота.

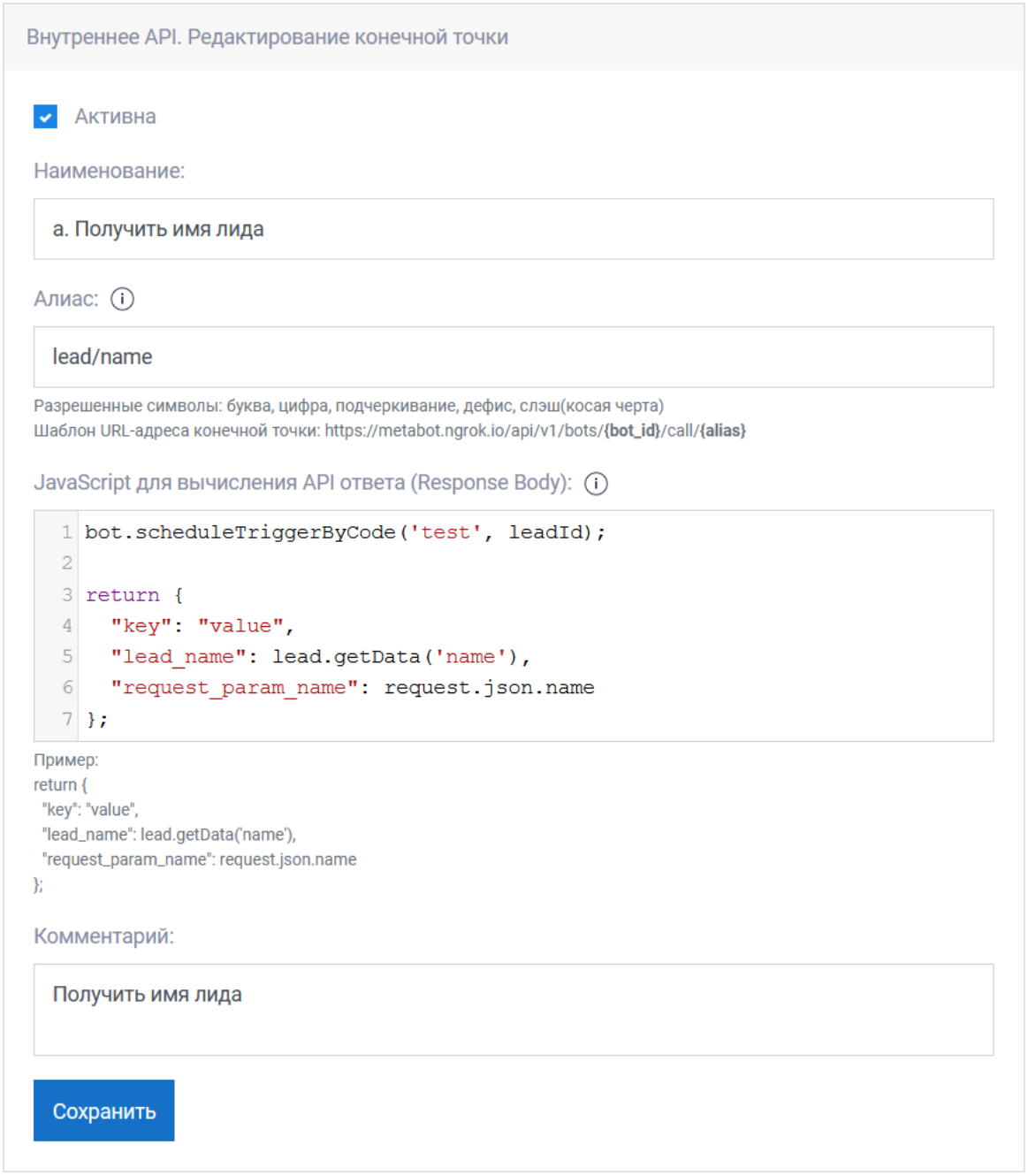
Кастомное внутреннее API является синхронным, это значит что для возвращения ответа по API не будет планироваться отдельного задания в очереди, поэтому очень важно не делать сложные вычисления в таком API.

Удобство синхронного API в том, что вы моментально получаете ответ на запрос, нет необходимости отдельно опрашивать результат выполнения задания или устанавливать отдельный колбэк. При этом, в JavaScript, являющимся телом API запроса можно также планировать любые другие «вложенные» задания (например триггеры, скрипты конструктора или рассылки), такие задания будут выполнены асинхронно, для них в очереди будут спланированы и выполнены отдельные задания.

Если вы хотите получить ответ от внешнего API позже, в момент выполнения отдельного асинхронного «вложенного» задания, то для этого вам необходимо вызвать внешний API в запущенном скрипте или триггере (с помощью внешнего эндпоинта или JS функции api.callEndpoint). Таким образом, вы можете создать любую цепочку связанных заданий и получить результат идентичный вызову коллбеков в необходимых точках.

Вызвать внешнее API можно с помощью JS функции **api.callEndpoint(...)**, подробности смотрите ниже, в разделе с описанием внешнего API.

Внутреннее API



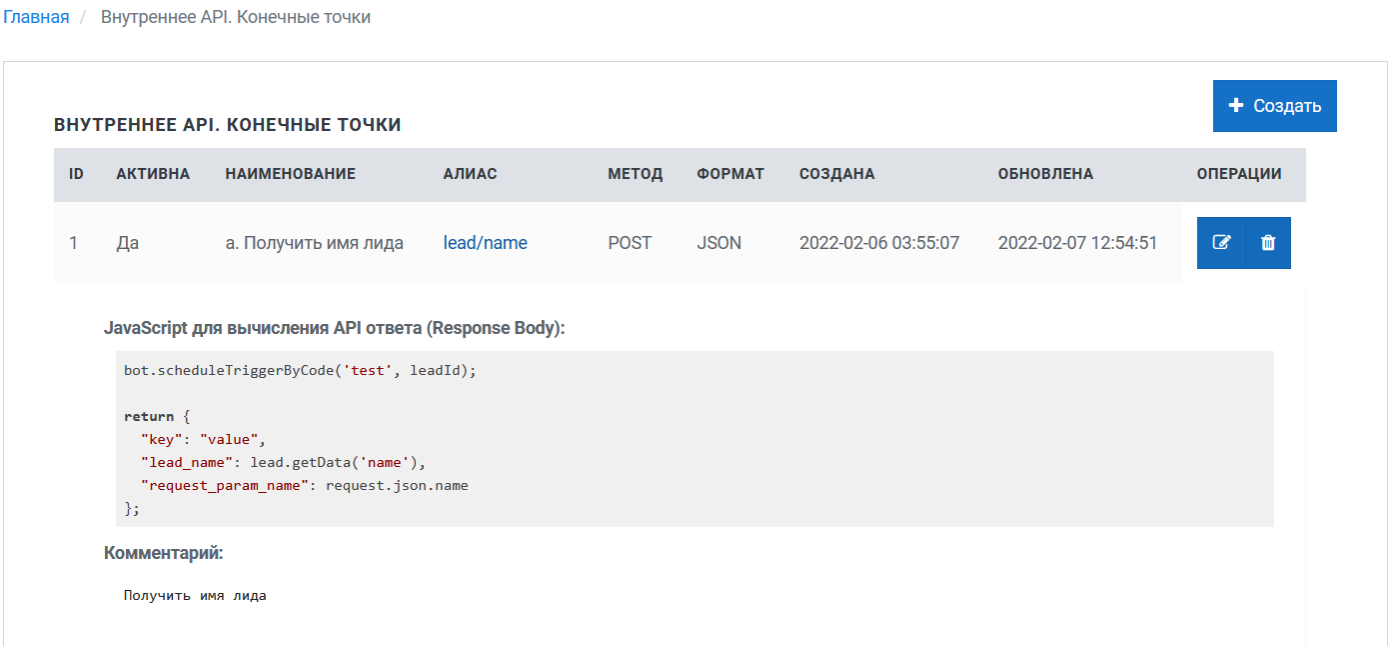
Структура конечной точки внутреннего API

Описание полей внутреннего эндпоинта представлена в таблице ниже.

Поле	Описание
Флаг активности	Если убрать флаг, то при обращению через API к данному endpoint будет возвращена ошибка.

Алиас	Дополнительный уникальный идентификатор. Используется как часть URL-адреса для обращения к конечной точке через API, а также для вызова в JavaScript функциях. Разрешенные символы: буква, цифра, подчеркивание, дефис, слэш (косая черта) Шаблон URL-адреса конечной точки: https://адресплатформы/api/v1/bots/**{bot_id}**/call/**{alias}** Для каждого бота создается свое пространство имен для API, что означает Алиас должен быть уникален в пределах бота.
JavaScript для вычисления API ответа (Response Body)	JavaScript для вычисления содержимого API ответа. Данный JS код будет выполнен при обращении к внутреннему API (по алиасу), результат выполнения кода будет являться телом ответа на API запрос. Это значит, что API ответ вычисляется на основе JavaScript. В JavaScript должен присутствовать ключевой оператор return для возврата значения из JS. Выражение которое указывается после return является ответом (Response Body) на API запрос. Можно возвращать: • Строку (будет сконвертирована в json, если формат валидный); • Массив; • Объект (в том числе JSON-объект). Пример: `return {` `"key": "value",` `"lead_name": lead.getData('name'),` `"request_param_name": request.json.name` `};` В JS можно выполнять любой код, например обращаться к пользователям, или устанавливать переменные в memory, планировать задания бота (триггеры, рассылки, скрипты), а также вызывать внешнее API с помощью api.callEndpoint или другие методов объекта api: post/get/postJson/getJson/request.
Комментарий	Любой текст. Доступно указание любых HTML-тэгов.

На данный момент доступен вызов кастомного внутреннего API только с методом POST и в формате JSON. Методы GET, PUT, PATCH, DELETE не поддерживаются.



Список всех конечных точек внутреннего API сортируется по следующим полям: наименование, алиас, id. Поэтому вы можете управлять сортировкой записей изменяя наименование или алиас. В будущем будет добавлена произвольная сортировка.

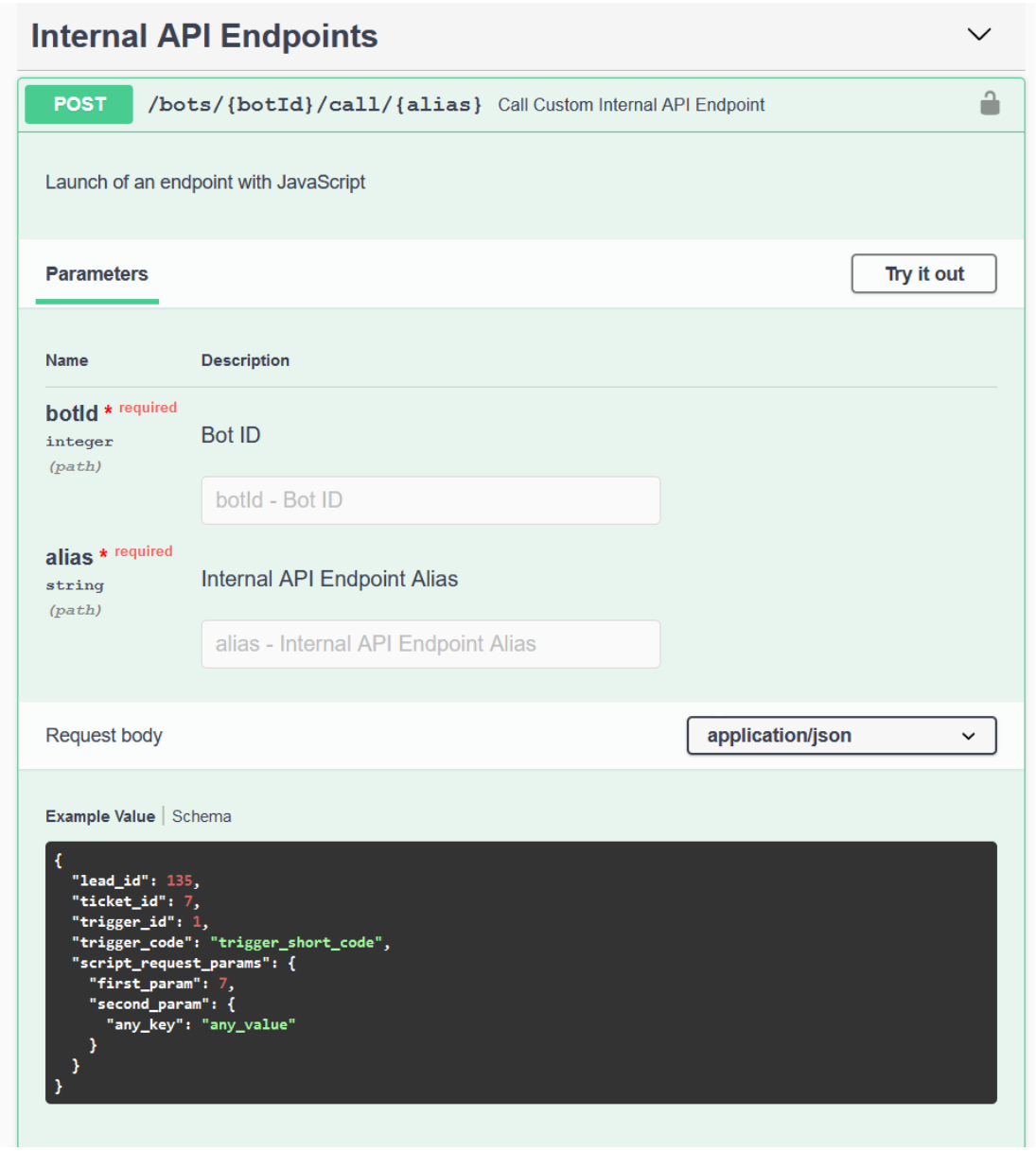
Вызов внутреннего API

Узнать структуру запроса и проверить работу внутреннего API можно с помощью Swagger, доступного по адресу платформы. Если платформа устанавливается в контуре вашей сети, то адрес сервера нужно узнать у вашей ИТ службы.

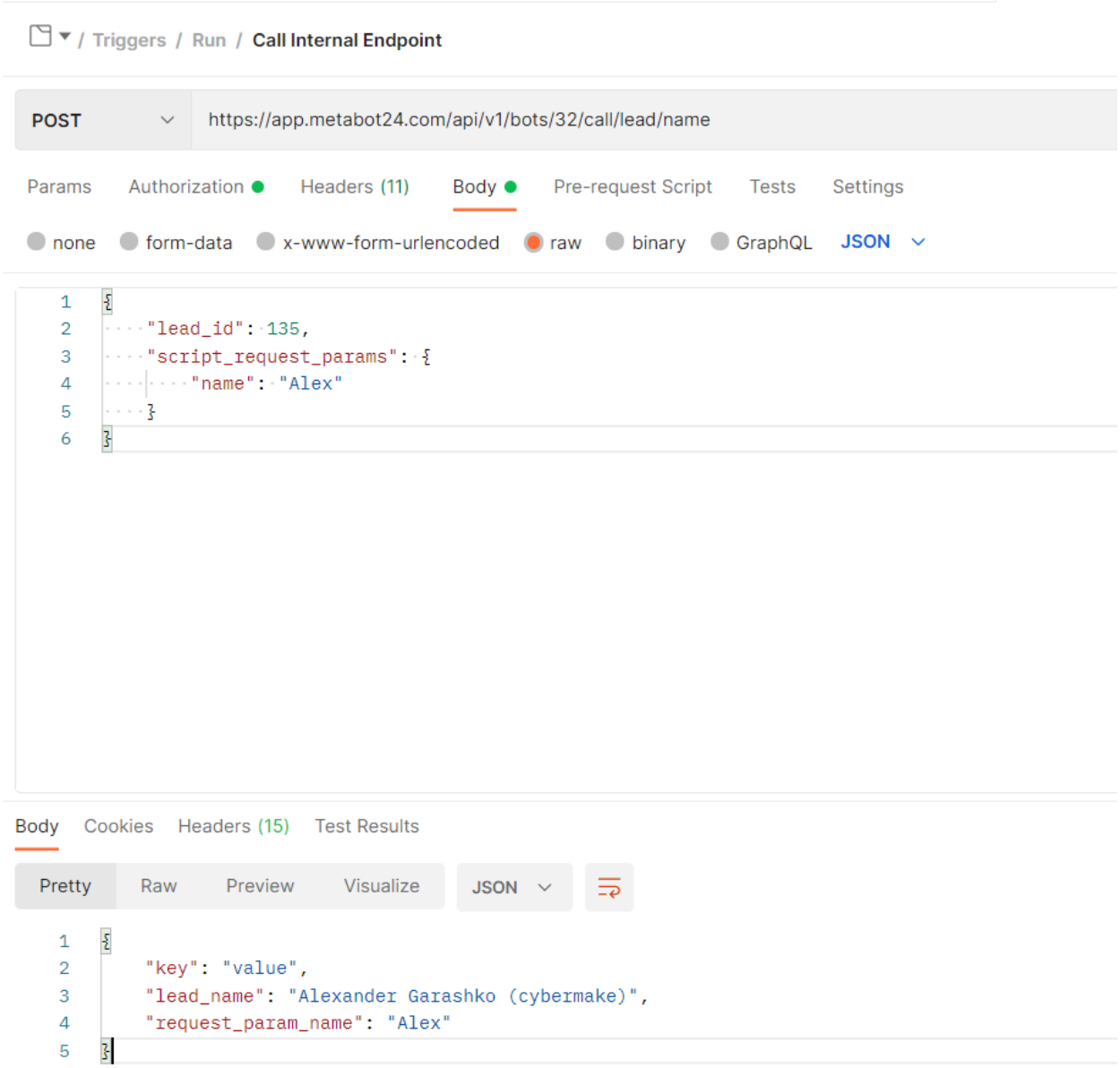
API публичного сервиса под управлением нашей команды можно посмотреть по ссылке.

Чтобы получить доступ к Swagger к публичному серверу, нужно **зарегистрировать аккаунт** на платформе по ссылке, затем подтвердить электронную почту и авторизоваться на платформе.

Внимание! При работе в Swagger платформы будут выполняться реальные API запросы, а не эмуляция API.



????? ????????? ?????? ??????????? API ????? c ??????? Postman ??? ?????? ??????? ????? ??? ???????????? API.



Параметры запроса

В URL запроса нужно указать идентификатор бота **botId** на платформе и **alias** внутреннего эндпоинта в боте.

В теле запроса размещаются данные запроса, например:

```
{
  "lead_id": 135,
  "ticket_id": 7,
  "trigger_id": 1,
  "trigger_code": "trigger_short_code",
  "script_request_params": {
```

```
    "first_param": 7,
    "second_param": {
      "any_key": "any_value"
    }
  }
}
```

С параметрами тела запроса к внутреннему эндпоинту можно ознакомиться в таблице ниже.

Поле	Обязательность заполнения	Описание
lead_id	-	В JS будет подготовлен и доступен полноценный объект lead, если у лида есть персона, то также будет доступен объект person.
ticket_id	-	В JS будет подготовлен и доступен полноценный объект ticket.
trigger_id или trigger_code	-	В JS будут доступны переменные triggerId, triggerName.
script_request_params	-	В JS будет подготовлены и доступны полноценные объекты request и last_request.

Авторизация запросов API

Для авторизации запросов чат-бота используется **Authorization Bearer Token**.

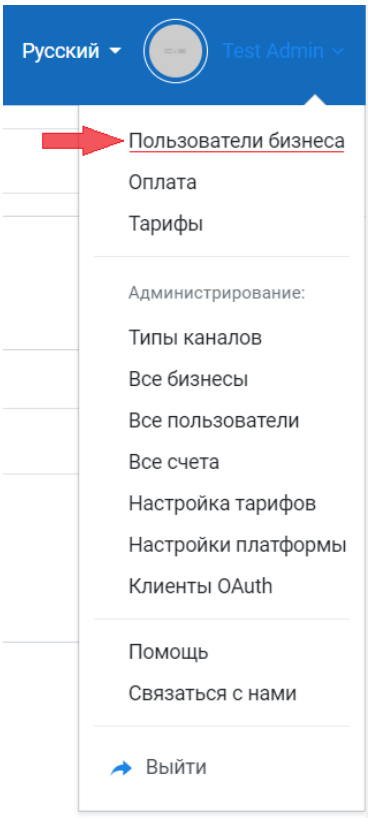
При каждом вызове API добавьте в заголовок авторизации <Access Token>. Параметры заголовков для всех запросов:

```
authorization: Bearer <Access Token>
Host: https://app.metabot24.com
```

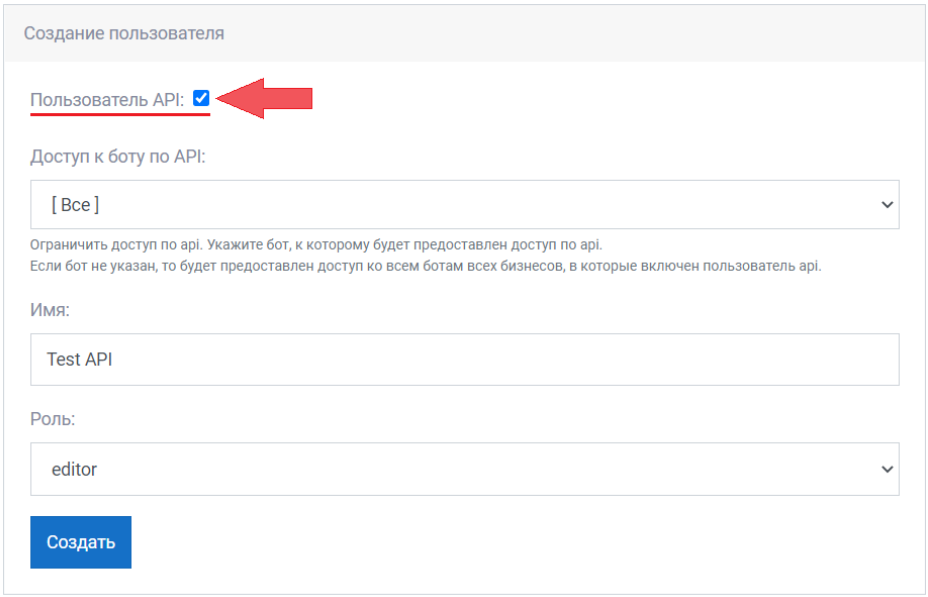
Генерация токена доступа к API

Чтобы получить API токен, нужно создать API пользователя и сгенерировать токен в в Админ-панели Metabot, для этого:

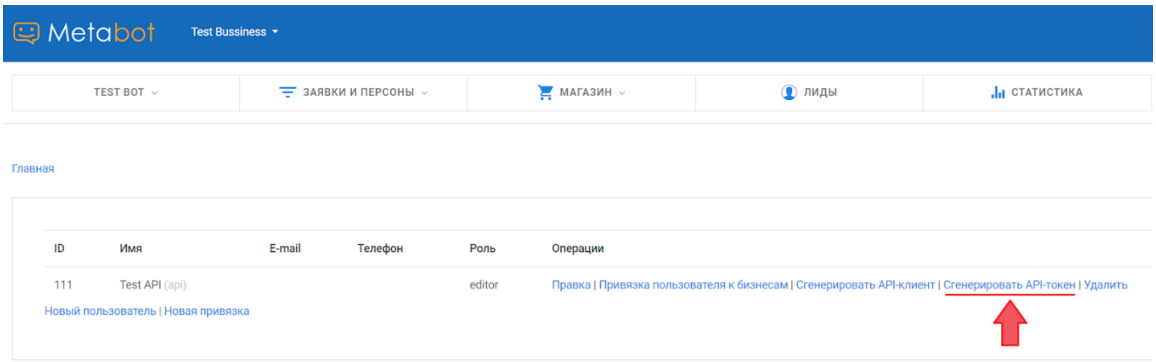
1. Создайте или привяжите нового пользователя в разделе **Пользователи бизнеса**.



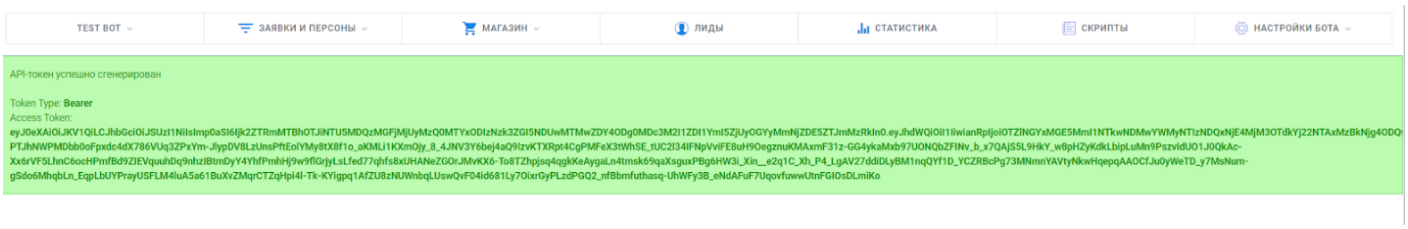
2. При создании пользователя включаем опцию **Пользователь API** и указываем имя пользователя. Если необходимо указываем роль пользователя и доступ к какому боту нужно получить по API, или оставляем поля по умолчанию.



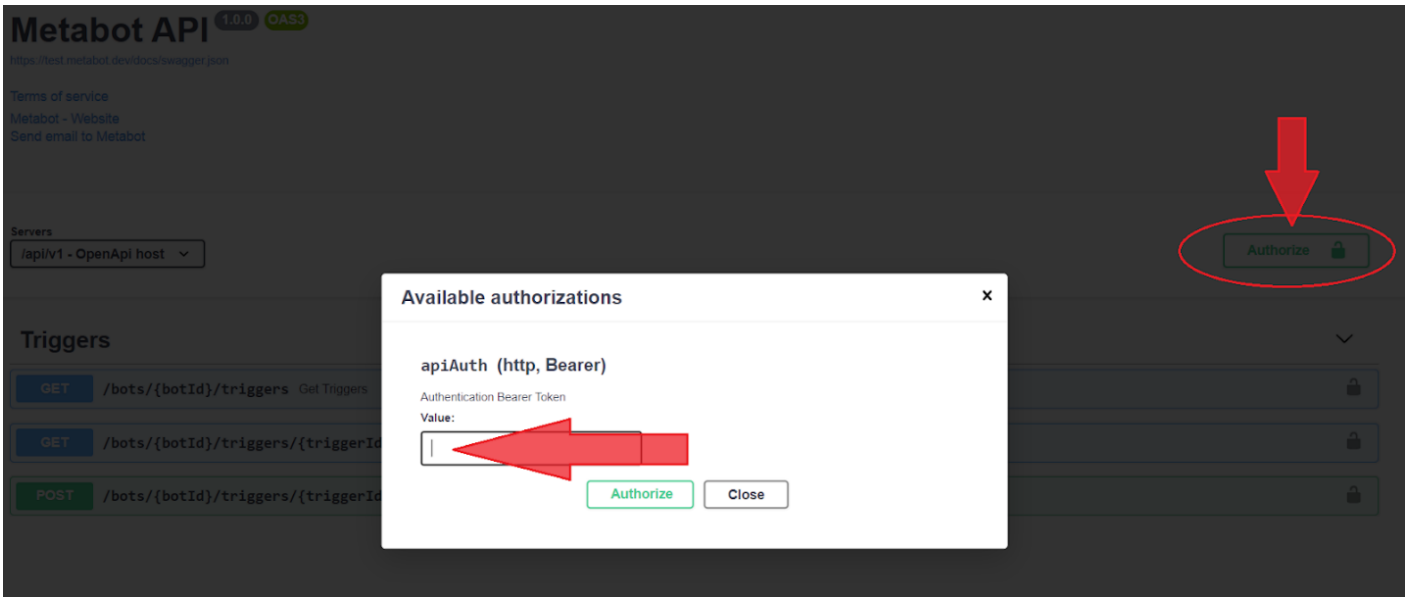
3. Нажмите на ссылку **Сгенерировать API-токен**.



4. Скопируйте полученный API-токен и используйте его в ваших запросах.



5. Можете использовать скопированный API-токен для API на странице Swagger (на странице /api/docs). Токен указывается на форме авторизации, которая открывается нажатием на кнопку **Authorize**.



> Если вдруг не найдете опцию создания API токена в вашем аккаунте, запросите токен в поддержке.

Внешнее API

ВНЕШНЕЕ API. КОНЕЧНЫЕ ТОЧКИ

ID	АКТИВНА	НАИМЕНОВАНИЕ	КОД	URL КОНЕЧНОЙ ТОЧКИ (ENDPOINT URL)	МЕТОД	ФОРМАТ
1	Да	1 Первый тест	1	https://webhook.site/c2c82a72-0390-4c6c-8313-d2ccee6b1a30	POST	JSON

JavaScript для вычисления API запроса (Request Body):

```
let requestParamName = null;
if (request.json) {
  requestParamName = request.json.name;
}

api.setHeaders({"some_param": "some value"});

return {
  "key": "value",
  "lead_name": lead.getData('name'),
  "request_param_name": requestParamName,
  "camelCaseParam": "cameCaseValue",
};
```

Заголовки HTTP запроса (HTTP Headers):

```
"{"token": "shmoken👍👧😅🤷😢👧😁😃"}"
```

Комментарий:

```
Комментарий Комментарий Комментарий
Комментарий Комментарий Комментарий
https://app.metabot24.com
https://app.metabot24.com
```

При создании новой конечной точки открывается окно создания со следующими параметрами:

Внешнее API. Редактирование конечной точки

☑ Активна

Наименование:

1 Первый тест

Код: ⓘ

1

URL конечной точки (Endpoint URL):

https://webhook.site/c2c82a72-0390-4c6c-8313-d2ccee6b1a30

JavaScript для вычисления API запроса (Request Body): ⓘ

```
let requestParamName = null;
if (request.json) {
  requestParamName = request.json.name;
}

api.setHeaders({"some_param": "some value"});

return {
  "key": "value",
  "lead_name": lead.getData('name'),
  "request_param_name": requestParamName,
  "camelCaseParam": "cameCaseValue",
};
```

Пример:

```
return {
  "key": "value",
  "lead_name": lead.getData('name'),
  "request_param_name": request.json.name
};
```

Метод(тип) HTTP запроса:

POST

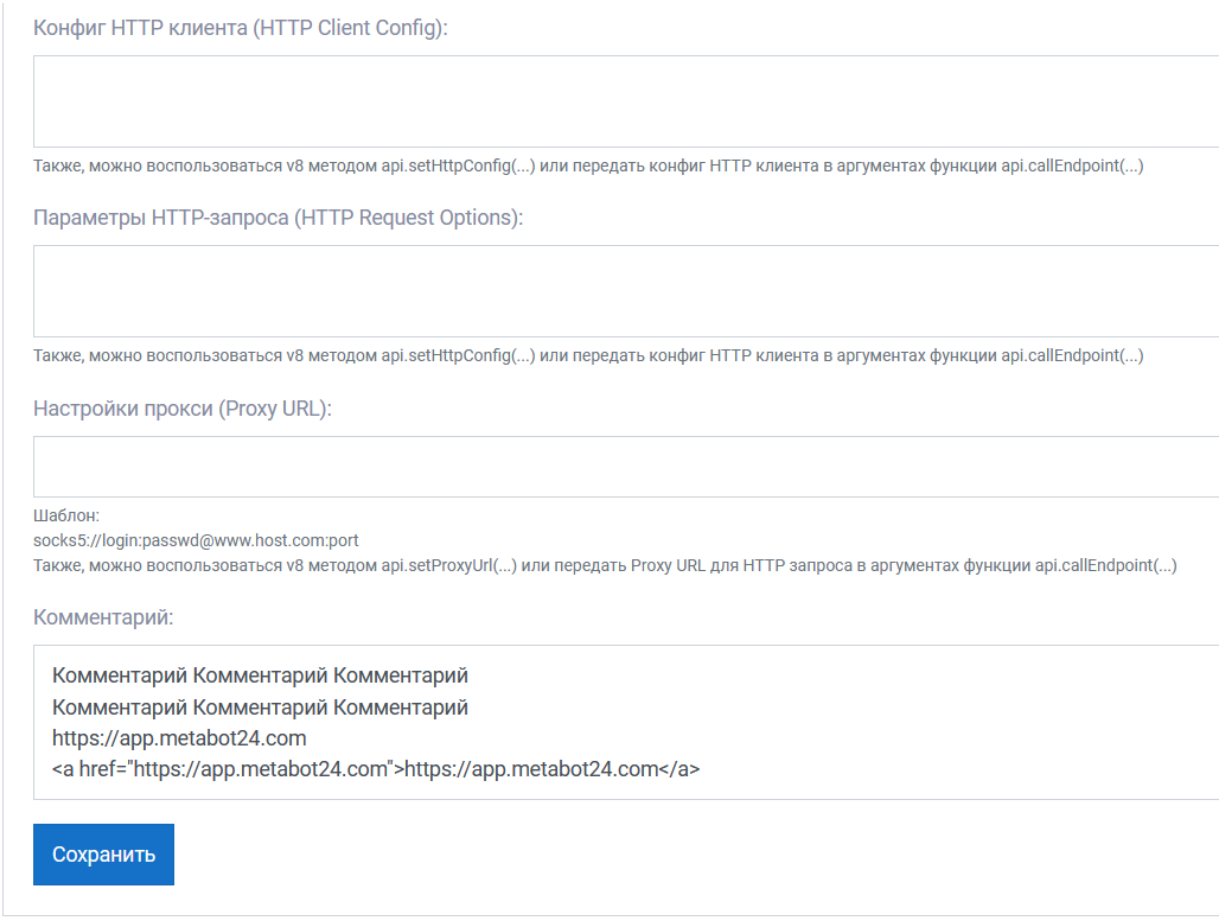
Формат HTTP запроса:

JSON

Заголовки HTTP запроса (HTTP Headers):

"{"token": "shmoken👍🙇😂🤷😟🙇😁😃"}"

Также, можно воспользоваться v8 методом api.setHeaders(...) или передать заголовки HTTP запроса в аргументах функции api.callEndpoint(...)



Вызов внешнего API из чат-бота

Метод для вызова внешнего API из JavaScript:

```
api.callEndpoint(
  string $externalApiEndpointCode,
  string|null $uri = null,
  string|null $method = null,
  string|null $format = null,
  array|object|string|null $body = null,
  array|object|string|null $headers = [],
  array|object|string|null $requestOptions = [],
  array|object|string|null $httpClientConfig = [],
  string|null $proxyUrl = null
): array|string|null
```

Пример №1 вызова внешнего API из JavaScript

```
response = api.callEndpoint("endpoint_1");
```

Параметры переданные в аргументах функции callEndpoint обладают наивысшим приоритетом, т.е. при подготовке параметров они идут перед теми, которые заданы другим способом, например, в настройках конечной точки или с помощью api сеттеров setHeaders / setRequestOptions / и т.д., или системными атрибутами бота.

При этом, все параметры суммируются в порядке приоритетов, за исключением тела запроса $body. Если аргумент $body указан в функции, то JS код из настроек конечной точки не учитывается, и в API в качестве тела запроса будет передано то, что указано в аргументе $body функции callEndpoint.

Таким образом, $body можно вычислить заранее (или хранить где-то в атрибутах или таблице) и передавать в аргументе функции или вычислять его в скрипте который вызывает api.callEndpoint, а не вычислять во вложенном JS самого эндпоинта. Если перекрываем $body — то JS из эндпоита вызываться не будет для вычисления тела запроса, т.к. тело перекрыто аргументом обладающим наивысшим приоритетом.

Пример №2 вызова внешнего API из JavaScript с перекрытием $body и других параметров

```
api.setHeaders({"Token": "my token"});
response = api.callEndpoint("endpoint_2", null, 'POST', 'TEXT', "MY BODY TEXT");
```

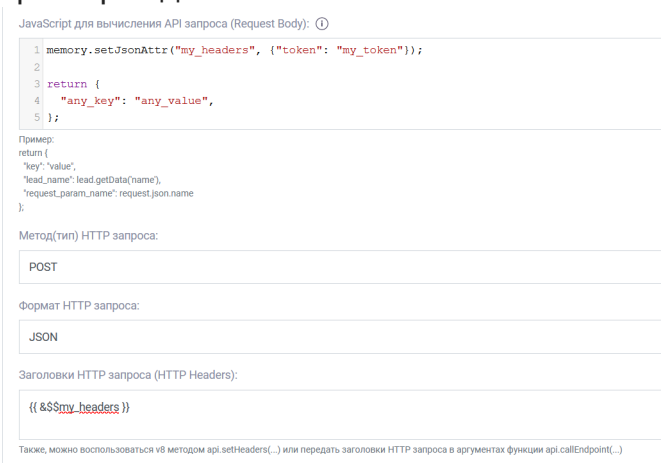
URL конечной точки заданный в настройках внешнего API (в интерфейсе) будет задействован, только если не указан $uri в аргументах функции (для второго по счету аргумента функции callEndpoint задан параметр null). Таким образом пустая строка в аргументе для URL перекрывает параметры указанные в настройках точки.

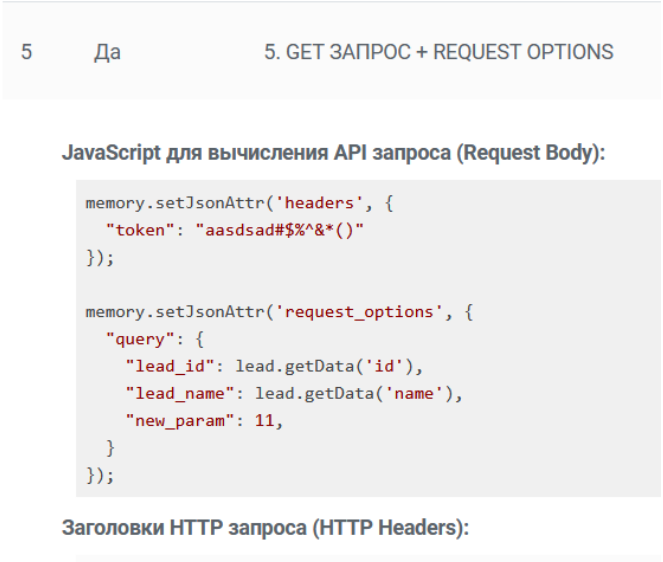
Структура конечной точки внешнего API

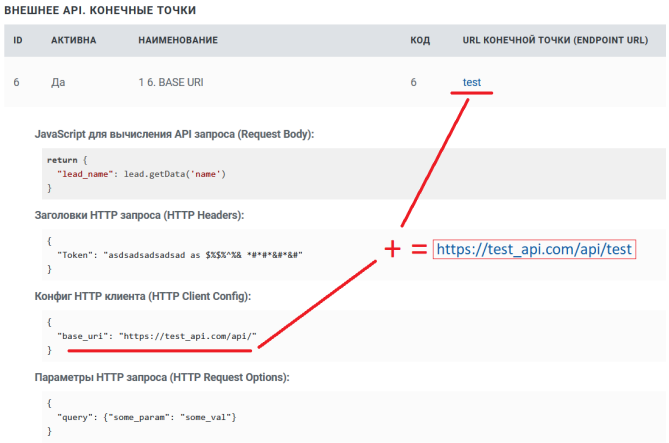
Описание полей внешнего эндпоинта представлена в таблице ниже.

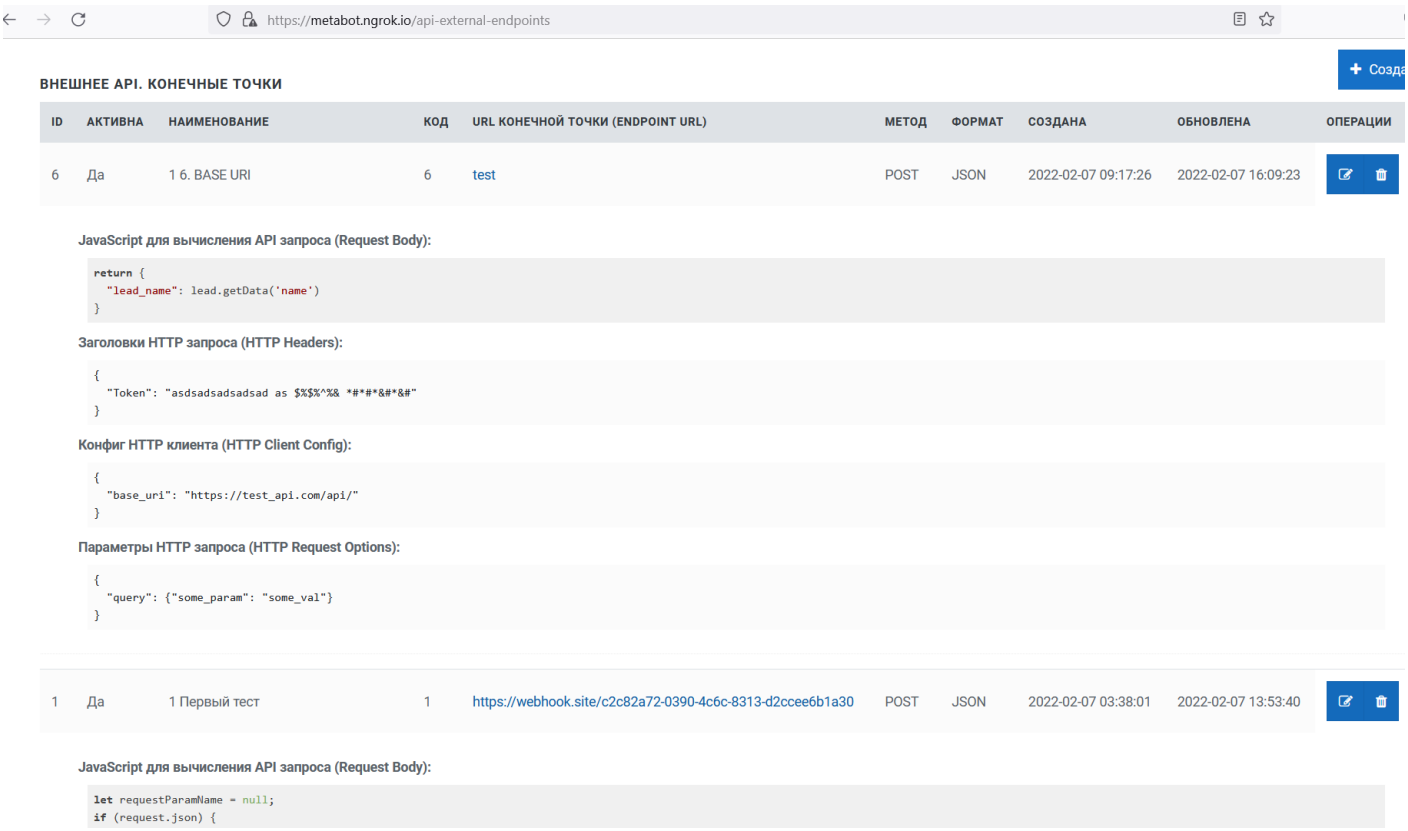
Поле	Описание
Флаг активности	Если убрать флаг, то вызов данного Endpoint не будет выполняться и метод api.callEndpoint(...) вернет null.
Код	Дополнительный уникальный идентификатор. Используется для указания объекта, например, в JavaScript функциях. В этом случае Код, используется как первый аргумент ($externalApiEndpointCode) метода api.callEndpoint(...). Для каждого бота создается свое пространство имен для внешнего API, это значит что Код должен быть уникален в пределах бота.

URL конечной точки (Endpoint URL)	Допустимо использование макропеременных.
JavaScript для вычисления API запроса (Request Body)	JavaScript для вычисления содержимого API запроса. Данный скрипт будет выполнен перед обращением к внешнему API, результат выполнения скрипта будет являться телом запроса к внешнему API. Это значит, что тело API запроса вычисляется на основе JavaScript. В JavaScript должен присутствовать ключевой оператор return для возврата значения из JS. Выражение которое указывается после return является телом API запроса (Request Body). Можно возвращать: • Строку (будет сконвертирована в json, если формат валидный); • Массив; • Объект (в том числе JSON-объект); • Любое другое выражение (если в конечной точке указан формат TEXT), для результата вычисления которого возможна автоматическая конвертация в строку (например число или булево). Пример (если формат запроса JSON): `return {` `"key": "value",` `"lead_name": lead.getData('name'),` `"request_param_name": request.json.name` `};` Другой пример (если формат запроса TEXT): `let a = false;` `return a;` В JS можно выполнять любой код, например обращаться к пользователям, или устанавливать переменные в memory, планировать задания бота (триггеры, рассылки, скрипты), а также вызывать внешнее API с помощью api.callEndpoint или другие методов объекта api: post/get/postJson/getJson/request.
Метод(тип) HTTP запроса	POST / GET / PATCH / и т.д.
Формат HTTP запроса	JSON / TEXT

Заголовки HTTP запроса (HTTP Headers)	Заголовки, которые будут переданы в HTTP запросе. В данном поле указывается JSON объект или массив. Допустимо использование макропеременных. Также, можно воспользоваться JS методом api.setHeaders(...) или передать заголовки HTTP запроса прямо в аргументах функции api.callEndpoint(...). Также доступна установка заголовков HTTP запроса с помощью атрибута бота **sys_api_headers**. ## Пример 1 для HTTP Headers  В JavaScript устанавливаем memory атрибут: ``` memory.setJsonAttr("my_headers", {"token": "my_token"}); ``` Далее используем этот атрибут в поле Headers с помощью макроподстановки: ``` {{ &$$my_headers }} ``` Обратите внимание, в данном примере выполняется полное заполнение JSON с помощью макроподстановки. ## Пример 2 для HTTP Headers JavaScript для вычисления API запроса (Request Body): ``` memory.setAttr('someValue', 'my value'); ``` Пример: return { "key": "value", "lead_name": lead.getData('name'), "request_param_name": request.json.name }; Метод(тип) HTTP запроса: GET Формат HTTP запроса: JSON Заголовки HTTP запроса (HTTP Headers): ``` { "some_param": "{{ &$someValue }}", "other_param": 11 } ```

Параметры HTTP запроса (HTTP Request Options)	Параметры HTTP запроса. В данном поле указывается JSON объект или массив. Допустимо использование макропеременных. Также, можно воспользоваться v8 методом api.setRequestOptions(...) или передать параметры HTTP запроса в аргументах функции api.callEndpoint(...) Также доступна установка параметров HTTP запроса помощью переменной бота **sys_api_request_options**. ## Пример 1 для HTTP Request Options  JavaScript: ``` memory.setAttr('leadName', lead.getData('name')); memory.setAttr('addQueryParams', ', "lead_id":' + lead.getData('id')); ``` HTTP Request Options: ``` { "query": { "lead_name": "{{ &$leadName }}", "new_param": 11 {{ &$addQueryParams }} } } ``` Пример 2 для HTTP Request Options 

Конфиг HTTP клиента (HTTP Client Config)	Параметры HTTP клиента. В данном поле указывается JSON объект или массив. Допустимо использование макропеременных. Также, можно воспользоваться v8 методом api.setHttpConfig(...) или передать конфиг HTTP клиента в аргументах функции api.callEndpoint(...) Также доступна установка конфига HTTP клиента с помощью переменной бота **sys_api_http_client_config**. Пример 1 для HTTP Client Config 
Настройки прокси (Proxy URL)	Используется для того, чтобы обращаться к определенному ресурсу с помощью прокси сервера. Шаблон для указания прокси: `socks5://login:passwd@www.host.com:port` Также, можно воспользоваться v8 методом api.setProxyUrl(...) или передать Proxy URL для HTTP запроса в аргументах функции api.callEndpoint(...). Также доступна установка прокси с помощью переменной бота **sys_api_proxy_url**.
Комментарий	Любой текст. Доступно указание любых HTML-тэгов.



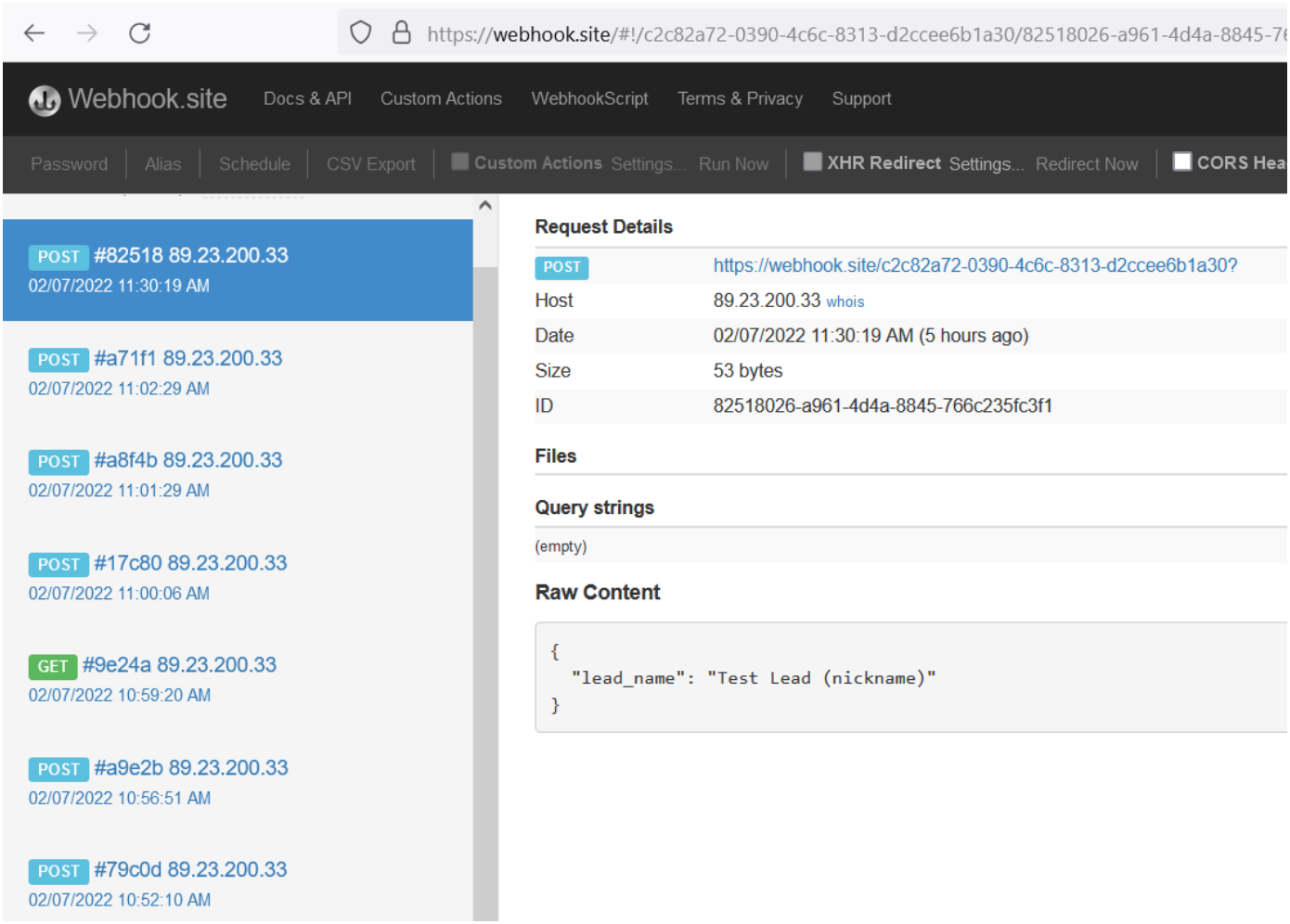
> Список всех конечных точек внешнего API сортируется по следующим полям: наименование, алиас, id. Поэтому вы можете управлять сортировкой записей изменяя наименование или алиас. В будущем будет добавлена произвольная сортировка.

Проверка работы внешнего API

Для тестирования внешнего API можно воспользоваться любым онлайн сервисом для отладки вебхуков (или развернуть аналогичный локально):

- https://stackoverflow.com/questions/5725430/http-test-server-accepting-get-post-requests
- https://webhook.site
- https://pipedream.com/requestbin
- http://ptsv2.com

Пример:



С API так же можно работать с помощью JS методов. Все методы и их описание вы можете посмотреть в **Справочнике по функциям JS**

Версия #6
Юрий Гарашко создал 2023-06-12 10:59:05 UTC
Ирина Петрова обновил 2025-11-11 07:01:29 UTC