Урок 3: Вывод фото в боте по REST API

Уважаемые разработчики! В этом уроке вы познакомитесь с основами разработки приложений с помощью платформы Metabot. Вместе мы создадим простой бот, который будет присылать в мессенджер по запросу изображение с сайта Unsplash.com.

Обращаясь к сторонним веб-сервисам с помощью REST API, мы создадим простой диалоговый сценарий, запрашивающий ключевое слово, по которому вы хотели бы найти изображение. Затем бот обращается к сервису API и возвращает найденное изображение. Далее мы добавим кнопку "Прислать следующее изображение". Таким образом, через бота вы сможете посмотреть поочерёдно все изображения, найденные на сервисе.

> Подробнее изучить работу бота вы можете с помощью нашего примера — бота в Telegram: **@Metabot101Metabot** или **Chat Widget**

В ходе урока вы узнаете:

- Как разрабатывается REST API интеграция с помощью вызова точек интеграции, называемых "endpoints";
- Как использовать JavaScript для обращения к сторонним ресурсам;
- Как работать с данными, которые хранятся в лиде (lead) и во временной памяти (memory);
- Как работать с режимом отладки кода, который позволит вам получать уведомления об ошибках выполнения вашего кода.

Приятного обучения!

Инструкция по разработке: Подготовка бота

Первым делом требуется создать бота, скрипт-приветствие в нем и добавить в скрипт сообщение с приветствием и меню.

> Как это сделать вы можете узнать из урока **"Hello Humans: ваше руководство по быстрому старту"**

Для нашего бота в начальном скрипте понадобится только команда отправки текста. Примерное содержимое команды может быть таким: "Приветствую! Это бот для просмотра фотографий на сервисе Unsplash.com".

Пункты меню можно создавать по мере необходимости, на начальном этапе меню можно отложить. Лучше сначала подключить необходимые каналы и маршруты к боту, запустить бота и включить режим отладки для нашего лида.

Подключение каналов

В данном боте будет создано два канала: Telegram и Metabot Widget

1. Создадим канал Telegram.

> Как это сделать вы можете узнать из инструкции **"Интеграция канала Telegram с платформой Metabot"**

2. Создадим канал Metabot Widget.

> Как это сделать вы можете узнать из инструкции **"Metabot Widget"**

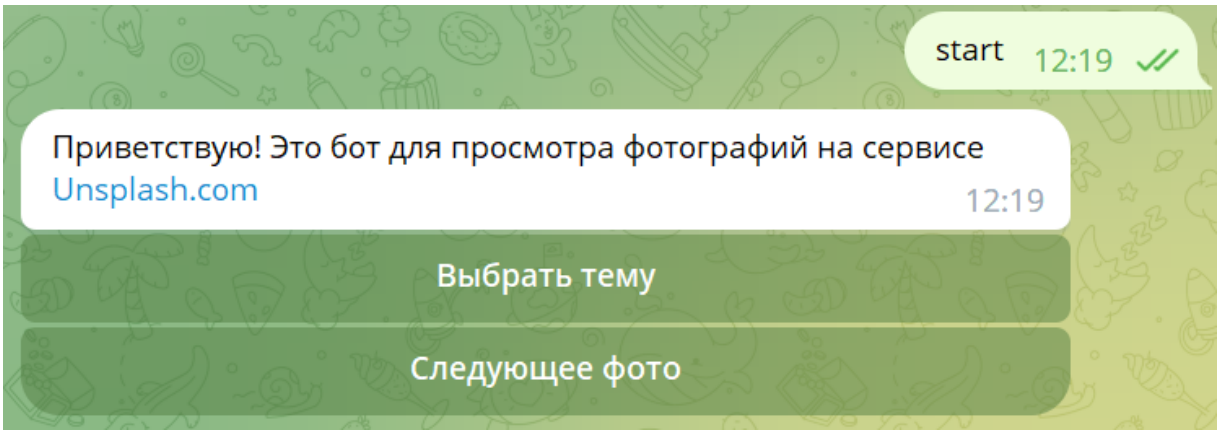
Подключение маршрутов

1. Создадим маршрут, который будет ссылаться на стартовый скрипт.

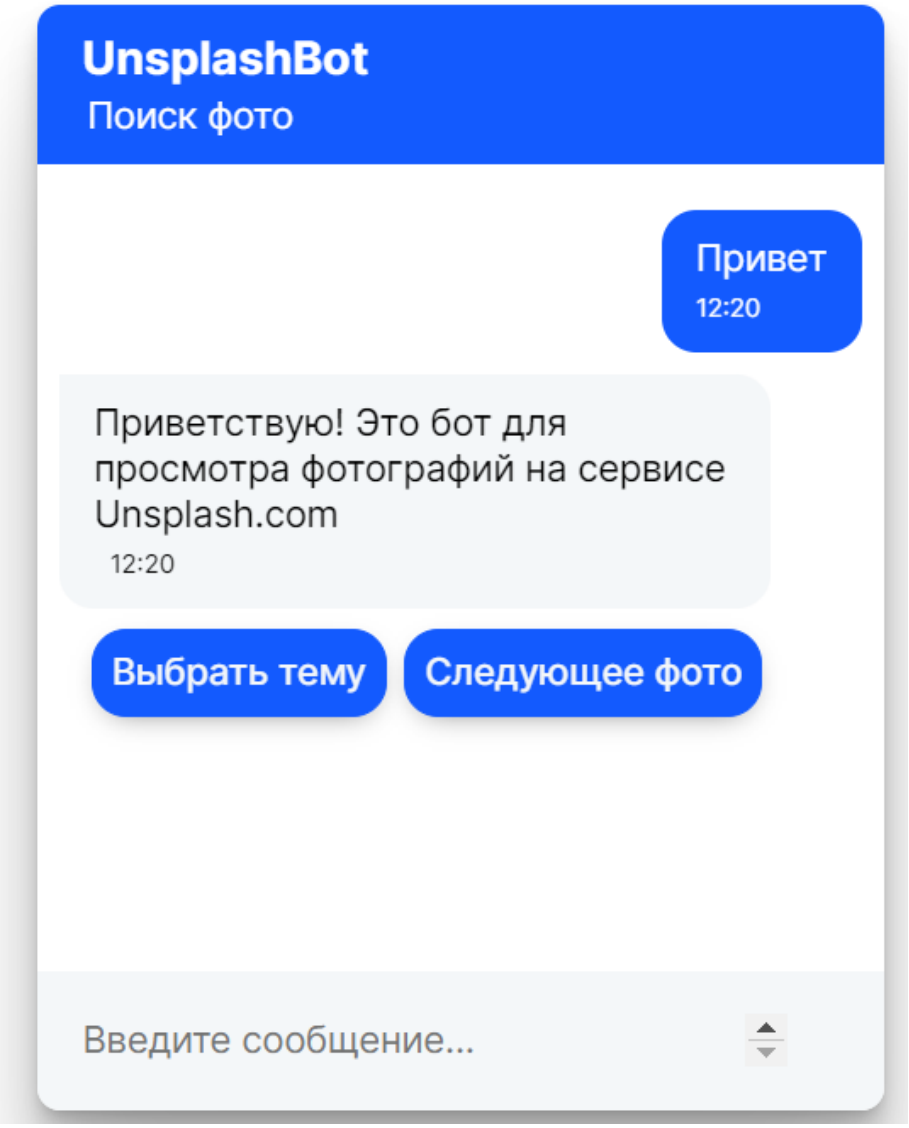
> Как это сделать вы можете узнать из инструкции **"Маршруты"**

Запускаем первую версию бота

1. Находим бот в Telegram и запускаем /start.



2. В справочнике каналов открываем канал «Metabot Widget» и в справочнике каналов последовательно нажимаем на кнопку «Предварительный просмотр».

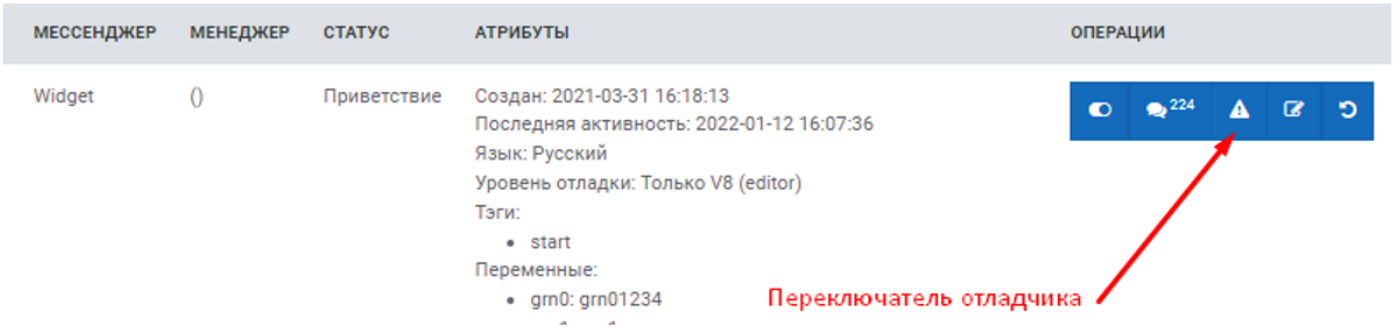


Если все работает можно включать режим отладки для обоих созданных лидов бота.

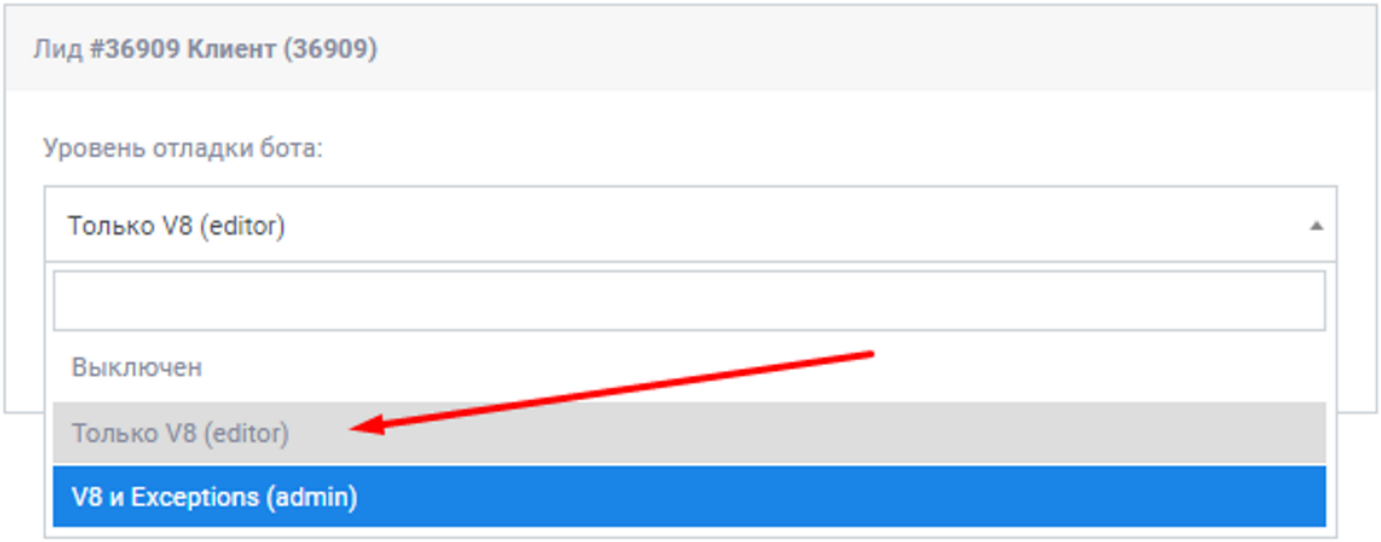
Включение режима отладки

Разработать бота для интеграции с внешними API без включения режима отладки практически невозможно, поэтому обязательно включите такой режим для себя. В режиме отладки различные ошибки отправляются в мессенджер или в Widget, если у лида, под которым вы ведете разработку, включена опция отладка бота.

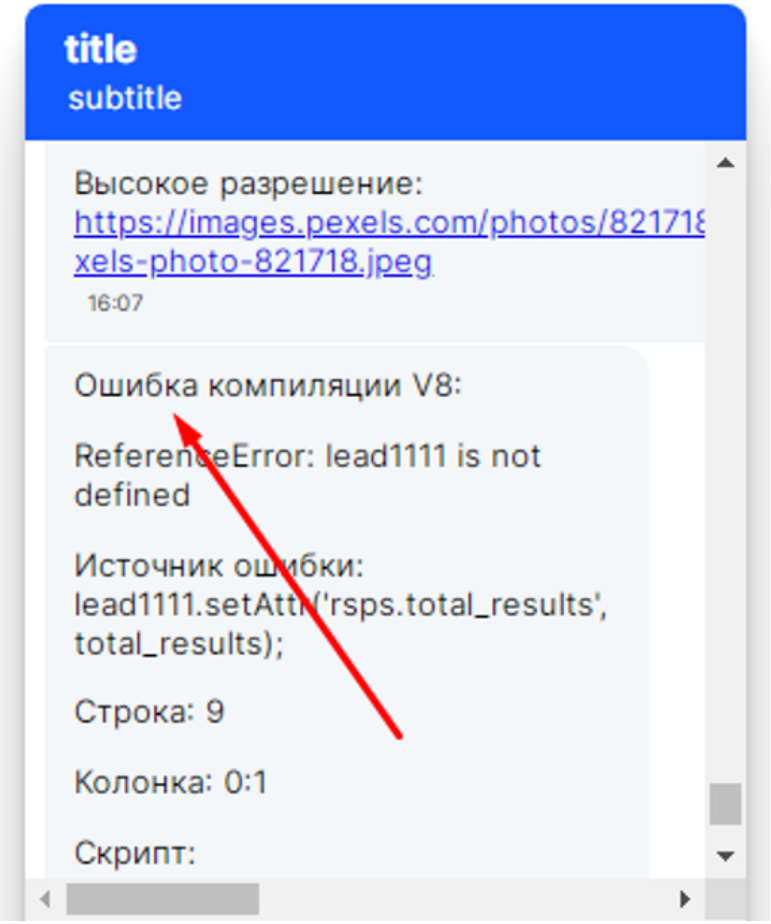
1. Для включения отладки следует зайти в меню "Лиды" и нажать на кнопку переключения отладчика.



2. Затем нужно выбрать уровень отладки. В данном случае выбирайте уровень "editor".



Если режим отладки включен, в мессенджер или виджет (в зависимости от канала, с которым связан лид), то в случае ошибок в JS-коде, будут выдаваться сообщения об ошибках. Если лид, под которым выполнялась отладка находится на продуктовом сервере, после завершения разработки отладчик следует отключить.



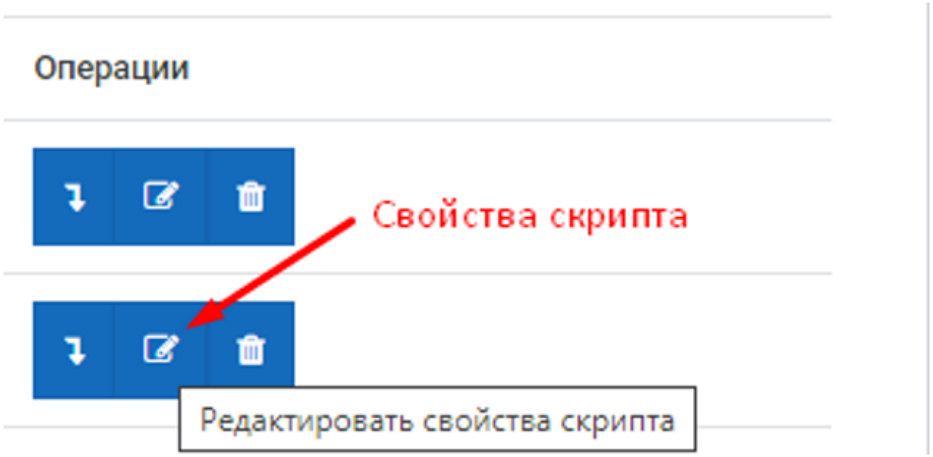
Расширяем скрипт с приветствием и меню

1. Добавляем пункты меню "Выбрать тему" и "Следующее фото" в стартовый скрипт.

> Как это сделать вы можете узнать из инструкции **"Создание меню"**

Если при создании пункта меню не указывать скрипт, то будет автоматически создан пустой скрипт с тем же именем, что и подпись кнопки.

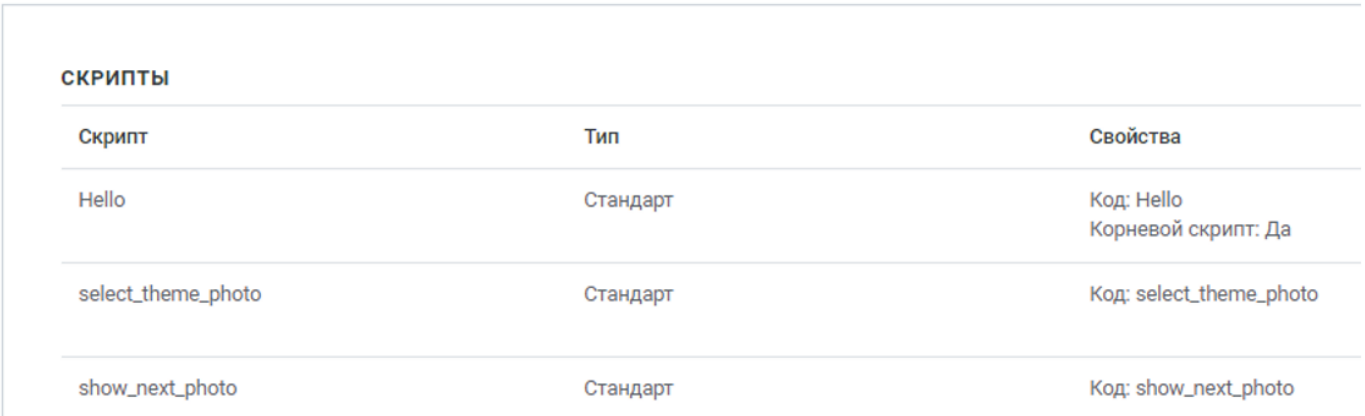
2. Следующим шагом отредактируем свойства автоматически созданных скриптов, а точнее их имя и JS-код. Это делается при помощи кнопки редактирования в меню скриптов.



- **Название** — это имя скрипта, которое может содержать в себе любые символы. Скриптам связанные с интеграциями, которые содержат большой объем JavaScript команд, рекомендуется присваивать англоязычные имена, так как это область работы JS программистов;
- **Код** — это дополнительный уникальный идентификатор. Используется для поиска скрипта, например, в JavaScript функциях, а также для поиска и запуска скрипта через NLP. Он всегда обозначается одним словом без пробелов.

3. Для скрипта выбора темы вводим в поля название и код значение **select_theme_photo**, а в скрипт переключения на следующее фото значение **show_next_photo**.

В итоге получаем следующие скрипты после изменения свойств:



Кнопки в боте появятся, но реакция на их нажатие отсутствует, так как созданные скрипты пока пустые. Их наполнением мы займёмся чуть позже.

Теперь мы готовы выполнить интеграцию с сервисом Unsplash.com.

Подключаемся к API Unsplash.com

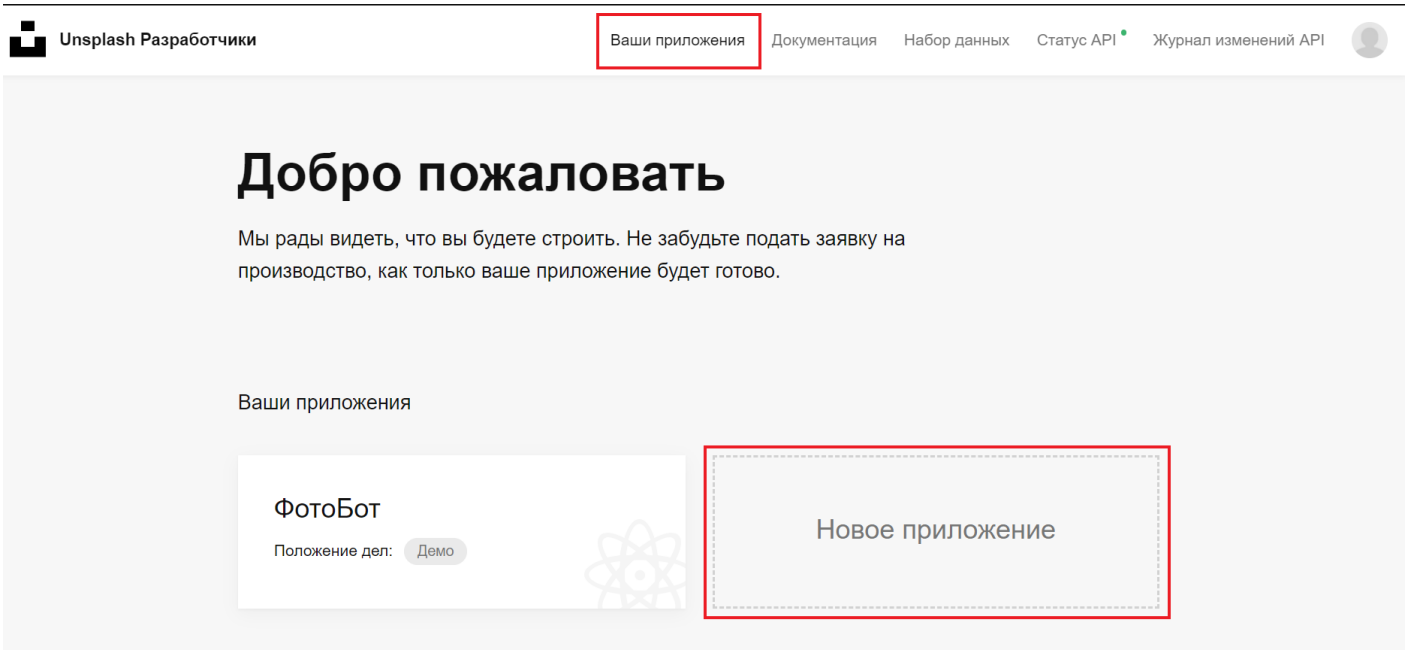
Для начала получим токен к API Unsplash.com.

1. Переходим к документации API Unsplash.com по ссылке:

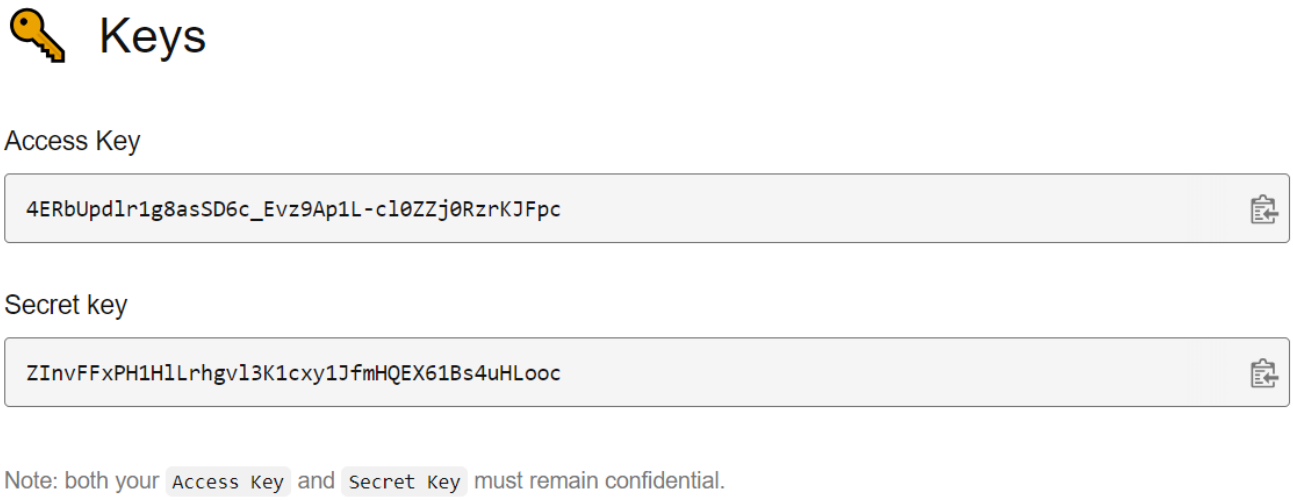
https://unsplash.com/documentation

2. Регистрируемся на платформе.

3. Для работы с API нам нужно получить специальный ключ-токен для доступа к методам API. Переходим в раздел "Ваши приложения" и нажимаем на окошко "Новое приложение".



4. Затем заполняем информацию о том для чего будет использован ключ, прокручиваем страницу ниже и как результат получаем Access Key. Он и будет использован для доступа к API.



5. В документации Unsplash указано два способа применения Access Key:

- При помощи заголовка авторизации:

```
Authorization: Client-ID YOUR_ACCESS_KEY
```

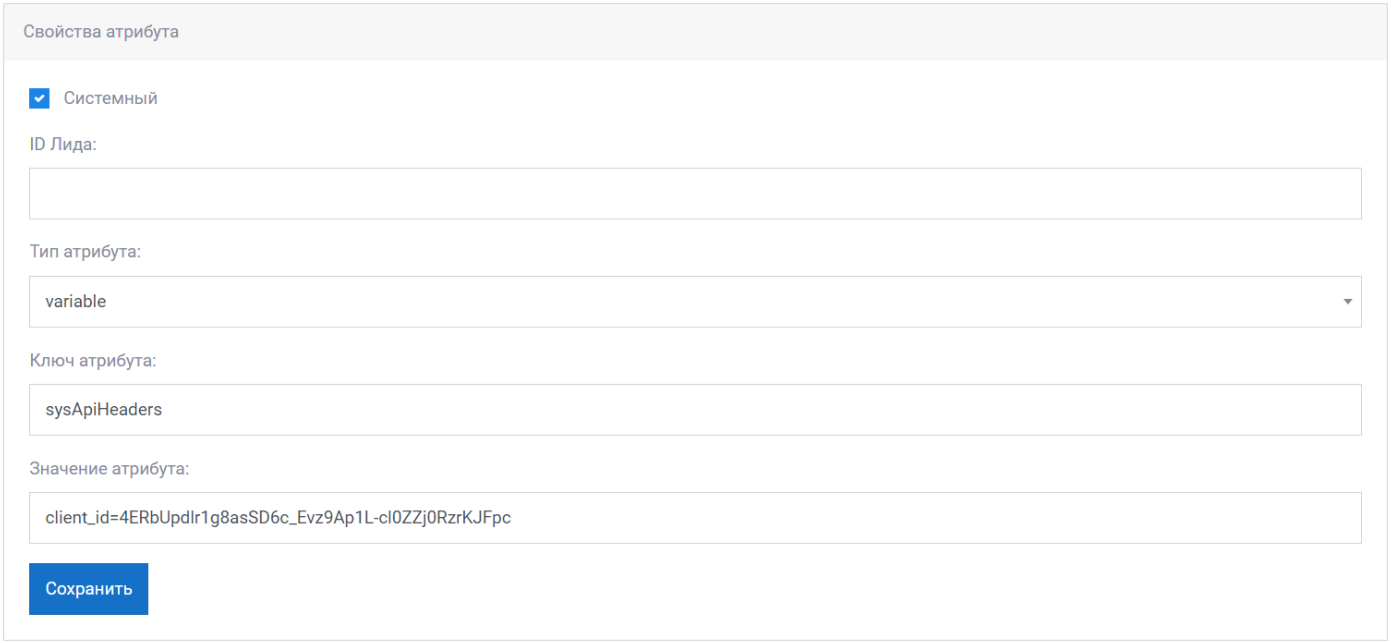
- Используя client_id параметр запроса:

```
https://api.unsplash.com/photos/?client_id=YOUR_ACCESS_KEY
```

Мы используем второй вариант и просто добавим Access Key как параметр URL. Для этого зайдем в «Настройки бота» меню «Атрибуты» создадим системный атрибут **sysApiHeaders** и присвоим ему значение:

```
client_id=4ERbUpdlr1g8asSD6c_Evz9Ap1L-cl0ZZj0RzrKJFpc.
```

> Подробнее про атрибуты вы можете узнать из инструкции **Атрибуты**



Исследуем endpoints API Unsplash.com

1. На главной странице документации в разделе Schema находим базовый URL для работы с фото.

Расположение

API доступен по адресу `https://api.unsplash.com/`. Ответы отправляются в формате JSON.

2. Аналогично предыдущему пункту, в «Настройки бота» меню «Атрибуты» заносим системный атрибут **sysApiHttpClientConfig** и присваиваем значение: **https://api.unsplash.com/**

В итоге имеем в нашем боте следующие атрибуты:

ТИП	НАЗВАНИЕ	ЗНАЧЕНИЕ
variable **[системный]**	sysApiHeaders	client_id=4ERbUpdlr1g8asSD6c_Evz9Ap1L-cl0ZZj0RzrKJFpc
variable **[системный]**	sysApiHttpClientConfig	https://api.unsplash.com/

Подробнее про работу с API на платформе Metabot вы можете узнать из инструкции **Конструктор API**

Заполняем скрипт select_theme_photo

Данный скрипт запрашивает тему фото, выполняет REST запрос к сервису и выводит первую выбранную фотографию.

1. Для реализации данного функционала создадим команду "Запросить значение". Зададим команде параметр "Имя переменной" = **photoTheme**. Именно в этом атрибуте будет храниться результат запроса.

> Подробнее о команде **"Запросить значение"** вы можете узнать из соответствующей инструкции.

2. Следующей командой будет "Выполнить JavaScript". Она будет содержать JS код для доступа к API.

```
// раздел 1: подготовка параметров REST запроса

var photoTheme = lead.getAttr('photoTheme');
var url = bot.getAttr("sysApiHttpClientConfig") + "search/photos?" +
bot.getAttr("sysApiHeaders") + "&query=" + photoTheme + "&per_page=1";

//-------------------------------------------------------

// раздел 2: подготовка вспомогательной переменной и начальная установка атрибутов лида

var total_results = 0;
lead.setAttr('url', url);
memory.setAttr('rsps.description', '');
lead.setAttr('rsps.totalResults', 0);
memory.setAttr('rsps.photographer', '');
memory.setAttr('rsps.full', '');
memory.setAttr('rsps.raw', '');
lead.setAttr('photoThemeIndex', 1);

//-------------------------------------------------------
```

В разделе **1** выбирается из атрибутов лида введенная ранее пользователем тема **photoTheme** и подготавливаются параметры REST запроса. Здесь же мы добавляем ранее созданные системные атрибуты в URL запроса. Так же добавляем в URL атрибуты запроса: тему фото и количество выводимых фото.

Ниже подробно расписаны все возможные атрибуты **запроса**:

Атрибут	Тип	Описание
query	string | required	Поисковой запрос. Например: Океан, Тигры, Груши и т.д.
orientation	string | optional	Желаемая ориентация фотографии. Текущие поддерживаемые ориентации: landscape, portrait or squarish.
order_by	string | optional	Как сортировать фотографии. По умолчанию: relevant. Допустимые значения latest и relevant.
color	string | optional	Желаемый цвет фото. Поддерживаемые цвета: black_and_white, black, white, yellow, orange, red, purple, magenta, green, teal, и blue.
collections	string | optional	Идентификаторы коллекций для сужения поиска. Если несколько, через запятую.
content_filter	string | optional	Ограничьте результаты по безопасности контента. По умолчанию: low. Допустимые значения low и high.
page	integer | optional	Номер страницы, которую вы запрашиваете. По умолчанию: 1.
per_page	integer | optional	Количество результатов, которые вы запрашиваете на странице. По умолчанию: 10.

В разделе **2** присваиваем начальные значения атрибутов:

- **url** — REST запрос;
- **total_results** – всего фото по заданной теме;
- **photo_theme_found** — признак, что запрос вернул фото;
- **next_page** — следующая страница;
- **photographer** – фотограф;
- **original** — ссылка на оригинальное фото;
- **medium** — ссылка на уменьшенную копию medium;
- **photo_theme_index** — текущиее фото в теме.

При этом одни переменные сохраняются в атрибутах лида, а другие во временном хранилище в памяти (в memory).

Примечание: Переменные, которые необходимы для выполнения алгоритма скрипта лучше сохранять в memory, чтобы не засорять атрибуты лида ненужными данными. В тоже время необходимо помнить, что при запуске нового скрипта по кнопке, по намерению NLP или по маршруту memory очищается, а lead очистить можно только принудительно командой. Поэтому переменные, которые используются в различных скриптах, сохранять в memory не нужно. В тоже время скрипт, который запускается из некоторого скрипта командой запустить скрипт имеет доступ к переменным в memory скрипта, который его запустил и всех остальных его запустивших.

Примечание: для отладки удобно временно сохранять переменную в lead вместо memory, так как переменные lead можно просматривать в справочнике атрибутов лидов.

Используются некоторые параметры из JSON объекта, который возвращает REST запрос.

Ниже приведена полная структура ответа:


```
{
  "total": 133,
  "total_pages": 7,
  "results": [
    {
      "id": "eOLpJytrbsQ",
      "created_at": "2014-11-18T14:35:36-05:00",
      "width": 4000,
      "height": 3000,
      "color": "#A7A2A1",
      "blur_hash": "LaLXMa9Fx[D%~q%MtQM|kDRjtRIU",
      "likes": 286,
      "liked_by_user": false,
      "description": "A man drinking a coffee.",
      "user": {
        "id": "Ul0QVz12Goo",
        "username": "ugmonk",
        "name": "Jeff Sheldon",
        "first_name": "Jeff",
        "last_name": "Sheldon",
        "instagram_username": "instantgrammer",
        "twitter_username": "ugmonk",
        "portfolio_url": "http://ugmonk.com/",
        "profile_image": {
```

```
          "small": "https://images.unsplash.com/profile-1441298803695-accd94000cac?ixlib=rb-0.3.5&q=80&fm=jpg&crop=faces&cs=tinysrgb&fit=crop&h=32&w=32&s=7cfe3b93750cb0c93e2f7caec08b5a41",
          "medium": "https://images.unsplash.com/profile-1441298803695-accd94000cac?ixlib=rb-0.3.5&q=80&fm=jpg&crop=faces&cs=tinysrgb&fit=crop&h=64&w=64&s=5a9dc749c43ce5bd60870b129a40902f",
          "large": "https://images.unsplash.com/profile-1441298803695-accd94000cac?ixlib=rb-0.3.5&q=80&fm=jpg&crop=faces&cs=tinysrgb&fit=crop&h=128&w=128&s=32085a077889586df88bfbe406692202"
        },
        "links": {
          "self": "https://api.unsplash.com/users/ugmonk",
          "html": "http://unsplash.com/@ugmonk",
          "photos": "https://api.unsplash.com/users/ugmonk/photos",
          "likes": "https://api.unsplash.com/users/ugmonk/likes"
        }
      },
      "current_user_collections": [],
      "urls": {
        "raw": "https://images.unsplash.com/photo-1416339306562-f3d12fefd36f",
        "full": "https://hd.unsplash.com/photo-1416339306562-f3d12fefd36f",
        "regular": "https://images.unsplash.com/photo-1416339306562-f3d12fefd36f?ixlib=rb-0.3.5&q=80&fm=jpg&crop=entropy&cs=tinysrgb&w=1080&fit=max&s=92f3e02f63678acc8416d044e189f515",
        "small": "https://images.unsplash.com/photo-1416339306562-f3d12fefd36f?ixlib=rb-0.3.5&q=80&fm=jpg&crop=entropy&cs=tinysrgb&w=400&fit=max&s=263af33585f9d32af39d165b000845eb",
        "thumb": "https://images.unsplash.com/photo-1416339306562-f3d12fefd36f?ixlib=rb-0.3.5&q=80&fm=jpg&crop=entropy&cs=tinysrgb&w=200&fit=max&s=8aae34cf35df31a592f0bef16e6342ef"
      },
      "links": {
        "self": "https://api.unsplash.com/photos/eOLpJytrbsQ",
        "html": "http://unsplash.com/photos/eOLpJytrbsQ",
        "download": "http://unsplash.com/photos/eOLpJytrbsQ/download"
      }
    },
    // больше фото ...
  ]
}
```


3. Продолжаем разрабатывать скрипт в той же команде.

```
// раздел 3: выполнение REST запроса

memory.setAttr('photoThemeFound', 0);
let jsonResponse = api.getJson(url);
if (jsonResponse) {

//-----------------------------------------------

// раздел 4: сохранение результата запроса в атрибутах лида

  var rsps = jsonResponse;
  totalResults = rsps['total'] * 1;
  if (totalResults > 0) {
    memory.setAttr('photoThemeFound', 1);
    lead.setAttr('rsps.totalResults', totalResults);
    lead.setAttr('rsps.photographer', rsps['results'][0]['user']['username']);
    lead.setAttr('rsps.description', rsps['results'][0]['description']);
    lead.setAttr('rsps.full', rsps['results'][0]['urls'][ 'full' ]);
    lead.setAttr('rsps.raw', rsps['results'][0]['urls'][ 'raw' ]);
  }

//-----------------------------------------------

// раздел 5

  memory.setJsonAttr("lastHttpResponse", api.getLastResponseContent());
  memory.setAttr("lastHttpResponseCode", api.getLastResponseCode());
}

//-----------------------------------------------
```

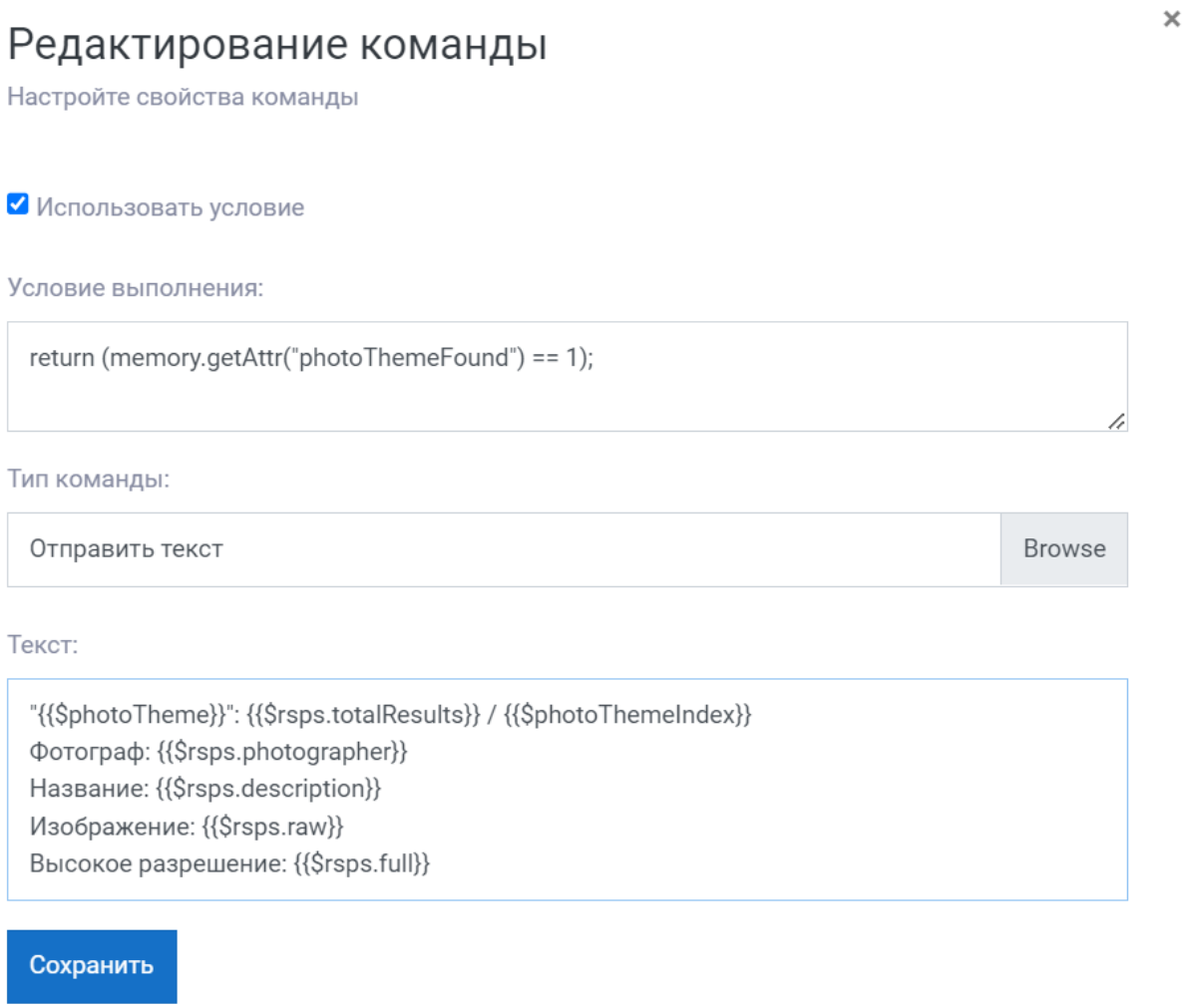
В разделе **3** выполняется REST запрос к сервису и проверяется что запрос выполнился.

В разделе **4** проверяется что в данной теме имеются фото и переносятся необходимые параметры REST запроса в атрибуты лида.

В разделе **5** для удобства отладки запоминаем в атрибуте лида полный ответ и код ответа сервиса.

После завершения отладки запоминание полного ответа в атрибуте лучше закомментировать или удалить, так как он занимает довольно много места и будет сохранен для каждого лида обратившегося в бота.

4. Далее выводим ответ в канал, используя данные, сохраненные в атрибутах лида. Для этого используем две команды вывода текста с условием. Одна срабатывает когда атрибут **photoThemeFound** равен 1, вторая, наоборот, когда не равен 1.



Так выглядят данные команды после создания:

Отправить текст	`return (memory.getAttr("photoThemeFound") == 1);`	"{{$photoTheme}}": {{$rsps.totalResults}} / {{$photoThemeIndex}} Фотограф: {{$rsps.photographer}} Название: {{$rsps.description}} Изображение: {{$rsps.raw}} Высокое разрешение: {{$rsps.full}}
Отправить текст	`return !(memory.getAttr("photoThemeFound") == 1);`	Фото по теме "{{$photoTheme}}" не найдены.

Вы можете скопировать данные команд из таблицы:

Условие выполнения	Текст команды
return (memory.getAttr("photoThemeFound") == 1);	"{{$photoTheme}}": {{$rsps.totalResults}} / {{$photoThemeIndex}} Фотограф: {{$rsps.photographer}} Название: {{$rsps.description}} Изображение: {{$rsps.raw}} Высокое разрешение: {{$rsps.full}}
return !(memory.getAttr("photoThemeFound") == 1);	Фото по теме "{{$photoTheme}}" не найдены.

Заполняем скрипт show_next_photo

1. Входим в редактор команд скрипта и набираем скрипт, который аналогично предыдущему состоит из трех частей:

- **1** извлечение из лида параметров запроса;
- **2** выполнение запроса и заполнение атрибутов в memory и lead;
- **3** сохранение переменных для отладки.

Команда запроса темы не нужна, так как она определена в скрипте выбора темы и сохранена в атрибуте **photoTheme**. При этом ранее сохраненный индекс фото увеличиваем на единицу, чтобы при запросе не выводилось одно и то же фото.

```
// раздел 1: подготовка параметров REST запроса

var photoTheme = lead.getAttr('photoTheme');
var photoThemeIndex = lead.getAttr('photoThemeIndex') * 1;
photoThemeIndex++;
var url = bot.getAttr("sysApiHttpClientConfig") + "search/photos?" +
bot.getAttr("sysApiHeaders")
+ "&query=" + photoTheme + "&page=" + photoThemeIndex + "&per_page=1";

//-------------------------------------------------------------

// раздел 2

lead.setAttr('url', url);
memory.setAttr('photoThemeFound', 0);
jsonResponse = api.getJson(url);
if (jsonResponse) {
  var rsps = jsonResponse;
  totalResults = rsps['total'] * 1;
```

```
  if (totalResults > 0 && photoThemeIndex <= totalResults) {
    memory.setAttr('photoThemeFound', 1);
    lead.setAttr('photoTheme', photoTheme);
    lead.setAttr('photoThemeIndex', photoThemeIndex);
    lead.setAttr('rsps.totalResults', totalResults);
    lead.setAttr('rsps.photographer', rsps['results'][0]['user']['username']);
    lead.setAttr('rsps.description', rsps['results'][0]['description']);
    lead.setAttr('rsps.full', rsps['results'][0]['urls'][ 'full' ]);
    lead.setAttr('rsps.raw', rsps['results'][0]['urls'][ 'raw' ]);
  }

//-------------------------------------------------------

// раздел 3

  memory.setJsonAttr("lastHttpResponse", api.getLastResponseContent());
  memory.setAttr("lastHttpResponseCode", api.getLastResponseCode());
}

//-------------------------------------------------------
```

2. Аналогично создаем две команды вывода данных с условиями для отправки в канал необходимых атрибутов лида. Одна срабатывает в случае успешной выборки фото, другая в противоположном случае.

Вы можете скопировать данные команд из таблицы:

Условие выполнения	Текст команды
return (memory.getAttr("photoThemeFound") == 1);	"{{$photoTheme}}": {{$rsps.totalResults}} / {{$photoThemeIndex}} Фотограф: {{$rsps.photographer}} Название: {{$rsps.description}} Изображение: {{$rsps.raw}} Высокое разрешение: {{$rsps.full}}
return !(memory.getAttr("photoThemeFound") == 1);	Фото не найдено.

Бот готов! Теперь можно переходить к его тестированию.

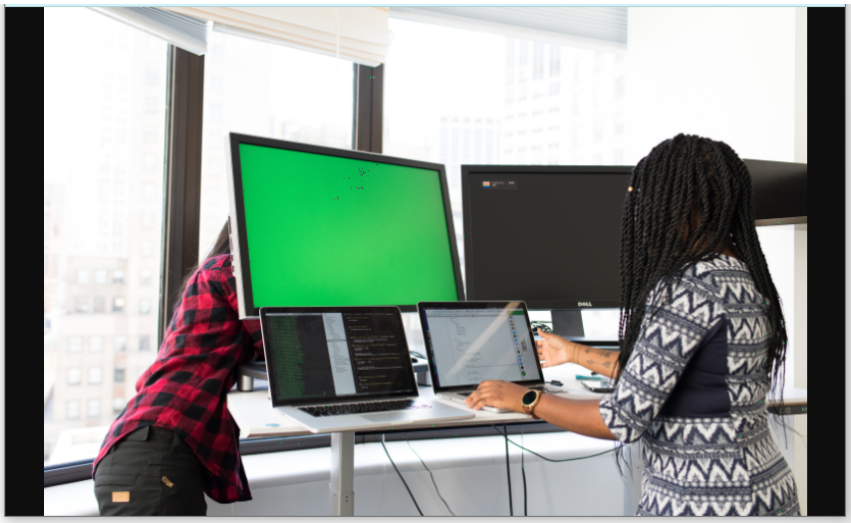
Запускаем бот!

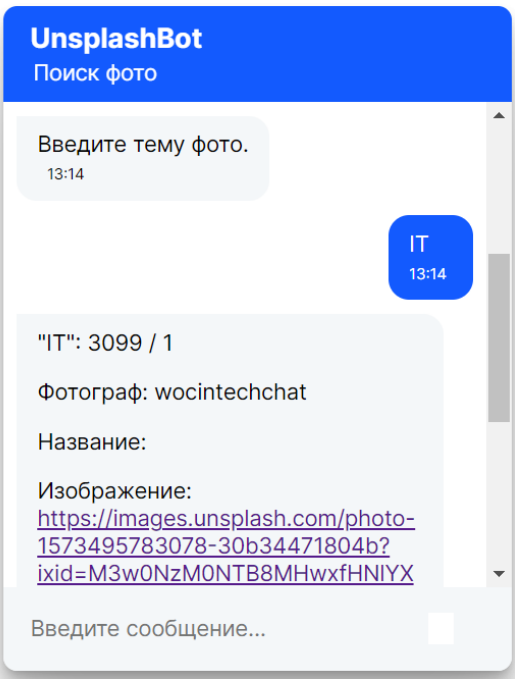
1. Запускаем бот в канале Telegram.

Запрашиваем у бота тему фото, которое мы хотим получить. Если фото по теме найдены, то бот выводит количество найденных фото по теме и индекс выведенного фото. Ниже указывается username фотографа, название фото и ссылки на него в обычном и HD формате.

2. Запускаем бот в канале виджета Metabot.

Бот в виджете работает по тому же принципу: запрос темы и вывод ссылок на фото.





Поздравляем с успешным созданием бота!

> Рекомендуем так же ознакомиться с остальными нашими уроками по созданию ботов, например **Автоматизируем службу поддержки**

Версия #23
Ирина Петрова создал 7 July 2023 09:30:35
Ирина Петрова обновил 20 March 2024 08:57:41