

# Урок 8: Диалоговый интерфейс к ИТ системам

В этом уроке вы познакомитесь с теорией и практическими примерами того как создавать современные диалоговые интерфейсы к системам предприятия.

Чат-бот выступает в качестве дополнительного сервиса, расширяющего основную систему, будто то ERP, Ecomm, Helpdesk или другая информационная система (далее просто — система), возможностью вести интерактивные диалоговые коммуникации с пользователями системы в мессенджерах, социальных сетях и онлайн-чате для сайта.

Что дают интеграции для бизнеса:

- Возможность реализовать бесшовный клиентский опыт по всему пути клиента, например, регистрация на сайте, прием тикетов в поддержку и др;
- Возможность отправлять сервисные и маркетинговые уведомления прямо из систем в мессенджеры на смартфоны клиентам при наступлении тех или иных событий в основных системах. Например, оживление брошенной корзины;
- Сможете установить прямой канал связи с каждым пользователем ваших веб-сайтов, порталов, веб-сервисов с поддержкой интерактивных диалоговых коммуникаций;
- Возможность создать виртуальную личность бренда или виртуального помощника, взаимодействующего с бэк-энд системами компании и публичными Интернет-сервисами;
- Альтернативу мобильному приложению в мессенджере. При желании сможете добавлять много функционала и очень быстро. Сможете оперативно тестировать множество гипотез.

Ниже описаны основные принципы и важные нюансы, на которых строится интеграция основных системы и чат-бота и обмен данными между ними.

## Коммуникации в чат-боте

Под коммуникацией подразумевается диалоговый сценарий или скрипт, заранее спроектированный и запускаемый на коммуникационной low-code платформе Metabot, на которой разрабатывается и выполняется чат-бот. Коммуникации могут быть абсолютно любыми и варьироваться от самых простых, таких как отправка текстового сообщения/уведомления, до более сложных и растянутых во времени, которые могут инициировать дополнительные вспомогательные коммуникации, например, напоминания,

подтягивать данные с веб-сайта по API и прочее.

Чат-бот — это одновременно как прямой канал связи с пользователями для отправки уведомлений, так и интерактивный интерфейс к системе в виде диалогового приложения, через которое пользователь взаимодействует с системой.

# Карты коммуникации

Для описаний коммуникационных процессов, которые происходят с участием и/или между пользователями систем используется инструмент под названием **карта коммуникаций** (communication map). Для построения карты коммуникаций вначале согласовываются сущности, с которыми могут происходить изменения: пользователи, заказы, обращения и другие.

Для каждой сущности (или процесса) составляется отдельная карта коммуникаций, в которой описываются события, которые могут произойти с сущностью (например, в виде этапов процесса). Затем, для каждого события согласовываются коммуникации, которые необходимо запустить в чат-боте во время наступления этих событий. Для каждой коммуникации описывается список кого нужно уведомить, какие действия дальше ожидаются от них и прочие детали.

Пример карты коммуникаций для личного аккаунта пользователя:

№	Этапы процесса с описанием / Событие	Нужна ли коммуникация в чат-боте и какая / Коммуникация	Детальное описание коммуникации
I. Активация / деактивация чат-бота			
1	Пользователь подключил чат-бот	Сообщаем чат-боту, что пользователь снова подключил чат-бот.  Коммуникация актуальна только для тех, кто использовал чат-бот ранее.	Чат-бот радостно здоровается с пользователем и интересуется как может помочь.  Также, чат-бот может предложить прислать свежие новости.  Также, чат-бот может проверить аккаунт пользователя и запланировать различные задачи, которые мы хотим, чтобы пользователь сделал, например, оставил отзывы о завершенных заявках.

2	Пользователь отключил чат-бот	Пользователь отключил чат-бот	Де-авторизуем пользователя в чат-бот. Сообщаем пользователю, что отключение прошло успешно, и чтобы продолжить работу нужно будет авторизоваться заново.
3	Пользователь временно остановил отправку сообщений в чат-бот	Пользователь временно остановил отправку сообщений в чат-бот	Чат-бот подтверждает пользователю, что он больше не побеспокоит до такого то времени, за исключением критически важных сообщений от компании. Сообщает, что пользователь может реактивировать чат-бот раньше когда пожелает.
4	Пользователь восстановил отправку сообщений в чат-бот	Сообщаем чат-боту, что пользователь восстановил чат-бот.	Чат-бот радостно приветствует пользователя и интересуется как он может помочь. Также, чат-бот может предложить прислать обзор активностей и новостей, произошедших с момента паузы.  Также, чат-бот может проверить аккаунт пользователя и запланировать различные задачи, которые мы хотим, чтобы пользователь сделал, например, оставил отзывы о завершенных заявках.

## II. События редактирования пользователя

5	Пользователь или Администратор изменил свои данные в своем личном кабинете	Сообщаем чат-боту, что пользователь отредактирован.	Чат-бот получает информацию об измененных полях. Вместе со списком изменений чат-боту сообщается ID события, идентифицирующее точку, где произошло событие изменения. Поскольку мест где могут меняться данные о пользователе может быть несколько (а также лиц изменяющих данные тоже несколько: администратор , обычный пользователь), то на основе уникального ID события, передаваемого в запросе с изменениями пользователя, на стороне чат-бота появляется дополнительная гибкость в планировании коммуникаций, для тех или иных случаев.
6	Администратор блокирует пользователя	Сообщаем чат-боту, что пользователь был заблокирован.	Чат-бот блокирует пользователя. Пользователю сообщаем, что аккаунт был заблокирован, сообщаем причину и перенаправляем в меню для неавторизованных пользователей
7	Администратор отменяет блокировку пользователя	Сообщаем чат-боту, что пользователь был разблокирован.	Чат-бот разблокирует пользователя. Пользователю сообщаем, что аккаунт был разблокирован и перенаправляем в меню, соответствующее его роли.

8	Администратор удаляет учетную запись пользователя	Сообщаем чат-боту, что пользователь был удален.	Чат-бот де-авторизует пользователя. Пользователю ничего не сообщаем. При следующей попытке взаимодействовать с чат-ботом, чат-бот уведомит пользователя, что учетная запись удалена и перенаправить в меню для неавторизованных пользователей.
---	---	---	--

Для каждой сущности или процесса для которого необходимо реализовать интерактивные коммуникации в чат-боте, составляется аналогичная карта, которая помогает спланировать коммуникации вместе с клиентом и разработчиками основной системы. Это рабочий документ, который помогает вам договориться и спланировать работы.

Приведем пример еще одной карты коммуникаций, в этот раз для реального проекта — чат-бота для Семейного сайта <https://fmlst.ru>, который выступил мобильным интерфейсом с конструктору семейных веб-сайтов. При желании, можете попробовать чат-бот по адресу: <https://t.me/MoyaSemiyaBot>.

Чат-бот позволил организовать создание веб-сайта прямо из мессенджера, а также является семейным коммуникатором — присылает уведомления о новых публикациях на семейном сайте всем членам семьи, которые авторизовались в чат-бота, позволяет разместить комментарий прямо под публикацией на сайте, рассылает комментарий всем членам семьи.

Примеры из карты коммуникаций для этого проекта выглядят следующим образом:

№	Этапы процесса с описанием / Событие	Нужна ли коммуникация в чат-боте и какая / Коммуникация	Детальное описание коммуникации
II. Уведомления членов семьи			

1	На семейном сайте размещена новая публикация	Нужно уведомить всех членов семьи, у которых подключен и активирован чат-бот.	Нужно прислать размещенную новость и картинку, если прикрепili изображение.
2	Член семьи прокомментировал публикацию	Нужно уведомить все членов семьи, у которых подключен и активирован чат-бот.	Нужно прислать сам комментарий и дать возможность на него ответить через Telegram.

## Точки интеграции

Каждой коммуникации соответствует **точка интеграции** (integration endpoint).

Коммуникации запускаются с помощью вызова нужной точки интеграции, название которой указывается в URL, а в теле запроса в **script\_request\_params** указываются параметры коммуникации, например, список ID всех пользователей, которых нужно уведомить.

Для обеспечения интерактивной коммуникации требуется реализация двусторонней интеграции системы и чат-бота. Команде разработки системы нужен API для вызова массовой и индивидуальной коммуникации с пользователями, а команде разработки чат-бота нужен API для чтения и изменения данных на портале в ходе диалога с пользователем.

Взаимодействия портала и чат-бота происходят по протоколу REST API, а точки взаимодействия, в которых происходит обращение одной системы к ресурсам другой называются точками интеграции (или endpoints).

У чат-бота есть внутренние точки интеграции (внутренние эндпоинты) для обращения к чат-боту из внешних систем, а также есть внешние точки интеграции (внешние эндпоинты) для обращения к внешним системам из чат-бота. В эндпоинте содержится программный код на JavaScript, в котором пишется код обращения к API, синхронизации данных, запуска коммуникации и прочее. Подробнее смотрите в разделе [Конструктор API](#).

Ниже показан пример описания точек интеграции для личного аккаунта пользователей:

№	Точка интеграции / Коммуникация	Параметры	Описание
---	------------------------------------	-----------	----------

1	Изменение данных пользователя
---	-------------------------------

Endpoint Alias: users/update

URL и Header:

```
Метод: POST
URL: {Server
URL}/bots/{botId}/call/users/update
Accept: application/json
Content-Type: application/json
```

Тело запроса:

```
{
  "script_request_params": {
    "userId": "<ID пользователя в основной
системе/required>",
    "lastname": "<Фамилия>",
    "firstname": "<Имя>",
    "patronymic": "<Отчество>",
    "role": "<Роль/required>",
    "active": "<Статус активности/bool>",
    "phone": "<Мобильный телефон>",
    "email": "<Email адрес>",
    "location": {
      "country": "<Страна местоположения>",
      "region": "<Регион>",
      "city": "<Город>"
    }
  }
}
```

Примеры ответов:

В случае успеха:

```
{
  "success": true
}
```

Пользователь не найден:

```
{
  "success": false,
  "message": "Пользователь не найден"
}
```

Запрос для информирования чат-бота об изменении сведений о пользователе. Вызывается каждый раз, когда на в основной системе редактируются данными самим пользователем, сотрудником или администратором.



2	Отправка флага о том, что пользователь хочет активировать чат-бот		
	<div>Endpoint Alias: users/connect-bot</div> <div>URL и Header:</div> <div><div>Метод: POST</div><div>URL: {Server</div><div>URL}/bots/{botId}/call/users/connect-bot</div><div>Accept: application/json</div><div>Content-Type: application/json</div></div> <div>Тело запроса:</div> <div><div>{</div><div>  "script_request_params": {</div><div>    "userId": &lt;ID пользователя в основной</div><div>  системе/required&gt;</div><div>  }</div><div>}</div></div> <div>Примеры ответов:</div> <div>В случае успеха:</div> <div><div>{</div><div>  "success": true</div><div>}</div></div> <div>Пользователь не найден:</div> <div><div>{</div><div>  "success": false,</div><div>  "message": "Пользователь не найден"</div><div>}</div></div>	<div>Передаем ID</div> <div>пользователя, которому</div> <div>активируем чат-бота.</div>	

Приведем еще несколько примеров:

№	Точка интеграции / Коммуникация	Параметры	Описание
1	Благодарим за покупку		

	<p>Endpoint Alias: transaction/thank-you</p> <p>URL и Header:</p> <div><p>Метод: POST</p><p>URL: {Server</p><p>URL}/bots/{botId}/call/transaction/thank-you</p><p>Accept: application/json</p><p>Content-Type: application/json</p></div> <p>Тело запроса:</p> <div><pre>{   "script_request_params": {     "userId": &lt;ID пользователя в основной системе/required&gt;,     "orderId": &lt;ID заказа/required&gt;   } }</pre></div> <p>Примеры ответов:</p> <p>В случае успеха:</p> <div><pre>{   "success": true }</pre></div> <p>Пользователь не найден:</p> <div><pre>{   "success": false,   "message": "Пользователь не найден" }</pre></div>	<p>Благодарим пользователя за покупку в нашем магазине.</p>
2	Уведомляем о новой публикации на сайте	

Endpoint Alias: post/new

URL и Header:

```
Метод: POST
URL: {Server
URL}/bots/{botId}/call/post/new
Accept: application/json
Content-Type: application/json
```

Тело запроса:

```
{
  "script_request_params": {
    "postId": <ID публикации/required>,
    "authorId": <ID автора
публикации/required>,
    "userIds": [
      <ID первого пользователя/required>,
      <ID второго пользователя>,
      ..
      <ID N-го пользователя>
    ]
  }
}
```

Уведомляем о новой публикации пользователей в массиве userIds

Примеры ответов:

В случае успеха:

```
{
  "success": true
}
```

Пользователь не найден:

```
{
  "success": false,
  "message": "Пост не найден"
}
```



Endpoint Alias: comment/new

URL и Header:

```
Метод: POST
URL: {Server
URL}/bots/{botId}/call/comment/new
Accept: application/json
Content-Type: application/json
```

Тело запроса:

```
{
  "script_request_params": {
    "postId": <ID публикации/required>,
    "authorId": <ID автора
публикации/required>,
    "commentAuthorId": <ID автора
комментария>,
    "comment": "<комментарий>",
    "userIds": [
      <ID первого пользователя/required>,
      <ID второго пользователя>,
      ..
      <ID N-го пользователя>
    ]
  }
}
```

Присылаем новый комментарий пользователям userIds.

Примеры ответов:

В случае успеха:

```
{
  "success": true
}
```

Пользователь не найден:

```
{
  "success": false,
  "message": "Пост не найден"
}
```

В таблицах выше описаны интеграции основной системы с вашим чат-ботом. По сути, это техническое задание для двух сторон: для разработчиков чат-бота (на прием запроса и запуск коммуникации) и для разработчиков основной системы (на вызов запроса в нужный момент в системе).

Затем (точнее — все вместе) вы аналогичным образом вы планируете и специфицируете интеграцию в обратную сторону — от чат-бота к основной системе. Обратите внимание, что в примерах выше основная система передает не полную информацию, например, в точке интеграции для уведомления о новой публикации на сайте есть только ID публикации, ID автора и список ID пользователей, которых нужно уведомить, но отсутствует название и текст публикации.

Всю дополнительную информацию, которая необходима чат-боту, чтобы обеспечить корректную коммуникацию, чат-бот запрашивает сам во встречных API запросах по мере необходимости. Вы, конечно, можете построить архитектуру иначе — передавать все сведения в основном запросе, чтобы в чат-боте не было необходимости подгружать дополнительную информацией — это ваше право.

Для согласования того какие API точки интеграции нужны для запросов от чат-бота к основной системе вы составляете аналогичную таблицу с точками интеграций в которой специфицируете все API, который нужны вам в чат-боте, в том числе запрос на изменений данных, не только на чтения.

№	Точка интеграции / Запрос	Параметры	Описание
1	Получение информации о пользователе по его ID, имейлу или телефону		

Запрос:

```
{
  "userId": "<ID
пользователя>",
  "email": "<Адрес
электронной
почты>",
  "phone":
"<Телефонный
номер>"
}
```

Пример ответа:

```
{
  "userId": 127,
  "surname":
"Александр",
  "name":
"Сергеевич",
  "patronymic":
"Пушкин",
  "role": "user",
  "isActive": true,
  "phone":
"7918777777",
  "email":
"address@domain.ru"
,
  "location": {
    "country":
"Россия",
    "region":
"Ставропольский
край",
    "city":
"Ставрополь"
  }
}
```

GET /user

Используется в авторизации и регистрации для поиска пользователя.

2	Отправка данных для регистрации нового пользователя		
	GET /user	<div>Запрос:</div> <div><pre>{   "firstname": "&lt;Имя&gt;",   "lastname": "&lt;Фамилия&gt;",   "middlename": "&lt;Отчество&gt;",   "role": "&lt;Роль&gt;",   "phone": "&lt;Мобильный телефон&gt;",   "email": "&lt;Email адрес&gt;",   "location": {     "country": "&lt;Страна местоположения&gt;",     "region": "&lt;Регион&gt;",     "city": "&lt;Город&gt;"   } }</pre></div> <div>Пример ответа 200 ОК:</div> <div><pre>{   "userId": 1177, }</pre></div>	Используется в регистрации.

# Синхронизация данных

Модель синхронизации данных в чат-боте — это важный аспект проекта, понимание которого позволит избежать ошибки в работе над коммуникациями и точками интеграции.



Основной момент заключается в том, что чат-бот не хранит в своей базе данных абсолютно все данные, которые есть в основной системе. Чат-бот подгружает и кэширует необходимые данные для обеспечения коммуникации.

Например, если нужно отправить комментарий 50 пользователям о новости, чат-бот подгрузит данные только по новости, сделает рассылку пользователям в чат-боте и на этом все. Когда в системе произойдут новые изменения, система запустит новую коммуникацию и чат-бот снова подтянет нужные данные.

В ходе диалога с чат-ботом пользователи могут внести изменения в систему. Например, получив новость пользователи могут прокомментировать ее. В этом случае, чат-бот вызовет нужные методы API. Основная система зафиксирует изменения в базе данных и затем запустит новую коммуникацию. Цикл повторится.

Подготовка данных для обеспечения коммуникации чат-ботом осуществляется двумя способами:

- Данные, необходимые для обеспечения коммуникации, подгружаются чат-ботом с помощью API портала по необходимости (pull on demand):
  - Это наиболее часто используемый метод для уведомлений связанных с бизнес-логикой;
  - В этом случае, основная система вызывает нужную коммуникацию, передает ID сущности, ID пользователей (или одного пользователя). Затем чат-бот по ID сущности, подгружает (pull'ит) по API данные необходимые для обеспечения коммуникации;
- Все данные, необходимые для обеспечения коммуникации, целиком и полностью передаются в чат-бот в момент вызова коммуникации порталом (push):
  - Например, можно использовать этот метод используется для синхронизации данных о пользователях. Каждый раз сведения об учетных записях пользователя меняются в основной системе, необходимо их отправлять чат-боту для актуализации.

Какую архитектуру выбрать зависит от многих факторов, включая от доступных разработчики ресурсов со стороны основной системы и необходимой заказчику гибкости решения и скорости движения. Для максимальной гибкости и скорости движения мы рекомендуем в запросе на коммуникацию передавать базовую информацию, такую как ID ресурса, а для остального запросить у разработчиков основной системы API, чтобы подтягивать данные по мере необходимости. Тогда если вам не хватает каких-то данных, вам не придется просить разработчиков основной системы модифицировать тело запроса — вы подгрузите данные по API.

## Авторизация и регистрация пользователей

Для того, чтобы иметь возможность вести коммуникацию с пользователями через чат-бот необходимо интегрировать учетные записи пользователей, предоставив пользователям портала функционал авторизации и регистрации в чат-боте.

После того, как пользователь портала привязал чат-бота к своей учетной записи, а говоря техническим языком, после того, как было установлено **однозначное соответствие** между идентификатором пользователя в чат-боте (personId/leadId в боте) с идентификатором пользователя в основной системе (userId в системе), можно переходить к коммуникациям.

Таблица соответствия ID пользователей двух систем хранится в чат-боте. Хранить ID пользователя в чат-боте в БД основной системы нет необходимости.

Для отправки коммуникации системе достаточно сообщить в запросе к чат-боту ID пользователя в системе (userId), которому необходимо отправить коммуникацию.

Суть интеграции заключается в использовании API для поиска пользователя системы по идентификатору:

1. Чат-бот запрашивает у пользователя ID, email или телефон, ищет пользователя в основной системе по API, и если находит, переходит к двухфакторной авторизации через SMS.
2. В Telegram можно предложить вариант входа без SMS, сравнивая номер, привязанный к Telegram, с номером, привязанным в основной системе.
3. После подтверждения идентификации чат-бот должен сообщить системе, что пользователь привязал чат-бот. Это делается с помощью метода API системы user/connect-bot.
4. Метод user/connect-bot устанавливает признак того, что пользователь привязал чат-бот. Теперь система может отправлять сообщения (коммуникации) этому пользователю в чат-бот.

Аналогично работает регистрация через чат-бот:

1. Сначала пользователь вводит свои данные, чат-бот ищет пользователя в системе, а если не находит, то создает новую запись с помощью API.
2. Система создает пользователя и возвращает идентификатор нового пользователя (userId).
3. Чат-бот выполняет запрос user/connect-bot, чтобы установить у пользователя признак, что чат-бот привязан.
4. Теперь канал прямой связи с пользователем установлен и можно вести диалог.

Возможны и другие варианты интеграции — все зависит от технических возможностей системы и доступных ИТ ресурсов.

## Структура API чат-бота

Узнать структуру запроса и проверить работу внутреннего API чат-бота можно с помощью Swagger, доступного по адресу платформы: <https://app.metabot24.com/api/docs>.

Доступ к Swagger могут получить только авторизованные пользователи, привязанные к проекту. Зарегистрируйтесь на платформе по адресу <https://app.metabot24.com>, подтвердите электронную почту, авторизуйтесь и затем перейдите по ссылке.

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

Ниже показан пример из Swagger.

# Internal API Endpoints



POST

/bots/{botId}/call/{alias} Call Custom Internal API Endpoint



Launch of an endpoint with JavaScript

## Parameters

Try it out

Name

Description

**botId** \* required

integer  
(path)

Bot ID

botId - Bot ID

**alias** \* required

string  
(path)

Internal API Endpoint Alias

alias - Internal API Endpoint Alias

Request body

application/json



Example Value | Schema

```
{
  "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"
    }
  }
}
```

Обращение к чат-боту осуществляется через API платформы "POST /bots/{botId}/call/{alias}", где:

1. В качестве метода используется **POST**.
2. Для идентификатора чат-бота **botId** на платформе укажите ID вашего бота на платформе, которого предварительно создайте в вашем аккаунте.
  1. ID вы сможете узнать, наведя мышкой на название вашего бота в панели управления.

3. Уникальный идентификатор **alias** точки интеграции (эндпоинта), который соответствует конкретной коммуникации.
  1. Узнать название точки интеграции можно воспользовавшись описанием точек интеграции.
4. Абсолютно все остальные параметры запроса всегда передаются в тело запроса в массив **script\_request\_params**.
  1. Остальные параметры, которые по умолчанию выдает Swagger, можете игнорировать.
  2. Чтобы узнать точный список параметров запроса, воспользуйтесь таблицами с описанием точек интеграции.
  3. Пример body запроса:

```
4. {
  "script_request_params": {
    "requestId": 100,
    "userId": 1707
  }
}
```

В качестве примера рассмотрим ситуацию когда исполнитель принял заявку и необходимо об этом уведомить заказчика. Пример спецификации точки интеграции для этого события показан ниже:

2	Уведомляем о новой публикации на сайте	
	<p>Endpoint Alias: <u>post/new</u></p> <p>URL и Header:</p> <pre>1 Метод: POST 2 URL: {Server URL}/bots/{botId}/call/post/new 3 Accept: application/json 4 Content-Type: application/json</pre> <p>Тело запроса:</p> <pre>1 { 2   "script_request_params": { 3     "postId": &lt;ID публикации/required&gt;, 4     "authorId": &lt;ID автора публикации/required&gt;, 5     "userIds": [ 6       &lt;ID первого пользователя/required&gt;, 7       &lt;ID второго пользователя&gt;, 8       .. 9       &lt;ID N-го пользователя&gt; 10    ] 11  } 12 }</pre>	Уведомляем о новой публикации пользователей в массиве userIds

Название точки интеграции (и одновременно название коммуникации) подчеркнуто красным, а содержимое **script\_request\_params** описано в центральной колонке.

# Авторизация запросов API

Адрес сервера API:

```
https://app.metabot24.com/api/v1
```

Для авторизации запросов чат-бота используется **Authorization Bearer Token**. При каждом вызове API добавляйте заголовок авторизации, пример:

```
Authorization: Bearer <Access Token>
Accept: application/json
Content-Type: application/json
Access Token предоставляется администратором Metabot.
```

## Обработка ошибок

В случае успеха, чат-бот возвращает код ответа **200 OK**, а в теле запроса, содержится информация об успешном выполнении, например:

```
{
  "success": true
}
```

Если есть ошибка, то также возвращается **200 OK** и дополнительно возвращается текст ошибки в поле **message**, например:

```
{
  "success": false,
  "message": "Сообщение об ошибке"
}
```

В каждом отдельном случае ответ API чат-бота может быть расширен, в случае необходимости, например:

```
{
  "success": false,
  "errorCode": 422,
  "message": "Ошибка валидации"
}
```

Все возможные коды ответа сервера:

Код ответа HTTP	Причина	Возможны ли доп. поля в теле ответа
200	Все хорошо. В теле запроса возможны дополнительные ответы.	Да
401	Отсутствуют данные для авторизации или не верный токен авторизации.	Нет
403	Нет прав для выполнения запроса. Например, у пользователя API, для которого выдан токен, нет прав доступа к чат-боту.	Нет
404	Неправильный URL запроса. Возникает, когда точки интеграции нет или URL совсем не верный. Если URL верный, то значит кастомный внутренний эндпоинт не найден по алиасу.	Нет
500	Внутренняя ошибка. Например, ошибка в JavaScript коде чат-бота.	Нет

Подробнее про интеграционные возможности платформы Metabot смотрите в разделе [Конструктор API](#).

# Архитектура чат-бота

В РАЗРАБОТКЕ. Обратитесь за шаблоном решения в поддержку.