Тур по платформе: Архитектура Metabot

«Людям не нужно еще одно приложение, они хотят лучший способ связи с бизнесом. Создавайте содержательные взаимоотношения с помощью обмена сообщениями».
— Шейн Мак, со-основатель XMTP, со-основатель Assit (приобретена Verint).

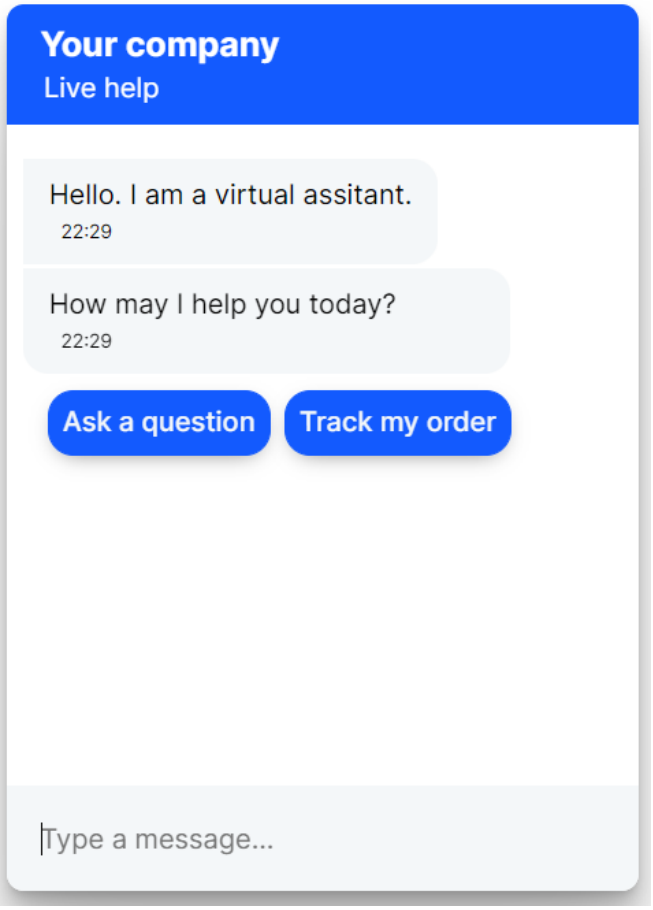
В этом подробном гиде мы рассмотрим, как работает платформа Metabot, на примере взаимодействия с чат-ботом. Мы последовательно пройдемся по ключевому функционалу платформы, рассмотрим, как различные компоненты связаны друг с другом, и порассуждаем на тему, как ведется разработка чат-ботов с глубокими интеграциями в бизнес. После того как вы прочтете данный текст, у вас будет полное и всеобъемлющее представление о том, как устроен Metabot и как ведется разработка на этой платформе.

Чтобы создавать решения с помощью платформы Metabot, вам необходимо понимать компоненты, из которых состоит платформа, как они взаимосвязаны и как их использовать. Архитектура ваших чат-ботов и приложений будет опираться на эти компоненты и их взаимосвязь.

Давайте рассмотрим гипотетический пример общения с участием Боба, Алисы и чат-бота. Боб — клиент компании, где Алиса работает в отделе поддержки клиентов, где они только что внедрили нового чат-бота.

Давайте представим, что Боб зашел на сайт компании, чтобы проверить статус своего заказа. Боб решил не тратить время на изучение сайта, и сразу обратился в чат, чтобы узнать, как проверить состояние своего заказа.

При нажатии на значок чата в правом нижнем углу экрана сайта открывается диалоговое окно, в котором чат-бот приветствует Боба.



Чат-бот

Чат-бот — это компьютерная программа или программное обеспечение, способное обрабатывать информацию о пользователе и общаться как человек. Он может быть оснащен искусственным интеллектом и работать на платформе Metabot.

Чат-боты являются мощным инструментом для облегчения взаимодействия с пользователями и предоставления им удобного способа связи с бизнесом. Они могут быть настолько простыми или сложными, насколько это необходимо, и могут быть адаптированы для различных платформ и нужд.

С технической точки зрения, чат-бот представляет собой **клиент-серверное приложение**. **Серверная часть** — это платформа Metabot, на которой хранятся состояние разговора и пользовательские данные, а также обрабатываются сообщения. **Клиентская часть** может быть мессенджером или приложением для социальных сетей, установленным на устройстве пользователя, либо окном живого чата на веб-сайте.

Клиент не хранит никаких данных или состояние разговора. Все, что Боб видит на экране, — это просто ввод и вывод, история разговора. В этом смысле современные чат-боты ничем не отличаются от консольной строки компьютера. **Состояние разговора** и **пользовательские данные** хранятся на сервере, хранение и обработку которых обеспечивает платформа Metabot.

В понятие "диалоговое приложение" вкладывается более глубокий смысл, чем "чат-бот". Это может включать специальные чат-боты с более широким функционалом и возможностями, благодаря графическим расширениям интерфейсов мессенджеров и интеграции с корпоративными системами.

Диалоговое приложение

Диалоговое приложение (или **dialog app**) представляет собой специализированный тип чат-бота, который обладает более широким функционалом и возможностями по сравнению с обычными чат-ботами. Оно может работать в мессенджерах или социальных сетях и интегрироваться с корпоративными системами и публичными Интернет-сервисами.

Диалоговые приложения являются хорошей альтернативой мобильным приложениям в ряде случаев, особенно когда используются графические расширения мессенджеров. Они могут реализовать практически любой функционал, за исключением некоторых специфических задач, которые требуют прямого доступа к устройству, таких как непрерывное отслеживание координат. Это делает диалоговые приложения более безопасными, так как они получают только те данные, которые пользователь сам отправляет.

Основные характеристики диалогового приложения:

- **Без необходимости скачивания и установки** — они работают в уже предустановленных мессенджерах и социальных сетях, таких как Telegram, WhatsApp;
- **Персонализированный и бесшовный диалоговый интерфейс** — интеграция с корпоративными системами и публичными Интернет-сервисами обеспечивает персонализированный доступ к бизнесу;
- **Идентификация пользователей** — все пользователи идентифицированы, и с каждым установлен прямой личный канал связи;
- **Push-уведомления** — диалоговые приложения могут отправлять push-уведомления на устройства пользователя;
- **Автоматизация коммуникационных бизнес-процессов** — они могут автоматизировать как внешние (диалоги с клиентами и партнерами), так и внутренние коммуникационные процессы (диалоги с сотрудниками);
- **Привычный разговорный интерфейс** — диалоговые приложения имеют разговорный интерфейс, к которому привыкли миллионы людей, упрощая таким образом онбординг новых пользователей.

Диалоговые приложения представляют собой мощный инструмент для современных компаний, позволяя обеспечить эффективное взаимодействие с клиентами и партнерами через уже существующие платформы мессенджеров и социальных сетей. Они обладают уникальными возможностями для персонализации, автоматизации и удобства в использовании, что делает их ценным решением для многих бизнес-потребностей.

> Диалоговое приложение также может называться "диалоговым ботом", "разговорным приложением" или "разговорным интерфейсом".

Каналы

Когда Боб щелкнул значок чата и развернул окно живого чата, его приветствовал чат-бот. Вместо окна живого чата на сайте Боб мог нажать на один из значков мессенджеров или социальных сетей, перейти в их приложение и там начать аналогичный диалог с компанией, например, в Telegram или WhatsApp.

Каналы коммуникации — это посредники, с помощью которых ваша компания общается с целевой аудиторией. Подумайте о «канале» чат-бота как об интерфейсе/системе, используемой конечным пользователем для взаимодействия с вашей компанией. Эти каналы не ограничиваются вашим сайтом; они распространяются на различные платформы, где ваша аудитория проводит время.

> Подключение чат-бота к каналам популярных цифровых платформ, чьи приложения установлены или можно установить на мобильные устройства пользователей, позволяет реализовать стратегию **подключенного диалогового опыта** (connected conversational experience) для вашего бренда или компании. Это обеспечивает бесшовное и последовательное взаимодействие, независимо от того, где происходит разговор.

Боты в Metabot являются омниканальными, то есть вы можете разработать чат-бот один раз, а затем подключиться к нескольким различным каналам одновременно. Вам не нужно будет создавать новый чат-бот, хотя в некоторых случаях вам нужно будет адаптировать своего бота к специфическим требованиям канала. Например, разные каналы могут разрешать прикреплять к сообщению разное количество кнопок, поэтому вам нужно будет создать диалоговые сценарии, учитывающие эту особенность.

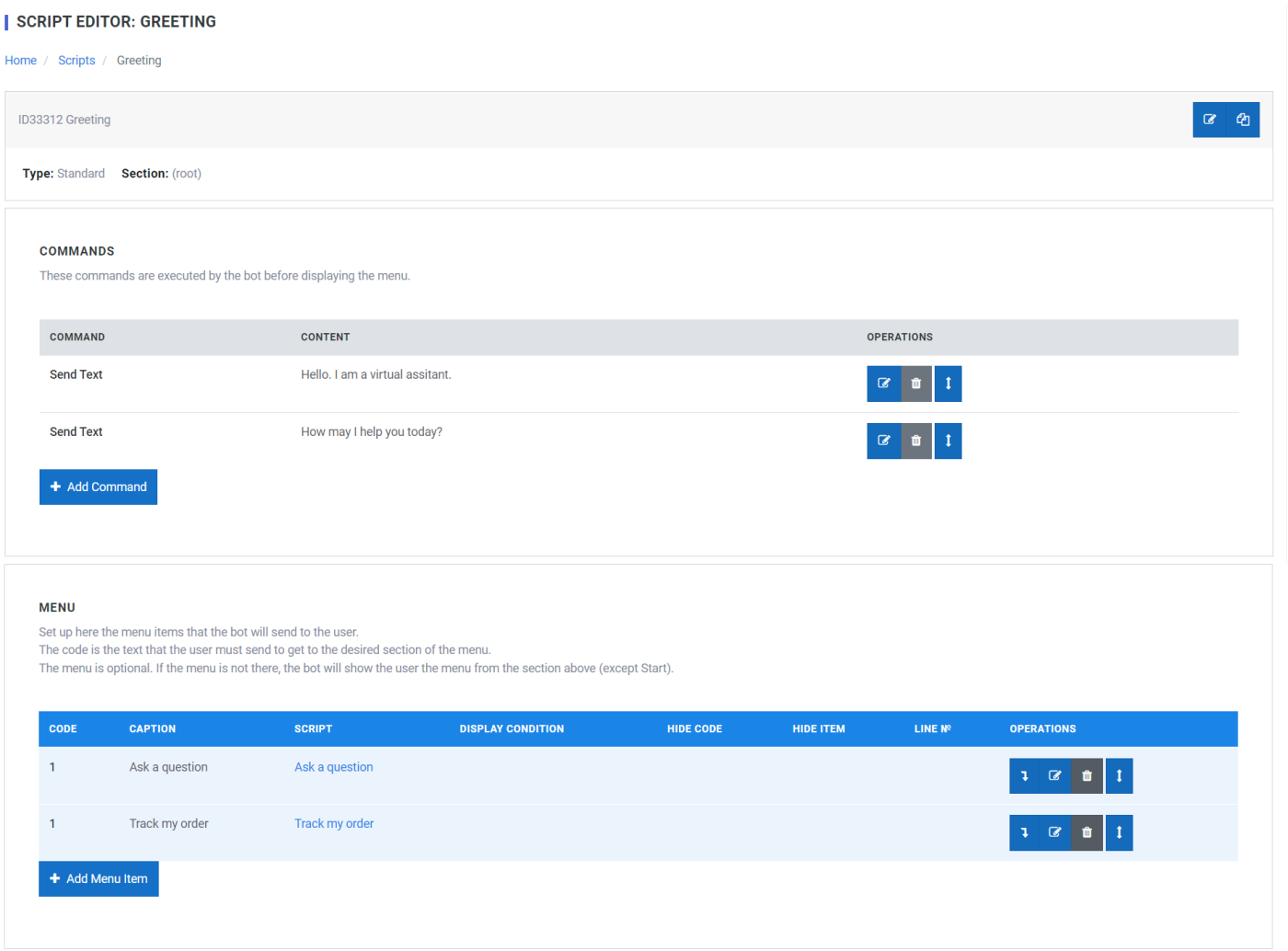
Этот омниканальный подход улучшает пользовательский опыт, предоставляя гибкость и удобство. Будь то ваши клиенты предпочитают общаться через социальные сети, мессенджеры или непосредственно на вашем сайте, они могут иметь последовательное и персонализированное взаимодействие с вашим брендом.

> Мы рекомендуем вам прочитать документацию по **Каналам**, чтобы понять, как лучше всего использовать их для ваших конкретных потребностей. Это поможет вам в выборе правильных каналов для вашей аудитории и обеспечит оптимизацию вашего чат-бота для каждой платформы, предоставляя действительно подключенный и увлекательный опыт.

Скрипты и команды

Когда Боб открыл окно чата, его поприветствовал чат-бот, прислав ему текст приветствия. Сообщил, что говорит бот, а также прислал две быстрых кнопки: "Задать вопрос" и "Отследить мой заказ".

В техническом плане, приветствие — это самый первый скрипт, состоящий из трех команд: два сообщения и кнопки меню. **Скрипт** — это сценарий диалога, который состоит из команд, обеспечивающих взаимодействие с пользователем. **Команды** — это инструкции, которые чат-бот выполняет в рамках скрипта, такие как отправка сообщения или запрос ввода.



В архитектуре Metabot скрипты и команды играют ключевую роль в создании чат-ботов и диалоговых приложений, позволяя создавать гибкие и адаптивные диалоговые интерфейсы.

Вот список команд, которые можно использовать в Metabot:

- **Отправить текст** — бот отправляет текстовое сообщение с эмодзи;
- **Отправить изображение** — бот отправляет пользователю картинку;
- **Отправить файл** — бот отправляет файл, например, документ;
- **Отправить Email** — бот отправляет email;

- **Установить статус лида** — бот меняет статус обращения в CRM;
- **Выполнить скрипт** — выполнить отдельный скрипт с командами бота;
- **Запросить значение** — запросить значение у пользователя и сохранить его в атрибутах лида;
- **Добавить тэги** — добавить тэг к обращению;
- **Удалить тэги** — удалить тэги в обращении;
- **Добавить контекст** — добавить контекст обращения;
- **Удалить контекст** — удалить контекст обращения;
- **Повторить вопрос** — повторить вопрос;
- **Выполнить триггер** — прямой вызов триггера с проверкой условия в триггере;
- **Выполнить JavaScript** — выполнить JavaScript;
- **Выполнить JavaScript Callback** — запустить JavaScript Callback;
- **Стоп** — остановить выполнение, не выводить меню;
- **Распознать NLP намерение** — по распознанному намерению будет найден и запущен соответствующий скрипт;
- **Перевести на оператора** — перевести диалог на оператора CRM;
- **Вернуть боту** — вернуть диалог от оператора обратно на бота.

> Подробнее ознакомиться с командами вы можете в **Командах**, а узнать как работать со скриптами в **Скриптах**.

Лиды и диалоги

Когда Боб впервые заговорил с чат-ботом компании, на платформе Metabot для него автоматически создалась запись. В этой записи будет храниться вся история диалога с Бобом, собранные в ходе диалога данные, а также состояние диалога. Если диалог в какой-то момент будет передан Алисе, сотруднице колл-центра, она также увидит всю эту информацию.

Когда пользователь впервые связывается с компанией через один из доступных каналов, в системе Metabot автоматически создается новый **лид**, и начинается разговор.

Лид в контексте платформы Metabot — это конечный пользователь вашего чат-бота, а в маркетинге это термин, обозначающий потенциального клиента. Вся история диалога сохраняется на платформе для каждого пользователя и доступна на вкладке "Лиды".

Состояние диалога — это ключевой момент в беседе, на котором пользователь остановился. Оно запоминает текущие данные разговора, а также сценарий и команду, на которой пользователь остановился. При переходе к следующей команде или сценарию состояние диалога автоматически меняется.

В следующий раз, когда диалог возобновится пользователем (или ботом), он продолжится в зависимости от состояния диалога. Вам не нужно заботиться о состоянии диалога, его возобновлении и тому подобном. Вы проектируете сценарии и управляете состоянием с

помощью команд — все остальное за вас сделает платформа.

На платформе есть специальная команда для остановки/сброса диалога. Ее вызов стирает состояние разговора, и когда пользователь снова говорит с ботом, диалог начинается с самого начала, то есть с приветствия. Однако рекомендуется никогда не прерывать диалог и спроектировать бота так, чтобы беседа никогда не достигла точки останова. Сброс диалога рекомендуется использовать только в целях отладки.

Если вам нужно обновить всего бота для всех пользователей, вы можете воспользоваться функцией "Сбросить все беседы".

> Рекомендуем ознакомиться с секций **Лиды**.

Персоны

Допустим, компания подключила чат-бот к Telegram и WhatsApp. Когда Боб, ради любопытства, написал в компанию в оба мессенджера, то платформа создала для него два лида. А как сделать так, чтобы наше приложение знало, что оба лида принадлежат одной **персоне**, т.е. одному человеку? Правильно — использовать функционал персон!

После того как вы познакомились с концепцией лида, вам необходимо познакомиться с концепцией персоны (persons). Чат-бот — это инструмент персонализации коммуникации, и понятие персоны является важным элементом.

Если лид — это аккаунт пользователя в конкретном канале (например, в Telegram или WhatsApp), то персона — это сам человек. У персоны может быть несколько лидов (аккаунтов в каналах), не так ли?

Мы создали персоны, чтобы иметь справочник людей для интеграции с внешними системами. В работе вы будете сталкиваться с ситуациями, когда вам нужно интегрироваться с внешней системой, в которой хранятся пользователи.

Мы рекомендуем устанавливать связку между персонами и пользователями во внешней системе. Это правильная структура связей. При разработке баз данных и бизнес-приложений в таблицах мы рекомендуем завязываться на запись о человеке, а не о его аккаунте в мессенджере. Например, в любой компании есть база данных у HR отдела со списками сотрудников.

У пользователя может поменяться номер, он может создать новый аккаунт в мессенджере — в этом случае вы просто поменяете у персоны лид, через который он работаете, и человек получит доступ к своим данным. Например, вы делали чат-бота для интернет-магазина и использовали ID лида как идентификатор заказчика. У одного заказчика поменялся Телеграм аккаунт и теперь он начал делать заказы под новым лидом. Если бы вы использовали ID персоны для идентификации заказчика, то информация о заказах

продолжала накапливаться под тем же самым аккаунтом.

Итак, в терминах платформы Metabot, персона — это запись с данными о человеке. У каждой персоны есть **роль** (person role). Вы можете создать набор ролей для каждого класса персона, например, гость, пользователь, сотрудник, администратор и другие.

Вы можете проверять в чат-боте принадлежность персоны к той или иной роли, например, для того чтобы контролировать и управлять уровнем доступа к тому или иному функционалу. Мы рекомендуем создавать разные скрипты с навигационным меню для каждой роли.

Редактирование персоны

Имя:

Фамилия:

Отчество:

E-mail:

Телефон:

Сотрудник / Клиент:

- ○ Сотрудник бизнеса (внутренняя персона)
- ◉ Клиент (внешняя персона)

Название организации:

Адрес организации:

Роль:

[Не выбрано]

Web-пользователь:

[Не выбрано]

Комментарий

> Рекомендуем подробнее ознакомиться с персонами и ролям в разделе **Персоны**.

Маршруты

Допустим, Боб впервые заходит в чат-бот и видит приветственное сообщение. Как бот определил, что нужно показать именно это сообщение? Ответ кроется в механизме, называемом **маршруты**.

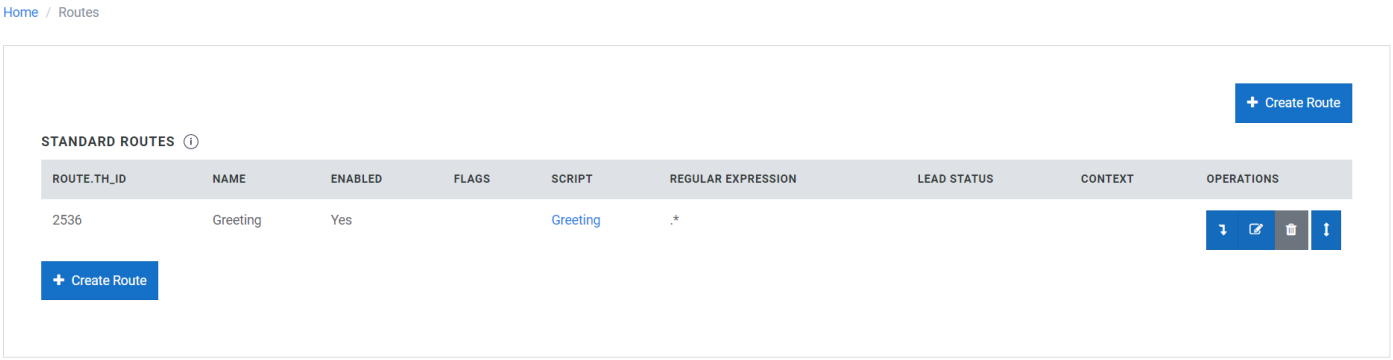
Маршруты — это точки входа в сценарий диалога. Маршруты работают за кулисами — слушают ввод пользователя и запускаются при соблюдении условий запуска маршрута. Запуск нового маршрута переключает состояние диалога на новый сценарий.

Существует несколько типов маршрутов:

- **Маршруты Начала Диалога** — обрабатываются только тогда, когда у пользователя еще нет состояния разговора, т.е. разговор с ботом еще не начался (или был принудительно остановлен или сброшен на платформе). Например, для приветствия и отображения приветственного меню;
- **Маршруты Взаимодействия** — работают, когда разговор уже идет. Например, когда пользователь уже "течет" по боту, а мы хотели, чтобы бот реагировал на секретное слово, например, «помощь» или «купон»;
- **Маршруты Передачи Управления** — работают, когда диалог передан человеку и бот замолкает. Например, когда пользователь обратился за помощью к человеку, но передумал и хочет позвать бота обратно;
- **Маршруты Deeplink (Диплинки)** — работают по внешним ссылкам или так называемым deeplink-ссылкам, когда мы заходим в бота по внешней ссылке, например, через рекламу или QR-код и хотим запустить конкретный диалоговый сценарий, связанный с этой ссылкой.

Маршруты имеют множество параметров, которые определяют, когда они должны сработать и какой скрипт запустить. Например, для примера Боба и Алисы мы создали маршрут с названием Greeting (приветствие), который запускает одноименный скрипт, когда пользователь пишет любой текст в чате. Регулярное выражение ".*" соответствует любому вводу.

Вы также можете перемещать маршруты вверх и вниз, чтобы изменить **приоритет маршрутов**. У вас может быть несколько маршрутов одновременно, но всегда срабатывает только один маршрут — тот, который успешно определился первым.



> Регулярное выражение, regular expression или сокращенно regex представляет собой специальный шаблон для сопоставления и поиска текста. Мы рекомендуем прочитать информацию в Интернет о регулярных выражениях для общего развития.

> Рекомендуем ознакомиться с разделом **Маршруты**.

Тэги и атрибуты

Представьте себе ситуацию, где чат-бот снова взаимодействует с пользователем по имени Боб. Боб только что подписался на рассылку и скачал файл. Чат-бот может сохранить эту информацию с помощью **тэгов** и **атрибутов**, что позволит более персонализировано и эффективно взаимодействовать в будущем.

Тэги и **атрибуты** являются основными компонентами, которые хранят данные, собранные чат-ботом во время взаимодействия с пользователем. Они обеспечивают надежный и гибкий способ управления взаимодействием с пользователями, позволяя более персонализировано и эффективно вести диалог. Понимая и используя эти компоненты, вы можете создать более интересный и отзывчивый опыт взаимодействия с чат-ботом.

Тэги — это простые ярлыки, которыми вы можете пометить пользователя. Например, "зарегистрировался", "скачал_файл" и т.д. Тэг либо есть, либо его нет.

Атрибуты — это переменные разных типов, в которых хранятся значения: строка, число, дата, булево, JSON. Например, "возраст", "город", "страна".

JSON — это многосоставное поле, в котором можно хранить массивы данных. JSON — наиболее популярный формат для обмена системами через REST API.

> JSON (JavaScript Object Notation, обычно произносится как /ˈdʒeɪsən/ JAY-sən) — текстовый формат обмена данными, основанный на JavaScript. Как и многие другие текстовые форматы, JSON легко читается людьми. Подробнее:
>
> https://ru.wikipedia.org/wiki/JSON

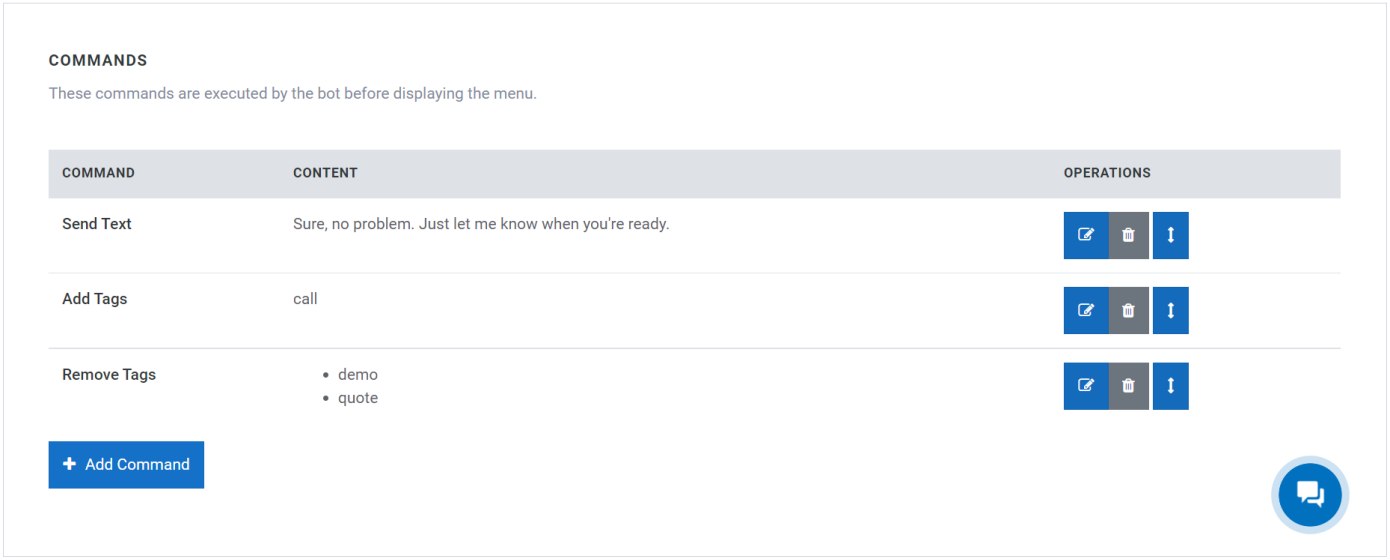
Пример JSON:

```
{
   "firstName": "Иван",
   "lastName": "Иванов",
   "address": {
       "streetAddress": "Московское ш., 101, кв.101",
       "city": "Ленинград",
       "postalCode": 101101
   },
   "phoneNumbers": [
       "812 123-1234",
       "916 123-4567"
   ]
}
```

Еще один пример, где названия полей указаны на русском языке:

```
{
   "имя": "Иван",
   "фамилия": "Иванов",
   "адрес": {
       "улица": "Московское ш., 101, кв.101",
       "город": "Ленинград",
       "индекс": 101101
   },
   "телефоны": [
       "812 123-1234",
       "916 123-4567"
   ]
}
```

Для добавления и удаления тэгов используйте команды **Добавить тэги** и **Удалить тэги** соответственно.



Для работы с тэгами лида используйте следующие JS-методы объекта lead:

Название метода	Метод
Проверить существование тэга	lead.**isTagExist**('some_tag')
Получить список тэгов лида	lead.**getAllTags**(): array

Для работы с атрибутами лида используйте следующие команды JS объекта lead:

Название метода	Метод
Установить значение переменной лида	**lead.setAttr(string $key, string $value): self**
Получить значение переменной лида	**lead.getAttr(string $key): string|null**
Получить массив всех переменных лида	**lead.getAllAttr(): array**
Проверить наличие переменной лида	**lead.isAttrExist(string $key): bool**

Объект **lead** является ключевым элементом во взаимодействии чат-бота с пользователями и доступен в JS из любого места на платформе, будь то скрипты, триггеры, API эндпоинты или условия. Через lead вы можете обратиться к текущему лиду, с которым ваш бот ведет диалог.

Ваш чат-бот может одновременно вести диалог с десятками, сотнями или даже тысячами лидов. Платформа Metabot работает асинхронно, обрабатывая каждый диалог в отдельном процессе, называемом **воркером** (worker, работником). Воркер выполняет для лида различные скрипты, триггеры и прочее. Когда вы используете lead в вашем коде, вы можете

быть уверены, что когда процесс дойдет до этого места, ваш код обратится к лиду, с которым идет диалог. Если же вам нужно в ходе диалога с одним лидом обратиться к другому лиду, то такое тоже возможно, но это делается специальным образом, о чем вы узнаете позже.

Атрибуты бота

Если вам необходимо где-то сохранить общие для всех лидов данные, воспользуйтесь хранилищем самого чат-бота. Атрибуты и тэги бота — это глобальное пространство имен, доступное для любого лида в ходе любого диалога.

В боте рекомендуем хранить общие для всех лидов данные, такие как настройки бота, результат вызова общих API и т.д., в то время как в лиде храните информацию по конкретному лиду.

В JavaScript вы также можете работать с атрибутами бота, используя следующие методы:

Название метода	Метод
Установить значение переменной бота	**bot.setAttr(string $key, string $value): self**
Получить значение переменной бота	**bot.getAttr(string $key): string|null**
Получить массив всех переменных бота	**bot.getAllAttr(): array**
Проверить наличие переменной бота	**bot.isAttrExist(string $key): bool**

Эти методы обеспечивают гибкость и мощь в управлении данными, позволяя вам создавать сложные и динамичные взаимодействия с пользователями через вашего чат-бота.

Условия

Представьте ситуацию, когда Боб пытается второй раз подписаться на новостную ленту. Нужно ли это делать? Конечно, нет. Чат-бот должен проверить, подписан ли Боб уже, и если да, то показать другое меню или как минимум вывести сообщение, что он уже подписан. Для реализации таких проверок в чат-ботах используются **условия**.

Условия — это код на JavaScript, который вы можете применить к различным объектам (командам, маршрутам, кнопкам, триггерам и т.д.) в качестве критерия выполнения или отображения этих объектов пользователю. Если условие истинно (true), то действие совершается; если ложно (false), объект игнорируется.

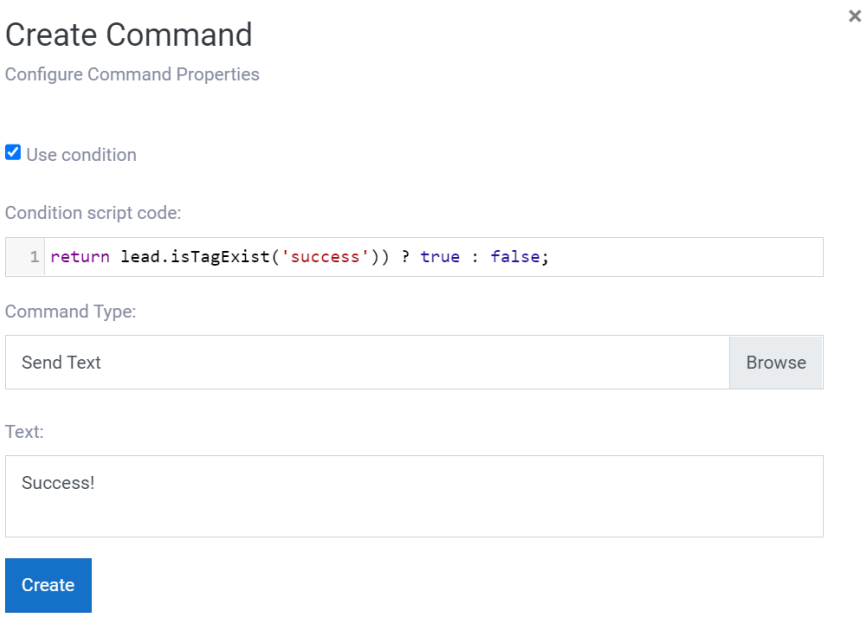
Это может быть полезно в различных сценариях. Например:

- Если вы хотите, чтобы бот спросил имя пользователя только в случае, если он еще не вводил его;
- Если вы хотите скрыть кнопку "Подписаться на ленту", когда пользователь уже подписан.

Примеры кода:

Код условия	Объяснение
return lead.getAttr('some_var')*1 == 100 ? true : false;	Если some_var равен числу 100, то условие истинно (true), иначе (false).
return !lead.isTagExist('some_tag');	Если у пользователя НЕ стоит some_tag тэг, то условие истинно. Восклицательный знак перед условием в JS означает инверсию булевой переменной.
return lead.isTagExist('some_tag');	Если у пользователя стоит тэг some_var, то будет истинное условие.
return bot.getAttr('some_var') == "yes";	Если в боте в атрибуте some_var записан текст "yes", то условие истинно.
return lead.getAttr('age') >= 18;	Если возраст пользователя больше или равен 18, то условие истинно.
return lead.isAttrExist('email');	Если у пользователя есть атрибут 'email', то условие истинно.

На скриншоте ниже показан пример использования JS кода в условиях вывода команды.



Условия являются мощным инструментом для создания гибких и адаптивных чат-ботов, которые могут реагировать на различные ситуации и предоставлять пользователю наиболее релевантный опыт.

JavaScript Low-Code

Вы уже познакомились с написанием условий на JavaScript. Теперь давайте рассмотрим более широкий аспект написания кода.

Представьте следующие ситуации:

- Боб хочет узнать статус заказа, который хранится не в боте, а в системе учета заказов;
- Боб хочет получать уведомления об изменении статуса заказа в Telegram или WhatsApp;
- Алиса хочет автоматизировать обработку клиентских запросов, интегрируя чат-бот с системой управления отношениями с клиентами (CRM).

Здесь без программирования не обойтись. Поддержка языка JavaScript является одной из ключевых особенностей платформы Metabot, и мы надеемся, что вам понравится наш low-code подход к созданию чат-ботов так же сильно, как он нравится нам!

Что такое Low-Code?

Low-code, или разработка с минимальным кодом, — это упрощенный подход к созданию программного обеспечения. Он включает в себя использование готовых компонентов, проверенных методологий, простых команд и визуальных инструментов.

Написание кода все еще играет важную роль, но не в таком объеме, как при традиционной разработке. Платформа Metabot заботится о многих рутинных задачах, которые обычно выполняют разработчики.

Преимущества Low-Code

Low-code подход хорош тем, что он позволяет бизнес-пользователям (маркетологам, дизайнерам, специалистам по работе с клиентами) управлять **цифровым опытом** вместе с технической командой.

Под "цифровым опытом" подразумевается взаимодействие между бизнесом/организацией и клиентом/пользователем с помощью информационных технологий и устройств. Это может включать в себя клиентский опыт (CX), опыт сотрудника (EX), опыт партнера и так далее.

Существуют различные подходы к разработке программного обеспечения: high code, low code, no code, и даже генерация кода с помощью ИИ (AI-generated code).

- **High Code** — традиционный подход с большим количеством кода. Здесь цифровым опытом полностью управляют разработчики, а циклы разработки самые длинные и дорогие;

- **Low Code** — упрощенный подход с использованием визуальных интерфейсов и минимальным количеством кода. Здесь цифровым опытом преимущественно управляет заказчик и команда специалистов отвечающая за коммуникации, которые в паре с программистами создают решение;
- **No Code** — решения без кода, создаваемые через конфигурацию и настройки. Здесь цифровой опыт может создаваться без программистов или при их минимальном участии. No Code решения обычно заточены на решение конкретных кейсов и не обладают универсальной гибкостью, как в случае с High Code и Low Code.

Metabot Platform — это low-code инструмент для командной работы, обеспечивающий гибкость и скорость создания и запуска коммуникационных проектов и чат-ботов.

Применение JavaScript в Metabot

JavaScript играет ключевую роль в разработке на Metabot. Вы можете использовать его в различных аспектах чат-бота, таких как:

- Условия запуска команд в скрипте;
- Условия вывода кнопок для выбора в меню;
- Манипуляции с данными: атрибуты, теги, кастомные таблицы и пр;
- Работа с триггерами, рассылками и отложенными задачами;
- Создание API для чат-бота;
- И многие другие случаи.

Основной момент, который вам нужно знать, чтобы понимать в чем смысл low code в рамках Metabot, заключается в следующем. Вы выстраиваете архитектуру вашего чат-бота преимущественно с помощью визуальных интерфейсов, но как только вам нужны алгоритмы вы используете вставки JS кода там где это необходимо.

Пример

После того как мы узнали, что Боб хочет отследить заказ, мы можем выполнить JS код, который обратится к корпоративной системе заказов по API и вернет статус заказа. Эта информация затем может быть выведена в чат-боте.

Ниже приведен пример кода, который выполняет запрос к корпоративной системе учета заказов, используя номер заказа, полученный от пользователя. Если статус заказа найден, он сохраняет информацию о заказе в атрибутах лида (заказчика), таких как статус, дата, имя клиента и общая сумма заказа.

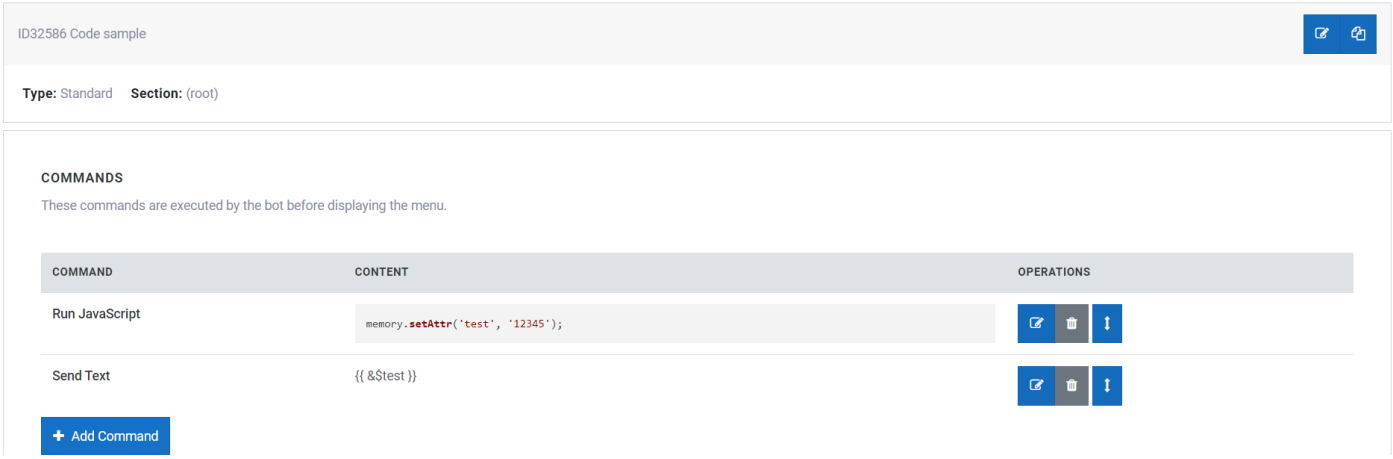
```
// Выполнение REST запроса для проверки статуса заказа
memory.setAttr('orderStatusFound', 0);
let orderNumber = lead.getAttr('orderNumber'); // Получение номера заказа от пользователя
let url = `https://corporate-system.example.com/api/orders/${orderNumber}`; // URL API
корпоративной системы
```

```
let jsonResponse = api.getJson(url);

if (jsonResponse) {
  // Сохранение результата запроса в атрибутах лида (заказчика)
  var orderInfo = jsonResponse;
  let orderStatus = orderInfo['status'];
  if (orderStatus) {
    memory.setAttr('orderStatusFound', 1);
    lead.setAttr('orderInfo.status', orderStatus);
    lead.setAttr('orderInfo.date', orderInfo['date']);
    lead.setAttr('orderInfo.customerName', orderInfo['customerName']);
    lead.setAttr('orderInfo.totalAmount', orderInfo['totalAmount']);
  }

  // Сохранение информации о последнем HTTP-ответе
  memory.setJsonAttr("lastHttpResponse", api.getLastResponseContent());
  memory.setAttr("lastHttpResponseCode", api.getLastResponseCode());
}
```

Ниже показаны скриншоты скрипта с JS командой:



При редактировании кода открывается редактор с подсветкой синтаксиса:



Как использовать JavaScript код? Процесс довольно прост. Если вам необходимо реализовать определенную логику, описать условия выполнения различных операций или сделать выборку пользователей, вы обращаетесь к JavaScript.

Как эффективно применять JavaScript? Чтобы максимально эффективно использовать JavaScript, рекомендуется ознакомиться со справочником объектов платформы, доступных для манипуляций из JS, а также изучить их свойства и методы.

Также вам всегда доступен справочник по JS функциям.

Могу ли я создавать чат-боты с помощью Metabot без знаний JavaScript? Конечно, это возможно. Многие задачи, такие как автоматизация контакт-центра, могут быть решены практически без JS или с использованием минимального набора базовых JS-операторов. Вы также можете воспользоваться нашими примерами, типовыми конфигурациями ботов, готовыми плагинами и интеграциями. Однако для сложных задач и интеграций потребуется либо умелое владение JS, либо помощь JS-разработчика.

Что такое JavaScript? JavaScript — это популярный язык программирования с огромным сообществом разработчиков по всему миру и множеством доступных примеров в Интернете. Если вы столкнулись с трудностью в написании определенного кода, вы можете воспользоваться поисковыми системами или инструментами, такими как GPT, чтобы найти или сгенерировать нужный код. Затем вы можете адаптировать его под Metabot. Стоит отметить, что платформа Metabot поддерживает урезанную версию JS без Promises, но включает в себя все основные элементы языка JS, такие как переменные, функции, классы, вызов REST API и многое другое.

Кастомные таблицы

Помните, как мы рассматривали примеры с Бобом, который хотел узнать статус заказа или получать уведомления об изменении статуса заказа? Возможно, вам захотелось узнать, как эти данные хранятся и организуются. Вот где вступают в игру кастомные таблицы и базы данных.

Если вы уже сталкивались с разработкой программного обеспечения или приложений, вы, возможно, подумали: "Тэги и атрибуты — это здорово, но мне нужны отдельные таблицы, а ещё лучше — полноценная база данных со множеством таблиц!" Не переживайте, мы позаботились об этом.

Для тех, кто только знакомится с разработкой приложений, важно понимать, что практически любое приложение, в каких бы целях оно не создавалось (для личного пользования, для вашего заказчика или вашего бизнеса), не обходится без сервиса базы данных.

Кастомные таблицы — еще один важный компонент платформы Metabot, который позволяет организовать хранение данных и доступ к ним на уровне алгоритмов чат-бота и API запросов к нему. Вы сможете создавать таблицы с нужным вам полями и затем с помощью JS методов объекта **table** работать с ними.

Ниже приведен список некоторых функций объекта **table** для работы с таблицами и объекта item для работы со строками таблицы:

Название метода	Метод
Создание записи	table.**createItem**(string $tableName, array|object $data): **ScriptCustomTableItem**
Поиск записей	table.**find**(string $tableName, array|object $columns = [], array|object $where = [], array|object $orderBy = [], ?int $limit = null, ?int $offset = null): **array**
Подсчет количества записей	table.**count**(string $tableName, array|object $where = []): **int**
Подсчет суммы по полю	table.**sum**(string $tableName, string $column, array|object $where = []): **int**
Поиск максимального значения по полю	table.**max**(string $tableName, string $column, array|object $where = []): **mixed|null**
Обновление записи	item.**update**(array|object $data): **bool**
Удаление записи	item.**delete**(): **bool**

Получить форматированную дату по названию поля	item.**getDateFormatted**(string $fieldName, string $format = 'Y-m-d H:i:s', ?string $timeZone = null): **string|null**
Получить форматированную дату из произвольной строки	item.**getDateFormattedFromString**(string $date, string $format = 'Y-m-d H:i:s', ?string $timeZone = null): **string|null**
Перезагрузить данные записи из БД	item.**reload**(): **bool**

Когда Боб запросил статус посылки, мы могли бы обратиться к таблице `**orders**` и с помощью `**table.find**` найти запись, например, с помощью следующего кода:

```
let id = lead.getAttr('orderId');
let items = table.find('orders', ['status'], [["orderId", "=", id]]);
```

Таким образом, мы получаем из таблицы `**orders**` столбец `**status**` заказа `**id**`, который был записан в атрибут `**orderId**`.

> Полный список функций и возможностей кастомных таблиц смотрите в разделе **Кастомные таблицы**.

Интеграции API

Пример из предыдущей секции про поиск заказа во внутренней таблице вашего чат-бота, созданной на Metabot, справедлив только для случаев, когда вы реализовываете бизнес-приложение полностью на Metabot.

Допустим, в предыдущем примере с Бобом, который хотел узнать статус своей посылки, вам нужно было обратиться к внешней системе логистики. Это можно было бы сделать с помощью следующего кода:

```
let trackingNumber = lead.getAttr('trackingNumber');
let url = `https://external-shipping-system.example.com/api/parcels/${trackingNumber}`;
let jsonResponse = api.getJson(url);
let parcelStatus = jsonResponse['status'];
lead.setAttr('parcelStatus');
```

Этот код обращается к внешней RESTful системе и получает статус посылки Боба, используя его трекинг номер.

>

> API (Application Programming Interface) — это набор правил и протоколов, которые позволяют одним программам взаимодействовать с другими. Если вы новичок в этой области, рекомендуем подробно изучить что такое API.

> REST (Representational State Transfer) — это архитектурный стиль взаимодействия компонентов в сети. RESTful веб-сервисы используют стандартные HTTP-методы. Если вы не знакомы с REST, рекомендуем ознакомиться с этим руководством по REST.

В реальности вы часто будете сталкиваться с заказчиками, у которых уже есть ERP, Ecomm или любая другая система для учёта и хранения данных, к которой вам нужно будет обращаться по API запросу. Это может включать работу с системами, такими как 1С:Предприятия, AmoCRM, Bitrix24, Microsoft Dynamics, OTRS и другие.

Вам нужно будет уметь интегрировать ваши чат-боты с этими системами, чтобы создавать бесшовный и персонализированный диалоговый опыт для ваших пользователей. Это требует профессионализма, определенных технических навыков и мастерства, а также умения работать с различными сторонами, включая маркетологов и других сотрудников заказчика. Но не волнуйтесь, наша методология глубоких интеграций мессенджеров в бизнес-процессы и платформа Metabot, помогут вам в этом.

Интеграции с системами делаются с помощью **конструктора API**. Платформа позволяет создавать **точки интеграции**, называемые **эндпоинтами (endpoints)**, которые описывают все точки взаимодействия вашего чат-бота с внешними системами и наоборот.

Интеграция сервисов осуществляется с помощью стандартных REST API интерфейсов. Если вы столкнулись с устаревшими системами и сервисами, вам, возможно, придется организовать обмен данными другим способом или работать с API в два шага (сперва авторизоваться под конкретным пользователем, а потом выполнять запросы от его имени).

Существует несколько способов вызова API, включая использование настроенных API endpoints и прямые вызовы объекта **`api`**, как в примере выше.

> Подробную информацию о том, как разрабатывать API интеграции с примерами, смотрите в разделе **Конструктор API**.

Исходящие коммуникации

Представьте себе ситуацию, где ваш чат-бот автоматически отправляет пользователю поздравительное сообщение по его дню рождения или напоминает о предстоящем мероприятии. Как это можно реализовать? Ответ на этот вопрос лежит в понимании **триггеров**, **рассылок** и **заданий** в контексте платформы Metabot.

Боты не только реагируют на команды пользователя, но и могут самостоятельно инициировать диалог, реагируя на определенные события или по расписанию. Давайте разберемся, как на Metabot создаются такие проактивные коммуникации.

Metabot имеет встроенный **планировщик** (scheduler), позволяющий добавлять отложенные **задания** (job) для будущего выполнения. Вы можете запланировать отправку контента после приветствия лида с ботом.

Также есть триггерная система, позволяющая создавать **триггеры**, которые автоматически выполняются при определенных событиях, например, при добавлении тега у лида.

Массовые рассылки сообщений называются **рассылки** (broadcast) и позволяют отправлять интерактивные скрипты выборке лидов. Таргетированная выборка осуществляется на JS, и если условие истинно, лид войдет в рассылку.

Все исходящие коммуникации добавляются в исходящую **очередь заданий** в планировщике cron scheduler, системном процессе платформы, отвечающем за обработку отложенных коммуникаций.

Этот раздел поможет вам понять, как использовать эти инструменты для создания динамичных и интерактивных чат-ботов, которые могут автоматически реагировать на различные события и взаимодействовать с пользователями в реальном времени.

Для разработки исходящих коммуникаций вам стоит овладеть следующими понятиями и функционалом:

- **Триггеры** (triggers) — программируемые события, которые приводят к выполнению нужного вам действия;
- **Броадкасты** (broadcasts) — массовые рассылки сообщений многим получателям одновременно;
- **Джобы** (jobs) — универсальный термин, означающий задачи или работы по отправке коммуникации.

Платформа поддерживает несколько способов запуска исходящих коммуникаций:

- **Запланировать вызов в коде JS** через объект bot с различными методами планирования:
  - Универсальный метод `bot.**scheduleJob**()` для планирования коммуникации;
  - Дополнительные методы на ваш выбор под те или иные случаи, например:
    - bot.**runScriptForLead**();
    - bot.**runTriggerForLead**();
    - bot.**runScriptByNameForLead**();
    - bot.**runTriggerByNameForLead**();
    - bot.**runScriptForPerson**();
    - bot.**runTriggerForPerson**();
    - bot.**runScriptByNameForPerson**();
    - bot.**runTriggerByNameForPerson**();

- **Создать триггер руками** в интерфейсе и задать условия автоматического выполнения запуска;
- **Создать и выполнить массовую рассылку** в интерфейсе платформы с JS условием для фильтрации лидов;
- **Обратиться к API бота** из внешних систем через API эндпоинты вашего бота;
- **Запланировать джобу** через API платформы, указав ID лида и объект, который хотите вызвать для этого лида, а также другие настройки.

Интерфейс может показаться не самым интуитивным, но он функциональный и максимально гибкий. С помощью отложенного запуска можно спроектировать любые растянутые во времени коммуникации.

> Подробную информацию по работе с триггерами с примерами кода смотрите в разделе **Триггеры**.

> Подробную информацию по работе с бродкастами смотрите в разделе **Рассылки**.

> Подробную информацию по работе с эндпоинтами смотрите в разделе **Конструктор API**.

Процессинг коммуникаций

Представьте, что в вашу компанию обратился не один клиент по имени Боб, а сразу несколько десятков Бобов. Каждый из них имеет свои вопросы, проблемы и запросы. Ваша задача — обслужить каждого Боба так, как будто он единственный. Именно так работает платформа Metabot.

На платформе Metabot в одном бизнесе могут одновременно работать несколько ботов, с которыми одновременно могут общаться большое количество пользователей. Бот — это особый класс программного обеспечения, построенного на асинхронной архитектуре. Следует отметить, что сами современные коммуникации стали асинхронными. Пользователь отправляет сообщение в мессенджер или чат на сайте, переключается на другую задачу в ожидании ответа.

Важно своевременно обслужить всех пользователей с минимальными задержками для каждого и доставить все сообщения в корректной последовательности. Происходит это благодаря асинхронной архитектуре платформы. Чтобы вы могли фокусироваться на решении прикладных и творческих задач, Metabot берет на себя все технически задачи по обработке входящего потока сообщений от пользователей, выполнения всех команды согласно логике вашего бота, учитывает состояния диалога, формирует и доставляет исходящие сообщения обратно в каналы, логирует выполнение для отладки.

Все входящие сообщения из канала принимаются сервером Metabot и сохраняются во **входящей очереди** (incoming queue). Очередь обрабатывается с помощью ядра (core), называемого также **процессингом коммуникаций** (communication processing) и после обработки исходящего сообщения отправляются обратно в канал к пользователю.

В процессинге коммуникаций одновременно участвует несколько **воркеров** (workers) — рабочих процессов, которые разбирают входящую очередь, обрабатывают сообщения, выполняют команды и отправляют исходящие коммуникации обратно в канал. На отправку сообщений выделена одна группа воркеров, на рассылки и триггеры — другая. Количество воркеров зависит от конфигурации сервера. Обычно их не более 10.

Разработка ядра — это наиболее сложная и критичная часть системы, за которую отвечают высококвалифицированные инженеры Metabot. Сейчас ядро — это закрытая разработка, но мы планируем сделать ядро расширяемым, чтобы сторонние разработчики могли адаптировать процессинг под свои нужды без страха нарушить его работу.

Важной особенностью платформы является то, что все процессы, запрограммированные в боте, гарантированно выполняются последовательно. Некоторые боты, которые мы встречали, созданные кустарным способом или на сторонних платформах реагировали на действия пользователей в произвольном порядке — достаточно было очень быстро выполнять команды в чат-боте. За счет процессинга коммуникаций на базе очередей, Metabot гарантирует точную последовательность выполнения команд в том порядке в котором пользователь или бот их инициирует.

Диалог осуществляется с помощью обмена **веб-хуками** (webhooks), которые каналы (например, мессенджеры) вызывают на платформе, когда пользователь взаимодействует с чат-ботом. В разделе Каналы мы показали, как устанавливаются веб-хуки.

Историю диалога (журнал сообщений) можно просмотреть через лид или контакт-центр. Также, администраторам доступен журнал веб-хуков и запросов к серверу Metabot.

Обработка естественного языка (NLP)

Отлично! Раз вы добрались до этой секции, значит теперь вы имеете хорошее представление об основных компонентах платформы Metabot и ее уникальной архитектуре. Уверен теперь вы понимает, как можно использовать Metabot и в каких задачах. Теперь давайте рассмотрим расширения платформы **обработкой естественного языка**, когда пользователь не путешествует по сценарию, а вводит текст в свободной форме.

Вы можете подумать, что можно использовать регулярные выражения (regex) в маршрутах для реакции на слова и даже фразы, но что делать, когда фраз и оборотов десятки и сотни, а также нужно учесть кучу синонимов и различных сленговых фраз? Регулярки однозначно не годятся. На помощь приходит нейросеть.

Обработка естественного языка (NLP, от англ. Natural Language Processing) — это область искусственного интеллекта, которая помогает компьютерам понимать, интерпретировать и использовать человеческий язык. В основе этого лежит идея о том, что наши компьютеры, программы и боты должны понимать нас так же легко, как мы понимаем друг друга.

В контексте чат-ботов, NLP помогает боту понимать ввод пользователя, даже если он не соответствует точным командам или фразам, которые были заранее запрограммированы. Например, если Боб спрашивает "Как мне отследить мой заказ?", бот, обученный с помощью NLP, поймет, что пользователь хочет узнать информацию о состоянии его заказа, даже если эта фраза не была заранее запрограммирована.

В Metabot есть готовая интеграция с Google **Dialogflow**. Dialogflow (сокращенно «DF») — это облачный сервис распознавания естественного языка от Google, который поддерживает различные языки, в том числе русский. У него есть бесплатные лимиты использования.

> Мы также планируем подключить DeepPavlov, Rasa и другие популярные NLP. Если у вас есть свой движок и вы хотели бы сотрудничать, напишите нам.

Чаще всего в реальных проектах NLP используется как инструмент навигации, чтобы направить пользователя в нужный нам сценарий диалога, либо в самом начале разговора, когда пользователь пришел к нам с проблемой или вопросом — задает вопрос сходу, либо когда в ходе разговора возник новый вопрос.

Также, с помощью NLP вы сможете извлекать **сущности** и **переменные** (entity extraction) из свободного ввода. **Сущности** — это справочники с набором значений, например, города, блюда, страны. Извлечение сущностей позволяет упростить пользовательский опыт и еще больше автоматизировать коммуникацию.

В качестве примера, вернемся к Бобу, который открыл чат с намерением проверить состояние своего заказа. Если бы в чате мы предусмотрительно не добавили быструю кнопку "Track my order", то вероятнее всего Боб спросил бы об этом в чате, например, написав одну из следующих формулировок:

- "Здравствуйте! Как мне отследить заказ?";
- "Где моя посылка? Трекинг номер ABC1-EFG7";
- "Мне два дня назад должен был прийти заказ, что случилось?";
- "Трекер".

Все эти вариации выражают одно и того же намерение получить информацию о заказе. Разные вариации вопроса на одну и ту же тему собираются в одну группу, которую называют **намерением** (intent).

Намерение (intent) — это разные вариации одного и того же запроса, в разных формулировках. Когда и зачем обычно люди пишут или звонят бизнесу? Когда и зачем вы это делаете лично? Разве не когда у вас возникла проблема, возник вопрос или вы хотите,

чтобы компания что-то сделала для вас? В любом случае, вы либо намереваетесь получить информацию от компании, либо намереваетесь сделать какое-то действие или намереваетесь побудить компанию на нужное вам действие.

У компании может быть от нескольких десятков до сотен намерений, в зависимости от сложности бизнеса, ассортимента услуг и продукции, количества целевых аудиторий. Намерения и формулировки могут меняться в течении года и с развитием компании.

Мы рекомендуем начинать внедрение чат-бота с того, что сначала внедрить каналы чата на сайт, мессенджеры и социальные сети. Внедрите каналы, научитесь или научите вашего клиента пользоваться ими и эффективно коммуницировать в них, накопите достаточное количество реальных диалогов с реальными пользователями. Затем выполните анализ диалогов, выделите популярные интенты и автоматизируйте их.

Вот как работает интеграция NLP на техническом уровне:

1. Чат-бот отправляет пользовательский ввод в NLP.
2. Получаем ответ в виде распознанного намерения и извлеченных сущностей. Ответ от NLP сохраняется в атрибутах лида и доступен для работы из любого скрипта в обычном режиме.
3. Чат-бот ищет скрипт с распознанным именем интента по точному совпадению имени интента или по маске. Если сценарий для намерения не найден, будет использоваться общий скрипт для всех интентов.
4. Если возникло исключение, будет использовать **Fallback** скрипт.
5. Остальное зависит от потребностей вашего проекта. Вы можете либо передать прямой ответ от NLP пользователю, запустить весь сценарий, либо ответить пользователю и снова передать управление NLP.

> Создание NLP ботов при помощи платформы Metabot и сервиса Dialogflow вы можете подробнее рассмотреть и опробовать на практике с помощью нашего урока **Автоматизируем службу поддержки**.

Переключение между чат-ботом и человеком

Представьте ситуацию: Боб, ваш клиент, общается с вашим чат-ботом, но вдруг столкнулся с сложным вопросом, который бот не может решить. В этом случае, бот может перевести разговор в службу поддержки, и тогда к диалогу подключится Алиса, оператор поддержки. Это позволяет обеспечить гибкость и эффективность обслуживания клиентов, сочетая автоматизацию и человеческое взаимодействие.

Чат-боты и автоматизация коммуникации — это замечательно, но полностью оставлять пользователей без заботы людей — это неправильно. Чат-боты могут выступать в роли

отличной автоматизации "нулевого" уровня поддержки, помогая с рутиной, но со сложными и эмоциональными проблемами и запросами может справиться только другой человек.

Не превращайте чат-бота в стену вокруг компании. Ведь вы бы сами хотели пользоваться услугами компании, с персоналом которой вы не можете поговорить, когда у вас возникнет сложная или нестандартная ситуация?

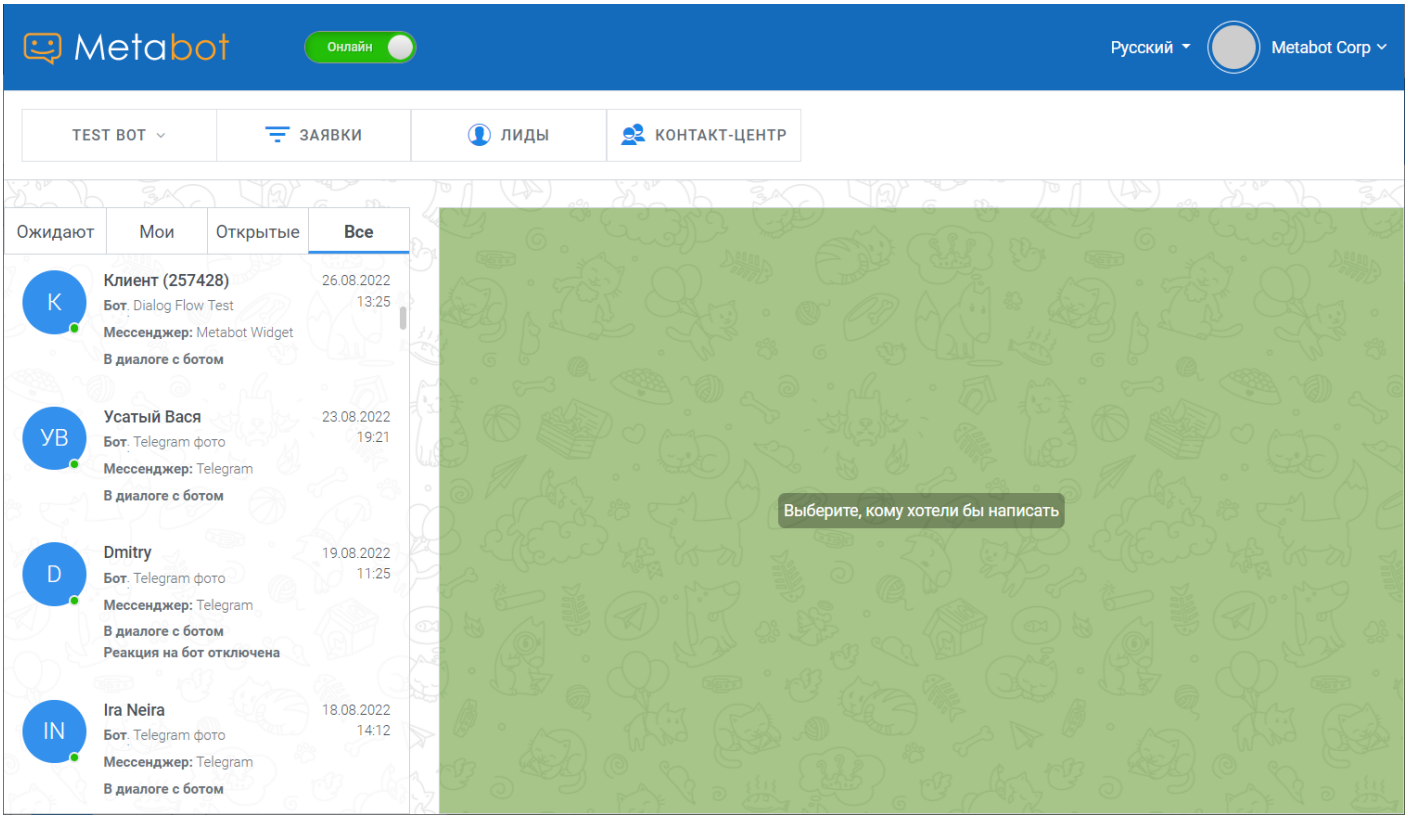
Для организации гибридной поддержки пользователей с помощью чат-бота и людей, используйте функцию перевода диалога на оператора в нужных точках диалоговых сценариев, которые называются **Перевести на оператора** и **Вернуть боту**.

Первая команда приостанавливает работу бота и переводит диалог в службу поддержки, а вторая возвращает бота обратно. Вторая команда нужна для тех случаев, когда пользователь позвал людей по ошибке, передумал и хочет вернуться обратно — вы можете создать маршрут для таких ситуаций, который будет слушать ввод пользователя и, когда он скажет, например, "бот", "меню" или "/start", вызовет скрипт, в котором будет команда возврата.

Для того чтобы люди могли принять участие в диалоге и обработать запрос, диалоги нужно куда-то передавать. Вы можете воспользоваться Контакт-центром Metadesk от Metabot или подключить стороннюю платформу, например, Livetex, Jivo, Webim, Umnico, Bitrix24, с которыми Metabot имеет готовую интеграцию.

> Если вам необходимо подключить чат-бот к другому рабочему месту оператора, напишите нам.

По завершению диалога платформа переведет диалог обратно чат-боту.



> Как организуется работа людей с автоматизациями в контакт-центре будущего и какими характеристиками должны обладать сотрудники, узнайте в **Контакт-центре завтрашнего дня**.

Пользовательские маршруты и аналитика

Представьте, что вы создали сложный лабиринт, в котором каждый коридор ведет к новому уровню взаимодействия с вашим бизнесом. Ваш чат-бот — это гид, который помогает пользователям найти правильный путь. Но как вы можете отследить, где именно находится каждый пользователь в этом лабиринте? Как вы можете определить, какие маршруты они выбирают и какие преграды встречают на своем пути? Вот где на помощь приходят **пользовательские маршруты и аналитика**.

Чат-боты — это мощный инструмент для гиперперсонализации. Они создают уникальный диалоговый интерфейс для каждого пользователя, адаптируясь под его потребности. Диалог постоянно течет и меняется, а диалоговые сценарии и цепочки отложенных исходящих сообщений позволяют вести каждого пользователя от одного сценария к другому автоматически, в том темпе, в котором удобно каждому пользователю.

Но если у нас есть бот со сложной системой коммуникаций, как мы можем узнать, где находятся наши пользователи в каждый момент времени? Или как нам узнать, по каким путям они двигались? Как нам оценить, насколько успешно мы спроектировали пользовательский опыт?

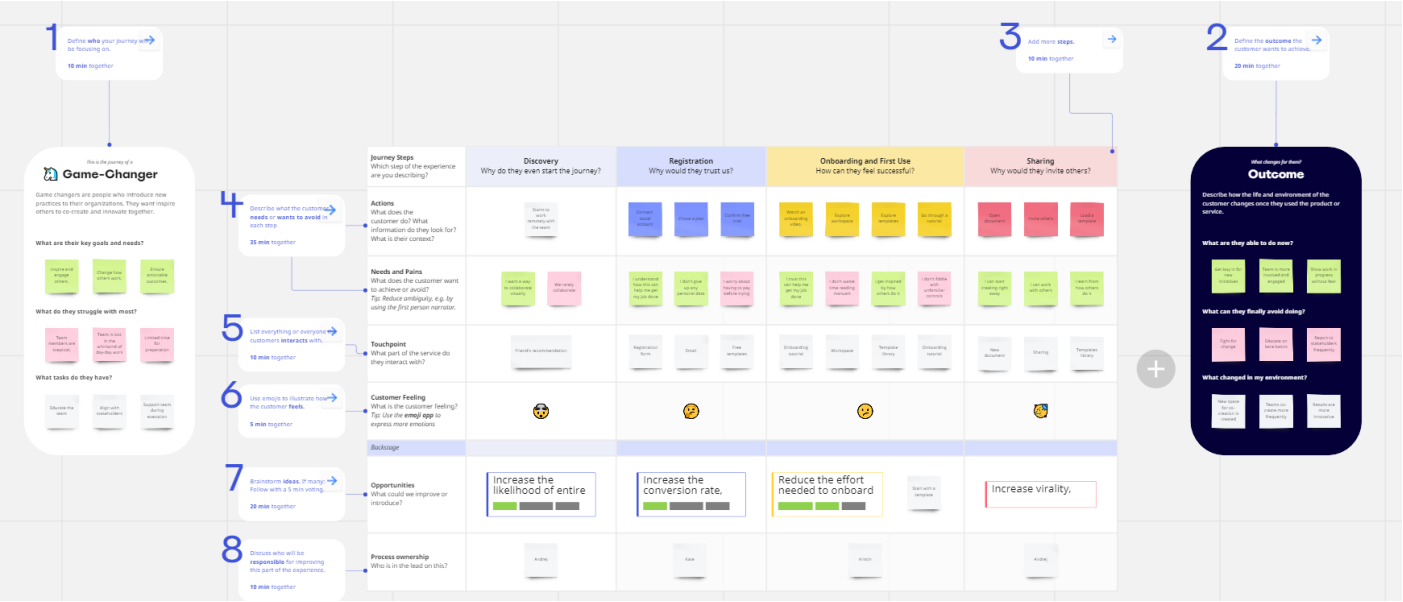
Для визуализации путей пользователей по вашему чат-боту вы можете воспользоваться дополнительным плагином — панелью аналитики **Dialog Journey** от команды Metabot. Этот плагин позволяет визуализировать пути пользователей по вашему чат-боту, а также позволяет фильтровать данные в таблицах по времени, пользователю, шагу и так далее.

Однако просто включить аналитику и все заработало не получится. Сначала вам нужно спроектировать сами пути для каждого класса персон (сегмента), путь которого вы хотели бы отслеживать.

Пути или как мы называем их — **путешествия** (journeys), состоят из **фаз/этапов** (phases/stages), которые в свою очередь состоят из **шагов** (steps).

Мы рекомендуем создать отдельную карту для планирования путешествия пользователей. Для этого вы можете воспользоваться популярным у маркетологов фреймворком, который называется **карты клиентских путей** (customer journey map или CJM). Это готовая и детально описанная методология анализа того, как клиент взаимодействует с бизнесом, которая помогает визуализировать путь клиента и потом командой работать над его улучшением.

Настоятельно рекомендуем освоить этот инструмент, потому что создание чат-ботов с помощью Metabot напрямую с ним связано. Когда вы внедряете чат-бот в бизнес, вы только тем и занимаетесь, что улучшаете клиентские пути. Ниже пример CJM карты, созданный Miro .



Обычно работа над чат-ботом начинается с вопросов заказчику:

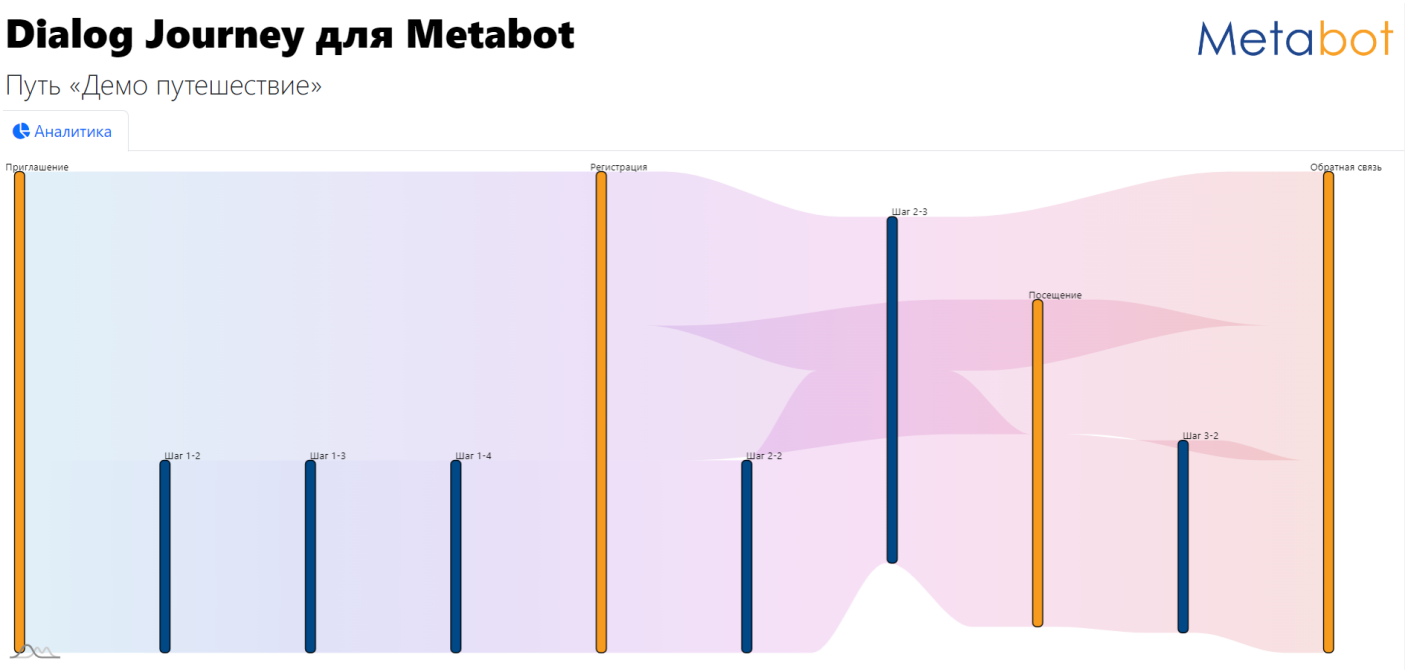
- Какие пути приносят 80% дохода?
- Какие пути приносят 80% проблем/жалоб?

Автоматизацию следует начинать с наиболее популярных путей. А чтобы улучшить пути, сперва их нужно определить — пройти по ним самостоятельно, собрать информацию о них от других и визуализировать в виде карты путей CJM. Тот путь, который проходят ваши пользователи по CJM, называется пользовательским опытом (user experience), о котором мы уже говорили выше.

Затем вы переходите к созданию чат-бота, который является ничем иным, как новым путем и опытом для пользователей. Двигаясь по пути, вы улучшаете шаг за шагом каждый этап пути и постепенно приходите к полноценному подключенному бесшовному персонализированному опыту со сквозной автоматизацией всего пути всех аудиторий бизнеса. Понимаете теперь, что такое чат-боты на самом деле и почему мы назвали их диалоговыми приложениями?

Итак, когда вы спроектировали новые пути и занялись их разработкой, теперь в коде вы должны сигнализировать аналитике Dialog Journey о том, что пользователь достиг нового шага.

Для этого вы вызываете код, который логирует шаг. Данные о передвижении пишутся в кастомные таблицы вашего бота, на основе этих данных строится аналитика.



Плагины и организация рабочего пространства

Представьте, что вы строите замок из конструктора. Каждый блок, каждый элемент имеет свое место и функцию. Вместе они создают единую структуру. Так же и с чат-ботами. Они состоят из множества различных элементов, каждый из которых выполняет свою роль. И чтобы не потеряться в этом многообразии, важно правильно организовать свое рабочее пространство.

В прошлой главе мы говорили о том, что чат-бот можно разбить на отдельные части, каждая из которых отвечает за определенный этап взаимодействия с пользователем. Таким образом, чат-бот можно представить как набор мини-ботов или мини-программ, каждая из которых решает свою задачу.

Например, на этапе знакомства с пользователем могут быть использованы сценарии, которые представляют компанию, собирают информацию о пользователе и сегментируют его. Затем может следовать этап регистрации или авторизации, создания учетной записи во внешней системе. Далее — этап выбора товара или услуги, формирование корзины покупок, оплата. И так далее.

Для того, чтобы не запутаться в этом многообразии, мы рекомендуем следовать следующим принципам:

- **Применяйте принципы абстракции, компонентного подхода и повторного использования ваших решений.** Это поможет вам быстро и эффективно масштабировать свои решения и использовать их в разных проектах;
- **Группируйте связанные общей целью скрипты в папки.** Это поможет вам легче ориентироваться в структуре вашего проекта;
- **Понятно и интуитивно называйте скрипты, триггеры, эндпоинты и другие объекты**, чтобы можно было легко понять их назначение;
- **Введите стандарт в наименование объектов и кодовых имен**. Например, для атрибутов можно использовать camelCase или snake_case;
- **Введите стандарт оформления кода.** Это поможет вам и вашей команде быстрее и легче понимать код друг друга;
- **Изучите, как создаются плагины и сниппеты**, чтобы размещать в них JS код, который можно подключать в скрипты и другие места в ваших ботах.

Плагины — это расширения платформы Metabot, создаваемые с помощью языков JS или PHP для повторного и многократного использования в разных местах в одном и том же боте или в абсолютно разных ботах.

> Подробнее о плагинах и сниппетах смотрите в разделе Плагины.

ChatGPT и генеративные алгоритмы

Ранее мы говорили о технологии обработки естественного языка (NLP), которая позволяет чат-ботам понимать и интерпретировать ввод пользователя. Однако, несмотря на все

преимущества NLP, она имеет свои ограничения. NLP основана на предопределенных шаблонах и правилах, что может привести к ограниченности в ответах и неполному пониманию контекста.

Генеративные алгоритмы, такие как ChatGPT, представляют собой значительный шаг вперед. Они не просто понимают ввод пользователя, но и способны генерировать подходящие ответы, обучаясь на больших объемах данных. Это позволяет создавать более естественные и нелинейные диалоги.

Представьте, что Боб, ваш пользователь, вступает в диалог с вашим чат-ботом. Он задает вопрос, на который нет готового ответа в вашем сценарии. Вместо того чтобы просто сказать "Извините, я не понимаю", ваш чат-бот использует генеративные алгоритмы искусственного интеллекта, чтобы создать подходящий ответ, основанный на обучении на большом объеме данных. Боб получает полезный ответ, и диалог продолжается гладко.

Однако стоит помнить, что генеративные алгоритмы не всегда являются лучшим решением. Они должны использоваться там, где они могут быть наиболее полезны. Кроме того, за сообщения, которые эти алгоритмы "генерируют", может последовать юридическая ответственность для вас или ваших клиентов.

> Если вам нужны нелинейные диалоги или экспертные ИИ-системы для консультирования, воспользуйтесь возможностями ChatGPT на платформе Metabot. Это мощный инструмент, который поможет вам создать более глубокие и персонализированные взаимодействия с вашими пользователями.

Личность бренда: Создание уникального образа в цифровом мире

Вспомните нашего Боба, который взаимодействовал с чат-ботом. Бот помог ему совершить покупку, ответил на все вопросы и даже предложил дополнительные товары. Но что если этот бот мог бы не просто отвечать на вопросы, но и общаться с Бобом, как настоящий сотрудник компании? Что если бы он имел свою уникальную "личность", которая отражала бы ценности и качества бренда?

Технологический прогресс за последние 50-70 лет привел к тому, что интерфейсы компьютера стали все более естественными и привычными для людей. Мы перешли от больших мейнфреймов и перфокарт к персональным компьютерам, мобильным устройствам и сенсорным экранам. Сегодня мы активно осваиваем разговорные интерфейсы, которые являются самыми естественными и привычными для нас.

Среда, в которой мы живем и взаимодействуем, также меняется с развитием технологий. Устное слово сменили книги и печатный пресс, затем пришло радио и телевидение, компьютеры, интернет, видеокамеры и мобильные устройства. Сегодня мы уже говорим о

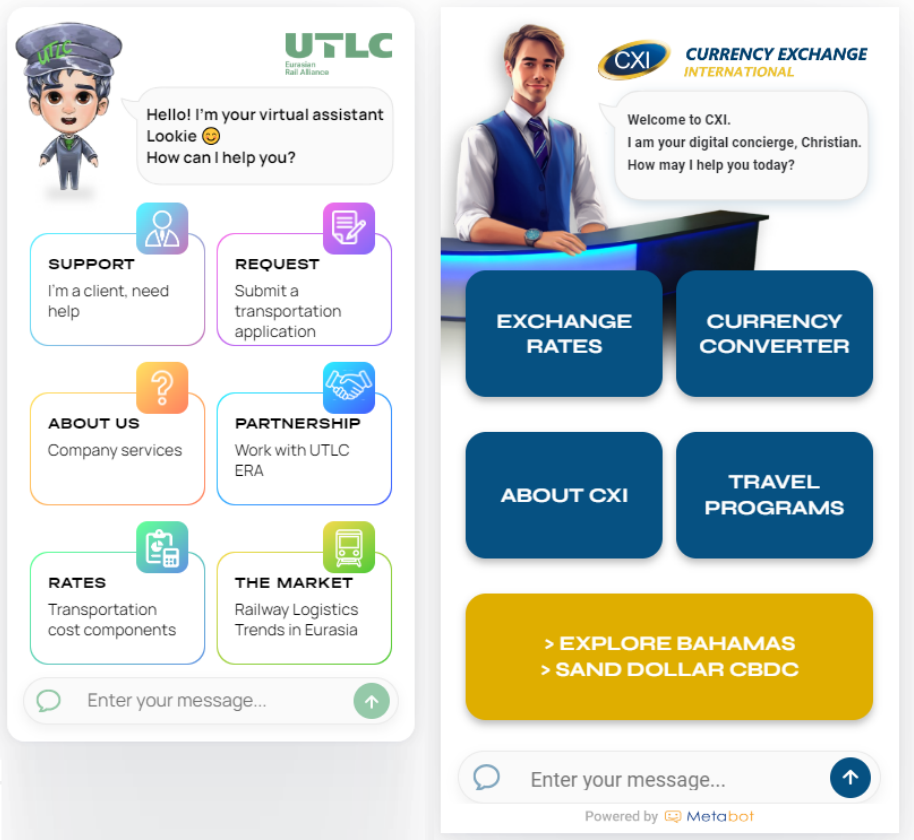
метавселенных, виртуальной и дополненной реальности, умных колонках, умных автомобилях и умных домах.

Смена парадигмы коммуникации неизбежно приводит к тому, что бизнес стремится к новым способам коммуникации и платформам. Сегодня никого уже не удивишь пабликом бренда в социальной сети или чат-ботом на сайте компании.

Однако, чтобы идти в ногу со временем и соответствовать современной парадигме коммуникаций, бизнесу необходимо не только использовать новые технологии, но и создавать уникальную "личность" бренда или компании в виде аватара. Этот аватар должен отражать ценности и качества целевой аудитории, пользователей услугами и продукции бренда или компании.

Создание "личности" бренда включает в себя разработку уникального портрета, определение тона голоса для текстов, характера общения и многого другого. Например, как вы хотите, чтобы бот обращался к вашей аудитории: на "Вы" или "ты"?

Metabot предлагает уникальные возможности для создания брендированного чат-виджета для вашего сайта, который будет отражать личность вашего бренда. Мы также предлагаем воспользоваться нашим фреймворком по проектированию цифровой личности, который поможет вам создать уникальный и запоминающийся образ вашего бренда в цифровом мире.



Кроме того, мы рекомендуем вам подумать о создании уникального голосового ассистента, который будет взаимодействовать с вашими клиентами. Это может быть особенно полезно для умных колонок и умных домов, где взаимодействие происходит в основном через голос.

> Для создания личности бренда воспользуйтесь нашим уроком по ссылке: [URL]

Что дальше?

Вау! Вы добрались до этого места, и за это мы вас приветствуем. Добро пожаловать в захватывающий новый мир чат-ботов! Мы ценим ваше время и усилия, потраченные на ознакомление с этим материалом.

Итак, что дальше? Вот несколько шагов, которые вы можете предпринять:

- **Начать практические уроки** — теперь, когда у вас есть теоретические знания, пришло время применить их на практике;
- **Скачать готовые шаблоны и примеры кода** — используйте предоставленные уроки в качестве руководства и воспользуйтесь готовыми шаблонами и примерами кода для ускорения процесса обучения;
- **Изучить базовые и продвинутые функции** — чтобы углубить свое понимание, изучите более подробно базовые и продвинутые функции Metabot, перейдя по предоставленным выше ссылкам;
- **Записаться на консультацию** — если вам нужна дополнительная помощь или у вас есть конкретные вопросы, не стесняйтесь записаться на консультацию с нами. Наши эксперты готовы помочь вам справиться с любыми проблемами, с которыми вы можете столкнуться;
- **Заказать решение «под ключ»** — если вы предпочитаете более отстраненный подход, мы предлагаем решения «под ключ», адаптированные под ваши конкретные потребности. Позвольте нам заниматься техническими аспектами, пока вы сосредоточены на своем бизнесе.

Помните, путь к освоению технологии чат-ботов — это марафон, а не спринт. Не торопитесь, практикуйтесь и не стесняйтесь просить помощи, когда это необходимо. Мы здесь, чтобы поддержать вас на каждом этапе. Удачи вам и приятного создания ботов!

Версия #79
Artem Garashko создал 2023-02-12 13:49:54 UTC
Ирина Петрова обновил 2025-11-11 07:22:56 UTC