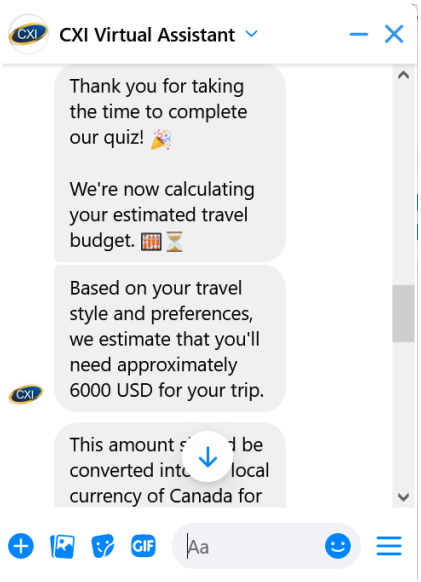
Урок 9: Создание интеллектуального квиза для путешественников — интеграция Manychat с Metabot

В этой статье мы рассмотрим, как интегрировать Manychat, отличный инструмент для построения воронок, с платформой Metabot в качестве Low-code инструмента для бэкенд-разработки. Manychat обладает удобным интерфейсом для работы с пользователями, но не поддерживает написание кода. Metabot идеально дополняет Manychat, предлагая возможности быстрой разработки и легкой интеграции с чат-ботами.



В этом уроке мы разработаем интерактивный квиз для путешественников. На протяжении квиза участники будут отвечать на вопросы, касающиеся характера их путешествия, а также характера затрат. Эта информация поможет нам оценить, сколько денег им стоит взять с собой в поездку.

В конце квиза, основываясь на предоставленных данных, наша система предоставит участникам приблизительную оценку требуемого бюджета для их путешествия. Это будет сделано с помощью специально разработанного алгоритма, который учитывает различные факторы, такие как длительность пребывания, количество путешественников, их предпочтения в расходах, сезонность и цель поездки.

Ниже представлен пример кода, который мы будем использовать для расчета бюджета. Этот код можно адаптировать под ваши задачи, будь то создание собственных калькуляторов расходов или оценка бюджета для различных мероприятий. Воспользуйтесь этим примером как отправной точкой для разработки собственных инструментов.

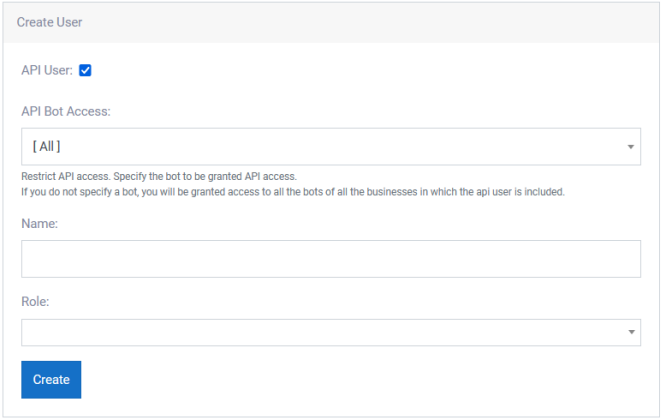
Видео-инструкция по интеграции ManyChat и Metabot

6a106086f3134bfa9d5703ffe9e5f2a7-1706838793059-with-play.gif

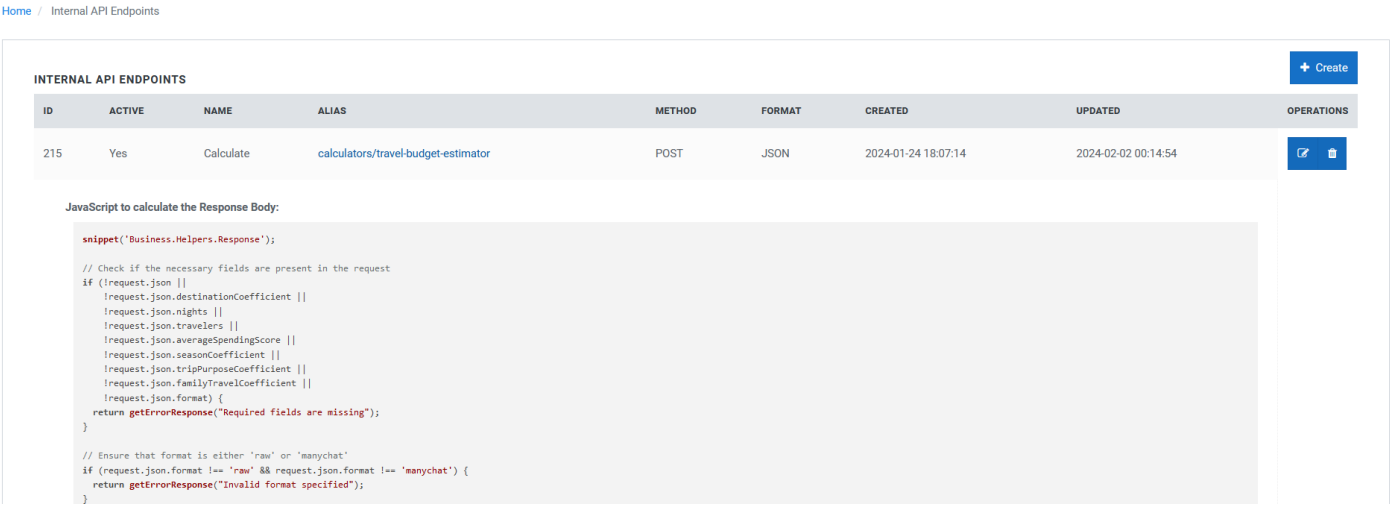
Обзор интеграции

1. Сначала нужно создать бота в системе Metabot.

2. Для авторизации запросов необходимо создать пользователя с типом API и сгенерировать токен.



3. В боте нужно создать endpoint, который будет обрабатывать запросы от Manychat.



4. Затем нужно написать код для обработки запросов. Пример кода показывает, как обработать запросы и вернуть результат в формате, подходящем для Manychat.

Пример кода эндпоинта

```
snippet('Business.Helpers.Response');

// Check if the necessary fields are present in the request
if (!request.json ||
    !request.json.destinationCoefficient ||
    !request.json.nights ||
    !request.json.travelers ||
    !request.json.averageSpendingScore ||
    !request.json.seasonCoefficient ||
    !request.json.tripPurposeCoefficient ||
    !request.json.familyTravelCoefficient ||
    !request.json.format) {
  return getErrorResponse("Required fields are missing");
}

// Ensure that format is either 'raw' or 'manychat'
if (request.json.format !== 'raw' && request.json.format !== 'manychat') {
  return getErrorResponse("Invalid format specified");
}

// Extract values from the request
const destinationCoefficient = request.json.destinationCoefficient;
const seasonCoefficient = request.json.seasonCoefficient;
```

```
const tripPurposeCoefficient = request.json.tripPurposeCoefficient;
const familyTravelCoefficient = request.json.familyTravelCoefficient;

// Base daily cost based on average spending score (example values)
let baseDailyCost;
if (request.json.averageSpendingScore <= 1.75) {
  baseDailyCost = 50; // Low spender
} else if (request.json.averageSpendingScore <= 2.5) {
  baseDailyCost = 100; // Moderate spender
} else if (request.json.averageSpendingScore <= 3.25) {
  baseDailyCost = 150; // High spender
} else {
  baseDailyCost = 200; // Ultra spender
}

// Calculating the total recommended amount
const totalAmount = (baseDailyCost * request.json.destinationCoefficient * request.json.nights
* request.json.travelers) *
                    seasonCoefficient *
                    tripPurposeCoefficient *
                    familyTravelCoefficient;

// Determine the response format
if (request.json.format === 'raw') {
  // Returning the calculated budget in raw format
  return getSuccessResponseWithJson({
    calculatedBudget: totalAmount
  });
} else {
  // Returning the calculated budget in Manychat format
  return {
    "version": "v2",
    "content": {
  /*    "messages": [
        {
          "type": "text",
          "text": `Ok! I got it`
        }
        // Additional messages can be added here
      ],*/
```

```
      "actions": [
        {
          "action": "set_field_value",
          "field_name": "budget_estimate_amount", // Custom field name in Manychat
          "value": totalAmount.toString()  // Setting the calculated budget as the value
        }
      ],
      "quick_replies": []
    }
  };
}
```

Обратите внимание, что мы возвращаем ответ в двух форматах:

- **Raw Format** — простой JSON. Для интеграции с Manychat он вам не понадобится;
- **Manychat Format** — формат, подходящий для динамических сообщений Manychat.

Код плагина Business.Helpers.Response

Плагин **Business.Helpers.Response** служит утилитой для стандартизации ответов API в бизнес-приложениях. Его основная цель - упростить создание последовательных и структурированных ответов для различных сценариев, возникающих во время взаимодействия с API.

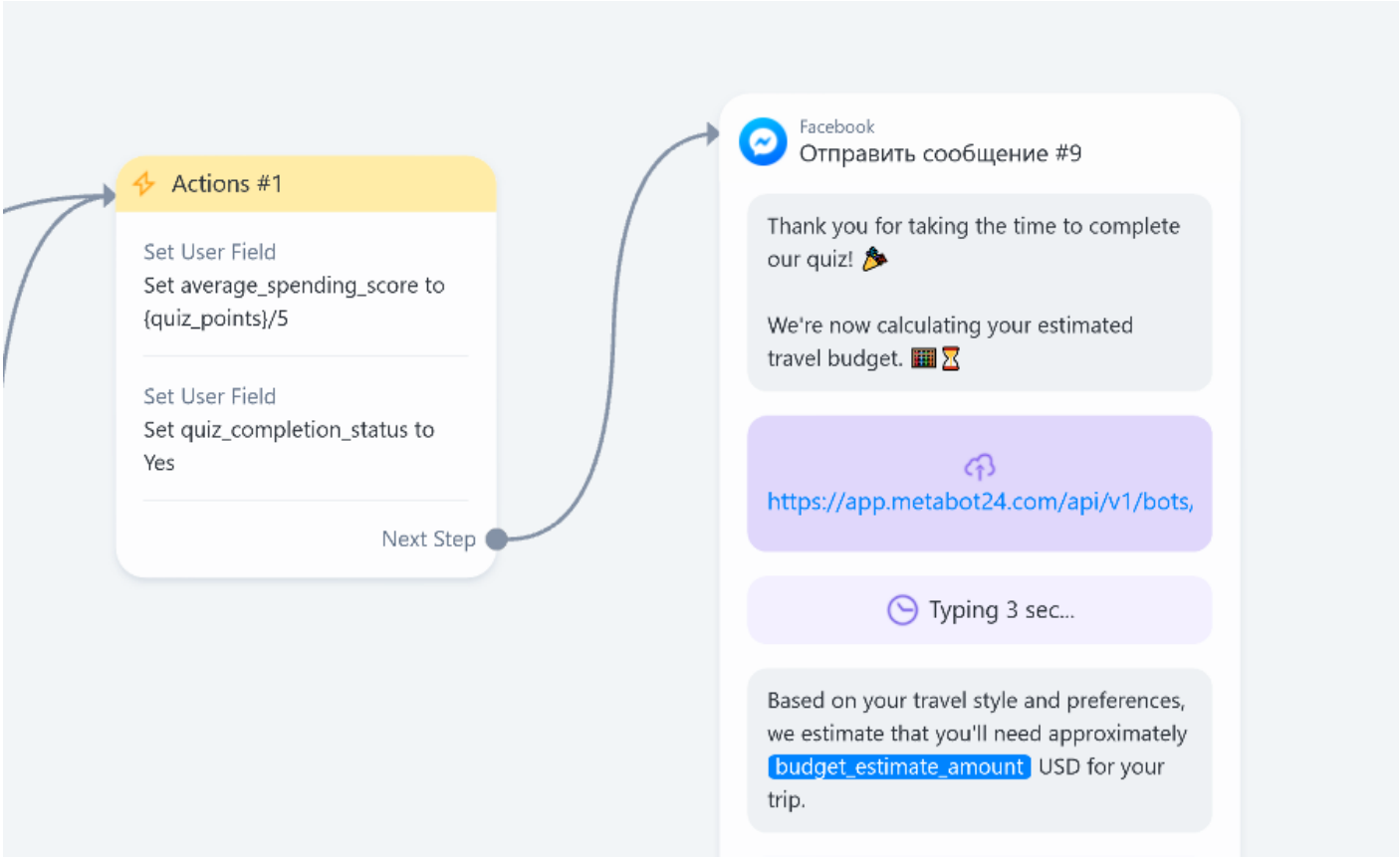
Код плагина доступен в Библиотеке плагинов. Создайте плагин с таким же названием в вашем бизнесе на платформе Metabot, скопировав код плагина.

Интеграция с ManyChat

1. В Manychat реализуйте команду для динамического формирования контента, указав URL endpoint MetaBot.

Подробнее про эту команду:

https://support.manychat.com/support/solutions/articles/36000143342-dev-tools-dynamic-block

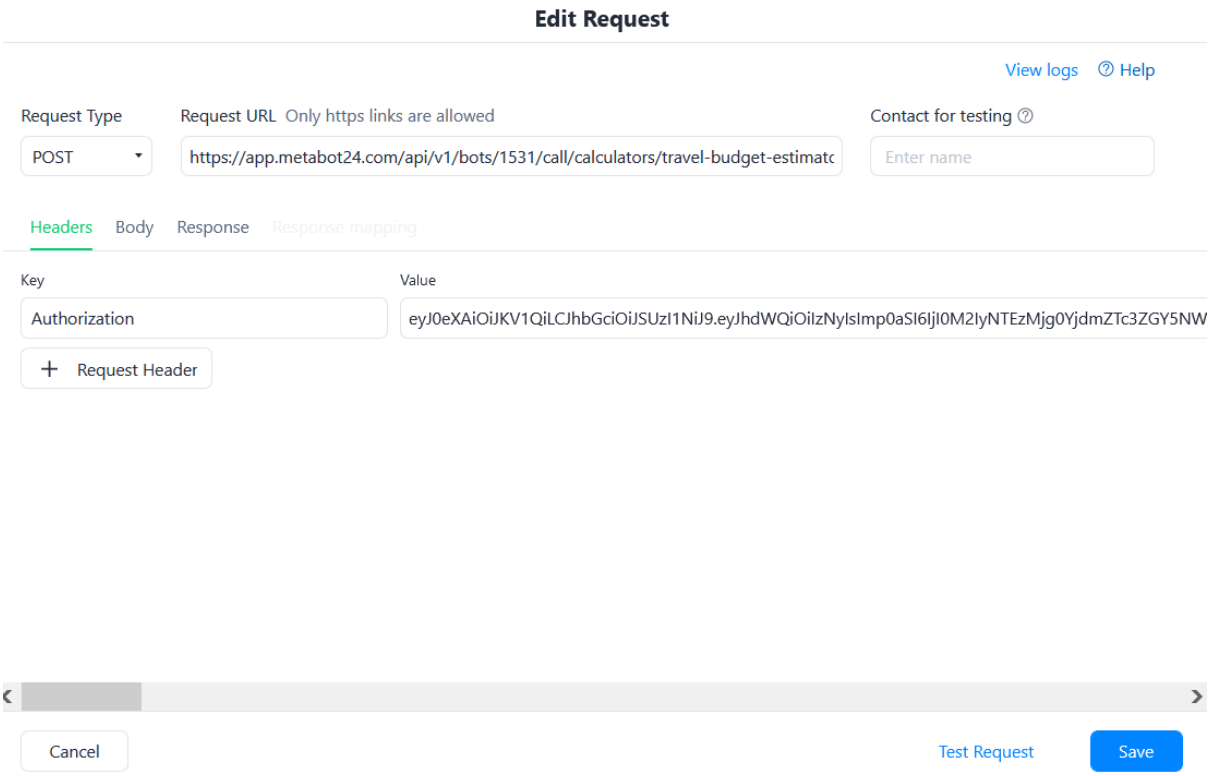


2. В Header запроса укажите токен авторизации, а в Body — параметры из воронки Manychat.

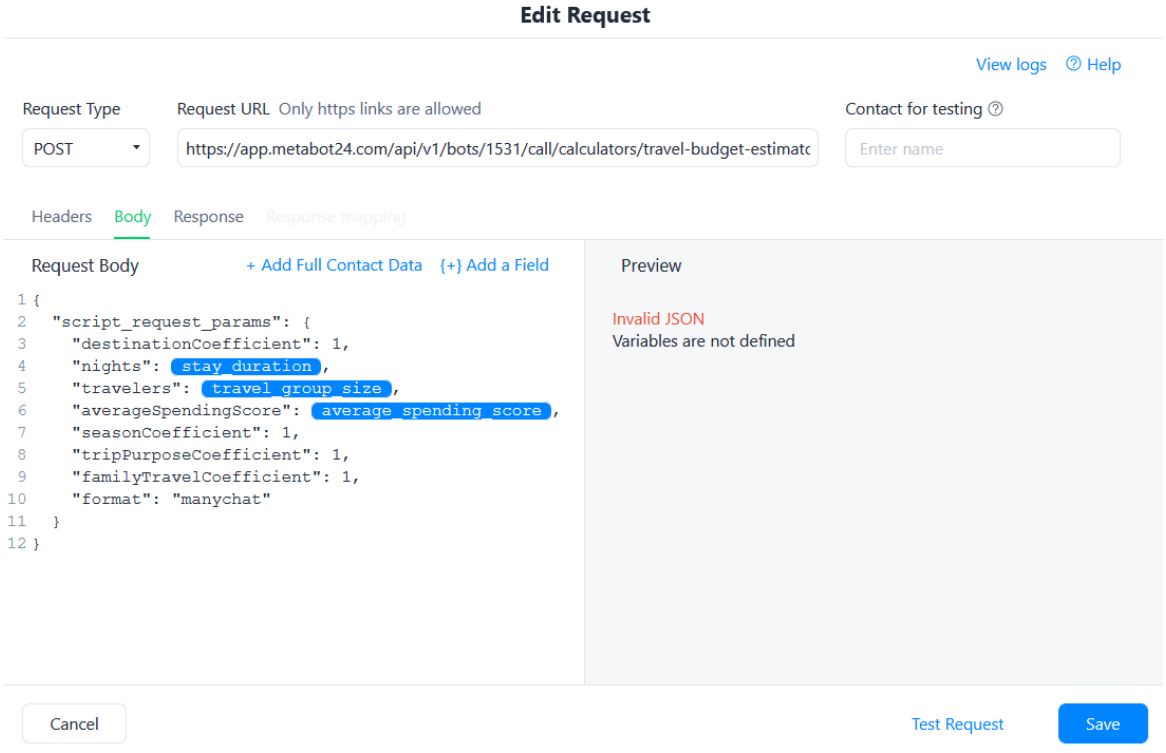
Подробнее про формат обмена REST API: https://manychat.github.io/dynamic_block_docs/

Metabot API Endpoint URL: https://app.metabot24.com/api/v1/bots/{BOT-ID}/call/{ENDPOINT-ADDRESS}

3. Настройте заголовок запроса:



4. Настройте тело запроса:



Пример тела запроса

```
{
  "script_request_params": {
```

```
    "destinationCoefficient": 1,
    "nights": 5,
    "travelers": 1,
    "averageSpendingScore": 2.25,
    "seasonCoefficient": 1.0,
    "tripPurposeCoefficient": 1.0,
    "familyTravelCoefficient": 1.0,
    "format": "manychat"
  }
}
```

Пример тела ответа

```
{
    "version": "v2",
    "content": {
        "actions": [
            {
                "action": "set_field_value",
                "field_name": "budget_estimate_amount",
                "value": "500"
            }
        ],
        "quick_replies": []
    }
}
```

> Всегда оборачивайте все запросы к эндпоинтам вашего бота на платформы Metabot в "script_request_params"!

> Интеграция Manychat и Metabot открывает новые возможности для создания продвинутых чат-ботов. Благодаря Low-Code подходу Metabot и удобному интерфейсу Manychat, можно быстро разрабатывать и интегрировать сложные функциональности, улучшая взаимодействие с пользователями.

Версия #22
Artem Garashko создал 2024-02-02 01:20:08 UTC
Ирина Петрова обновил 2024-03-20 08:58:26 UTC