Web App форма с использованием keyboard-кнопки (только для Telegram)

> Инструкция для разработчика бота.

> Данный вид формы работает только в Telegram.

> Ключевая особенность: форма открывается в всплывающем окне на котором расположен WebView (встроенный браузер). Для отправки данных в Metabot API не требуется дополнительный backend.

> Актуальный исходный код веб-формы реализующий все три вида форм смотрите по ссылке: go-to-the-mars.html

> Пример работы веб-формы приведенной выше смотрите в Telegram боте https://t.me/metabot_test_form_bot

Отличие данного варианта от универсальной формы в виде ссылки, в том, что форма работает как Web App, т.е. это специальный режиме браузера встроенного в Telegram. Этот режим гарантирует возврат в бота после заполнения формы, а также не позволит открыть ссылку во внешнем браузере (а также включает дополнительный функционал для взаимодействия веб-страницы с ботом). В обычном же режиме для универсальной формы, нет гарантий, что когда веб-страница будет закрыта, пользователя перекинет в бота.

Краткое описание принципа работы бота:

1. В бот отправляется keyboard-кнопка (в формате Telegram Web App). При нажатии на кнопку открывается всплывающее окно, в которое встроен Web View, пользователь не может открыть эту ссылку во внешнем браузере, а также просмотреть исходный код формы (если только не откроет сам Telegram в браузере).
2. После заполнения формы и нажатия на кнопку Отправить данные, все данные будут отправлены в Metabot, в команду JavaSctript Callback, Internal API endpoint в данном случае не требуется. Для понимания смотрите исходный код веб-формы.
3. Данные пришедшие с формы доступны в команде JavaSctript Callback в виде обычного вебхука. Необходимо сохранить их в таблице.
4. Далее необходимо вызвать скрипт бота для продолжения беседы, этот скрипт работает с сохраненными данными формы, запрашивает подтверждение, что все введено верно, а также предлагает заполнить форму повторно.

Принцип работы, по шагам:

- Бот генерирует ссылку с уникальным хэш-кодом, привязанным к лиду.
- Бот отправляет эту ссылку в виде inline-кнопки в Telegram-мессенджер.
- Пользователь бота нажимает на кнопку, открывается Web App с формой (т.е. пользователь остается в Telegram, в мессенджере открывается встроенный браузер).
- Пользователь бота заполняет форму и нажимает кнопку для отправки формы.
- Форма отправляет данные на backend сайта, где размещена форма.
- Backend сайта отправляет данные формы в API Metabot, в данные должен быть включен уникальный хэш лида.
- Metabot принимает и сохраняет данные заполненной формы.
- Если все ок, то страница с формой в браузере должна быть закрыта, это делается с помощью простого JS кода (подробности рассматриваются на отдельной странице документации с описанием создания HTML-формы).

Таблицы для работы с формами

Чтобы рассматриваемый пример бота с формами работал, вам необходимо создать две кастомные таблицы.

Таблица: Хэш-коды лидов

> Таблица необходима для хранения соответствий между лидом и хэшом для лида.

При каждом вызове формы выполняется поиск хэша по лиду, если хэш не найден, то формируется новый. После отправки формы, запись с хэш-кодом удаляется, чтобы ссылка на каждый новый опрос содержала уникальный хэш. Поиск предыдущего хэша необходим, если есть формы, ожидающие заполнения, для случаев, если лид закрыл форму, а затем спустя время время решил ее заполнить, нажав на туже кнопку вызова формы. При этом в

таблицу заложено поле (expire_at) - дата истечения хэш-кода, вы можете использовать это поле для реализации времени жизни хэш-кода, но в таком случае в веб-форме необходимо предусмотреть уведомление о том, что форма истекла, или закрывать ее автоматом после истечения, или формировать в боте новую ссылку, если лид пытается отправить форму, хэш которой истек. Это требование не обязательно, может реализовываться по мере необходимости и зависит от конкретной задачи.

Структура таблицы:

ID	АКТИВНА	СИХР. СТРУКТУРЫ	ОТОБРАЖАТЬ В МЕНЮ	НАИМЕНОВАНИЕ	ЗАГОЛОВОК В ИНТЕРФЕЙСЕ
69	Да	Да	Да	lead_form_hash	Leads Hashes for Forms

Структура таблицы (JSON словарь):

АКТИВНО	СИХР. СТРУКТУРЫ	НАИМЕНОВАНИЕ	ЗАГОЛОВОК	ТИП	ОБЯЗАТЕЛЬНОЕ	ЗНАЧЕНИЕ ПО УМОЛЧАНИЮ	РАЗМЕР	ТОЧНОСТЬ	ПОДСКАЗКА	ВСПЛЫВАЮЩАЯ ПОДСКАЗКА
Да	Да	id	ID	AUTOINCREMENT	Да	NULL				
Да	Да	form	Form	TEXT	Да					
Да	Да	lead_id	Lead ID	BIG_INT	Да	NULL				
Да	Да	hash	Hash	TEXT	Да					
Да	Да	expire_at	Expire At	DATETIME		NULL				
Да	Да	created_at	Created At	CREATED_AT	Да	NOW				

Таблица: Данные форм

> Таблица необходима для хранения данных заполненных форм.

Данная таблица реализована как универсальная, все данные пришедшие с формы сохраняются в текстовом поле в виде JSON, поэтому она может работать с любой формой, но работать с строкой в виде JSON в JS неудобно, тк для доступа к данным приходится выполнять JSON.parse(), вы можете нормализовать данные, реализуя для каждой формы бота отдельную таблицу с необходимыми полями для хранения, например такими как: Имя, Фамилия, Адрес и т.д., чтобы в дальнейшем было проще работать с данными и выполнять поиск по таблице. Также, в вашем случае таблица для хранения данных может и не потребоваться, если всплывающая форма очень простая и предназначена, например, для заполнения одного поля, например, адреса с авто комплитом от Dadata или для выбора выпадающего списка или даты в календаре. Т.е. форма может реализовывать даже один компонент для заполнения которого необходим HTML + JS.

Структура таблицы:

ID	АКТИВНА	СИХР. СТРУКТУРЫ	ОТОБРАЖАТЬ В МЕНЮ	НАИМЕНОВАНИЕ	ЗАГОЛОВОК В ИНТЕРФЕЙСЕ
70	Да	Да	Да	form_results	Form Results

Структура таблицы (JSON словарь):

АКТИВНО	СИХР. СТРУКТУРЫ	НАИМЕНОВАНИЕ	ЗАГОЛОВОК	ТИП	ОБЯЗАТЕЛЬНОЕ	ЗНАЧЕНИЕ ПО УМОЛЧАНИЮ	РАЗМЕР	ТОЧНОСТЬ	ПОДСКАЗКА	ВСПЛЫВАЮЩАЯ ПОДСКАЗКА
Да	Да	id	ID	AUTOINCREMENT	Да	NULL				
Да	Да	form	Form	TEXT	Да					
Да	Да	lead_id	Lead ID	BIG_INT	Да	NULL				
Да	Да	data	Data	TEXT						
Да	Да	created_at	Created At	CREATED_AT	Да	NOW				
Да	Да	updated_at	Updated At	UPDATED_AT		NULL				

Чтобы посмотреть исходный код внутреннего API, сначала перейдите в бота по ссылке на любой из скриптов.

Нажмите в предупреждающем сообщении на ссылку для смены активного бота, а затем перейдите на страницу с внутренним API в настройках бота или по ссылке.

Если у вас нет доступа к указанному скрипту, то запросите шаблон бота в технической поддерже Metabot.

Скрипт для формирования ссылки и отправки кнопки в мессенджер

Пример скрипта.

Скрипт отправки ссылки состоит из единственной команды бота:

- **Выполнить JavaScript Callback**:

КОМАНДЫ

Эти команды бот выполняет до того как покажет меню.

КОМАНДА	СОДЕРЖИМОЕ	ОПЕРАЦИИ
Выполнить JavaScript Callback	**Запустить Callback сразу при старте этой команды:** Да **JavaScript Callback:** `// Код скрипта`	

```
const menuHelper = require('Common.TelegramComponents.MenuHelper')
let md5 = require('Common.Utils.Md5')

//
-------------------------------------------------------------------------------------------------
-----
// Инициализация компонента
if (menuHelper.isFirstImmediateCall()) {
  // !!! ИНИЦИАЛИЗАЦИЯ !!!

  // Скрываем предыдущее меню с кнопкой формы
  menuHelper.removeInlineKeyboard()

  // Сбрасываем ID последнего сообщения
  menuHelper.clearLastTelegramMessageId()

  // -------------------------
  //ГЕНЕРИРУМ ССЫЛКУ С ХЭШЕМ
  let expireAt = new Date()
  expireAt.setSeconds(expireAt.getSeconds() + 300)

  let hashItems = table.find('lead_form_hash', [], [
    ['form', '=', 'mars'],
    ['lead_id', '=', leadId],
  ])

  // Todo: Можно вынести в сниппет
  let leadHash
  if (hashItems.length > 0) {
```

```
    leadHash = hashItems[0].hash
  } else {
    const salt = 'YourSuperSecretString' + (new Date()).getTime()
    leadHash = md5(salt + leadId)

    table.createItem('lead_form_hash', {
      'form': 'mars',
      'lead_id': leadId,
      'hash': leadHash,
      'expireAt': expireAt
    })
  }

  // Вы можете вынести эту ссылку в атрибуты бота
  let formUrl = 'https://app.metabot24.com/testWidgets/go-to-the-
mars.html?mode=tg_keyboard&q=' + leadHash
  // ---------------------------

  // Отправляем кнопку с формой при инициализации
  let message = 'Для продолжения, пожалуйста заполните форму'
  let buttons = [
    [
      {
        "text": "Заполнить форму",
        "web_app": {
          "url": formUrl
        }
      }
    ]
  ]
  let apiAdditionalParams = {
    "reply_markup": {
      "keyboard": buttons,
      "resize_keyboard": true // Для того чтобы кнопка не была слишком большой по высоте
    }
  }
  menuHelper.sendMessage(message, null, null, apiAdditionalParams)

  return false
}
```

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

//
-----------------------------------------------------------------------------------------------
-----
// ЛОГИКА КОМПОНЕНТА
//
-----------------------------------------------------------------------------------------------
-----

//
-----------------------------------------------------------------------------------------------
-----
// Обработка нажатых кнопок или входящего текста

let webhook = bot.getWebhookPayload()
let webAppData = null
let data = null
let rawJsonData = null

if (typeof webhook.message == 'object') {
  if (typeof webhook.message.web_app_data == 'object') {
    webAppData = webhook.message.web_app_data

    //Example:
    //"web_app_data": {
    //  "button_text": "Заполнить форму",
    //    "data": "{...}"
    //}

    if (typeof webAppData.data === 'string') {
      rawJsonData = webAppData.data
      data = JSON.parse(rawJsonData)
      if (typeof data !== 'object') {
        rawJsonData = null
        data = null
      }
    }
```

```
  }
}

// --------------------------------
if (data !== null && typeof data === 'object' && typeof data.q === 'string') {
  // СОХРАНЕНИЕ ДАННЫХ
  // КОД ИДЕНТИЧНЫЙ API INTERNAL ENDPOINT
  // TODO: МОЖНО ВЫНЕСТИ В СНИППЕТ

  // Доп защита
  // Все равно проверяем хэш (на случай чтобы не взломали)
  // А также чтобы почистить формы ожидающие заполнения по лиду (хэши по лиду)
  // Можно не выполнять эту проверку здесь
  // Todo: Можно вынести в сниппет
  let hashItems = table.find('lead_form_hash', [], [
    ['form', '=', 'mars'],
    ['hash', '=', data.q],
  ])

  let found = 0

  if (hashItems.length > 0) {
    formLeadId = hashItems[0].lead_id
    hashItems.forEach(hashItem => hashItem.delete())
  }

  // Доп защита
  // Проверяем что к нам пришла форма принадлежащая лиду
  // Можно не выполнять эту проверку здесь
  if (formLeadId === leadId) {
    // Удаляем предыдущие ответы
    // Todo: Можно вынести в сниппет
    oldResults = table.find('form_results', [], [
      ['form', '=', 'mars'],
      ['lead_id', '=', formLeadId],
    ])
    if (oldResults.length > 0) {
      oldResults.forEach(oldResult => oldResult.delete())
    }
```

```
    // Предварительно сохраняем данные в таблице
    table.createItem('form_results', {
      "form": "mars",
      "lead_id": formLeadId,
      "data": rawJsonData
    })

    // Удаляем кнопку калькулятора, если она есть !
    // Внимание! Удалять можно только с отправкой сообщения!
    menuHelper.removeReplyKeyboardWithMessage('Спасибо, данные получены!')

    //bot.scheduleScriptByCode('after_submit', formLeadId)

    // Выходим из цикла и запускаем скрипт уведомления о принятой форме
    return {
      "break": true,
      "run_script_by_code": "after_submit"
    }
  } else {
    // Можно вывести сообщение - что чтото прилетело из того что не ожидали
    // или выслать письмо администратору
    // ...
  }
}
// --------------------------------

// Todo: Здесь можно повторно отправить кнопку, тк она скрывается с экрана если отправить
сообщение в бота
bot.sendText('Пожалуйста, заполните форму, для этого нажмите на кнопку в нижней части
мессенджера,'
             + ' после этого откроется форма, которую вам необходимо заполнить'
             + ' и нажать кнопку на форме "Отправить заявку"')

return false

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

Обратите внимание на строки с кодом запуска скрипта, после сохранения данных с формы:

```
    // Выходим из цикла и запускаем скрипт уведомления о принятой форме
    return {
      "break": true,
      "run_script_by_code": "after_submit"
    }
```

Этот код может быть заменен на "return true" для выхода из цикла и на команду бота - "Выполнить скрипт", которую в таком случае необходимо разместить после команды JavaScript Callback. Подробнее смотрите в описании команды Выполнить JavaScript Callback.

Скрипт выполняемый после сохранения данных

Данный скрипт запускается из JavaScript Callback, код которой приведен выше.

Данный скрипт нужен чтобы пользователь продолжил диалог с ботом, а также демонстрирует как использовать данные сохраненные в таблицу результатов и вывести их в виде текста.

Пример скрипта.

Скрипт отправки ссылки состоит из следующих команд бота и пунктов меню:

- Команда **Выполнить JavaScript**:

```
const menuHelper = require('Common.TelegramComponents.MenuHelper')

// -------------------------------------------------------------
// Удаляем кнопку с формой для Inline формы, если она выведена

if (menuHelper.hasLastTelegramMessageId()) {
  // Удаляем кнопку с ссылкой на форму
  menuHelper.removeInlineKeyboard()

  // Сбрасываем ID последнего сообщения
  menuHelper.clearLastTelegramMessageId()
}
// -------------------------------------------------------------

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

```
// Для Inline кнопки с формой
// Удаляем кнопку с формой , если она выведена
// Удаляем здесь - тк после заполнения формы в JS-Callback мы уже не попадаем
// Тк прием данных с формы осуществляет через Internal Endpoiint
// Для keyboard кнопки ссылку на форму с кнопкой здесь удалять не нужно,
// тк для keyboard кнопки мы возвращаемся в JS-Callbak после заполнении формы
if (menuHelper.hasLastTelegramMessageId()) {
  // Удаляем кнопку
  menuHelper.removeInlineKeyboard()

  // Сбрасываем ID последнего сообщения
  menuHelper.clearLastTelegramMessageId()
}
// ---------------------------------------------------------------

// Если получаем данные напрямую, без таблицы form_results
//let data = request.json

// Если получаем данные из таблицы form_results
// Todo: Можно вынести в сниппет
let data = table.find('form_results', [], [
  ['form', '=', 'mars'],
  ['lead_id', '=', leadId]
])

if (data.length > 0 && typeof(data[0].data) === 'string' && data[0].data.length > 0) {
  data = JSON.parse(data[0].data)
} else {
  data = {}
  bot.sendText('Ошибка! Данные не найдены')
}

// Для определения какую кнопку показывать в меню ниже
// См условие аунктов меню текущего скрипта
memory.setAttr('mode', '')
if (typeof(data.mode) === 'string' && data.mode.length > 0) {
  memory.setAttr('mode', data.mode)
}
```

```
/*
// Сообщение о результате отработки формы, отправляется в бота
// Но проблема что система распознает сообщение как вебхук от лида
// Поэтому использовать можно только в JS-Callback и игнорировать вебхук вручную
// Или через регулярку маршрута, запуская отдельны скрипт
//
// Данное сообщение также автоматом закрывает web-view
// Но пробелм с закрытием нет, это выполняется в JS итак с помощью
// кода window.Telegram.WebApp.sendData(JSON.stringify(formData));
// или при выполнении window.Telegram.WebApp.close()
//
// Обычно используется для игры в web_view для вывода очков после победы или проигрыша
if (data.tg_query_id && data.tg_query_id.length > 0) {
  //https://core.telegram.org/bots/api#sentwebappmessage
  //https://core.telegram.org/bots/api#inlinequeryresultarticle
  bot.sendPayload('answerWebAppQuery', {
    "web_app_query_id": data.tg_query_id,
    "result": {
      "type": "article",
      "id": "1", // Unique identifier for this result, 1-64 Bytes
      "title": "Форма заполнена",
      //"description": "Description",
      "input_message_content": {"message_text": "Спасибо! Данные сохранены!"}
    }
  })
}
*/

memory.setJsonAttr('data', data)
memory.setAttr('is_qualified', data.is_qualified ? 'Да' : 'Нет')
memory.setAttr('has_experience', data.has_experience ? 'Да' : 'Нет')
```

- Команда **Отправить текст**:

```
Ваши данные:
Имя: {{ &$$data.name }}
Email: {{ &$$data.email }}
Возраст: {{ &$$data.age }}
Профессия: {{ &$$data.specialization}}
Ваш адрес проживания: {{ &$$data.address}}
```

```
Прошел курсы в Центре подготовки космонавтов: {{ &$is_qualified }}
Я уже летал в космос (имею опыт): {{ &$has_experience}}

Все данные в виде JSON:
{{ &$$data }}
```

- Команда **Отправить текст**:

```
Все верно?
```

- Пункты меню данного скрипта, с условиями:

КОД	НАДПИСЬ	СКРИПТ	УСЛОВИЕ ВЫВОДА
1	Да, все верно!	Подтвердил сохранение	

2	Нет, заполнить повторно	Форма в виде ссылки	`let mode = memory.getAttr('mode')`
			`if (mode !== 'tg_inline' && mode !== 'tg_keyboard') {`
			`  return true`
			`}`

2	Нет, заполнить повторно	Форма в виде inline кнопки	`let mode = memory.getAttr('mode')`
			`if (mode === 'tg_inline') {`
			`  return true`
			`}`

3	Нет, заполнить повторно	Форма в виде keyboard кнопки	`let mode = memory.getAttr('mode')`
			`if (mode === 'tg_keyboard') {`
			`  return true`
			`}`

Условия кнопок необходимы для вызова скрипта соответствующего текущему формату формы.

Условие для вывода универсальной формы в виде ссылки:

```
let mode = memory.getAttr('mode')
if (mode === 'tg_inline') {
  return true
}
```

Условие для вывода формы в Web App при нажатии на inline-кнопку:

```
let mode = memory.getAttr('mode')
if (mode !== 'tg_inline' && mode !== 'tg_keyboard') {
```

```
  return true
}
```

Условие для вывода формы в Web App при нажатии на keyboard-кнопку:

```
let mode = memory.getAttr('mode')
if (mode === 'tg_keyboard') {
  return true
}
```

Версия #2
Юрий Гарашко создал 2023-06-14 13:53:53 UTC
Ирина Петрова обновил 2025-11-11 07:20:20 UTC