Pyrus

Интеграция сервиса **Pyrus** с **Metabot** позволяет подключить бота в задачи по Формам в любых каналах связи. При этом, все новые диалоги вначале будут отправляться боту. По сценарию бот, предварительно собрав информацию и запросив контактные данные, далее может переключить диалог на оператора.

История диалогов с ботами автоматически сохраняется в задаче.

Настройка интеграции в Metabot

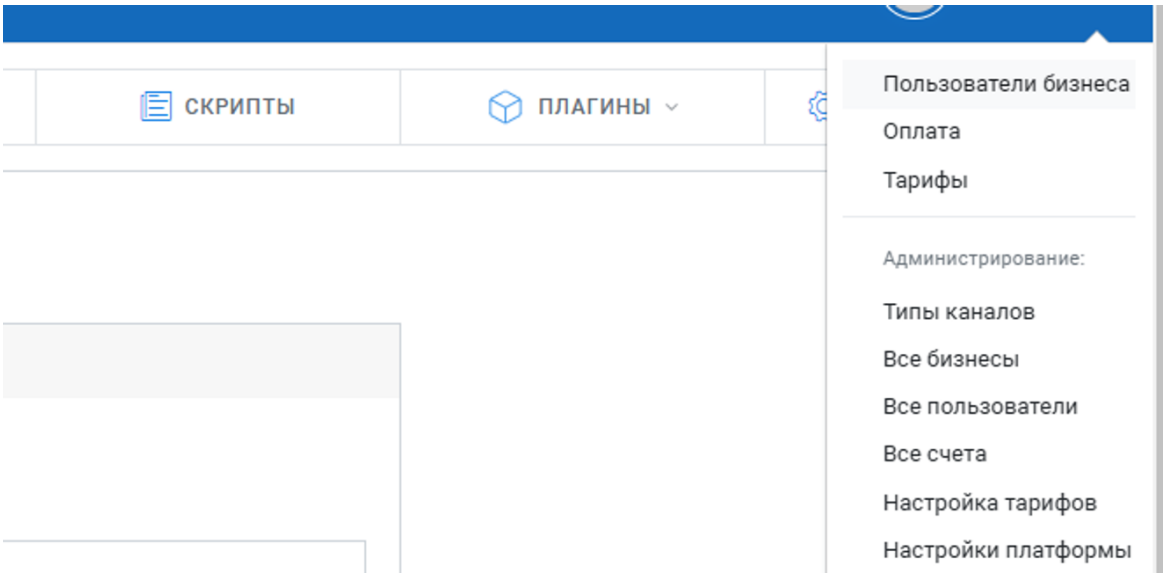
Настройка аккаунта

Для подключения интеграции **Pyrus** нужен аккаунт на **Metabot Platform**. Регистрация аккаунта (бизнеса) бесплатная и далее тарификация зависит от активности пользователей в боте.

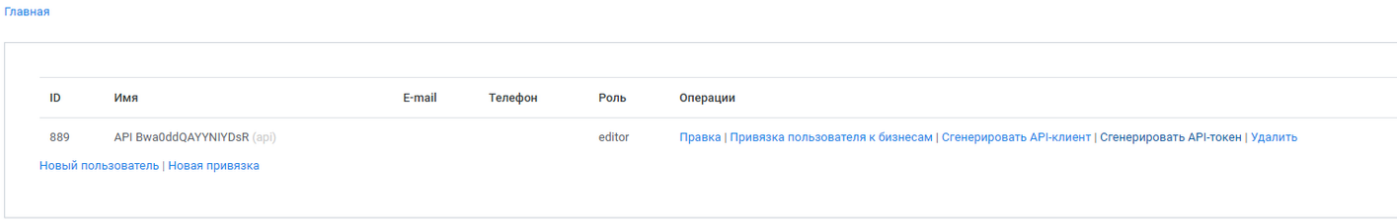
Создайте чат-бота в своем аккаунте, придумав ему название и проставив все необходимые настройки.

> Уроки с шаблонами можно посмотреть в **документации Metabot**, там же вы найдете инструкции по подключению бота в каналы.

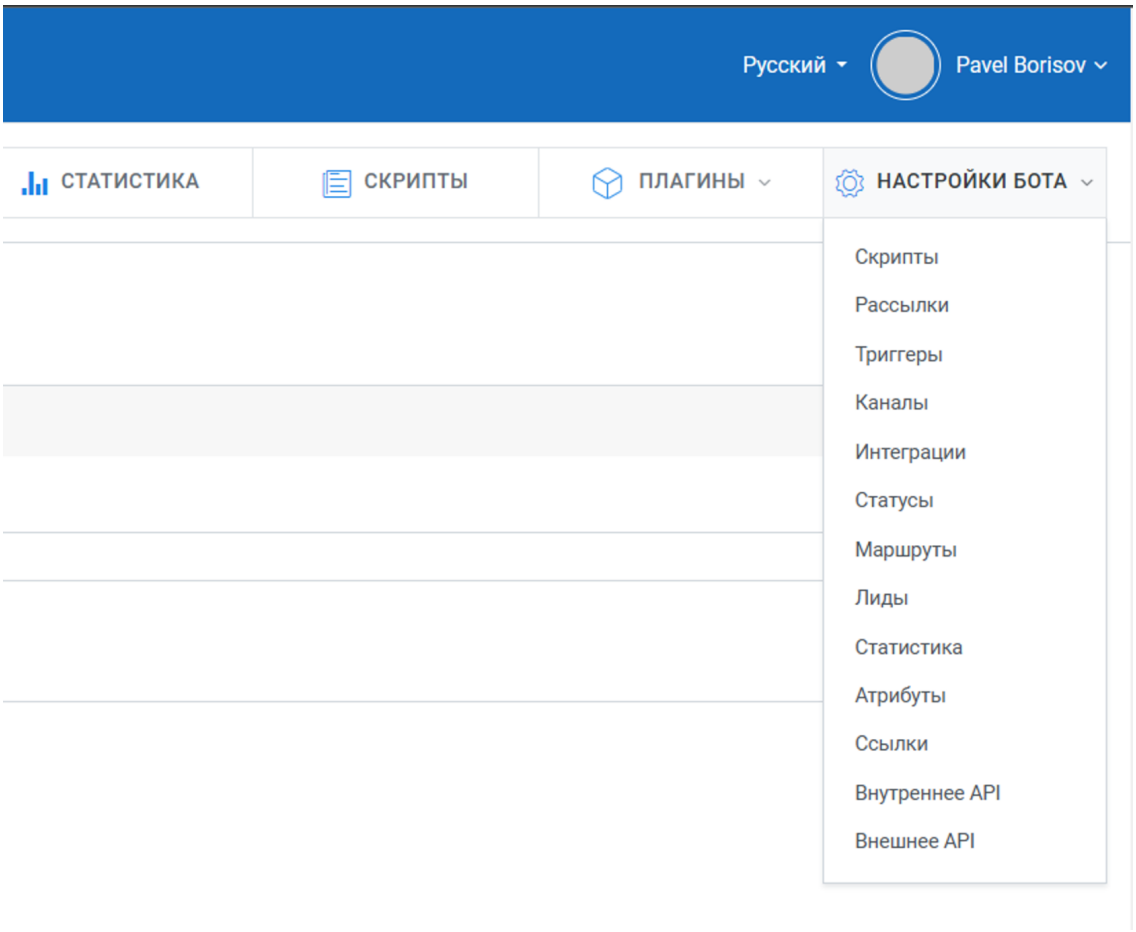
1. Для настройки интеграции с Pyrus в бизнес-аккаунте должен быть создан пользователь с правами доступа **Пользователь API**, для этого обратитесь в службу поддержки Metabot через раздел **Связаться с нами** или через виджет на платформе оставить обращение.



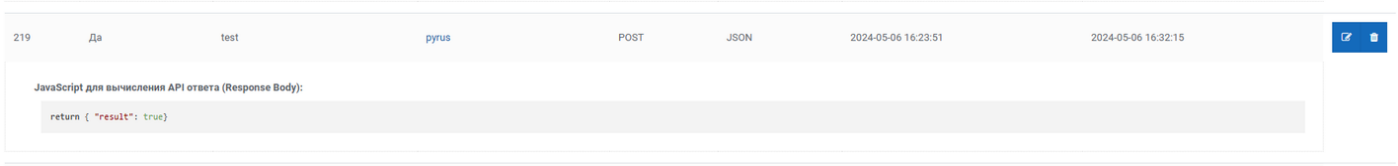
2. В разделе **Пользователи бизнеса** для появившегося пользователя сгенерируйте API-токен, нажав соответствующую кнопку. Скопируйте и сохраните появившийся в зеленом информационном окне токен.



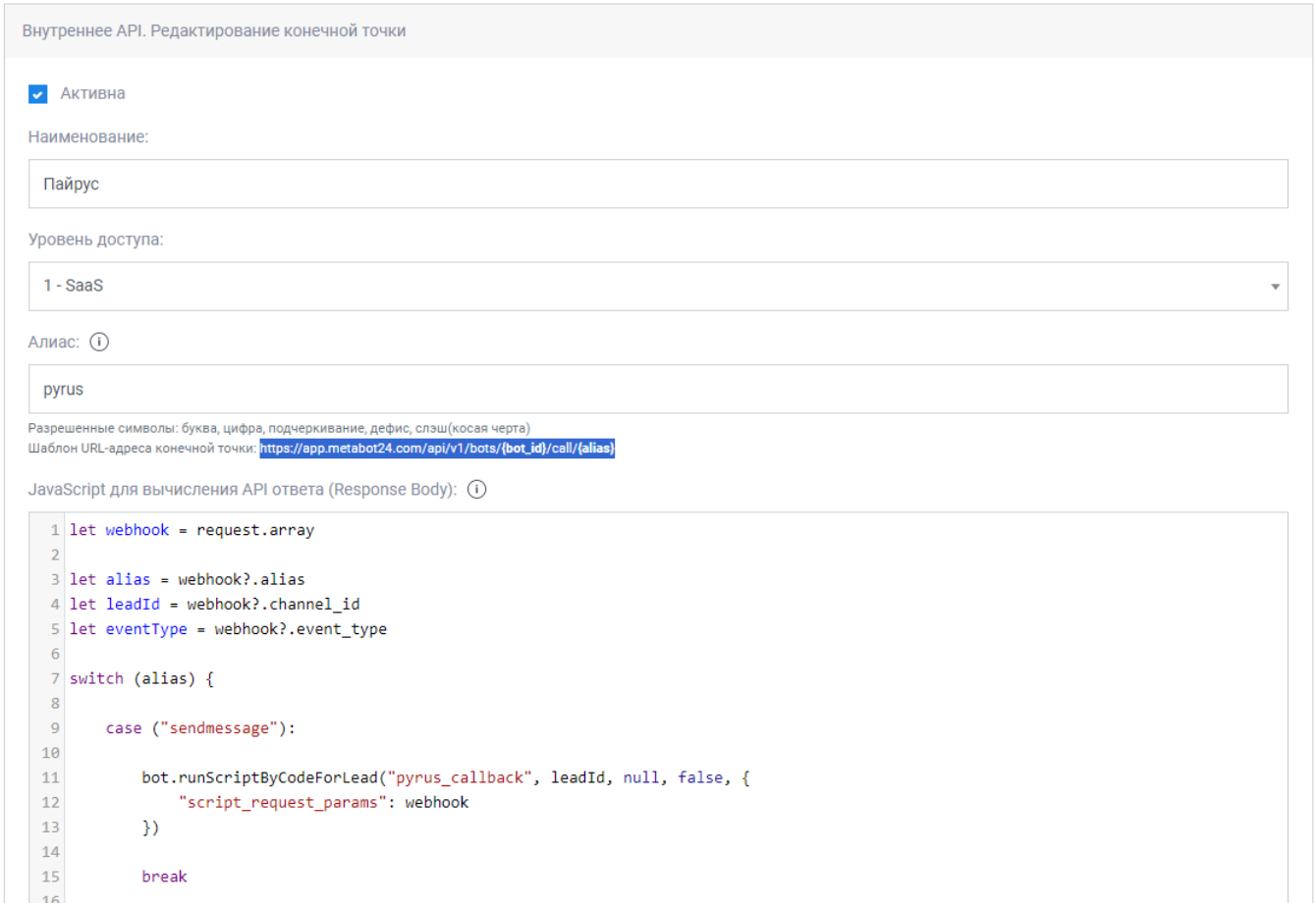
3. Перейдите в раздел **Настройки бота — Внутреннее API**.



4. Создайте внутренний endpont (конечную точку) с алиасом — **pyrus**.



5. Скопируйте получившуюся ссылку: https://app.metabot24.com/api/v1/bots/{bot_id}/call/{alias} — замените в ней **bot_id** на ID своего бота.



Для того чтобы узнать ID, необходимо нажать на редактировать бота в списке ботов и скопировать его ID (это число расположенное в url между /bot/ и /edit). А алиас нужно заменить на **pyrus** (без фигурных скобок).

В Response Body добавьте код:

```
let webhook = request.array
```

let alias = webhook?.alias let leadId = webhook?.channel_id let eventType = webhook?.event_type

switch (alias) {

```
case ("sendmessage"):

    bot.runScriptByCodeForLead("pyrus_callback", leadId, null, false, {
        "script_request_params": webhook
    })

    break
```

```
case ("event"):

    leadId = findMetabotLeadId(webhook?.mappings)

    // Если тип close
    if (eventType == "Close") {
        bot.runScriptByCodeForLead("pyrus_close_task", leadId, null, false, {
            "script_request_params": webhook
        })
    }

    break
```

}

return { "result": true }

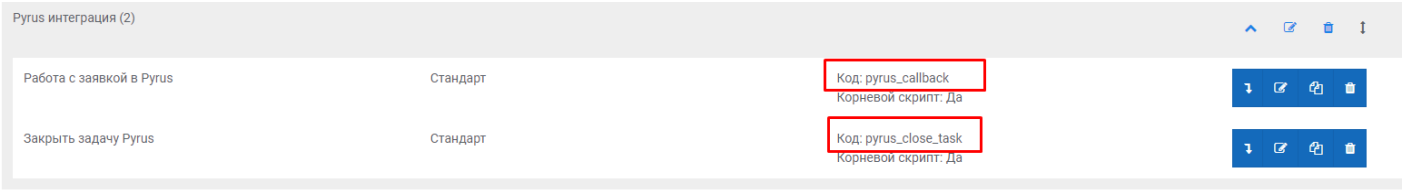
function findMetabotLeadId(mappings) { let result = mappings.find(mapping => mapping.code === "MetabotLeadId") if (result) { let value = result.value.split(',')[0] return parseInt(value, 10) } return null }

> Номер формы будет отображен после подключения аккаунта в Pyrus.

6. Сохраните создание эндпоинта.

Настройка чат-бота

В списке скриптов создайте отдельный раздел для команд, которые будут работать в связке с эндпоинтом.



Название раздела и скриптов может быть произвольным, но коды ОБЯЗАТЕЛЬНО нужно перенести в предлагаемом виде:

- **pyrus_callback** — запускает перевод на общение из задачи;
- **pyrus_close_task** — завершает диалог и возвращает в бота.

В скрипте перевода диалога на Pyrus необходимо создать: команду **Выполнить JavaScript Callback** с кодом:

Команду Выполнить JavaScript Callback с кодом

```
require('Common.DB.Select')
let db = CommonDBSelect
```

let pyrus = require("Common.Integrations.Pyrus") let pyrusRequestData = request.array

pyrus.accountId = bot.getAttr("sys_pyrus_account_id")

let attachments = bot.getAllAttachments() let guiIds = [] let mappings = lead.getJsonAttr("mappings")

let whJob = bot.getWebhookJob() let eventType = whJob.event_type

// Если из виджета, достаём по другому if (eventType === 'user_attachments') { attachments = JSON.parse(whJob.payload)?.payload?.attachments || [] }

if (attachments.length) {

```
for (let i = 0; i &lt; attachments.length; i++) {

    let url = attachments?.[i]?.url
    let giuId = pyrus.getGuiIdFile(url)
    guiIds.push(giuId)

}
```

}

switch (true) { case Boolean(Object.keys(pyrusRequestData).length): let messageText = pyrusRequestData?.message_text || "" let attachmentIds = pyrusRequestData?.attachment_ids || []

```
    if (attachmentIds.length) {

        let messageAttachments = pyrus.getFileUrl(attachmentIds[0]) // todo: Сдлеать
передачу всех файлов и возврат массива через for
        let sendAttachmentData = [{
```

```
            "type": "file",
            "url": messageAttachments?.url
        }]

        bot.sendMessage(`Оператор ${pyrus.getOperatorName()}: Файл`, [],
sendAttachmentData)

        if (bot.getLastResultPayload()?.ok == false) {

            pyrus.sendMessage(`Файл ${attachmentIds[0]} не доставлен. Попробуйте изменить
формат, сжать или отправить ссылку`, [], [], "БОТ")
        }

        return false

    } else if (messageText === "/stop") {
        pyrus.setForwarded(false) // Показываем, что диалог возвращён боту
        bot.sendMessage("Диалог с оператором завершён")
        bot.runScriptForLead(3137, leadId)
        pyrus.sendMessage(`Диалог с пользователем завершён`, [], [], "БОТ")

    } else if (messageText.includes("/go")) {
        let numberPattern = /\d+/
        let numberMatch = messageText.slice(messageText.indexOf("/go") +
3).match(numberPattern)
        let scriptCode = parseInt(numberMatch[0], 10)

        pyrus.setForwarded(false) // Показываем, что диалог возвращён боту
        bot.sendMessage(`Оператор ${pyrus.getOperatorName()} перевёл вас в сценарий
бота.`)
        bot.runScriptForLead(scriptCode, leadId)
        pyrus.sendMessage(`Диалог с пользователем завершён. Пользователь переведён на
выбранный сценарий`, [], [], "БОТ")

    } else {
        bot.sendMessage(`Оператор ${pyrus.getOperatorName()}: ${messageText}`)
    }
    break
```

```
default:
    let incomingMessage = bot.getIncomingMessage()

    if (isFirstImmediateCall){
        bot.sendMessage("Перевод диалога на оператора")
        incomingMessage = "Пользователь отправил новый вопрос"
    }

    if (checkIncomingMessage(incomingMessage)) {
        bot.runScriptForLead(3137, leadId)
    } else if (pyrus.isForwardedToOperator()) {
        pyrus.sendMessage(incomingMessage, mappings, guiIds)
    } else {

        // Примечанеи: Если оператор небыл подключен ранее, то диалог достанется с даты создания лида. Если был, то с даты завершения диалога. Либо с даты которую передадите в функции
        let lastCommunicateDate = pyrus.getLastCommunicateDate()
        let lastMessages = db.getMessagesByDate(lastCommunicateDate)
        pyrus.sendGroupMessages(lastMessages)

        if (guiIds.length) {
            pyrus.sendMessage(incomingMessage, [], guiIds) // Отправляем последнее сообщение с файлами
        }

        if (mappings.length) {
            pyrus.sendMessage("Поля формы обновлены", mappings) // Отправляем поля
        }
    }
    pyrus.setForwarded(true) // Показываем, что диалог передан оператору Pyrus
    return false
```

}

// Проверка слова на совпадения function checkIncomingMessage(word) { const patterns = [/^бот$/, /^меню$/, /^/start$/];

```
    word = word.toLowerCase();

    for (let pattern of patterns) {
        if (pattern.test(word)) {
            return true;
        }
    }

    return false;
}
```

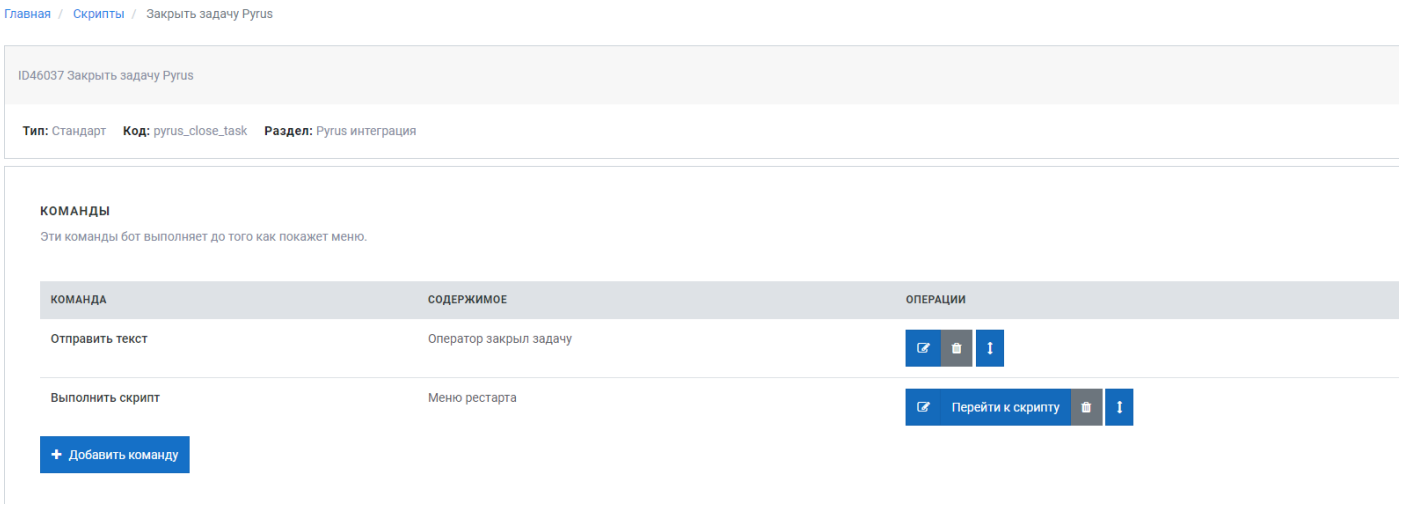
> В коде используется функция pyrus.getOperatorName(), которая возвращает значение — "Имя" или пустое значение. Также в коде callback есть текст "Оператор", который можно заменить на значение на любом языке, это используется в мультиязычных чат-ботах и когда интерфейс pyrus отличен от русского.

> В коде используется системный атрибут бота, в котором необходимо изменить значение, подставив цифровое значение после подключения.

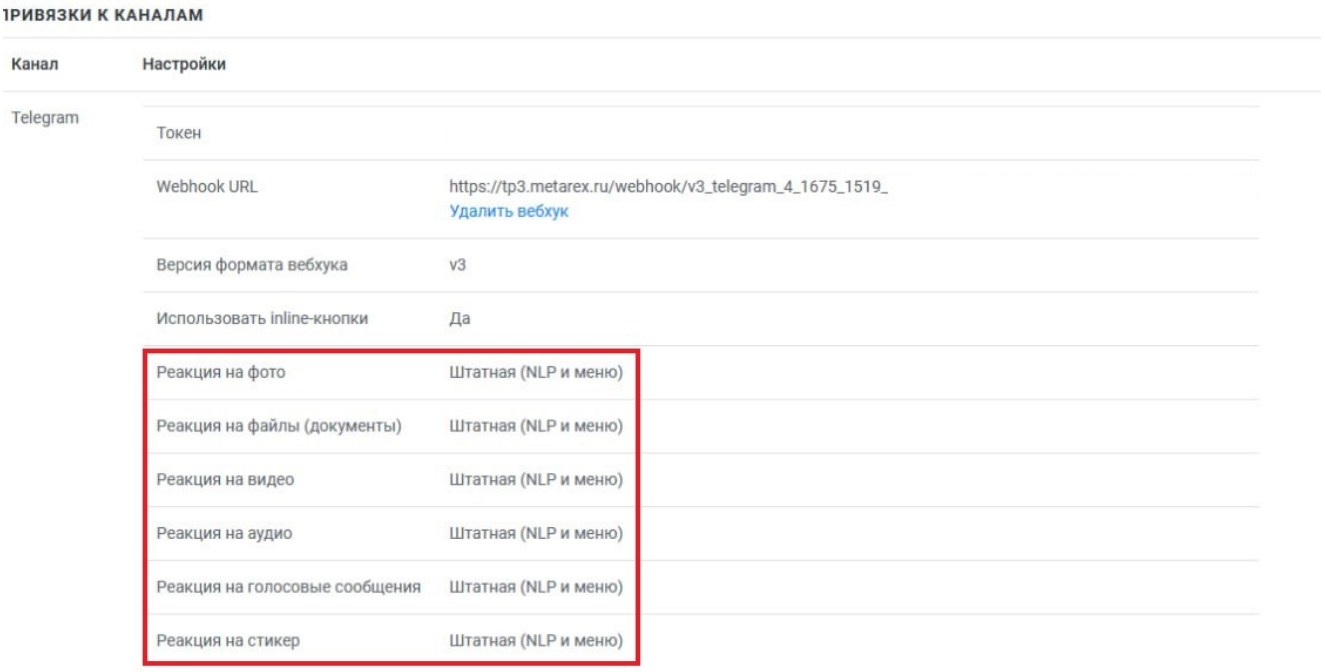
ЛИД	ТИП	НАЗВАНИЕ	ЗНАЧЕНИЕ
	variable **[системный]**	sys_pyrus_account_id	000000

> Именно этот скрипт отвечает за беспрерывное общение в задаче.

И завершить его может либо кодовые слова, прописанные в переменной **patterns**, либо закрытие задачи, которое можно настроить в боте с переводом в любой сценарий.



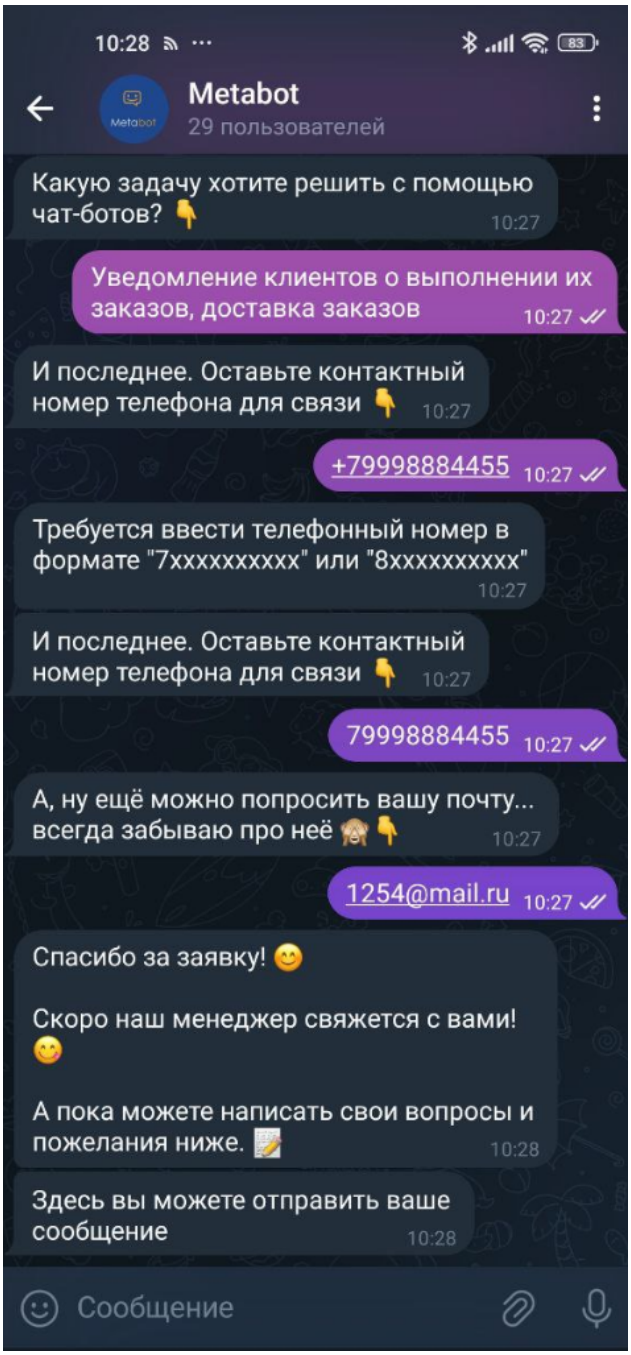
> Если вы хотите, чтобы ваш бот в Telegram мог принимать файлы (фото, аудио и т.д.), не забудьте указать это при создании канала.



Далее настройки следует произвести в самом Pyrus.

Настройка расширения в Pyrus

Чтобы Pyrus отслеживал сообщения, приходящие при передаче диалога от чат-бота. созданного на платформе Metabot, и автоматически создавал по ним заявки, включите интеграцию вашего аккаунта в Metabot с формой обработки клиентских обращений.

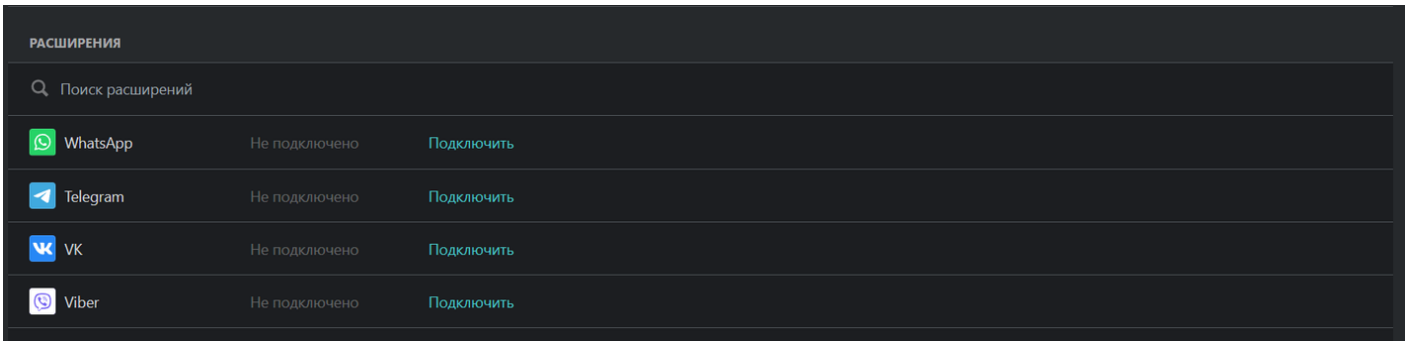


Бот преобразует ваши разговоры в любом канале, подключенном на платформе Metabot в комментарии к задачам в Pyrus и позволяет вам управлять ими так же, как и обычными задачами.

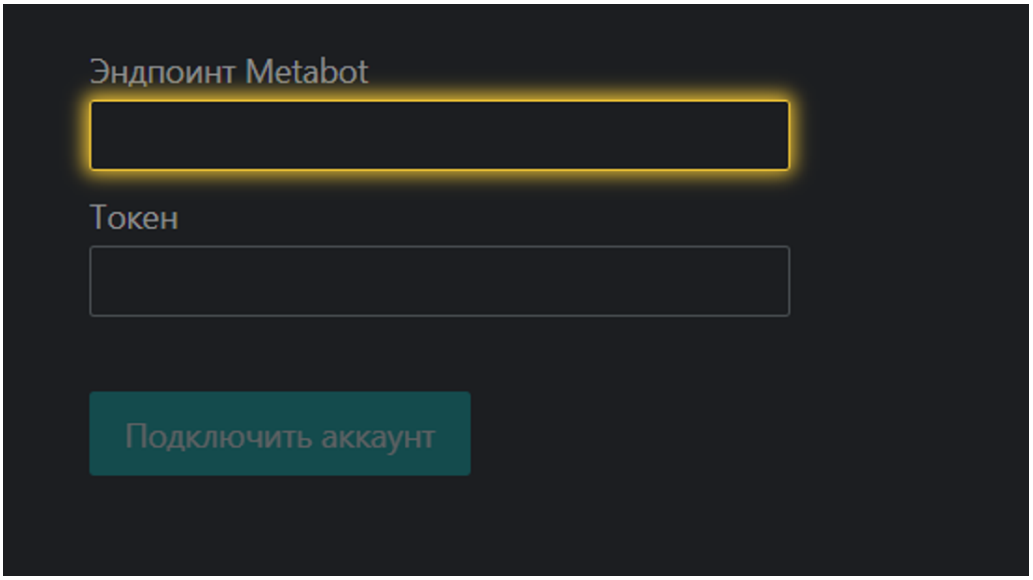
Сотрудники смогут отвечать клиентам из Pyrus прямо в тот канал связи, в котором ранее обращался пользователь (Telegram, Viber, VK, WhatsApp, виджет на сайте), завершать задачи, добавлять других участников в разговор и использовать любой другой функционал Pyrus.

Чтобы сообщения клиентов из чат-бота попадали в задачи Pyrus, подключите расширение **Metabot** к нужной форме — новой или уже существующей.

Перейдите в настройки формы и в списке расширений найдите **Metabot** и нажмите **Подключить.**

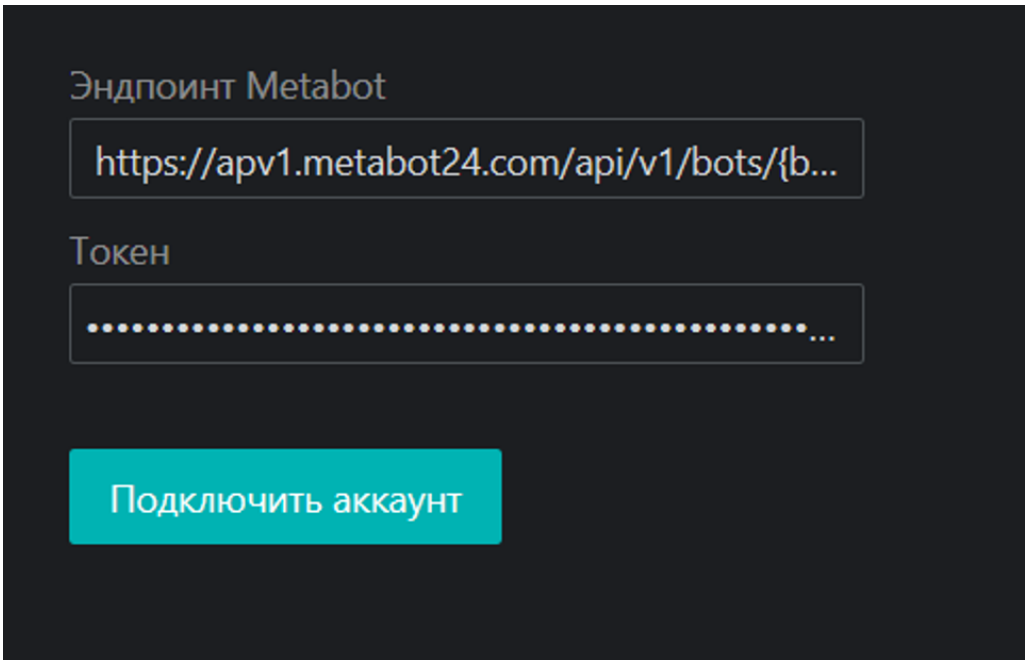


В настройках расширения необходимо указать **Эндпоинт Metabot** и **Токен**.



В поле **Эндпоинт Metabot** вставляем ссылку, которую берём во внутреннем API (https://app.metabot24.com/api-internal-endpoints) с предварительно измененными параметрами в п.3.

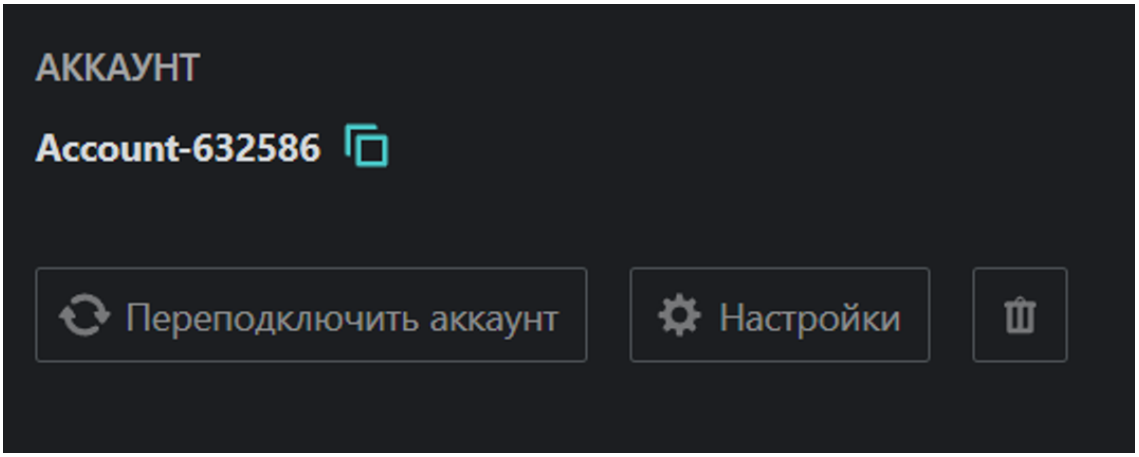
В поле **Токен** вставляем значение, которое генерируем в разделе **Пользователи бизнеса** https://app.metabot24.com/user, в п.2.



Нажмите кнопку **Подключить аккаунт**.

Генерируется номер аккаунта, это значит что аккаунт успешно подключен.

> Номер формы — это цифры после **Account**.



Теперь, когда клиент будет общаться с ботом, вы получите его в Pyrus. А ваши ответы он увидит в сообщениях в том канале связи, в котором он общался с ботом.

Все API запросы будут записываться в **Лог**.

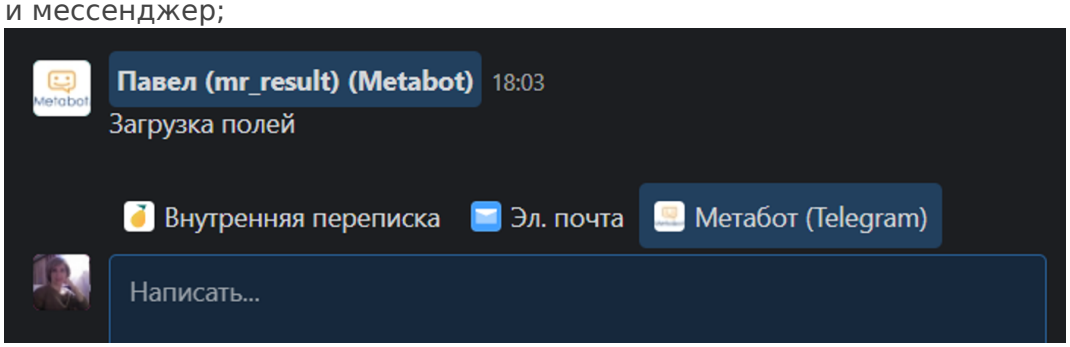
> Логи можно обрабатывать с помощью JS кода в блоке эндпоина, который вы указали.

Возможности интеграции

Отправлять и принимать сообщения

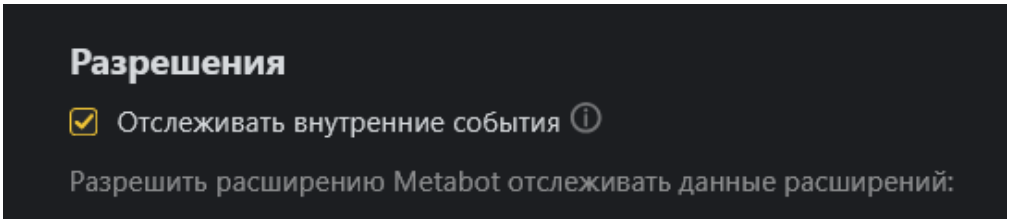
При запуске на платформе Metabot скрипта **callback_pyrus** происходит следующее:

- Если это был первый запрос пользователя, то при переходе диалога в Pyrus вся история его переписок поступит комментарием в задачу по Форме, к которой подключено расширение. В блок с отправкой сообщений пропишется имя аккаунта и мессенджер;



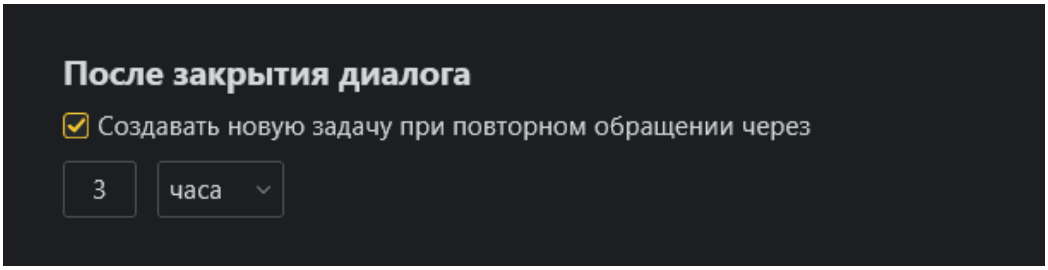
- Если это не первый запрос пользователя, то отправиться только само сообщение от пользователя;
- Если менеджер закрыл диалог с пользователем, т.е. закрыл задачу, нажав на кнопку **Завершить**, а человек подал новый запрос, то загружается вся история с момента последнего диалога;

> Для корректного завершения задачи и возврата диалога боту необходимо, чтоб в настройках интеграции для нужной формы было выдано разрешение на отслеживание внутренних событий.

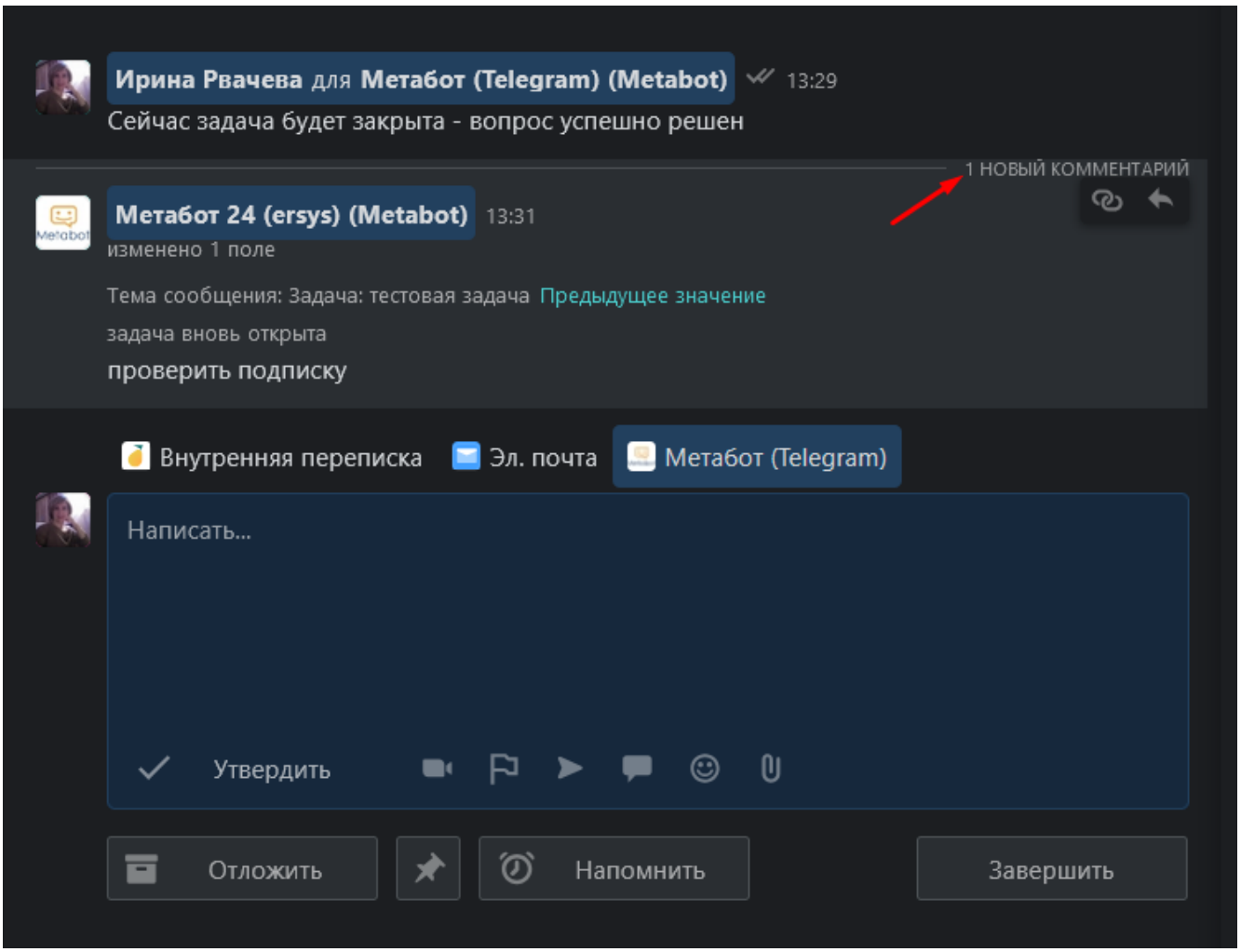


- Если менеджер закрыл диалог с пользователем, прошло время больше количества часов/дней, которые были указаны в настройках Формы и пользователь задал вопрос повторно, то создается **новая** задача без истории;

> Часы/дни можно изменить в настройках формы.

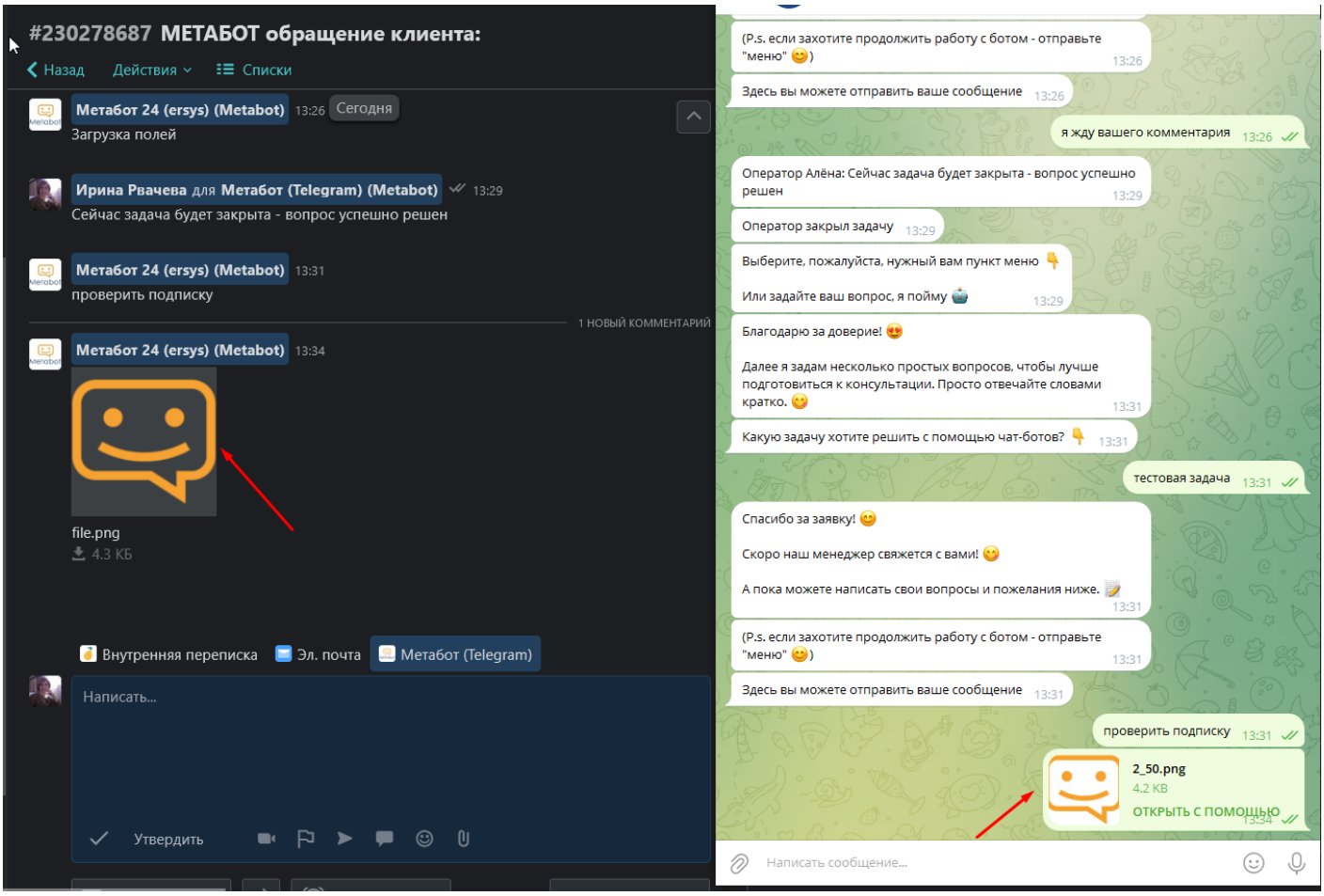


> Происходит вновь открытие задачи и добавляется новый комментарий



- Файлы можно отправлять в чат-бот и принимать из чат-бота в каналах связи: **Telegram** и **Виджет на сайте** только в том случае, когда открыт диалог с пользователем.

> В настройках канала обязательно должна быть включены следующие опции: Реакция на фото, файлы и видео - **Штатная (NLP и меню)**

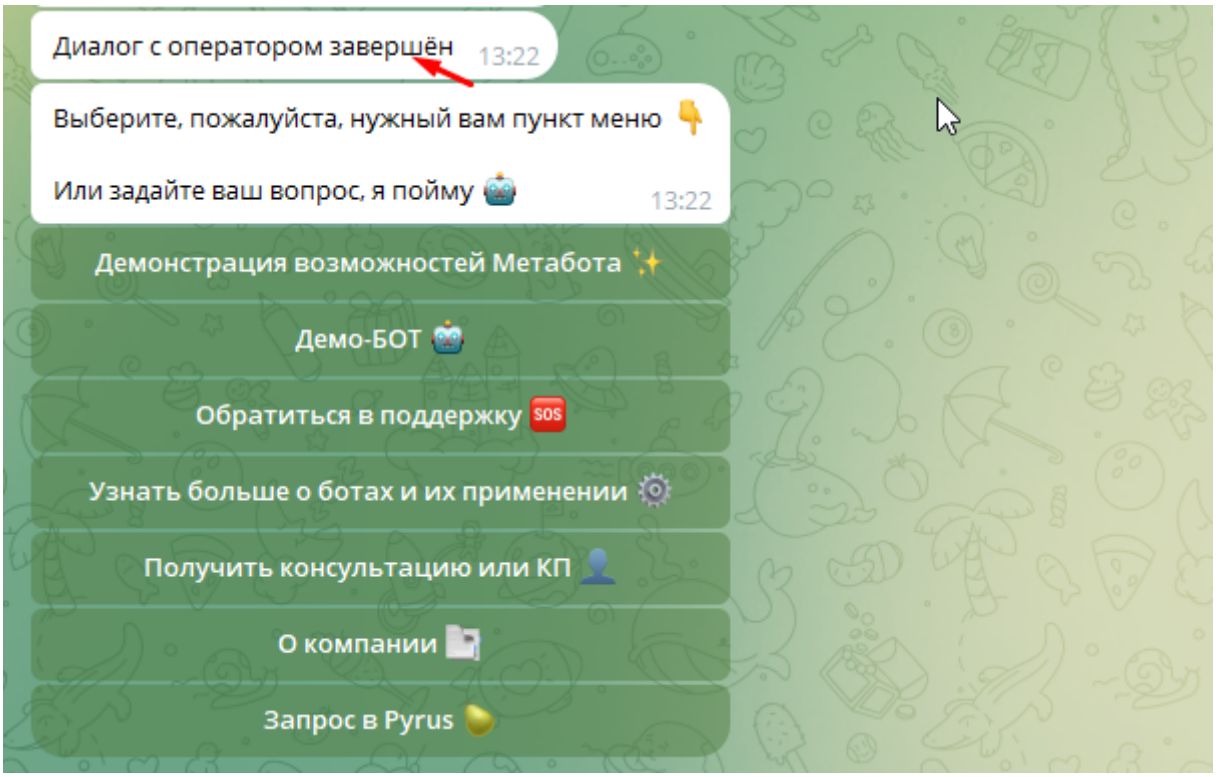


- На платформе можно настроить **Маршрут** бота с регулярными выражениями. Например, когда пользователь пишет **“бот"** или **“меню"** (без кавычек), то для его диалога запускается скрипт с меню и обновляется атрибут, в котором указывается что диалог закрыт.

Работа менеджера

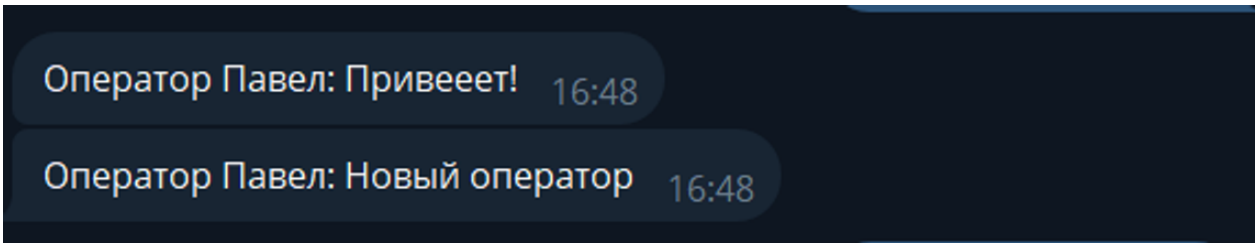
- Менеджер из задачи в Pyrus может останавливать диалог, написав команду — **/stop** ;

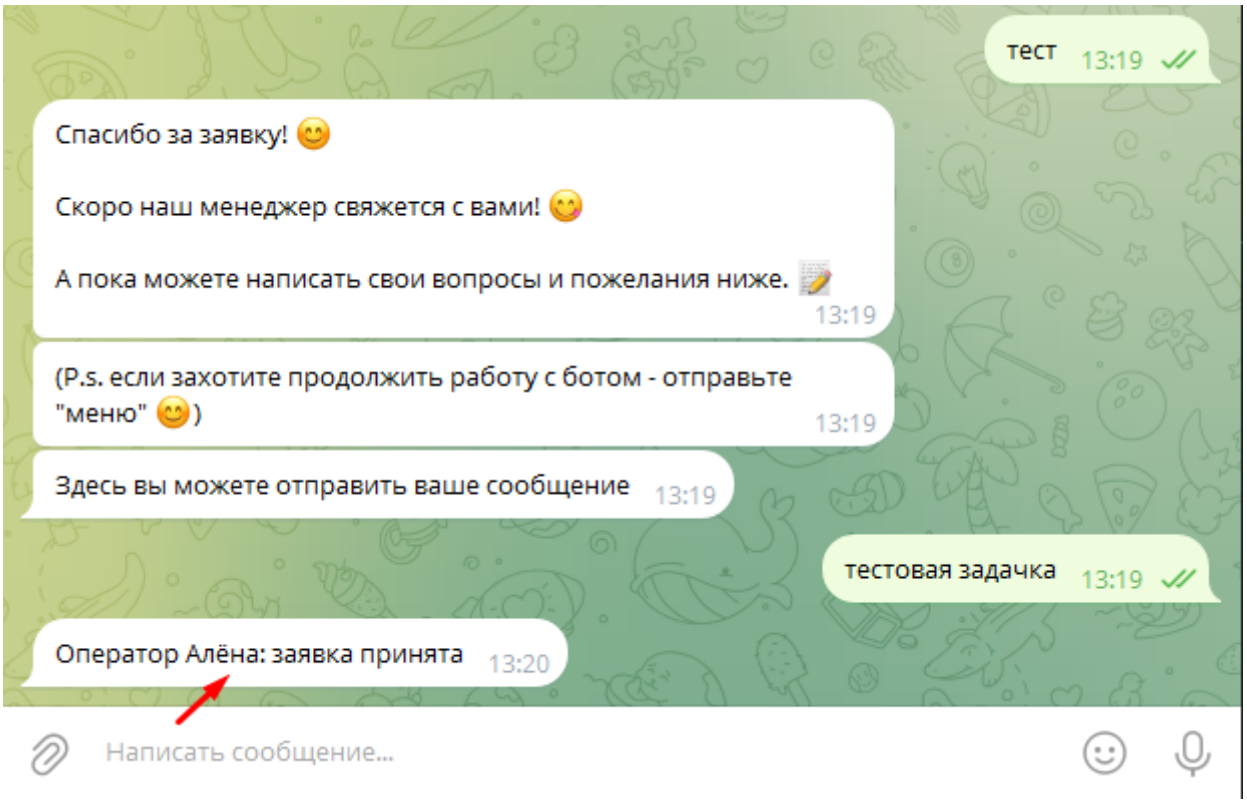
> Пользователю отправляется сообщение, что его диалог Завершен и запускается скрипт **Главного меню**. Задача в Pyrus при этот действии не закрывается.



- Менеджер из задачи в Pyrus может переводить пользователя в определенную точку сценария на запуск скрипта, написав команду — **/go + ID<скрипта>**;
- В диалоге пользователя в мессенджере отображается **Ответственный менеджер**, который ведет в донное время диалог с пользователем.

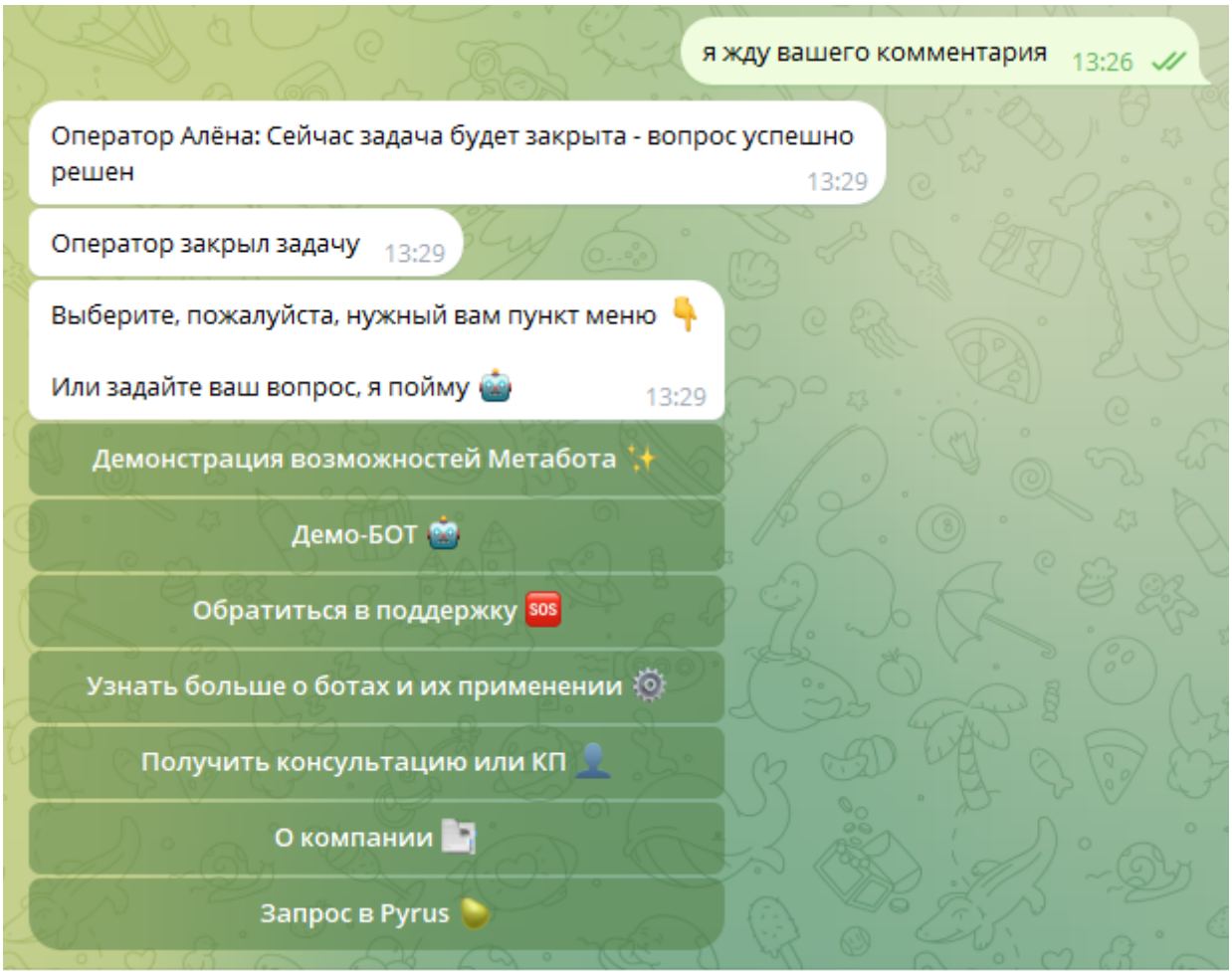
> Имя оператора меняется после смены **Ответственного** в Pyrus и после отправки сообщения пользователем.





Закрытие задачи

При закрытии менеджером задачи по кнопке **Завершить**, она закрывается и на платформе **Metabot**. Пользователю отправляется сообщение и запускается скрипт **Главного меню**.



Запрос и передача данных

Для того чтобы передать контактные или запрошенные у пользователя данные в соответствующие поля задачи по Форме, необходимо сохранить эти данные в переменную **mappings**.

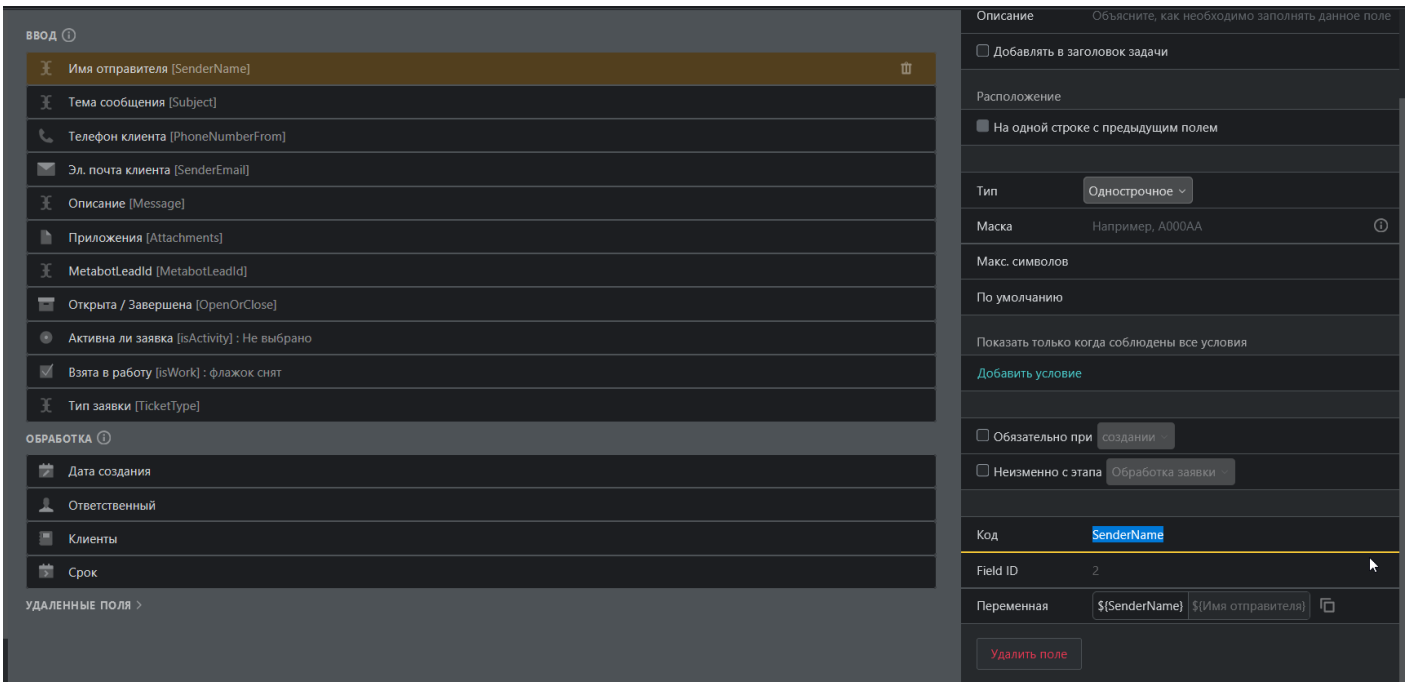
```
let mappings = [{
        "code": "SenderName",
        "value": lead.getAttr('clientName')
    }, {
        "code": "Subject",
        "value": "Задача: " + lead.getAttr('Задача')
    }, {
        "code": "PhoneNumberFrom",
        "value": lead.getAttr('sys_phone')
    },
    {
        "code": "SenderEmail",
        "value": lead.getAttr('sys_email')
    },
```

```
    {   "code": "CustomFields1",
        "value": lead.getAttr('источник')
    },
    {
        "code": "Message",
        "value": "Компания: " + lead.getAttr('company') + '\n' +
                 "Сфера деятельности: " + lead.getAttr('Ниша')
    },
    {
        "code": "MetabotLeadId",
        "value": leadId.toString()
    }
]
```

lead.setJsonAttr("mappings", mappings)

Как получить коды полей code

1. Перейдите в раздел **Формы**.
2. Выберите необходимую форму.
3. Перейдите в раздел **Настройки**.
4. Откройте вкладку **Шаблон**.
5. В настройках шаблона просмотрите или измените коды полей.

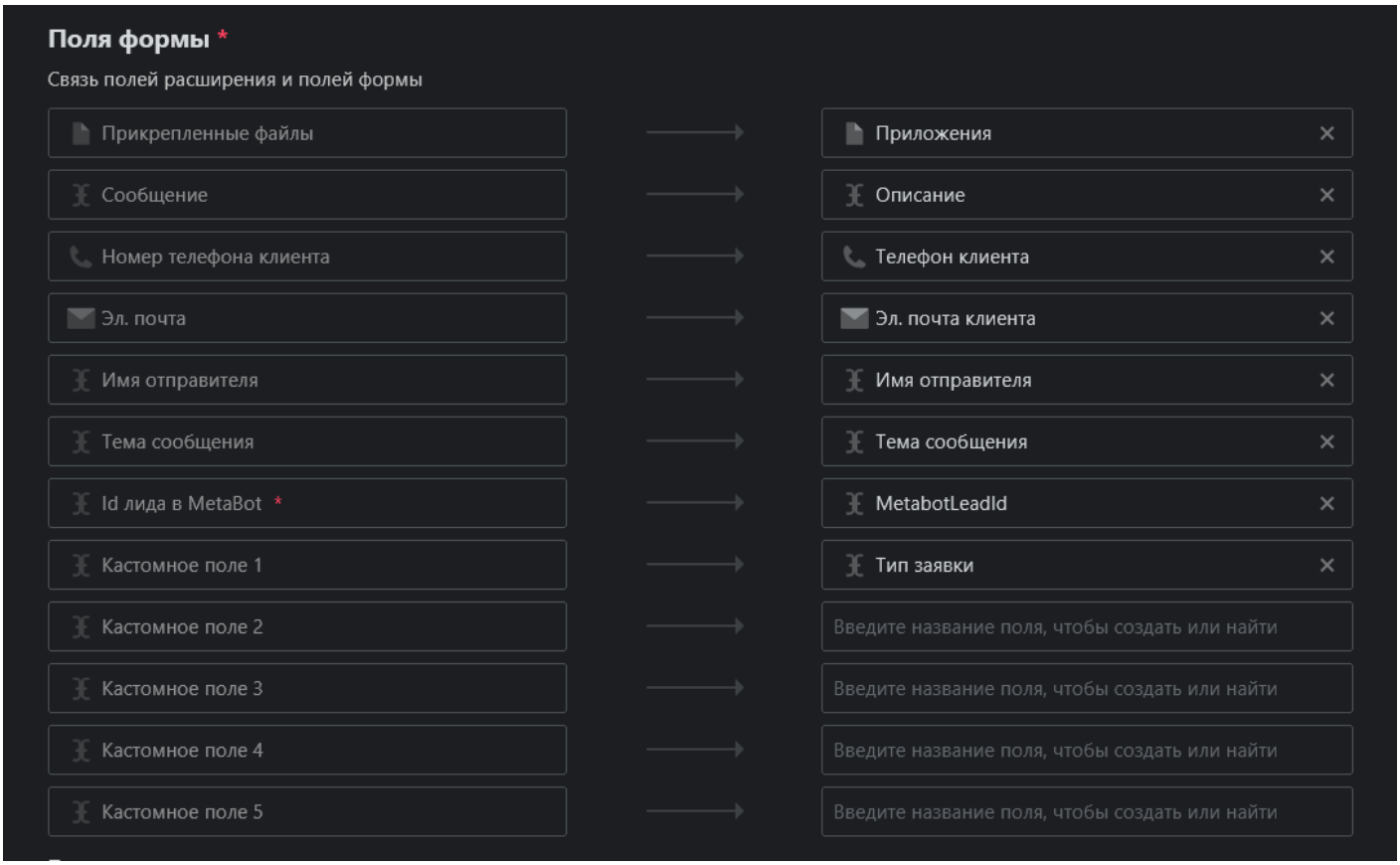


> На данный момент доступны **ТОЛЬКО** следующие типы полей с соответствующими кодами:
>
> - `SenderName` — Имя отправителя;
> - `Subject` — Тема сообщения;
> - `PhoneNumberFrom` — Телефон клиента;
> - `SenderEmail` — Эл. почта клиента;
> - `Message` — Описание;
> - `Attachments` — Файлы;
> - `MetabotLeadId` — MetabotLeadId (передача ID лида Metabot, с которым связана задача) ***обязательное поле***;
> - `OpenOrClose` — Открыта / Завершена.
>
> 5 кастомный текстовых поля, которые можно переименовать, сохранив коды без изменений:
>
> - CustomFields1 — Кастомное поле 1, тест;
> - CustomFields2 — Кастомное поле 2, тест;
> - CustomFields3 — Кастомное поле 3, тест;
> - CustomFields4 — Кастомное поле 4, тест;
> - CustomFields5 — Кастомное поле 5, тест.
>
> Все остальные добавляются по запросу.

> Обязательным для заполнения полем является только поле **MetabotLeadId**, менять код поля запрещено! По сопоставлению данного поля происходит связка лида в Метабот и в Pyrus.

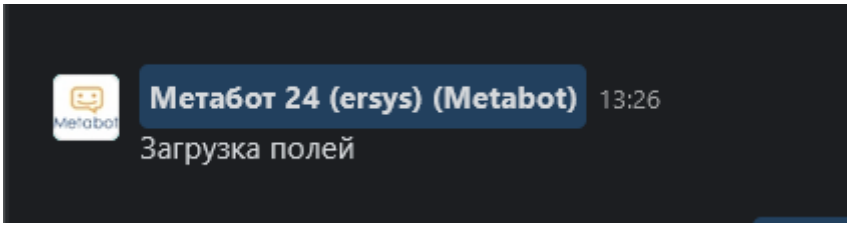
Здесь можно прописать любые данные, которые получает чат-бот во время диалога с пользователем.

В текущей настройке расширения доступны следующие переменные, которые будут записаны в соответствующие поля формы.



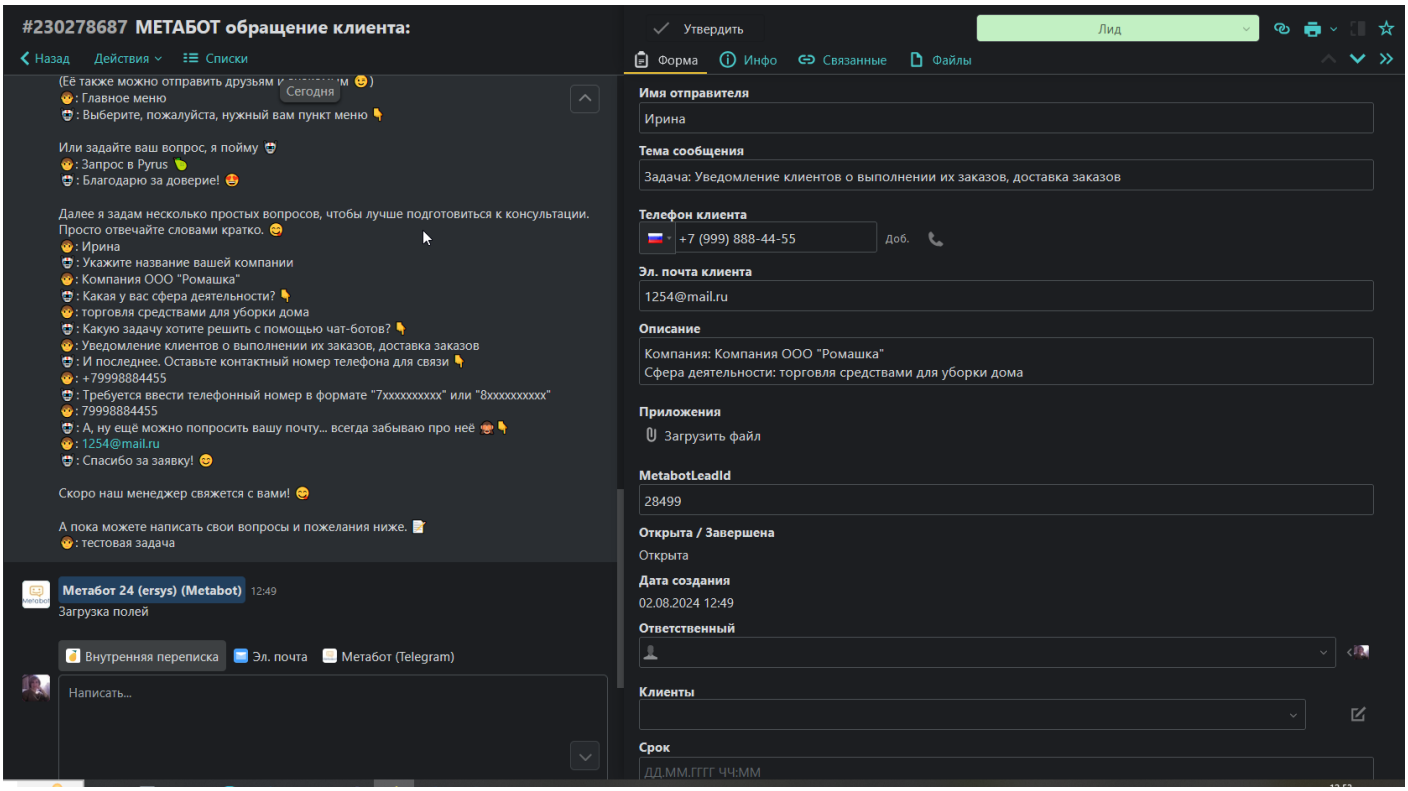
> Набор полей и переменных может быть изменен при разработке чат-бота на платформе Metabot, для этого необходимо обратиться в службу поддержки Metabot через раздел **Связаться с нами** или через виджет на платформе оставить обращение.

При передаче и обновлении данных в Pyrus отображается комментарий по данному действию.

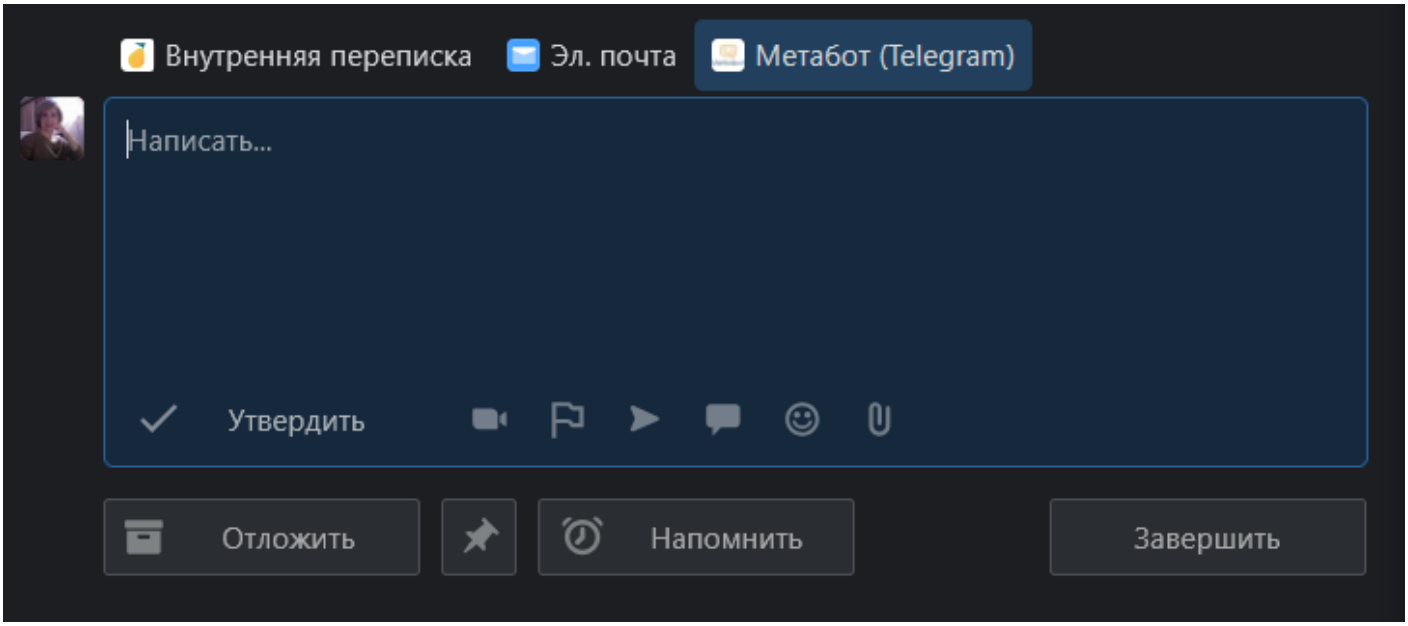


Работа в Форме задачи в Pyrus

При переводе диалога из канала связи чат-бота в Pyrus, в преднастроенной форме создается задача. В блоке Форма по полям прописываются переменные, которые собраны ботом в ходе диалога при текущем обращении пользователя или при ранее созданных обращениях. Эти переменные хранятся на платформе Metabot и передаются в Pyrus в момент работы скрипта интеграции.



Ответственный менеджер ведет переписку с пользователем в блоке **Metabot**.



> Для мобильного приложения Pyrus на данный момент не доступна возможность отвечать в сторонние каналы, в том числе и в Метабот. Доработка взята в дорожную карту развития Pyrus, но сроки не известны.

Версия #50
Ирина Рвачева создал 2024-07-31 10:53:02 UTC
Artem Garashko обновил 2026-06-10 11:03:22 UTC