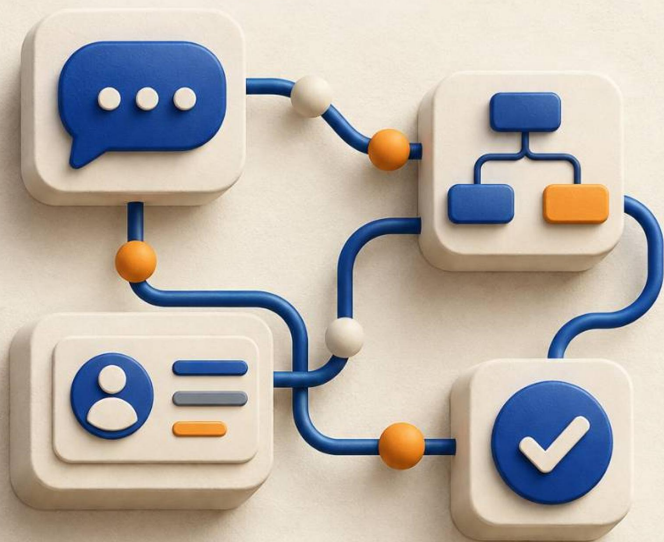


ZERO-CODE

**Продажа ботов
и автоматизаций
ПОД КЛЮЧ**



СЕРГЕЙ СТЕПАНОВ

Сергей Степанов

Zero-code. Продажа ботов и автоматизаций под ключ

<https://litres.ru/74573977>

SelfPub; 2026

Аннотация

Хотите создавать ботов и автоматизации для бизнеса и продавать их под ключ? «Zero-Code» показывает путь от первой рабочей схемы до сдачи проекта заказчику — без глубокого знания Python.

Вы разберётесь, как собирать решения в Make, n8n и Telegram-конструкторах, подключать нейросети, передавать заявки в amoCRM и Битрикс24, предотвращать дубли и восстанавливать работу после сбоев. Пошаговые примеры помогут понять не только настройки, но и логику каждого действия.

Отдельная часть посвящена продаже услуг: поиску клиентов, диагностике задачи, расчёту стоимости, подготовке предложения и передаче системы сотрудникам. Внутри — заполненные примеры брифа, спецификации и документов для приёмки.

Книга для фрилансеров, веб-разработчиков и IT-энтузиастов, которые хотят освоить работу ИИ-интегратора и научиться доводить автоматизацию до полезного для бизнеса результата.

Содержание

Введение	6
Глава 1. Что покупает заказчик	9
Автоматизация как завершённая работа	9
Четыре зоны ответственности	11
Как выбрать первую специализацию	12
Когда проект лучше не брать	13
Глава 2. Как найти задачу с измеримой пользой	14
Разговор о фактической работе	14
Карта процесса на одном листе	16
Исходные показатели	17
Выбор первого улучшения	18
Практическое задание	19
Глава 3. Как превратить разговор в спецификацию	20
Граница между просьбой и обязательством	20
Обязательные данные	22
Проверяемые критерии	23
Правила изменения объёма	24
Документирование решений	25
Глава 4. Данные и события без глубокого программирования	26
Из чего состоит цепочка	26
Объекты и массивы	27

Сопоставление полей	28
Чтение незнакомого ответа	30
Начинайте с ограничений	31
Пробная сборка перед обещанием	35
Глава 6. Рабочее окружение и доступы	36
Разделение учебной и рабочей среды	36
Кто владеет аккаунтами	37
Секреты и подключения	38
Реестр зависимостей	39
Проверка готовности	40
Результат и условия учебной сборки	41
Шаг 1. Подготовьте входные данные	42
Шаг 3. Проверьте доступ к Битрикс24	45
Шаг 4. Сформируйте тело запроса без ручной склейки	46
Шаг 5. Верните подтверждённый результат	48
Шаг 6. Обработайте неопределённый результат	50
Что делает первая версия	52
Шаг 1. Создайте бота и получите чат сотрудника	53
Шаг 2. Разделите типы входа	54
Шаг 3. Нормализуйте обращение	55
Шаг 4. Создайте постоянный журнал	56
Конец ознакомительного фрагмента.	57

Сергей Степанов

Zero-code. Продажа ботов и автоматизаций под ключ

**Zero-Code: Продажа ботов и автоматизаций под
ключ**

Сергей Степанов

Введение

Владелец небольшой сервисной компании получает заявки из мессенджера, с сайта и по телефону. Часть обращений сотрудники переносят в CRM вручную. Иногда карточка появляется без номера телефона, иногда один запрос превращается в две сделки, иногда клиенту обещают перезвонить и забывают. Компания может купить ещё один сервис, но сама покупка не определит, кто должен обработать заявку и что считать успешным результатом.

ИИ-интегратор решает эту задачу на стыке процессов и технологий. Он выясняет, как компания работает, собирает связи между системами, ограничивает действия нейросети, проверяет сбои и передаёт заказчику решение, которым сотрудники смогут пользоваться. Для многих проектов не требуется глубокое знание Python. Однако придётся понимать структуру данных, права доступа, последовательность операций и ответственность за изменения.

Эта книга предназначена для IT-энтузиастов, веб-разработчиков и фрилансеров, которые хотят продавать такие решения малому бизнесу. Если вы уже умеете создавать сайты, полезным следующим шагом станет подключение заявок к CRM. Если начинаете с нуля, сначала потребуется освоить несколько технических понятий. Они вводятся по мере необходимости и сразу применяются в учебных задачах.

Название Zero-Code обозначает способ сборки основной логики через визуальные инструменты. Оно не обещает, что любая интеграция обойдётся без формул, запросов к API и помощи разработчика. Иногда небольшая программная доработка проще и надёжнее десятка обходных действий. Профессиональная работа предполагает умение замечать эту границу.

Проекты стоимостью от 30 000 до 150 000 рублей рассматриваются как возможные коммерческие предложения разного состава. Эти суммы не являются установленными рыночными тарифами или обещанием заработка. На стоимость влияют обследование процесса, количество систем, сложность согласований, требования к надёжности и условия поддержки. Все финансовые примеры в книге условные: их задача — показать расчёт, а не выдать прогноз дохода.

Технические примеры сверены с документацией платформ при подготовке книги в октябре 2026 года. Они описывают учебные настройки и ожидаемые результаты; фактический запуск требует доступных аккаунтов и проверки в вашей среде. Названия элементов интерфейса, тарифы и доступность отдельных интеграций могут меняться. Поэтому инструкции объясняют назначение настроек и признаки правильного результата. До покупки подписки проверяйте, что нужные функции доступны в конкретном аккаунте и регионе.

Через книгу проходит учебная компания «ТехСервис»,

которая принимает заявки на ремонт бытовой техники. Компания вымышленная. Её бот собирает контакт, уточняет проблему, передаёт обращение менеджеру и сохраняет сведения в CRM. Дополнительные примеры показывают, как тот же подход применять к заказам, внутренним запросам и работе с документами. Разработка ведётся на тестовых данных: реальные сведения клиентов подключаются только после согласования доступа и правил обработки.

Результатом обучения должна стать не коллекция красивых схем, а готовность выполнить ограниченный проект: описать его границы, собрать рабочую цепочку, подтвердить результат тестами, назначить цену и передать систему заказчику. Два первых проекта дают конкретные маршруты Make и n8n. Затем вы добавите анкету, CRM и нейросетевую обработку, а заполненный комплект рабочих документов поможет определить состав коммерческого заказа.

Глава 1. Что покупает заказчик

Автоматизация как завершённая работа

В учебном «ТехСервисе» менеджер переписывает из Telegram в CRM имя, телефон, вид техники и описание неисправности. На одно обращение условно уходит четыре минуты. Бот, который только красиво отвечает на вопросы, не убирает эту работу. Чтобы решить исходную задачу, нужно получить необходимые сведения, проверить их и создать связанную карточку обращения.

Разница между демонстрацией и услугой возникает в момент, когда решение включают в реальную работу. На демонстрации можно вручную отправить идеальное сообщение и показать созданную сделку. В эксплуатации человек пришлёт фотографию вместо текста, дважды нажмёт кнопку, изменит номер телефона или напишет после полуночи. Заказчик покупает обработку оговорённых ситуаций, а также понятный порядок действий для остальных.

Опишите результат через изменение процесса. Формулировка «умный бот с нейросетью» ничего не сообщает о границах. Формулировка «после подтверждения контакта бот

создаёт обращение в CRM и уведомляет дежурного менеджера» позволяет обсудить данные, срок и проверку. Нейросеть может быть частью решения, но не заменяет описания результата.

Четыре зоны ответственности

Сначала интегратор исследует процесс: получает примеры заявок, узнаёт роли сотрудников, выясняет исключения. Затем проектирует решение: выбирает систему хранения, правила маршрутизации и точки человеческого контроля. После этого собирает и проверяет цепочку. Наконец, организует передачу и поддержку: объясняет сотрудникам, где искать ошибки и как восстановить работу.

Новичок часто оценивает только время сборки. Допустим, соединение трёх модулей заняло два часа. Но ещё потребовались встречи, доступ к CRM, согласование полей, исправление дублей и обучение администратора. Эти работы входят в проект, даже если не видны на полотне автоматизации. Их нужно учитывать в сроках и цене.

У исполнителя и клиента разные полномочия. Интегратор предлагает схему, но не решает самостоятельно, когда компания обязана вернуть деньги или какую скидку дать спорному клиенту. Такие правила утверждает уполномоченный представитель бизнеса. Если правило не определено, автоматизация должна передать вопрос сотруднику, а не придумать решение.

Как выбрать первую специализацию

Удобнее начинать с повторяющейся задачи в одной группе компаний. Например, принимать заявки для ремонтных служб, связывать формы сайтов с CRM для небольших агентств или формировать внутренние уведомления для учебных центров. Это позволяет глубже понять одинаковые ограничения и быстрее подготовить полезную демонстрацию.

Специализация не требует отвергать все остальные запросы. Она нужна для первого понятного предложения. «Подключаю заявки с вашего сайта к действующей CRM» легче проверить и купить, чем «автоматизирую бизнес любыми нейросетями». При этом выбранная услуга должна быть вам технически доступна: не обещайте голосового администратора, если пока собрали лишь текстовую форму.

Оцените задачу по трём признакам: повторяемость, доступность данных и ограниченность последствий ошибки. Перенос заявки обычно проще первого проекта по автоматическому возврату платежей. Сводка для менеджера безопаснее автономного изменения коммерческих условий. Такой порядок уменьшает стоимость обучения на реальном заказе.

Когда проект лучше не брать

Откажитесь от фиксированной цены до обследования, если клиент не может показать исходные данные, не назначает ответственного сотрудника или требует гарантировать рост продаж. Автоматизация влияет на отдельные этапы. Продажи зависят также от спроса, цены, качества услуги и работы менеджеров.

Отказ может быть частичным. Вместо полной перестройки предложите ограниченную диагностику: разобрать один канал заявок, описать проблемы и подготовить спецификацию. Если доступы или правила так и не появятся, у вас останется оплаченная завершённая работа, а у клиента — основание для следующего решения.

Для первой услуги запишите одним предложением: какой процесс меняется, что запускает автоматизацию, какой результат видит сотрудник и где требуется его подтверждение. Пока это предложение не получается конкретным, к выбору платформы переходить рано.

Глава 2. Как найти задачу с измеримой пользой

Разговор о фактической работе

На первой встрече попросите сотрудника показать последнюю обычную заявку. Пусть он объяснит, откуда она пришла, куда переносил данные, кому сообщил и как узнал, что обращение обработано. Такой показ полезнее обсуждения абстрактной «цифровизации»: он обнаруживает конкретные действия и пропущенные связи.

В «ТехСервисе» выясняется, что перенос данных занимает немного времени, а основная задержка возникает между уведомлением и назначением ответственного. Если автоматизировать только копирование, скорость первого ответа почти не изменится. Поэтому отдельно измеряют время поступления, регистрации и первого содержательного контакта.

Не превращайте встречу в экзамен для сотрудников. Вопрос «почему вы до сих пор делаете это вручную?» вызывает защиту и скрывает детали. Лучше уточнить: «Что мешает выполнить этот шаг сразу?» Ответ может выявить отсутствие доступа, неполные сведения или зависимость от руко-

водителя. Эти ограничения нужно включить в проект.

Карта процесса на одном листе

Запишите последовательность действий обычными глаголами: получить сообщение, уточнить контакт, проверить зону обслуживания, создать обращение, назначить менеджера, подтвердить получение. Рядом укажите исполнителя каждого действия и систему, где остаётся результат. Если результат нигде не фиксируется, измерять его будет трудно.

Для каждого перехода задайте вопрос об исключении. Что происходит, если телефон отсутствует? Если клиент уже есть в CRM? Если услуга не оказывается в его районе? Если менеджер не работает сегодня? Необязательно автоматизировать каждое исключение, но должно быть понятно, кому оно передаётся.

Отметьте источник истины — систему, сведения которой считаются определяющими. Для этапа сделки это может быть CRM; для списка услуг — утверждённый каталог; для свободных окон — календарь. Если бот и менеджер используют разные прайс-листы, правильная интеграция не исправит противоречие между ними.

Исходные показатели

Перед запуском соберите небольшую исходную выборку. Для каждой заявки зафиксируйте время поступления, время регистрации, факт потери, наличие обязательных полей и первый ответ. Укажите период и способ отбора. Не смешивайте обычную неделю с праздничным пиком, если хотите понять влияние изменения.

Среднее значение бывает недостаточно. Из десяти обращений девять могут обрабатываться за пять минут, а одно — через сутки. Поэтому полезно отдельно показать число заявок с задержкой больше согласованной границы. Саму границу определяет бизнес с учётом режима работы, а не интегратор произвольно.

Потерянная заявка и проигранная продажа — разные вещи. Если карточка не появилась в CRM, можно подтвердить техническую потерю. Если клиент отказался после консультации, причина может не иметь отношения к автоматизации. Сохраняйте это различие в отчётах и коммерческих обещаниях.

Выбор первого улучшения

Условно компания получает 450 обращений в месяц. Копирование каждого занимает четыре минуты: $450 \times 4 = 1800$ минут, или 30 часов. При условной внутренней стоимости часа 600 рублей получается 18 000 рублей времени. Это оценка ресурса, а не автоматическая экономия денег: зарплата может остаться прежней.

Денежный эффект появляется, если высвобождённое время уменьшает оплачиваемые переработки, позволяет обойтись без дополнительного найма или используется для работы, приносящей подтверждённую маржу. Предложите владельцу выбрать, какой эффект для него реален. Не складывайте стоимость времени и дополнительную прибыль без проверки, что они отражают разные результаты.

Сравните несколько задач. У приёма заявок может быть высокая частота и понятный результат. У сложной аналитики — привлекательная демонстрация, но плохие исходные данные. Для первого проекта обычно полезнее задача, где результат можно подтвердить небольшим числом проверок.

Практическое задание

Возьмите один знакомый процесс и опишите его без названий сервисов. Найдите действие, которое повторяется, требует переноса сведений и заканчивается наблюдаемым результатом. Соберите три обычных случая и три исключения. На этом материале сформулируйте предложение с измеримым критерием.

Если получить примеры невозможно, используйте вымышленные записи и прямо обозначьте учебный статус. Такая работа помогает освоить инструменты, но не даёт оснований утверждать, что реальному бизнесу уже рассчитан эффект. Для коммерческого проекта обследование всё равно потребуется.

Глава 3. Как превратить разговор в спецификацию

Граница между просьбой и обязательством

Клиент говорит: «Хочу, чтобы бот отвечал клиентам и всё заносил в CRM». В этой просьбе не определены виды вопросов, обязательные сведения, правила создания карточек и роль менеджера. Если начать сборку сразу, каждое уточнение будет восприниматься как исправление вашей ошибки.

Спецификация описывает поведение, которое стороны смогут проверить. Для учебного проекта граница такова: бот работает с личными текстовыми сообщениями, собирает контакт и описание проблемы, предлагает утверждённые категории техники и создаёт одну заявку после подтверждения. Он не диагностирует поломку, не назначает окончательную стоимость ремонта и не обещает время выезда.

Укажите также, что считается вне проекта. Например, голосовые сообщения, платежи, перенос старой базы и несколько филиалов. Это не повод составлять длинный список запретов. Достаточно зафиксировать смежные задачи, которые клиент с большой вероятностью может подразуме-

вать.

Обязательные данные

Задайте для каждого поля название, тип, источник, допустимые значения и поведение при отсутствии. Телефон — строка, потому что знак плюса и ведущие цифры нельзя терять. Категория техники — значение из утверждённого списка. Описание проблемы — текст ограниченной длины. Время поступления — дата с определённой временной зоной.

Сведения о клиенте и сведения о заявке храните отдельно. Один человек может обратиться по нескольким вопросам. Если заменять предыдущую проблему новой в карточке контакта, история обращений исчезнет. Связанный контакт и отдельная сделка или заявка сохраняют смысл данных.

Заранее согласуйте, какие поля заполняет клиент, какие вычисляет система, а какие оставляет менеджеру. Ответственный сотрудник обычно определяется правилом маршрутизации. Нейросеть не должна выбирать его из произвольного текста, если есть утверждённый график.

Проверяемые критерии

Критерий «бот работает быстро» замените наблюдаемым условием. Например: на согласованном тестовом наборе каждое подтверждённое обращение получает номер заявки; обязательные поля заполнены; повтор того же события не создаёт новую карточку; неуспешная запись попадает в очередь разбора. Время обработки задайте после пробной сборки и проверки внешних сервисов.

Для ответов нейросети критерии должны отражать задачу. Если модель классифицирует сообщения, проверяют совпадение с размеченными категориями и корректную передачу неопределённых случаев человеку. Красивый ответ не является доказательством правильной классификации.

Критерии приёмки не обязаны обещать бесконечную доступность. Отдельно опишите нормальный режим, допустимые ограничения и действия при отказе внешней системы. Так клиент поймёт, чем готовое решение отличается от сервиса, за который подрядчик принимает круглосуточную ответственность.

Правила изменения объёма

После согласования клиент может попросить добавить запись в календарь. Это разумное развитие, но оно меняет исходный результат. Запишите запрос, оцените влияние на сроки и бюджет, предложите отдельный этап. Не спорьте о том, «маленькая» ли функция: оценивать нужно новые состояния и последствия ошибки.

Полезно разделять уточнение и расширение. Исправление неверного отображения согласованного телефона — устранение дефекта. Поддержка второго языка или нового канала — дополнительное требование. Если исходная формулировка неоднозначна, сначала согласуйте толкование и только потом обсуждайте оплату.

Документирование решений

После встречи отправьте краткую запись: задача, результат, ограничения, сведения, которые предоставляет клиент, и вопросы без ответа. Укажите дату и версию. Когда обсуждение продолжается в мессенджере, переносите принятые решения в основную спецификацию, иначе важная договорённость потеряется среди сообщений.

Для «ТехСервиса» можно записать: «Версия 1.2. Контакт подтверждается отдельной кнопкой. При отсутствии района заявка создаётся со статусом “Нужно уточнение”. Окончательная цена сообщается менеджером». Такая запись устраняет сразу несколько будущих споров.

Согласованная спецификация становится основой оценки и тестов. Если вы не знаете, каким опытом подтвердить выполнение пункта, вероятно, он сформулирован слишком расплывчато. Исправьте его до разработки: после запуска двусмысленность обойдётся дороже.

Глава 4. Данные и события без глубокого программирования

Из чего состоит цепочка

Триггер — событие, которое начинает выполнение. Действие — операция с данными или внешней системой. Условие определяет, какая ветка будет выполнена. Визуальная платформа соединяет эти элементы, но логика остаётся такой же, как в обычной программе.

Пример: клиент отправил сообщение — триггер получил событие; цепочка извлекла текст; условие проверило, завершён ли сбор контакта; затем система либо задала уточнение, либо создала заявку. Число блоков само по себе ничего не говорит о качестве. Один блок может скрывать сложную операцию, а десять — выполнять простую последовательность.

Каждое выполнение должно иметь прослеживаемый идентификатор. Это технический номер, по которому связывают входное событие, запись в журнале и карточку CRM. Он нужен для поиска ошибок и защиты от повторов. Имя клиента для этой цели не подходит: имена повторяются и меняются.

Объекты и массивы

Объект хранит именованные поля. У заявки есть `request_id`, `service_type` и `problem_text`. Массив — упорядоченный набор элементов, например список вложений. Если вы ожидаете один объект, а сервис вернул массив, прямое сопоставление полей может не сработать.

В JSON, распространённом формате обмена, объект заключают в фигурные скобки, массив — в квадратные. Пример учебной записи: `{"request_id":"R-104","service_type":"washer","problem_text":"сливает воду","confirmed":true}`. Значение `true` является логическим признаком, а не текстом «да».

`null` означает отсутствие значения. Пустая строка, ноль и `false` имеют собственный смысл. Нулевая цена может означать бесплатную услугу, `false` — отказ, а пустой телефон — незаполненное поле. Если проверять всё одним условием «значение есть», вы смешаете разные ситуации.

Сопоставление полей

Mapping, или сопоставление, связывает выход одного шага с входом другого. В поле телефона CRM выбирают значение подтверждённого контакта, а не произвольное последнее сообщение. Для названия сделки можно использовать шаблон с категорией и номером заявки.

Проверяйте путь к полю на конкретном событии. В сообщении Telegram текст и контакт находятся в разных структурах. У нажатия кнопки структура снова другая. Поэтому цепочка должна сначала определить тип события и лишь затем читать соответствующие значения.

Ошибки сопоставления часто не вызывают технического отказа. CRM успешно принимает строку с неправильным содержанием. Чтобы заметить проблему, сравнивайте исходное сообщение и полученную карточку, а не только зелёную отметку выполнения модуля.

HTTP

и

API

API — набор правил, по которым одна программа обращается к другой. Визуальный коннектор использует API внутри себя. Если нужного действия нет в готовом модуле, можно настроить HTTP-запрос вручную, не создавая отдельное приложение.

В запросе задают метод, адрес, авторизацию, заголовки и тело. GET обычно читает сведения, POST часто создаёт объект или запускает действие, PATCH обычно изменяет часть объекта. Конкретный смысл определяется документацией сервиса: нельзя выбирать метод только по привычке.

Код ответа помогает понять результат. 401 обычно связан с авторизацией, 403 — с правами, 429 — с ограничением частоты. При этом успешный HTTP-ответ ещё не гарантирует бизнес-успех: некоторые API возвращают описание ошибки внутри тела ответа. Проверяйте оба уровня.

Чтение незнакомого ответа

Возьмите тестовый ответ и найдите три вещи: где расположен полезный объект, где его идентификатор и как сервис сообщает об ошибке. Затем сопоставьте только необходимые поля. Попытка перенести всю структуру сразу увеличивает риск перепутать вложенные значения.

Если поле отсутствует, выясните, это нормальная ситуация или дефект. Например, у нового контакта может не быть email. Для создания заявки это допустимо, если email не обязателен. Но отсутствие идентификатора после операции создания означает, что результат ещё не подтверждён.

Глава 5. Как выбрать

Make

,

n

8

n

или конструктор

Начинайте с ограничений

Для учебного «ТехСервиса» нужны Telegram, хранение состояния диалога, запись в CRM, журнал и уведомления. Перед выбором платформы проверьте наличие этих функций, условия использования, способ оплаты и правила доступа к данным. Привычный инструмент бесполезен, если клиент не может законно и стабильно его использовать.

Не переносите различия платформ в универсальную шкалу «лучше — хуже». В одном проекте важнее скорость первого запуска, в другом — размещение в контролируемой среде, в третьем — удобство редактора диалогов для сотрудников. Составьте небольшой список обязательных свойств и проверьте его на пробном аккаунте.

Отдельно оцените человека, который будет поддерживать систему. Если клиент не имеет администратора сервера, самостоятельный хостинг добавляет обязательства подрядчику. Если внутренний специалист уже обслуживает инфраструктуру, такой вариант может оказаться оправданным.

Где удобен

Make

Make позволяет собирать сценарии из модулей приложений, условий и преобразований. Для первого переноса заявок полезны готовые подключения и доступная история запусков. Во время прототипирования легко увидеть, на каком

шаге значение изменилось или исчезло.

Проверяйте модель потребления ресурсов на реальном сценарии. Количество заявок не равно числу оплачиваемых единиц: одна заявка может пройти несколько модулей, веток и обращений к модели. Точные правила зависят от используемых функций и текущего тарифа. Пробное выполнение даст более полезную оценку, чем расчёт по числу значков.

Уточните ограничения очереди, хранения истории и повторной обработки. Если важные события сохраняются только короткое время, может потребоваться отдельный журнал. Само наличие истории в кабинете ещё не означает, что через месяц вы сможете восстановить любой потерянный запрос.

Где удобен

n

8

n

n8n собирает рабочие процессы из узлов и позволяет использовать готовые интеграции, HTTP-запросы и выражения. Он подходит, когда требуется более подробное управление данными или размещение в собственной инфраструктуре. Возможность настройки не отменяет необходимость обслуживания этой инфраструктуры.

Облачный вариант и самостоятельное размещение сравнивайте по полной стоимости. Во втором случае учитываются сервер, резервные копии, обновления, защита доступа и

время администратора. Бесплатность отдельной редакции не превращает эксплуатацию в бесплатную услугу.

Перед коммерческим использованием проверьте действующие лицензионные условия. Разработка автоматизаций для клиента и продажа доступа к размещённой платформе могут относиться к разным моделям использования. Не обещайте собственный многопользовательский сервис только потому, что удалось установить приложение.

Где удобен

Telegram

-конструктор

Конструктор диалогов полезен, когда основная работа состоит из меню, вопросов, переменных и передачи оператору. Проверьте поддержку нужных типов событий, внешних запросов, экспорта данных и работы с несколькими ботами. Функции, доступные в рекламной демонстрации, могут зависеть от тарифа.

При выборе создайте короткий пробный диалог: категория услуги, контакт, подтверждение, передача данных во внешний сценарий. Затем измените ответ и повторите нажатие. Если конструктор не позволяет понять, какие сведения хранятся и когда отправляются, сложную CRM-интеграцию будет трудно отлаживать.

Комбинированная архитектура часто оправдана: конструктор ведёт диалог, а Make или n8n обрабатывает заявку. Но каждый дополнительный сервис создаёт отдельную точку

отказа и расходы. Добавляйте компонент только при понятной пользе.

Пробная сборка перед обещанием

Проверьте самый рискованный переход: например, создание связанного контакта и сделки в выбранной CRM. Если он не работает с доступными правами, красивое меню не приблизит проект к завершению. Для проверки достаточно вымышленных данных и ограниченного тестового доступа.

Сохраните результат пробной сборки: вход, выход, обнаруженное ограничение и выбранное решение. Такой технический эксперимент уменьшает неопределённость сметы. Он также помогает объяснить клиенту, почему одна архитектура стоит дороже другой.

Выбранная платформа должна обслуживать конкретный процесс. После проверки ограничений её можно фиксировать в спецификации вместе с зависимостями и регулярными расходами. Это станет основой рабочего окружения, а затем — первого сценария.

Глава 6. Рабочее окружение и доступы

Разделение учебной и рабочей среды

Для обучения создайте отдельного бота, тестовую воронку и набор вымышленных контактов. Не проверяйте создание сделок на действующей базе только потому, что доступ уже получен. Некоторые действия запускают уведомления, роботов CRM и другие процессы, последствия которых не видны в вашем сценарии.

Удобно различать три состояния проекта: прототип, приёмочный стенд и рабочая среда. Прототип может быть упрощённым. Приёмочный стенд должен отражать согласованные правила. Рабочая среда использует реальные доступы и имеет ограничения на изменения. Для небольшого проекта это могут быть отдельные сценарии и боты, а не три дорогих сервера.

В названиях указывайте назначение и среду: «Заявки — тест» и «Заявки — рабочий». Одного цвета или расположения папки недостаточно. Когда исполнитель устал или клиент переключил аккаунт, понятное имя снижает вероятность ошибочного запуска.

Кто владеет аккаунтами

Рабочий бот, CRM и подписки желательно оформлять на клиента. Интегратор получает роль с необходимыми правами. Это упрощает передачу: при завершении договора заказчик сохраняет систему, а не зависит от вашего личного аккаунта.

Если по договорённости инфраструктурой владеет подрядчик, отдельно опишите оплату, экспорт, прекращение обслуживания и доступ к данным. Клиент должен понимать, какие части останутся у него после окончания сотрудничества. Подмена этого решения устным «потом перенесём» создаёт коммерческий риск.

Не просите основной пароль, если сервис поддерживает отдельного пользователя, роль или интеграцию. Пароль владельца часто даёт полномочия, которых проекту не требуется. Минимальные права уменьшают масштаб возможной ошибки.

Секреты и подключения

Токен бота и API-ключ дают программе право действовать от имени соответствующей интеграции. Их хранят в предусмотренном платформой механизме credentials или connections, а не в тексте заметок, таблице заявок и скриншоте для портфолио.

Секрет может находиться не только в заголовке запроса. У входящего вебхука Битрикс24 чувствительная часть входит в адрес. Поэтому журналирование полного адреса способно раскрыть доступ. При демонстрации скрывайте и параметры авторизации, и такие адреса.

Если секрет попал в публичный материал, недостаточно удалить картинку. Отзовите или замените доступ, обновите подключение и проверьте работу. Старый секрет мог быть скопирован раньше, чем материал исчез. Порядок замены должен быть известен до первого инцидента.

Реестр зависимостей

В небольшом служебном документе перечислите систему, владельца аккаунта, назначение подключения, ответственного за оплату и порядок восстановления. Значения секретов туда не вставляйте. Реестр нужен, чтобы при сбое найти человека, который способен восстановить доступ.

Добавьте сведения о сроке подписки и контакте администратора. Просроченная оплата платформы и программный дефект могут выглядеть одинаково для клиента: заявки перестали проходить. Без реестра вы потратите время на отладку исправного сценария.

Проверка готовности

Перед сборкой убедитесь, что тестовый бот получает сообщения, CRM допускает создание учебной карточки, внешнее подключение сохраняется и история выполнения видна нужному сотруднику. Затем проверьте выход из аккаунта и повторный вход. Некоторые проблемы проявляются только после истечения короткого сеанса.

Если функция требует тарифа, которого пока нет, зафиксируйте это как зависимость. Не строите коммерческое обещание на временном пробном доступе без обсуждения оплаты после его окончания. Готовое окружение должно воспроизводиться, иначе даже простая автоматизация останется одноразовой демонстрацией.

Глава 7. Проект в

Make

от формы до сделки в Битрикс24

Результат и условия учебной сборки

Соберём сценарий, который получает заполненную заявку через вебхук, сохраняет её в журнал, создаёт сделку в тестовой CRM и возвращает номер созданного объекта. Это первый самостоятельный проект. Здесь нет нейросети и диалога: источник уже собрал необходимые сведения. Позже этот же вход можно подключить к конструктору бота.

Используйте отдельную тестовую воронку Битрикс24. Для первого запуска достаточно сделки с описанием и внешним номером; контакт будет подключён после освоения API в следующем техническом разделе. Такое разделение позволяет сначала проверить передачу заявки, затем добавить поиск человека и связь сущностей.

Понадобятся аккаунт Make с доступом к Webhooks, Data store и HTTP, тестовый портал CRM и REST-доступ с правом создания сделки. Доступность API, региональные ограничения и способы оплаты проверьте до сборки. Не подключайте учебную цепочку к рабочей форме компании.

Шаг 1. Подготовьте входные данные

Создайте новый сценарий и добавьте Webhooks — Custom webhook. Назовите вебхук «ТехСервис учебный вход». Полученный адрес храните в закрытых настройках источника. Он понадобится при отправке тестового POST-запроса, но не должен попадать в портфолио.

В HTTP-клиенте, например Postman, выберите POST, вставьте адрес вебхука и задайте заголовок Content-Type со значением application/json. В Body выберите raw и JSON. Отправьте следующий объект, предварительно включив ожидание данных через Run once:

```
{  
  "request_id": "FORM-001",  
  "source": "test_form",  
  "client_name": "Тестовый клиент",  
  "contact_phone": "TEST-PHONE-001",  
  "service_type": "washer",  
  "problem_text": "Не сливает воду",  
  "confirmed": true  
}
```

TEST-PHONE-001 — специально не телефон. В этом проекте он передаётся как текст в описании, без звонков и создания телефонного поля. Когда подключите настоящий контакт, замените учебный маркер допустимым тестовым зна-

чением, согласованным владельцем CRM. Нельзя брать случайный реальный номер.

Откройте выход вебхука. Убедитесь, что `request_id` и `problem_text` определены как строки, `confirmed` — как логическое значение. Если вместо отдельных полей пришла одна строка, проверьте Content-Type и валидность JSON. Повторно определите структуру по корректному образцу, прежде чем подключать остальные модули.

Шаг 2. Сохраните заявку до обращения к CRM

Создайте Data store с названием `ts_requests`. Ключ записи будет равен `request_id`. Добавьте текстовые поля `state`, `source`, `client_name`, `contact_phone`, `service_type`, `problem_text`, `crm_id` и `last_error`. Для учебного журнала `crm_id` удобно хранить как строку: пустое значение означает, что ID пока не получен.

После вебхука поставьте Router с двумя ветками. На допустимой ветке установите фильтр: `request_id` не пустой, `problem_text` не пустой, `confirmed` равен `true`. На второй ветке задайте обратное условие и Webhook response с кодом 400 и телом `{"status": "invalid_input"}`. Обе ветки не должны срабатывать одновременно; фильтр первой ветки не ставится перед Router, иначе недопустимые данные исчезнут до ответа.

В допустимой ветке добавьте Data store — Check the existence of a record. Выберите `ts_requests` и сопоставьте Key

с `request_id` входного вебхука. Поставьте Router: новая запись идёт на создание, существующая — на чтение через `Get a record`. В новой ветке используйте `Add/replace a record`: `Key` равен `request_id`, `state` равен `received`, остальные поля берутся из входа, `crm_id` и `last_error` пустые.

До теста включите последовательную обработку вебхуков в настройках сценария. В разных версиях настройка может называться `Sequential processing` или `Process data in order`. Она нужна, чтобы две доставки не выполнили проверку отсутствия одновременно. Ограничение действует внутри этого сценария: второй сценарий, ручной запуск или иной записывающий процесс могут обойти его. Во время учебного опыта оставьте единственного исполнителя для `ts_requests`.

Шаг 3. Проверьте доступ к Битрикс24

Создайте входящий вебхук CRM с минимально необходимыми правами. Его базовый адрес содержит домен портала, пользователя и секрет. В HTTP — Make a request укажите POST к методу `crm.item.fields`, Content-Type `application/json` и тело `{"entityTypeId":2}`. Это запрос схемы полей сделки, без создания объекта.

Посмотрите ответ. Зафиксируйте, какие поля обязательны именно в вашем портале. Если воронка, этап или ответственный требуются правилами проекта, получите их реальные идентификаторы и добавьте в конфигурацию. Числа из чужого примера не подходят. Для пустого тестового портала можно сначала проверить создание с полями ниже и системными значениями по умолчанию.

В учебной записи адрес HTTP-модуля складывается из закрытого базового адреса REST-вебхука и имени `crm.item.add`. Не вставляйте секрет в названия модулей. У HTTP-модуля включите разбор JSON-ответа, если такая настройка доступна в его версии. Сначала добейтесь чтения схемы: создание сделки с неработающими правами только усложнит диагностику.

Шаг 4. Сформируйте тело запроса без ручной склейки

Добавьте JSON — Create JSON и создайте структуру по следующему образцу:

```
{  
  "entityTypeId": 2,  
  "fields": {  
    "title": "Учебная заявка FORM-001",  
    "comments": "He сливает воду. Контакт TEST-  
PHONE-001",  
    "originatorId": "ts_form",  
    "originId": "FORM-001"  
  }  
}
```

Затем замените образцы динамическими значениями: title собирается из текста «Учебная заявка» и request_id, comments содержит problem_text и контактный маркер; originId равен request_id. originatorId оставьте постоянным ts_form. JSON-модуль должен экранировать кавычки и переносы строк. Не собирайте тело обычной строкой с подставленным клиентским текстом.

В HTTP-модуле crm.item.add передайте выход Create JSON как тело application/json. Если используемая версия HTTP предлагает собственный конструктор JSON, можно

сопоставить поля в нём вместо отдельного модуля. В обоих случаях массивы, числа и логические значения должны сохранить типы.

Пример сокращённого успешного ответа: `{"result": {"item": {"id": 104, "title": "Учебная заявка FORM-001"}}}`. Число 104 вымышлено. Извлеките `result.item.id` из фактического ответа, затем обновите запись `ts_requests: state` равен `crm_created`, `crm_id` равен полученному ID. Перед обновлением поставьте Router: успешная ветка требует отсутствия поля `error` и непустого `result.item.id`. Другой ответ направьте в `review_required`, сохранив причину; в этой ветке подтверждение `created` не отправляется. При обновлении сохраните исходные поля заявки; не заменяйте их пустыми значениями.

Шаг 5. Верните подтверждённый результат

Последним модулем новой ветки добавьте Webhook response: код 200, Content-Type application/json. Тело сформируйте через JSON-модуль: status равен created, request_id берётся из входа, crm_id — из ответа CRM. Пример результата: {"status":"created","request_id":"FORM-001","crm_id":"104"}

Откройте сделку вручную. Проверьте описание, внешний номер и воронку. Только после этого отправьте FORM-002 с другим текстом. В CRM должны появиться две разные заявки с соответствующими описаниями. Если текст совпал, значит где-то остался образец вместо сопоставления.

В ветке существующей записи Get a record должен получить сохранённый state и crm_id. Если ID есть, верните 200 со status already_created и прежним crm_id. При этом ветка не вызывает crm.item.add. Повторите FORM-001: число сделок не должно увеличиться.

Если запись существует, но crm_id пустой, верните 202 со status review_required и сохранённым request_id. Такой ответ означает необходимость проверки, а не создание сделки. В этой учебной версии незавершённые заявки восстанавливаются вручную по следующей процедуре. Нельзя отправлять их снова на создание только потому, что ID отсутствует в

журнале.

Шаг 6. Обработайте неопределённый результат

На HTTP-модуль создания поставьте обработчик ошибки. При timeout либо неопределённом сетевом отказе обновите state до review_required и запишите короткую причину в last_error. Сначала убедитесь на отключённом тестовом подключении, что обработчик действительно выполняется и исходная запись сохраняется. После записи ошибки верните Webhook response с кодом 202 и телом {"status":"review_required","request_id":"[исходный номер]"}, сформированным через JSON. Завершите обработчик Retry с Automatically complete execution равным No; в прежних интерфейсах аналогичная директива называлась Break. В настройках включите Store incomplete executions. Сохранённое выполнение не запускайте повторно до сверки CRM: ручная кнопка повтора также может создать дубль. Не используйте Resume с вымышленным ответом CRM.

Для сверки создайте отдельный диагностический HTTP-запрос к crm.item.list. Тело: {"entityTypeId":2,"filter":{"=originatorId":"ts_form","=originId":"FORM-001"},"select":["id","title","originatorId","originId"]}. В filter подставьте номер восстанавливаемой заявки. В учебном запросе ожидается малый результат; для общего списка учитывайте пагинацию.

Один найденный объект позволяет перенести его id в журнал и завершить обработку. Несколько объектов требуют ручного разбора. Если ничего нет, проверьте правильность фильтра на заведомо существующей FORM-002, затем повторите чтение после паузы. Решение о повторном создании принимает человек после проверки CRM и журнала. originId помогает поиску, но сам по себе не объявляется уникальным ограничением CRM.

Проверки проекта: новая заявка, повтор прежнего номера, другой номер с тем же текстом, неподтверждённые данные, кавычки в описании и отказ CRM. Сохраните результат каждого опыта. После этого у вас есть воспроизводимая передача формы в CRM с ограниченным режимом восстановления, а не только схема на экране.

Глава 8. Проект

Telegram

-бота в

n

8

n

с журналом и передачей менеджеру

Что делает первая версия

Второй проект принимает одно текстовое обращение, сохраняет его в таблице, отправляет сотруднику и подтверждает получение клиенту. Бот отвечает на /start отдельным пояснением. Фотографии и прочие неподдерживаемые сообщения получают просьбу написать текстом. Нейросеть подключается после проверки этого маршрута.

Здесь одно обычное текстовое сообщение считается отдельным обращением. Это сознательно простой режим: бот не собирает многошаговую анкету, не обещает выезд и не создаёт запись клиента в CRM. Если пользователь отправил три отдельных сообщения, получится три обращения. Для объединения сообщений потребуется состояние диалога, которое рассматривается дальше.

Шаг 1. Создайте бота и получите чат сотрудника

В BotFather создайте отдельного учебного бота командой `/newbot`. В `n8n` добавьте Telegram credentials и сохраните токен там. Создайте Telegram Trigger, выберите событие Message и эти credentials. Отправьте боту `/start`, запустив ожидание тестового события.

Откройте JSON-выход триггера. Запишите `message.chat.id` вашего тестового сотрудника: это получатель служебных уведомлений. Сотрудник должен сам открыть бота и отправить сообщение до первой попытки отправки ему. Для клиента `chat_id` всегда извлекается из его события, а служебный получатель задаётся отдельной конфигурацией.

Во время учебного опыта не используйте один токен одновременно в SendPulse, Make и `n8n`. У бота одна регистрация `webhook`, и новое подключение может заменить прежнее. Для собственного хостинга `n8n` должен иметь корректный публичный HTTPS-адрес, доступный Telegram.

Шаг 2. Разделите типы входа

После триггера добавьте Switch с режимом Rules. Первое правило: `message.text` равно `/start`. Второе правило: `message.text` существует и не равно `/start`. Для остальных данных используйте `fallback output`. Назовите выходы «Начало», «Обращение» и «Другой формат». Настройте отправку только в первый совпавший выход, если редактор предлагает вывод во все совпавшие.

В «Начало» добавьте Telegram — Message — Send Message. Chat ID задайте выражением `{{ $json.message.chat.id }}`. Текст: «Опишите проблему одним сообщением. Бот передаст обращение менеджеру. Дату и стоимость ремонта подтверждает сотрудник». Отключите разметку сообщения либо используйте обычный текст.

В `fallback` поставьте такой же узел ответа с текстом «Для этой версии нужна текстовая заявка. Напишите описание одним сообщением». В этой ветке таблица обращений не пополняется. Убедитесь, что сообщение с фотографией не попало в правило текста; подпись фотографии обрабатывается отдельно только после сознательного расширения входа.

Шаг 3. Нормализуйте обращение

На текстовой ветке добавьте Edit Fields с именем «Нормализация». В ручном сопоставлении создайте четыре строки: request_id, chat_id, problem_text и state. Для request_id используйте выражение `{{ 'tg_' + String($json.update_id) }}`, для chat_id — `{{ String($json.message.chat.id) }}`, для problem_text — `{{ $json.message.text }}`, для state задайте received.

Такой request_id подходит только этому режиму «одно обновление — одно обращение» и одному боту. При подключении нескольких ботов добавьте собственный постоянный префикс каждого бота. В многошаговом диалоге номер заявки создаётся отдельно: update_id там определяет доставку конкретного события, а не всю анкету.

После нормализации поставьте IF: длина problem_text после trim больше нуля и не превышает учебный предел 1000 символов. Для второго условия можно использовать `{{ $json.problem_text.length }}`. В false-ветке попросите сократить описание, не записывая заявку. Предел выбран для упражнения; рабочий предел согласуйте с бизнесом.

Шаг 4. Создайте постоянный журнал

В разделе Data tables создайте ts_telegram_requests. Добавьте строковые столбцы request_id, chat_id, problem_text, state и manager_message_id. Системные id и даты таблицы не заменяют request_id. Пустой manager_message_id означает, что подтверждённого результата уведомления пока нет.

После IF добавьте Data Table: Resource Row, Operation Get, таблица ts_telegram_requests. Conditions: request_id Equals { { \$json.request_id } }; Limit поставьте 1. В настройках узла включите Always Output Data, чтобы отсутствие строки не остановило новую заявку. После него поставьте IF с проверкой существования id возвращённой строки.

В true-ветке прочитайте state. Для notified верните клиенту прежний request_id и остановите маршрут. Для остальных состояний отправьте обращение на ручную сверку: повтор создания записи и автоматическая повторная отправка менеджеру в этой версии запрещены. Так потеря подтверждения не становится причиной серии одинаковых уведомлений.

Конец ознакомительного фрагмента.

Текст предоставлен ООО «Литрес».

Прочитайте эту книгу целиком, [купив полную легальную версию](#) на Литрес.

Безопасно оплатить книгу можно банковской картой Visa, MasterCard, Maestro, со счета мобильного телефона, с платежного терминала, в салоне МТС или Связной, через PayPal, WebMoney, Яндекс.Деньги, QIWI Кошелек, бонусными картами или другим удобным Вам способом.