12+

АНДРЕЙ ЛАДЫГИН

ИИ-агенты без программирования

СДЕЛАЙ, А НЕ СОВЕТУЙ



Андрей Ладыгин

ИИ-агенты без программирования. Сделай, а не советуй

«Издательские решения»

Ладыгин А.

ИИ-агенты без программирования. Сделай, а не советуй / А. Ладыгин — «Издательские решения»,

Чат-боты отвечают. Агенты — делают. Эта книга — практикум по сборке ИИ-агентов без программирования. Вы научитесь ставить задачу в терминах результата, подключать инструменты, настраивать доступы, выстраивать проверки и контролировать выполнение. Шесть проектов проведут вас от первого агента-оператора до цифрового сотрудника, который закрывает задачу и отчитывается о результате. Не совет — действие.

© Ладыгин А.
© Издательские решения

Содержание

Введение 6
Почему совет — не работа 7
Что делает агент 8
Почему n8n 9
Как читать книгу 10
Кому подойдёт книга 11
Кому не подойдёт 12
Часть I. Архитектура агента, который действует 13
Глава 1. Что такое агент в n8n 13
Проблема 13
Идея решения 13
Схема workflow 13
Разбор по шагам 14
Промпт для модели 14
Ограничения и риски 15
Глава 2. Задача, а не промпт 16
Проблема 16
Идея решения 16
Схема работы 16
Разбор по шагам 16
Промпт для модели 17
Ограничения и риски 18
Глава 3. Инструменты агента: руки, глаза и доступы 20
Проблема 20
Идея решения 20
Схема workflow 20
Разбор по шагам 20
Промпт для модели 21
Ограничения и риски 22
Часть II. n8n с нуля — без программирования 24
Глава 4. Интерфейс n8n и первые сценарии 24
Проблема 24
Идея решения 24
Разбор интерфейса 24
Триггеры: что запускает сценарий 25
Чтение данных между нодами 25
Expressions без кода 25
Типичные ошибки новичка 26
Практика 26
Результат главы 27
Конец ознакомительного фрагмента. 28

ИИ-агенты без программирования
Сделай, а не советуй

Андрей Ладыгин

© Андрей Ладыгин, 2026

ISBN 978-5-0071-5782-7
Создано в интеллектуальной издательской системе Ridero

Введение

Понедельник, 9:40. Менеджер открывает чат с ИИ и пишет: «Помоги разобрать входящие за выходные, там условно 60 писем». ИИ отвечает бодро: «Конечно! Вот план: отсортируйте письма по важности, ответьте на срочные, остальные перенесите в папку». Совет хороший. Проблема в том, что менеджер всё равно садится и делает это руками. ИИ посоветовал — работу сделал человек.

Этот разрыв знаком каждому, кто пробовал автоматизировать работу через ИИ. Модель отлично формулирует, что нужно сделать, и почти никогда не делает это за вас. Она не открывает почту, не сортирует письма, не создаёт задачи, не отвечает клиенту. Она рассуждает о работе, а не выполняет её.

Почему совет — не работа

Совет живёт в чате. Как только вы закрыли окно, результат исчез. Письма остались неразобранными, задачи не созданы, клиент не получил ответ. Совет — это текст. Работа — это изменение состояния: в почте, в CRM, в календаре, в таблице.

Совет не знает вашего контекста. Модель не видит ваши правила, ваших клиентов, ваши критерии срочности, ваши запреты на автоматические ответы. Она даёт универсальный ответ на частную задачу.

Совет не проверяет результат. Он не скажет: «Я разобрал 60 писем, 12 срочных, 3 не смог классифицировать, посмотри их». Он скажет «готово» — и вы поверите. А через день найдёте два пропущенных письма от ключевого клиента.

Что делает агент

Агент — это не чат. Это сценарий, в котором модель получает задачу, вызывает инструменты, проверяет результат и отчитывается. Схема простая:

получил задачу → спланировал шаги → вызвал инструменты (почта, поиск, таблица, CRM) → проверил результат → выполнил действие → отчитался человеку.

Разница видна на простом примере. Задача: разобрать входящие за выходные.

Советчик отвечает: «Вот план, как разобрать письма».

Исполнитель делает: открывает почту, читает 60 писем, делит их на срочные, обычные и мусор, создаёт задачи в CRM по срочным, готовит черновики ответов, спорные письма откладывает на проверку и присылает отчёт: «60 писем. 12 срочных, задачи созданы. 38 обычных, ответы в черновиках. 7 мусор. 3 спорных — вот они, посмотри».

Вы получаете не план, а результат.

Где агенты полезны, где опасны

Полезны там, где задача повторяется, данные структурированы, а результат можно проверить. Разбор почты, сортировка заявок, досье на клиента, отчёт за неделю, черновики ответов в поддержке, контент на согласование. Во всех этих случаях агент экономит часы, а человек проверяет результат по понятным критериям.

Опасны там, где ошибка дорого стоит и её трудно заметить. Финансовые решения, юридические документы, кадровые вопросы, автоматические ответы клиентам без проверки. Модель может уверенно выдать неверный результат: она не отличает факт от правдоподобной догадки. В таких сценариях обязателен человек, который подтверждает действие.

Два правила, которые действуют на протяжении всей книги.

Если агент работает с персональными данными, обработка должна соответствовать 152-ФЗ: согласие, цели, сроки хранения и защита данных.

Если агент создаёт рекламный контент, его публикация требует маркировки интернет-рекламы: токен ERID и передача данных в ЕРИР через ОРД.

Есть третий класс задач: агент справится, но это того не стоит. Если задача выполняется раз в месяц и занимает десять минут, автоматизация отнимет больше, чем сэкономит. Агент нужен там, где работы много и она повторяется.

Почему n8n

n8n — среда, в которой агент собирается из готовых блоков без программирования. Логика собирается из готовых блоков, а данные подставляются выражениями без кода. Вы соединяете триггер (что запускает агента), ноды (что он делает) и инструменты — sub-ноды Tool, которые дают агенту доступ к действиям и данным. Модель подключается как sub-нода Chat Model к агенту или цепочке и принимает решение, что делать дальше.

n8n даёт контроль. Вы видите каждое действие агента в журнале запусков и можете разобрать любой шаг. Вы ставите проверку перед рискованным шагом, ограничиваете доступ к данным, получаете отчёт о каждом запуске. Это не чёрный ящик, который сам решает, что делать с вашей почтой.

И ещё одно: n8n не требует программирования. Вам нужно понимать логику процесса, а не синтаксис. Если вы умеете объяснить сотруднику, как обрабатывать заявку, вы сможете собрать агента, который делает это сам.

Как читать книгу

Каждая глава заканчивается работающим агентом. Вы не просто читаете, вы собираете: подключаете ноды, вставляете промпты, проверяете результат на тестовых данных. К концу книги у вас будет шесть агентов полной сборки и ещё несколько сценариев схемами в приложении 5. Все они выполняют реальные задачи.

Не пропускайте практику. Теория здесь только для того, чтобы вы понимали, почему агент устроен именно так. Если пропустить сборку, книга превратится в набор советов — ровно то, от чего мы уходим.

Кому подойдёт книга

Предпринимателям, которые хотят снять с себя рутину. Руководителям, которым нужно, чтобы процессы работали без постоянного контроля. Аналитикам, маркетологам и специалистам поддержки, которые каждый день делают одно и то же руками. Всем, у кого есть n8n или готовность его освоить и кто устал от чат-ботов, дающих советы.

Кому не подойдёт

Тем, кто ищет готовое решение под ключ без настройки. Тем, кто хочет автоматизировать всё сразу. Тем, кто не готов проверять результат и нести за него ответственность.

Ответственность за последствия работы агента остаётся на человеке или организации, которая его эксплуатирует. Агент делает работу, но отвечает за неё человек.

Что вы соберёте в книге

Агент фиксации задач: принимает текст, определяет, является ли он задачей, записывает в таблицу.

Агент-исследователь: собирает досье на клиента или компанию из открытых источников.

Агент-оператор заявок: классифицирует обращение, создаёт задачу в CRM, готовит ответ клиенту.

Агент-контентщик: готовит черновик статьи или поста и отправляет на согласование.

Агент поддержки и продаж: отвечает на типовые вопросы и передаёт сложные случаи человеку.

Финальный агент: под ваш личный рабочий процесс, собранный по всем правилам книги.

Ещё два сценария — отчётник и контролёр задач — даны схемами в приложении 5.

Дальше — к первому агенту. Начнём с архитектуры: что делает агента исполнителем, а не советчиком.

Часть I. Архитектура агента, который действует

Глава 1. Что такое агент в n8n

Проблема

Вы уже пробовали автоматизировать работу. Скорее всего, это выглядело так: настроили бота, который отвечает на частые вопросы, или цепочку, которая пересылает письма в таблицу. Через неделю выяснилось, что бот отвечает не на те вопросы, а цепочка ломается на письмах без темы. Вы вернулись к ручной работе и решили, что автоматизация — это для крупных компаний с программистами.

Проблема была не в инструменте. Проблема в том, что вы собрали чат-бота или пересылку, а не агента. Чат-бот отвечает. Пересылка передаёт. Агент — решает задачу и выполняет её.

Идея решения

Агент в n8n — это workflow, в котором есть пять обязательных элементов. Если хотя бы одного нет, это не агент, а просто нода с моделью.

Первый элемент — триггер. Он запускает агента: по расписанию, по вебхуку, по новому письму, по сообщению в чате.

Второй — модель. Она принимает решение, что делать: классифицировать, извлечь, ответить, выбрать инструмент.

Третий — инструменты. Это sub-ноды Tool, которые дают модели доступ к действиям и данным: прочитать почту, найти в интернете, записать в таблицу, создать задачу, отправить сообщение.

Четвёртый — память. Sub-нода Memory хранит контекст: что уже сделано, о чём говорили, какие правила применяли. В первом агенте мы её пока не подключаем.

Пятый — правила и контроль. Это не отдельная нода, а система: критерии готовности, ограничения доступа, точки, где человек подтверждает действие, обработка ошибок.

Главный принцип: агент без инструментов — это просто чат. Модель без доступа к почте, таблице и CRM может только рассуждать. Дайте ей инструменты — и она начнёт действовать.

Схема workflow

Триггер → AI Agent (с sub-нодами Chat Model, Memory, Tool) → проверка результата → действие → отчёт человеку.

Для первого агента схема проще: Schedule Trigger → AI Agent с sub-нодой Tool Google Sheets → Telegram.

Schedule Trigger запускает агента каждый час. AI Agent получает задачу и инструмент — записать строку в таблицу. Если задача распознана, агент вызывает Tool, строка появляется в таблице. Затем Telegram отправляет уведомление вам.

Важно: Tool — это sub-нода. Она не соединяется стрелкой с другими нодами, а подключается к AI Agent в разъёме инструментов.

Разбор по шагам

Schedule Trigger. Запускает workflow по расписанию. Настройте интервал: для первого агента достаточно «каждый час». На выходе — пустой объект с датой запуска.

AI Agent. Основная нода. На вход получает задачу текстом. Внутри — модель, память и инструменты. На выходе — результат работы: подтверждение действия и структурированные данные.

Chat Model (sub-нода). Подключается к AI Agent. Выбираете доступную универсальную модель, которая будет принимать решения. Chat Model не работает сама по себе, только как часть агента или цепочки.

Memory (sub-нода). Подключается к AI Agent. Хранит историю запусков. В этой главе не подключаем: задача разовая, контекст не нужен. Но знать о ней нужно, иначе агент будет каждый раз начинать с нуля.

Tool Google Sheets (sub-нода). Даёт агенту действие: записать строку в таблицу. Описание инструмента: «Запиши строку в таблицу Задачи с колонками дата, задача, статус». Модель читает описание и решает, когда вызвать инструмент.

Telegram (нода уведомления). Получает результат агента и отправляет вам сообщение: «Агент выполнил задачу, строка добавлена». Это точка контроля: вы видите, что произошло, без открытия n8n.

Промпт для модели

Ты — агент, который фиксирует рабочие задачи в таблице.

На вход ты получаешь текст задачи. Твоя работа:
1. Определить, что это задача, а не вопрос или спам.
2. Извлечь суть задачи одной строкой.
3. Определить статус: новая, в работе, выполнена.
4. Если это задача, вызови инструмент Google Sheets и запиши строку.
5. Верни результат строго в JSON.

Формат ответа:
{
«is_task»: true или false,
«summary»: «суть задачи одной строкой»,
«status»: «новая» или «в работе» или «выполнена»,
«reason»: «почему ты так решил, одна фраза»
}

Если текст не является задачей, не вызывай инструмент. Верни is_task: false и объясни причину в reason.

Не выдумывай данные. Если не хватает информации, верни status: «новая» и укажи в reason, чего не хватает.

Правило «верни JSON» ключевое. Без него модель вернёт свободный текст, и вы не сможете предсказуемо проверить результат и построить уведомление. JSON делает ответ агента проверяемым.

Ограничения и риски

Модель ошибается в классификации. Письмо от коллеги «посмотри отчёт» может быть задачей, а может быть просьбой. На первом этапе ставьте ручную проверку: агент записывает в таблицу, вы просматриваете раз в день.

Модель может выдумать данные. Если в тексте нет срока, модель может его придумать. Промпт выше запрещает это, но проверяйте результат на первых запусках.

Schedule Trigger сработает, даже если задач нет. Агент вернёт is_task: false и не вызовет Tool. Это нормально, но следите, чтобы таблица не заполнялась мусором.

Tool может вернуть ошибку. Настройте Retry on Fail для AI Agent: 3 попытки, задержка 1000 мс, максимальная задержка 5000 мс. Если ошибка повторяется, агент остановится, и вы увидите её в журнале.

Human-in-the-loop

В этом агенте человек проверяет таблицу раз в день. Если агент ошибся с классификацией, вы исправляете статус вручную. Через неделю вы увидите, где модель ошибается чаще, и скорректируете промпт.

Не давайте агенту право удалять строки или менять статус выполненных задач. Только добавлять новые.

Практика

— Создайте новый workflow в n8n.

— Добавьте Schedule Trigger, установите интервал «каждый час».

— Добавьте AI Agent. Подключите к нему Chat Model.

— Добавьте sub-ноду Tool типа Google Sheets. Опишите действие: «Запиши строку в таблицу Задачи с колонками дата, задача, статус».

— Вставьте промпт из раздела выше в поле системного сообщения агента.

— Настройте Retry on Fail для AI Agent: 3 попытки, задержка 1000 мс.

— Добавьте ноду Telegram. В сообщении используйте выражение с результатом агента.

— Сохраните и запустите вручную. Проверьте: появилась ли строка в таблице, пришло ли уведомление.

— Откройте журнал выполнения и убедитесь, что Tool Google Sheets был вызван.

— Запустите ещё раз с текстом, который не является задачей. Убедитесь, что строка не добавилась и Tool не вызывался.

Результат главы

Вы собрали первого агента, который принимает текст, определяет, является ли он задачей, и записывает результат в таблицу через инструмент. У агента есть триггер, модель, инструмент и уведомление. Пока без памяти и без сложной логики, но принцип понятен: агент действует, а не советует.

Чек-лист

— Workflow запускается по расписанию и вручную.

— У агента есть минимум один инструмент — sub-нода Tool.

— Промпт требует JSON и запрещает выдумывать данные.

— Результат записывается в таблицу, а не остаётся в чате.

— Настроен Retry on Fail.

— Вы получаете уведомление о каждом запуске.

— В журнале видно, вызывался ли Tool.

Глава 2. Задача, а не промпт

Проблема

Вы открываете чат с моделью и пишете: «Помоги с отчётом». Модель отвечает бодро, но бесполезно: предлагает структуру, спрашивает, какой отчёт нужен, просит уточнить данные. Вы уточняете, она снова спрашивает. Через двадцать минут вы всё равно садитесь и делаете отчёт руками, а переписка остаётся в истории как напоминание о потерянном времени.

Знакомо? Причина не в модели. Причина в том, что вы дали ей промпт, а не задачу. Промпт — это просьба поговорить. Задача — это описание результата и условий, при которых он считается выполненным. Пока вы формулируете «помоги», «подскажи», «расскажи», любой ИИ будет советовать. Как только вы формулируете «сделай вот это, вот в таком виде, по таким критериям», тот же ИИ начнёт действовать.

Идея решения

Задача для агента состоит из четырёх частей. Если хотя бы одной нет, агент начнёт угадывать, и результат будет непредсказуемым.

Первая часть — результат. Что должно появиться на выходе: строка в таблице, задача в CRM, письмо в черновиках, файл с отчётом. Не «проанализируй», а «верни таблицу с колонками». Не «ответь клиенту», а «подготовь черновик ответа и сохрани в папку Черновики».

Вторая часть — входные данные. Что агент получает: текст письма, ссылку, файл, запись из CRM. Чем точнее вы опишете вход, тем меньше агент будет додумывать.

Третья часть — критерии готовности. Как понять, что работа выполнена. «Письмо классифицировано, если в нём есть тема и отправитель». «Отчёт готов, если в нём есть цифры за неделю и подпись руководителя». Без критериев агент будет считать готовым всё, что угодно.

Четвёртая часть — ограничения. Что агенту нельзя: удалять данные, отвечать клиенту без подтверждения, менять статус выполненных задач, выходить за пределы таблицы. Ограничения — это не про недоверие к модели, а про предсказуемость.

Формула простая: результат + вход + критерии + ограничения = задача, которую можно выполнить и проверить.

Схема работы

Задача текстом → агент разбирает её на шаги → вызывает инструменты → проверяет результат по критериям → выполняет действие или возвращает на доработку → отчитывается.

На каждом шаге агент отвечает на вопрос: «Это соответствует критериям готовности?» Если нет — не двигается дальше. Если да — выполняет действие и фиксирует результат.

Разбор по шагам

Результат. Опишите его так, как будто объясняете сотруднику. Не «сделай красиво», а «верни таблицу с колонками дата, клиент, сумма, статус». Не «подготовь ответ», а «подготовь текст ответа до 100 слов, без обещаний скидок, с подписью менеджера». Чем конкретнее результат, тем меньше вариантов у модели.

Вход. Опишите, что именно агент получает. Если это письмо — укажите, какие поля важны: отправитель, тема, тело, вложения. Если это заявка — какие поля обязательны, а какие

могут отсутствовать. Если поле отсутствует, агент должен не выдумывать, а вернуть статус «требует уточнения».

Критерии готовности. Сформулируйте их как проверки. «Готово, если: в результате есть все обязательные поля; сумма совпадает с исходными данными; статус один из трёх допустимых». Критерии должны быть проверяемыми: либо да, либо нет. Не «качественно», не «хорошо», не «профессионально». Критерии могут проверяться моделью или отдельной нодой вроде IF — так надёжнее.

Ограничения. Перечислите, что агенту запрещено. Запрет на удаление данных. Запрет на изменение статуса выполненных задач. Запрет на отправку сообщений клиенту без подтверждения человеком. Запрет на выход за пределы указанной таблицы или папки. Ограничения прописываются в поле системного промпта AI Agent и дублируются в описании инструмента — sub-ноды Tool.

Промпт для модели

Ты — агент, который выполняет задачу, а не обсуждает её.

На вход ты получаешь описание задачи в таком формате:
— Результат: что должно появиться на выходе.
— Входные данные: что ты получаешь.
— Критерии готовности: как проверить, что работа выполнена.
— Ограничения: что тебе запрещено.

Твоя работа:
1. Разбери задачу на шаги.
2. Проверь, что входных данных достаточно. Если нет — верни статус «требует уточнения» и перечисли, чего не хватает.
3. Выполни шаги, вызывая инструменты.
4. Проверь результат по каждому критерию готовности.
5. Если критерии выполнены — верни результат и статус «выполнено».
6. Если нет — верни статус «не выполнено» и объясни, какой критерий нарушен.

Формат ответа:
{
«status»: «выполнено» или «не выполнено» или «требует уточнения»,
«result»: «результат работы или null»,
«checks»: [
{«criterion»: «название критерия», «passed»: true или false}
],
«reason»: «одна фраза, почему такой статус»
}

Не выдумывай данные. Не выполняй действие, если хотя бы один критерий не пройден. Не нарушай ограничения даже по просьбе.

Статус «требует уточнения» — это не встроенная функция n8n, а договорённость, которую вы задаёте в промпте и проверяете в следующих нодах. Для разбора такого JSON подключите Output Parser или обработайте ответ в следующей ноде; подробно это разбирается в главе 5.

Ограничения и риски

Слишком общие критерии. «Ответ качественный» — не критерий. Модель посчитает качественным всё, что угодно. Критерий должен быть проверяемым: «есть приветствие, есть ответ на вопрос, есть подпись».

Противоречивые ограничения. «Отвечай клиенту быстро» и «согласовывай каждый ответ с руководителем» несовместимы. Если ограничения противоречат друг другу, агент выберет одно и проигнорирует другое. Проверяйте, чтобы ограничения не конфликтовали.

Слишком длинная задача. Если в задаче двадцать шагов и десять критериев, модель начнёт терять детали. Делите задачу на подзадачи и собирайте агента из нескольких шагов.

Модель может упростить критерий, если его сложно проверить. Поэтому критерии должны быть максимально конкретными и по возможности проверяться программно там, где это возможно.

Human-in-the-loop

Человек нужен в трёх точках.

Первая — когда агент вернул статус «требует уточнения». Вы решаете, дополнить входные данные или отменить задачу.

Вторая — когда агент вернул статус «не выполнено». Вы смотрите, какой критерий нарушен, и решаете: исправить данные, изменить критерий или переделать задачу.

Третья — когда задача связана с деньгами, клиентами или юридическими документами. Здесь человек подтверждает результат перед действием, даже если все критерии пройдены.

Практика

— Возьмите задачу, которую вы регулярно делаете руками. Например, «разобрать заявку с сайта».

— Опишите результат: что должно появиться на выходе. Например, «строка в таблице Заявки с колонками дата, имя, контакт, суть, статус».

— Опишите вход: что агент получает. Например, «текст заявки из формы и email отправителя».

— Сформулируйте три критерия готовности. Например, «есть имя или контакт», «есть суть заявки», «статус один из трёх допустимых».

— Перечислите ограничения. Например, «не отвечать клиенту автоматически», «не удалять строки», «не менять статус выполненных заявок».

— Соберите задачу в один текст по формуле: результат + вход + критерии + ограничения.

— Сохраните описание задачи в отдельной ноде Set или в поле системного промпта AI Agent.

— Вставьте этот текст в промпт из раздела выше и проверьте на трёх примерах: нормальная заявка, заявка без контакта, спам.

— Убедитесь, что агент возвращает статус «требует уточнения» на заявке без контакта и «не выполнено» на спаме, если критерии не проходят.

— Если агент вернул «не выполнено» на нормальной заявке, проверьте, какой критерий не прошёл, и скорректируйте либо критерий, либо промпт.

— Запишите, какие критерии пришлось уточнить после проверки. Это нормально: первая версия задачи почти всегда требует правок.

Результат главы

Вы научились формулировать задачу так, чтобы агент выполнял её, а не обсуждал. У вас есть формула из четырёх частей и промпт, который заставляет модель проверять результат по критериям и не выполнять действие, если критерии не пройдены. Это основа для всех агентов книги: без задачи нет исполнителя, есть только собеседник.

Чек-лист

— Задача описана через результат, а не через процесс.
— Входные данные перечислены явно.
— Критерии готовности проверяемые: да или нет.
— Ограничения сформулированы и не противоречат друг другу.
— Промпт требует статус и проверку по критериям.
— Агент возвращает «требует уточнения», если данных не хватает.
— Агент не выполняет действие, если хотя бы один критерий не пройден.
— Понятно, как разобрать JSON с проверками в n8n.

Глава 3. Инструменты агента: руки, глаза и доступы

Проблема

Агент из первой главы умел одно: записать строку в таблицу. Вы запускаете его на реальной задаче — обработать заявку с сайта — и выясняется, что таблицы мало. Нужно проверить, есть ли уже такой клиент в CRM, создать задачу менеджеру, отправить клиенту короткое подтверждение, поставить напоминание. Агент этого не умеет: у него нет доступа к CRM, почте и календарю.

Вы добавляете доступы. Через неделю обнаруживаете, что агент создал три одинаковые сделки по одной заявке, отправил клиенту письмо без вашего ведома и записал в журнал строку, которую потом никто не может найти. Инструменты у агента появились, а контроля — нет.

Инструмент — это не просто «доступ». Это действие, у которого есть последствия. Поэтому инструменты подключают по правилам: минимальные права, проверка перед действием, защита от повторов, журнал.

Идея решения

Инструменты агента делятся на три группы.

Глаза. Инструменты, которые только читают: поиск в интернете, чтение письма, получение записи из CRM, чтение таблицы. Они ничего не меняют, поэтому их подключать проще.

Руки. Инструменты, которые меняют состояние: записать строку, создать сделку, отправить сообщение, поставить задачу, добавить событие в календарь. Здесь нужны ограничения и проверка.

Доступы. Это не отдельная группа инструментов, а способ их подключения: credentials, права сервисного аккаунта, ограничения по папкам, таблицам и полям. Доступы определяют, что агент технически может сделать, даже если попросит.

Правило простое: сначала дайте агенту глаза, потом руки, и только с ограничениями. Не наоборот.

Схема workflow

Webhook (заявка с сайта) → AI Agent с инструментами: поиск клиента в CRM (глаза), создание задачи (руки), черновик ответа клиенту (руки с подтверждением) → журнал агента → уведомление человеку.

Для первого сценария достаточно трёх инструментов: HTTP Request Tool для проверки CRM, Google Sheets Tool для журнала, Telegram Tool для уведомления. Точные названия Tool-версий нод зависят от версии n8n — уточните в документации.

Разбор по шагам

Webhook. Принимает заявку с сайта. На выходе — поля формы и уникальный идентификатор заявки. Этот идентификатор дальше используется для защиты от повторов. Тестовый URL и продакшен-URL могут вести себя по-разному: проверяйте сценарий на тестовом, перед запуском переключайтесь на продакшен.

AI Agent. Получает заявку и набор инструментов. В системном промпте описано, какие инструменты для чего предназначены и что запрещено.

HTTP Request Tool (глаза). Проверяет, есть ли клиент в CRM по email или телефону. Только чтение. Если клиент найден, агент получает его ID и не создаёт дубль. В описании инструмента укажите только нужные поля: имя, контакт, ID, статус. Точное название Tool-версии уточните в документации n8n.

Google Sheets Tool (руки и журнал). Записывает строку в журнал агента: время, ID заявки, действие, статус. Журнал нужен не для красоты, а чтобы разобрать любой запуск. Отдельно ведите таблицу «Обработанные заявки» с уникальным request_id — по ней проверяются дубли.

Telegram Tool (уведомление). Сообщает вам, что заявка обработана, и прикладывает ссылку на запись в CRM. Это точка контроля.

Черновик ответа клиенту. Агент готовит текст, но не отправляет его. Отправка — отдельный шаг с подтверждением человеком. Это human-in-the-loop для внешней коммуникации.

Порядок действий в промпте: проверка дублей → проверка клиента → подготовка черновика → запись в журнал → уведомление. Если поменять порядок, защита от дублей перестанет работать.

Промпт для модели

Ты — агент обработки заявок с сайта.

На вход ты получаешь заявку: имя, контакт, текст обращения, ID заявки.

Твои инструменты:
1. Поиск клиента в CRM — только чтение. Используй, чтобы проверить, есть ли клиент.
2. Журнал агента — запись строки. Используй для фиксации каждого действия.
3. Уведомление в Telegram — сообщение владельцу процесса.

Порядок работы:
1. Проверь, нет ли уже заявки с таким request_id в таблице «Обработанные заявки». Если есть — верни статус «дубль» и остановись.
2. Проверь, есть ли клиент в CRM по контакту.
3. Если клиент найден — используй его ID, не создавай нового.
4. Если клиент не найден — верни статус «требует создания клиента» и не создавай его сам.
5. Подготовь черновик ответа клиенту до 80 слов, без обещаний скидок и сроков.
6. Запиши в журнал: ID заявки, действие, статус.
7. Отправь уведомление владельцу процесса.

Запрещено:
— отправлять сообщения клиенту самостоятельно;
— удалять записи;
— создавать сделки без подтверждения человека;
— передавать в модель пароли, ключи и токены.

Формат ответа:
{

«request_id»: «ID заявки»,
«client_found»: true или false,
«client_id»: «ID клиента или null»,
«draft_reply»: «текст черновика»,
«actions»: [«список выполненных действий»],
«status»: «обработано» или «требует уточнения» или «требует создания клиента» или «дубль»
}

Ограничения и риски

Дубли. Webhook может сработать дважды, или агент повторит действие после ошибки. Защита — идемпотентность: перед созданием записи проверяйте, нет ли уже строки с таким request_id в таблице «Обработанные заявки». Если есть — не создавайте новую. Это не встроенная функция n8n, а правило, которое вы реализуете проверкой.

Избыточные права. Если сервисный аккаунт CRM имеет право удалять сделки, агент технически сможет это сделать. Выдавайте минимальные права: чтение там, где нужно только чтение; запись — только в нужные поля.

Секреты в промпте. Ключи и токены хранятся в credentials n8n, а не в тексте промпта. Модель не должна видеть пароли. Если инструмент требует ключ, он подставляется n8n автоматически.

Ошибка инструмента. HTTP Request может вернуть таймаут, Google Sheets — ошибку доступа. Настройте Retry on Fail для критических нод: 3 попытки, задержка 1000 мс. Если проверка CRM не удалась, агент не создаёт запись и возвращает статус «требует уточнения».

Лишние данные. Не передавайте агенту всю CRM. Ограничьте выборку полями, которые нужны для задачи: имя, контакт, ID, статус.

Если агент получает и записывает персональные данные, обработка должна соответствовать 152-ФЗ: согласие, цели, сроки хранения и защита данных. Передача персональных данных во внешние сервисы и ИИ-модели требует оценки правовых оснований и защиты канала.

Human-in-the-loop

Человек подтверждает отправку любого сообщения клиенту. Агент готовит черновик, человек проверяет и отправляет. Ответственность за отправленное клиенту сообщение остаётся на человеке или организации, которая его отправила.

Человек создаёт нового клиента в CRM, если агент вернул статус «требует создания клиента». Автоматическое создание клиентов без проверки — риск дублей и ошибок в данных.

Человек разбирает записи со статусом «требует уточнения» и ошибки в журнале. Журнал агента с персональными данными хранится по тем же правилам, что и остальные данные: ограниченный доступ и срок хранения.

Практика

— Создайте новый workflow с Webhook. Настройте тестовый URL.

— Добавьте AI Agent. Подключите Chat Model.

— Добавьте Tool для проверки CRM. Если CRM нет, используйте Google Sheets как имитацию: поиск строки по email.

— Создайте таблицу «Обработанные заявки» с колонкой request_id.

— Добавьте Tool для записи в журнал агента. Колонки: время, request_id, действие, статус.

— Добавьте Tool для уведомления в Telegram.

— Вставьте промпт из раздела выше в системное сообщение агента.

— Настройте Retry on Fail для Tool проверки CRM: 3 попытки, задержка 1000 мс.

— Отправьте тестовую заявку через Webhook. Проверьте: нашёлся ли клиент, появилась ли строка в журнале, пришло ли уведомление.

— Отправьте ту же заявку повторно с тем же request_id. Убедитесь, что агент вернул статус «дубль» и не создал новую запись.

— Отправьте заявку с новым контактом. Убедитесь, что агент вернул «требует создания клиента» и не создал запись сам.

— Откройте журнал выполнения n8n и проверьте, какие инструменты вызывались. Это технический журнал; журнал агента в таблице — бизнесовый, они дополняют друг друга.

Результат главы

Вы подключили агенту глаза и руки: проверку CRM, журнал и уведомление. Научились ограничивать доступы, защищаться от дублей и оставлять след каждого действия. Агент по-прежнему не отправляет сообщения клиенту сам — это делает человек после проверки черновика.

Чек-лист

— У агента есть минимум один инструмент-«глаза» и один инструмент-«руки».

— Доступы выданы по минимуму: чтение отдельно, запись отдельно.

— Ключи хранятся в credentials, а не в промпте.

— Есть таблица «Обработанные заявки» с request_id.

— Настроена защита от дублей по request_id.

— Ведётся журнал агента.

— Настроен Retry on Fail для критических инструментов.

— Отправка клиенту требует подтверждения человеком.

— В журнале выполнения видно, какие инструменты вызывались.

— Добавлены оговорки про персональные данные и ответственность за отправленное сообщение.

Часть II. n8n с нуля — без программирования

Глава 4. Интерфейс n8n и первые сценарии

Проблема

Вы открыли n8n и увидели пустой холст. Слева — панель с нодами, справа — кнопки, сверху — меню. Непонятно, с чего начать, что нажимать и как понять, что сценарий вообще работает. Половина обучающих видео начинается словами «создадим workflow» и заканчивается через двадцать минут готовым результатом, а что произошло между этими моментами — не показано.

Эта глава закрывает разрыв. Мы разберём интерфейс по частям, соберём первый сценарий из пяти нод и научимся читать данные между ними. Без программирования: только ноды, поля и выражения.

Идея решения

Workflow в n8n — это схема из нод, соединённых стрелками. Каждая нода делает одно действие: получает данные, обрабатывает, отправляет. Данные передаются от ноды к ноде в виде объектов. Чтобы собрать сценарий, нужно ответить на три вопроса: что запускает workflow, что происходит на каждом шаге, что получается на выходе.

Интерфейс устроен вокруг этих трёх вопросов. Слева — библиотека нод, в центре — холст, справа — настройки выбранной ноды. Снизу — панель запуска и журнал выполнений. Если запомнить, где что лежит, сборка перестаёт быть загадкой.

Разбор интерфейса

Холст. Центральная область, где вы соединяете ноды. Перетаскивайте ноды из левой панели, соединяйте точки на краях нод. Стрелка показывает направление данных.

Панель нод слева. Здесь ноды сгруппированы по категориям: триггеры, действия, обработка данных, ИИ. В строке поиска можно найти ноду по названию. Если не знаете, какую ноду выбрать, начните с поиска по действию: «webhook», «schedule», «telegram».

Настройки ноды справа. Открываются по клику на ноду. Здесь вы заполняете поля: адрес, метод, таблицу, текст сообщения. Вкладка «Parameters» — основные поля, вкладка «Settings» — дополнительные: повторные попытки, таймауты, заметки. Названия вкладок могут отличаться в зависимости от версии n8n.

Кнопка запуска. Находится на панели workflow; её расположение зависит от версии n8n. Запускает workflow вручную. Для триггеров вроде Webhook ручной запуск открывает тестовый URL, на который нужно отправить запрос.

Журнал выполнений. Список запусков с результатом: успешно, ошибка, сколько времени заняло. Открывается кликом по запуску в панели Executions или по кнопке Executions в меню workflow. Внутри видно, какие ноды сработали и какие данные прошли через каждую.

Сохранение. Workflow сохраняется кнопкой Save. Без сохранения изменения остаются только в браузере и пропадут при закрытии.

Триггеры: что запускает сценарий

Триггер — первая нода в workflow. Без него сценарий не запустится сам.

Manual Trigger. Запуск вручную. Удобен для отладки и первых сценариев. По расписанию он не запускается.

Schedule Trigger. Запуск по расписанию: каждый час, каждый день, каждую неделю. Настройка простая: интервал и время.

Webhook. Запуск по внешнему запросу. Используется, когда сценарий должен срабатывать на событие: заявка с сайта, сообщение от сервиса, нажатие кнопки.

Chat Trigger. Запуск из чат-интерфейса. Используется для агентов, с которыми общаются текстом.

Email Trigger. Запуск по новому письму. Требует подключения почтового аккаунта; настройка зависит от почтового сервиса: для IMAP нужны credentials и адрес сервера, для Gmail — OAuth.

Для первого сценария возьмём Manual Trigger: проще всего проверить, что всё работает.

Чтение данных между нодами

Данные в n8n передаются объектами. Каждая нода получает на вход массив объектов и возвращает массив объектов. Чтобы обратиться к полю из предыдущей ноды, используется выражение.

Выражение — это текст в двойных фигурных скобках, который n8n заменяет на значение. Например, {{json. email}} означает «возьми поле email из текущего объекта». {{json [«текст письма»]}} — то же самое, но для поля с пробелом в названии.

Если нужно обратиться к конкретной ноде по имени, используется $ («Имя ноды').item. json. поле. Это полезно, когда между нодой-источником и текущей нодой есть промежуточные шаги.

Главное правило: перед тем как писать выражение, откройте журнал выполнения и посмотрите, какие поля реально пришли на вход ноды. Не угадывайте названия полей — смотрите.

Expressions без кода

Expression в n8n — это не программирование, а подстановка значений. Три конструкции покрывают почти все задачи новичка.

Первая — обращение к полю: {{$json. поле}}.

Вторая — объединение текста и полей: Новая заявка от {{json.name}}, контакт: {{json. email}}.

Третья — условное выражение, которое читается как «если — то, иначе»: {{$json.status === «новая»? «срочно»: «обычная»}}. Если статус «новая», вернётся «срочно», иначе «обычная».

Больше на старте не нужно. Сложные преобразования делаются отдельными нодами: Set для изменения полей, IF для ветвления, Code для нестандартной логики — но Code в этой книге мы избегаем.

Типичные ошибки новичка

Забыли сохранить workflow. Изменения остались в браузере и пропали. Сохраняйте после каждого шага.

Не тот триггер. Поставили Webhook, но не отправили запрос — сценарий не запустился. Для отладки используйте Manual Trigger.

Поле называется иначе. Пишете {{$json. email}}, а в данных поле Email или e-mail. Откройте журнал и посмотрите точное название.

Нода не подключена. Нода стоит на холсте, но стрелка от предыдущей к ней не проведена. Данные до неё не дойдут.

Запуск без тестовых данных. Нажали Run, но на вход ничего не пришло. Для Manual Trigger добавьте тестовые данные вручную или используйте ноду Set с примером.

Блок «Если не работает»

Если workflow не запускается — проверьте, что первая нода является триггером и выбрана для запуска.

Если нода вернула ошибку — откройте её настройки и проверьте обязательные поля. Красная подсветка показывает, что не заполнено.

Если данные не дошли до ноды — проверьте стрелку подключения и журнал выполнения: возможно, предыдущая нода вернула пустой массив.

Если выражение вернуло пустое значение — откройте журнал и посмотрите точное имя поля. Скорее всего, оно отличается от того, что вы написали.

Если Google Sheets вернул ошибку доступа — проверьте, что credentials Google Sheets действительны и у аккаунта есть доступ к таблице.

Если сценарий сработал, но результат не тот — проверьте, какая нода последняя в цепочке и что она возвращает. Часто ошибка не в выражении, а в том, что выбрана не та нода.

Практика

Соберём сценарий: «письмо → извлечение сути → запись в таблицу → уведомление». Без ИИ, чтобы сосредоточиться на интерфейсе. Используйте только тестовые данные, не подставляйте реальные контакты клиентов.

— Создайте новый workflow. Назовите его «Первый сценарий».

— Добавьте Manual Trigger. Это точка запуска.

— Добавьте ноду Set. В ней создайте поле email со значением test@example.com и поле со значением «Прошу прислать коммерческое предложение». Это имитация письма.

— Добавьте ноду Set ещё раз. Переименуйте её в «Извлечение сути». Создайте поле summary со значением «Запрос коммерческого предложения». Поле email пробросьте из предыдущей ноды через выражение {{$json. email}}.

— Перед подключением Google Sheets и Telegram настройте credentials в n8n: для Google Sheets — доступ к таблице, для Telegram — токен бота.

— Добавьте ноду Google Sheets. Подключите таблицу с колонками email, summary, дата. Настройте запись строки: email из предыдущей ноды, summary из предыдущей ноды, дата через выражение {{$now}}. Если формат даты не подходит, задайте его в настройках поля.

— Добавьте ноду Telegram. Настройте сообщение: «Новая заявка: {{json.summary}} от {{json. email}}».

— Соедините ноды стрелками: Manual Trigger → Set → Set «Извлечение сути» → Google Sheets → Telegram.

— Сохраните workflow.

— Нажмите Run. Откройте журнал выполнения и проверьте, что прошли все ноды.

— Проверьте таблицу: появилась ли строка. Проверьте Telegram: пришло ли сообщение.

— Откройте журнал и посмотрите, какие поля пришли на вход каждой ноде. Запишите точные названия полей.

— Измените значение в первой ноде Set на другой текст. Запустите снова. Убедитесь, что сценарий отработал с новыми данными.

В реальном сценарии вместо первого Set будет Email Trigger, а вместо второго — AI Agent. Но логика та же: получить данные, извлечь суть, записать, уведомить.

Результат главы

Вы разобрали интерфейс n8n, собрали первый сценарий из пяти нод и научились читать данные между ними. Поняли, что такое триггер, выражение и журнал выполнения. Дальше любая нода будет добавляться по той же логике: что на входе, что на выходе, как проверить.

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.