

AI-АГЕНТЫ НА GEMINI

Как создавать, настраивать и запускать собственных AI-агентов

Практическое руководство

От первого запроса до работающего агента в реальных сценариях

Содержание

Введение	1
Часть I. Понять агентов	4
Глава 1. Что такое AI-агент и чем он отличается от чат-бота	5
Глава 2. Как «думает» и действует агент	8
Глава 3. Почему именно Gemini	10
Глава 4. Из чего состоит агент	12
Часть II. Подготовка рабочего места	14
Глава 5. Получаем доступ к Gemini за пять минут	15
Глава 6. Первый запрос: как разговаривать с моделью	17
Глава 7. Системная инструкция: должностная инструкция агента	19
Глава 8. Параметры модели: тонкая настройка поведения	21
Часть III. Собираем агента по кирпичикам	23
Глава 9. Промпт-инжиниринг для агентов	24
Глава 10. Память агента	26
Глава 11. Инструменты: даём агенту руки	28
Глава 12. Планирование, циклы и самопроверка	31
Глава 13. Подключение внешних сервисов	33
Часть IV. Живые сценарии	35
Глава 14. Агент-SMM: автопостинг и ведение соцсетей	36
Глава 15. Агент поддержки клиентов	41
Глава 16. Агент-ресёрчер и аналитик	43
Глава 17. Бизнес-ассистент: почта, календарь, документы	45
Глава 18. Агент для продаж и интернет-магазина	47
Глава 19. Личный агент для повседневных задач	49
Часть V. Без кода	51
Глава 20. No-code: агент без единой строчки кода	52
Часть VI. Запуск и эксплуатация	54
Глава 21. Развёртывание: от ноутбука до облака	55
Глава 22. Стоимость, лимиты и экономия	56
Глава 23. Безопасность, приватность и этика	58
Глава 24. Типичные ошибки и как их избежать	60
Заключение	62
Приложение А. Чек-лист запуска агента	63
Приложение В. Шпаргалка по Gemini API	64
Приложение С. Глоссарий	66
Приложение D. Полезные ресурсы	67

Введение

Эта книга о том, как заставить искусственный интеллект не просто разговаривать, а работать — доводить задачи до результата, пока вы занимаетесь другими делами.

Ещё несколько лет назад «поговорить с нейросетью» означало задать вопрос и получить ответ. Удобно, но пассивно: модель отвечала — и замирала в ожидании следующего вопроса. Сегодня всё иначе. Языковые модели научились **принимать решения, пользоваться внешними инструментами и выполнять цепочки действий**. На этой способности выросло целое направление — **AI-агенты**.

AI-агент — это программа, которая получает цель и самостоятельно ищет путь к ней. Она рассуждает, планирует шаги, вызывает нужные сервисы (поиск в интернете, отправку письма, публикацию поста, запрос к базе данных), проверяет промежуточный результат и продолжает работу, пока задача не будет выполнена. По сути, это сотрудник, которого вы описываете словами, а не нанимаете по трудовому договору.

В центре этой книги — **Gemini**, семейство языковых моделей от Google. Gemini хорошо подходит для агентов по трём причинам: у него удобный и современный программный интерфейс, встроенная поддержка **вызова функций** (function calling) — механизма, который превращает модель в «руки» агента, а также щедрый бесплатный тариф, которого достаточно, чтобы собрать и обкатать первого агента, не потратив ни рубля.

Для кого эта книга

Я писал её так, чтобы она была полезна трём разным читателям одновременно:

- **Разработчику**, который хочет быстро разобраться в архитектуре агентов и получить рабочие примеры кода на Python, не продираясь через сотни страниц документации.
- **Предпринимателю и специалисту**, который видит, сколько рутины можно снять с людей, и хочет понять, что реально автоматизируется уже сегодня, сколько это стоит и с чего начать.
- **Человеку без опыта программирования**, который готов разобраться пошагово. Для него в книге есть отдельная часть про no-code — сборку агентов вообще без единой строчки кода.

Если вы относите себя к последней категории — не пугайтесь фрагментов кода. Их можно пропускать: рядом с каждым примером есть объяснение обычными словами, что именно происходит и зачем. А в части V показано, как собрать похожего агента мышкой в визуальном конструкторе.

Чему вы научитесь

К концу книги у вас будет не набор разрозненных фактов, а цельная картина и практический навык. Вы сможете:

Объяснить простыми словами, что такое агент, чем он отличается от чат-бота и где он реально приносит пользу, а где — лишняя сложность.

Получить доступ к Gemini, написать первый запрос и настроить поведение модели под конкретную задачу.

Дать агенту память, инструменты и доступ к внешним сервисам — от Google-таблиц до соцсетей.

Собрать несколько готовых агентов из живых сценариев: SMM-агента для автопостинга, агента поддержки, агента-аналитика, бизнес-ассистента.

Запустить агента так, чтобы он работал без вас — по расписанию, в облаке, с контролем расходов и безопасности.

Как устроена книга

Материал идёт от простого к сложному, шестью частями.

- **Часть I. Понять агентов** — что это такое, как агент «думает» и из каких деталей состоит.
- **Часть II. Подготовка рабочего места** — доступ к Gemini, первый запрос, настройка модели.
- **Часть III. Собираем агента по кирпичикам** — промпты, память, инструменты, планирование, внешние сервисы.
- **Часть IV. Живые сценарии** — шесть подробных примеров агентов для реальных задач, включая соцсети.
- **Часть V. Без кода** — как собрать агента в визуальных конструкторах.
- **Часть VI. Запуск и эксплуатация** — развёртывание, деньги, безопасность, типичные ошибки.

В конце — приложения: чек-лист запуска, шпаргалка по API, глоссарий и подборка ресурсов. К ним удобно возвращаться, когда книга уже прочитана, а агент собирается.

Небольшое предупреждение о версиях

Мир ИИ меняется очень быстро. Названия моделей, цены и мелкие детали интерфейсов будут обновляться уже после выхода книги. Поэтому упор здесь сделан на **принципы**, которые живут годами: как устроен агент, как он рассуждает и действует. Конкретные команды приведены в актуальном на момент написания виде, но перед запуском всегда сверяйтесь с официальной документацией — в приложении есть ссылки.

И последнее. Лучший способ читать эту книгу — с открытым ноутбуком. Не обязательно повторять каждый пример, но собрать хотя бы одного простого агента своими руками стоит: именно в этот момент абстрактные идеи превращаются в навык. Поехали.

Часть I. Понять агентов

Прежде чем собирать агента, важно понять, что мы вообще собираем. Эта часть — про идею и устройство. Кода здесь почти нет, зато после неё всё остальное ляжет легко.

Глава 1. Что такое AI-агент и чем он отличается от чат-бота

Представьте двух помощников. Первый умеет только отвечать на вопросы: вы спрашиваете — он говорит. Спросили «какая погода в Москве?» — он честно ответит, что не знает, потому что у него нет доступа к прогнозу, но зато красиво расскажет, как в принципе устроена погода. Второй помощник другой: услышав тот же вопрос, он сам открывает сервис погоды, смотрит данные и возвращает вам конкретный ответ — плюс 12 градусов, к вечеру дождь, возьмите зонт.

Первый — это **чат-бот**. Второй — **агент**. Разница не в вежливости и не в уме, а в способности **действовать**.

Определение

AI-агент — это программа на основе языковой модели, которая получает цель и самостоятельно выстраивает и выполняет последовательность действий для её достижения, используя доступные ей инструменты и память.

Разберём ключевые слова этого определения по частям, потому что в них — вся суть.

- **Получает цель.** Агенту не говорят «сделай шаг 1, потом шаг 2». Ему ставят задачу: «подготовь и опубликуй пост о нашей новой услуге». Как её решить — агент придумывает сам.
- **Самостоятельно выстраивает действия.** Он раскладывает цель на подзадачи: придумать текст, подобрать хэштеги, выбрать время, отправить в нужную соцсеть.
- **Использует инструменты.** У агента есть «руки»: функции, которые он может вызвать, — публикация поста, поиск в сети, запрос к базе, отправка письма.
- **Использует память.** Он помнит контекст: о чём был прошлый пост, как зовут клиента, какие правила у бренда.

Пять признаков настоящего агента

Чтобы не путать агента с обычным ботом, которому просто прикрутили красивое слово в названии, держите в голове пять признаков. Чем их больше — тем «агентнее» ваша система.

Автономность. Агент действует без пошагового контроля человека. Вы задаёте цель, он сам решает, как её достичь.

Целенаправленность. У него есть понятная задача-результат, а не просто «поболтать».

Использование инструментов. Он умеет вызывать внешние функции и сервисы, а не только генерировать текст.

Память и контекст. Он опирается на прошлые шаги и внешние данные, а не отвечает «с чистого листа» каждый раз.

Итеративность. Он проверяет промежуточный результат и при необходимости повторяет или корректирует действия.

Наглядное сравнение

Свойство	Чат-бот	AI-агент
Что делает	Отвечает на сообщение	Выполняет задачу до результата
Действия	Только текст	Вызывает инструменты и сервисы
Инициатива	Ждёт вопрос	Сам планирует шаги
Память	Часто в пределах диалога	Кратко- и долгосрочная
Контроль	Нужен человек на каждом шаге	Работает автономно по цели
Пример	«Расскажи про наш продукт»	«Опубликуй анонс продукта в трёх соцсетях»

Откуда у модели способность действовать

Сама по себе языковая модель по-прежнему умеет только одно: предсказывать следующий кусочек текста. Волшебства нет. Способность «действовать» появляется благодаря простому, но мощному приёму: модели дают список инструментов и учат её **отвечать не текстом для человека, а командой для программы.**

Вместо того чтобы написать «сейчас я проверю погоду», модель выдаёт структурированный ответ вроде «вызови функцию ``get_weather`` с параметром ``city = Москва``». Программа-обёртка (её называют **оркестратором** или **исполнителем**) видит эту команду, действительно вызывает функцию, получает результат и возвращает его модели. Модель смотрит на результат и решает, что делать дальше: вызвать ещё один инструмент или сформулировать финальный ответ. Этот цикл «модель ↔ инструменты» и превращает болтуна в работника.

Главная мысль главы

Агент — это языковая модель плюс три вещи: цель, инструменты и цикл, в котором она рассуждает, действует и проверяет результат. Всё остальное в этой книге — детали того, как правильно собрать эти три вещи вокруг Gemini.

Когда агент нужен, а когда — нет

Важная и честная оговорка: агент — не всегда правильный ответ. Если задача решается одним запросом к модели («перепиши это письмо вежливее»), никакой агент не нужен — это лишняя сложность, лишние расходы и лишние места, где что-то может сломаться.

Агент оправдан, когда выполняются хотя бы два из условий ниже:

- Задача состоит из **нескольких шагов**, которые заранее не всегда известны.

- Нужен доступ к **внешним данным или действиям** — сеть, база, отправка сообщений.
- Задача **повторяется** много раз, и её хочется снять с человека.
- Результат нужно **проверять и при необходимости переделывать**.

Держите этот фильтр в голове на протяжении всей книги. Он убережёт вас от соблазна строить космический корабль там, где хватило бы самоката.

Глава 2. Как «думает» и действует агент

В основе почти любого агента лежит один и тот же повторяющийся цикл. Разные фреймворки называют его по-разному, но суть одна: **наблюдение** → **рассуждение** → **действие** → **наблюдение результата** — и так по кругу, пока цель не достигнута. Разберём его на живом примере.

Цикл агента на примере

Задача: «Узнай, сколько сейчас стоит доллар, и, если он дороже 90 рублей, напиши мне в Telegram предупреждение». Вот как агент проходит цикл.

Наблюдение. Агент видит цель и список доступных инструментов: ``get_usd_rate()`` и ``send_telegram(text)``.

Рассуждение. «Чтобы ответить, мне нужен текущий курс. Сначала вызову `get_usd_rate``».

Действие. Агент вызывает ``get_usd_rate()``. Оркестратор выполняет функцию и возвращает: 92.4.

Наблюдение результата. Агент видит: курс 92.4, это больше 90.

Рассуждение. «Условие выполнено, надо отправить предупреждение».

Действие. Вызывает ``send_telegram("Доллар подорожал до 92.4 Р")``.

Завершение. Инструмент отработал, цель достигнута — агент формулирует итог для человека.

Обратите внимание: агент **не знал заранее**, придётся ли ему отправлять сообщение. Это решилось по ходу, на основе полученных данных. Именно эта способность ветвиться в зависимости от результата отличает агента от простого сценария «если — то», записанного программистом вручную.

Три популярные схемы рассуждения

За годы развития агентов сложилось несколько типовых «схем мышления». Знать их полезно: они помогают понимать, почему агент ведёт себя так, а не иначе.

ReAct: рассуждай и действуй

Самая распространённая схема. Модель чередует мысли и действия: сначала коротко рассуждает («мне нужно узнать X»), затем вызывает инструмент, смотрит на результат, снова рассуждает — и так далее. Название ReAct — это сокращение от Reasoning + Acting. Большинство агентов на Gemini работают именно так, потому что function calling ложится на эту схему идеально.

Планирование заранее

Агент сначала строит полный план из шагов, а потом выполняет его по пунктам. Подходит для задач, где последовательность более-менее предсказуема: «собери отчёт из трёх источников и пришли в PDF». Минус — если реальность отклонилась от плана,

агент может упрямо идти по устаревшему маршруту, поэтому план полезно пересматривать.

Рефлексия: проверяй сам себя

После выполнения агент критически смотрит на свой результат: «Я точно ответил на вопрос? Нет ли ошибок?» — и при необходимости переделывает. Это дороже (больше запросов к модели), но заметно повышает качество там, где цена ошибки высока.

Совет практика

Не выбирайте схему заранее из любви к красивым словам. На практике вы почти всегда начинаете с простого ReAct, а планирование и рефлекссию добавляете точно — только к тем шагам, где без них качество проседает.

Что агент видит на самом деле

Полезно один раз честно представить, что «видит» модель на каждом шаге. Это не магия и не сознание — это просто длинный текст (его называют **контекстом**), который мы каждый раз собираем и отправляем модели. В нём обычно четыре части:

- **Системная инструкция** — кто такой агент, какая у него роль и правила.
- **Описание инструментов** — какие функции доступны, что они делают, какие принимают параметры.
- **История диалога и действий** — что уже было сказано и сделано, какие пришли результаты.
- **Текущий запрос или цель** — что нужно сделать прямо сейчас.

Когда вы это осознаёте, многое становится понятным. Например, почему агент «забывает» что-то: просто эта информация не попала в контекст. Или почему он не пользуется инструментом: возможно, вы плохо описали, что этот инструмент делает. Практически вся отладка агентов сводится к вопросу: «А что именно сейчас видит модель?»

Глава 3. Почему именно Gemini

Языковых моделей сегодня много, и хороший агент можно построить на разных. Но у Gemini есть набор качеств, которые делают его особенно удобным стартом — особенно если вы только входите в тему.

Gemini коротко

Gemini — семейство моделей от Google DeepMind. Внутри семейства есть модели разного размера и назначения. Их принято делить по «весовым категориям»:

- **Flash-модели** — быстрые и дешёвые. Идеальны для агентов: большинство шагов агента — это простые решения «какой инструмент вызвать», и гонять ради них тяжёлую модель незачем.
- **Pro-модели** — более мощные и «умные», но медленнее и дороже. Их берут для сложного рассуждения, анализа, генерации длинных и качественных текстов.
- **Специализированные модели** — для генерации изображений, работы со звуком и видео, встраивания текста в векторы (пригодится для памяти — об этом в главе 10).

Практическое правило: **соберите агента на Flash-модели, а на Pro переключайте только те шаги, где реально не хватает качества.** Это экономит и деньги, и время ответа.

О названиях моделей

На момент написания книги в ходу линейки Gemini 2.5 (Flash и Pro) и более новые модели Gemini 3. Google регулярно выпускает новые версии и меняет названия. Не заучивайте конкретную строку вроде «gemini-2.5-flash» как истину на века — просто помните логику «flash = быстрый и дешёвый, pro = мощный», а точное имя актуальной модели всегда есть в списке в Google AI Studio.

Пять причин выбрать Gemini для агентов

Щедрый бесплатный тариф. Google AI Studio позволяет получить API-ключ и начать эксперименты бесплатно, в пределах суточных лимитов. Для обучения и прототипа этого более чем достаточно — вы соберёте первого агента, не привязывая карту.

Встроенный вызов функций. Function calling поддерживается «из коробки» и удобно ложится на Python: вы описываете обычную функцию, а модель сама понимает, когда и с какими параметрами её вызвать.

Большое окно контекста. Gemini умеет держать в «голове» очень длинные тексты. Для агента это значит, что можно передавать много истории, инструкций и данных, не изобретая сложных ухищрений.

Мультимодальность. Модель понимает не только текст, но и изображения, звук, документы. Агент может, например, посмотреть на скриншот и решить, что делать.

Встроенные инструменты Google. Есть готовые возможности вроде поиска в Google прямо из модели (grounding) и выполнения кода — их не нужно писать самому.

Два способа работать с Gemini

Технически к Gemini можно обращаться двумя путями, и важно с самого начала не запутаться.

- **Gemini Developer API (через Google AI Studio).** Самый простой путь. Заходите на сайт, получаете ключ, пишете несколько строк кода. Это то, что мы используем в книге по умолчанию.
- **Vertex AI / корпоративная платформа (через Google Cloud).** Более «взрослый» путь для компаний: расширенные настройки безопасности, интеграция с облаком, командная работа. Логика та же, отличается способ подключения и авторизации.

Для обучения и большинства проектов малого бизнеса хватает первого пути. Когда агент вырастет в серьёзный продукт с требованиями по безопасности — присмотритесь ко второму.

Новые «агентные» интерфейсы Google

Помимо классического вызова модели, Google развивает специальные интерфейсы и готовых агентов (например, для глубоких исследований). Они удобны, но быстро меняются. В книге мы учимся собирать агента из базовых кирпичей — function calling и цикла рассуждения. Освоив это, вы легко разберётесь в любой новой «агентной» надстройке, потому что под капотом у всех она устроена одинаково.

Глава 4. Из чего состоит агент

Соберём общую картину. Любой агент, каким бы сложным он ни был, складывается из пяти деталей. Если держать их в голове, любой чужой (и свой) агент разбирается по полочкам за минуту.

Пять деталей агента

1. Модель — «мозг»

Языковая модель (у нас — Gemini), которая рассуждает и принимает решения. Она не хранит состояние сама по себе и не выполняет действий — она только на основе полученного контекста решает, что сказать или какой инструмент вызвать. Всё остальное — обвязка вокруг неё.

2. Инструменты — «руки»

Набор функций, которые агент может вызвать: отправить письмо, опубликовать пост, найти данные, посчитать, записать в таблицу. Инструменты — это то, что связывает агента с реальным миром. Без них он остаётся собеседником, каким бы умным ни был.

3. Память — «блокнот»

Место, где хранится контекст: короткая память (что было в этом диалоге), долгая (факты о пользователе, правила, прошлые задачи) и, часто, векторная база знаний — для поиска по большому объёму документов. Память — это то, что превращает агента из «одноразовки» в помощника, который помнит вас и ваши предпочтения.

4. Оркестратор — «нервная система»

Программа, которая крутит цикл: собирает контекст, отправляет его модели, ловит команду на вызов инструмента, выполняет инструмент, возвращает результат модели, повторяет. Это тот самый механизм из главы 2. В простых случаях его пишете вы сами в десяток строк, в сложных — берёте готовый фреймворк.

5. Цель и правила — «должностная инструкция»

Системная инструкция, которая описывает, кто такой агент, что он должен и чего не должен делать, каким тоном общаться, где его границы. Это самая недооценённая деталь: именно от качества инструкции чаще всего зависит, будет агент полезным или капризным.

Как детали работают вместе

Соединим всё в одну картинку словами. Пользователь ставит **цель**. **Оркестратор** упаковывает цель вместе с **правилами**, описанием **инструментов** и нужными кусками **памяти** и отдаёт всё это **модели**. Модель рассуждает и либо просит вызвать инструмент, либо даёт финальный ответ. Если инструмент — оркестратор его выполняет, кладёт результат в память/контекст и снова зовёт модель. Круг замыкается, пока цель не достигнута.

Деталь	Роль	Пример в SMM-агенте
Модель	Принимает решения	Решает, о чём и как написать пост
Инструменты	Выполняют действия	<code>publish_telegram()</code> , <code>get_trends()</code>
Память	Хранит контекст	Тон бренда, темы прошлых постов
Оркестратор	Крутит цикл	Ваш скрипт или n8n-сценарий
Цель и правила	Задают рамки	«Пиши дружелюбно, без кликбейта»

Ориентир на всю книгу

Дальше мы соберём каждую из этих пяти деталей по отдельности, а потом — в живых сценариях — соединим их в работающих агентов. Если в какой-то момент вы почувствуете, что потерялись, вернитесь к этому списку из пяти деталей и спросите: какую из них я сейчас настраиваю?