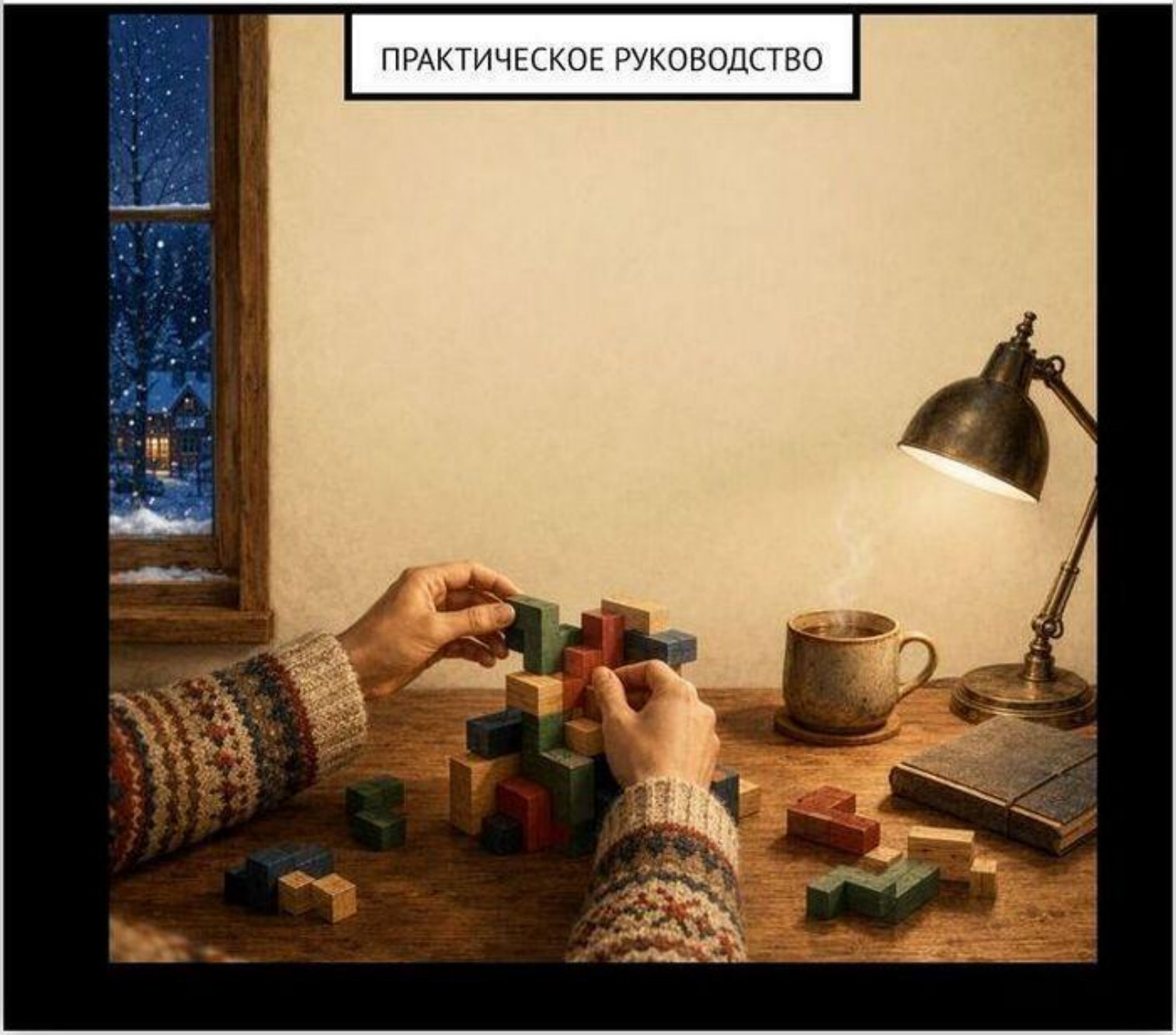
12+

АНДРЕЙ ЛАДЫГИН

Китайские ИИ для вайб-кодинга



Андрей Ладыгин

Китайские ИИ для вайб-кодинга. Практическое руководство

«Издательские решения»

Ладыгин А.

Китайские ИИ для вайб-кодинга. Практическое руководство / А. Ладыгин — «Издательские решения»,

Как перестать зависеть от недоступных зарубежных ИИ и начать создавать работающие проекты? Это практическое руководство по доступному ИИ-стеку, который работает в России без блокировок. Вместо «слепого вайба» вы освоите инженерный подход: критерии приемки, контроль и защиту от технического долга. Пройдите путь от простой считалки до ИИ-консультанта по документам, собрав личный набор инструкций и навыков для агентов.

© Ладыгин А.
© Издательские решения

Содержание

Предисловие 6
Глава 1. Почему это вообще работает — и почему не всегда 10
Глава 2. Что к чему: карта инструментов 14
Глава 3. Сколько это стоит на самом деле 18
Глава 4. Начинаем с малого: считалка, которая считает 22
Конец ознакомительного фрагмента. 24

Китайские ИИ для вайб-кодинга
Практическое руководство

Андрей Ладыгин

Иллюстрации ИИ GPT Image 2
Иллюстрация обложки ИИ GPT Image 2

© Андрей Ладыгин, 2026

ISBN 978-5-0071-5651-6
Создано в интеллектуальной издательской системе Ridero

Предисловие

«В книге упоминаются названия программных продуктов, товарные знаки и инструменты (Z, ZCode, GLM, Ollama и др.), которые принадлежат их respective правообладателям. Упоминание этих инструментов носит исключительно информационный и обучающий характер и не является рекламой или официальным партнерством. Автор не аффилирован с компаниями-разработчиками данных решений».

Вторник. К утру аккаунт перестал работать: оплата не прошла, и вместе с ней исчез доступ к инструменту, в котором вы разрабатывали последние месяцы. Калькулятор для клиента, страница, которую уже почти доделали, привычка работать «как привыкли» — всё это к обеду испарилось. А в пятницу что-то сдавать.

Если эта ситуация знакома вам хотя бы в уменьшенном виде — аккаунт заблокировали, карта не прошла, интерфейс перестал отвечать, цена за месяц выросла в три раза и стала абсурдной, — эта книга писалась с вами в виду.

Что это за книга

Это не обзор моделей. И не каталог инструментов. Это последовательность задач, которые растут в сложности, и у каждой задачи есть три вещи: понятная постановка, артефакт, который вы собираете руками, и критерии, по которым результат принимают или не принимают.

К концу книги у вас будут собраны:

— считалка, которая считает, и у которой проверяется каждый итог;

— генератор ссылок с метками и файловый помощник, который не ломается на кривых данных;

— страница, которую не стыдно показать, — и понятные критерии, почему она готова;

— валидатор таблицы и панель показателей, которая показывает только то, от чего зависит решение;

— интерактивный тренажёр, где логика ветвления выверена, а не «на глаз»;

— помощник по документам, который отвечает со ссылкой на источник;

— личный набор: файл инструкций для ИИ, копилка запросов, три собственных навыка, два подключения к внешним данным, схема распределения работы между агентами;

— конвейер, который исследует, пишет и проверяет контент без постоянного присмотра;

— сервис заявок: приём, сортировка, ответ, хранение;

— опубликованный сайт, который работает, а не лежит;

— и консультант, который знает ваши документы и говорит, откуда взял ответ.

Много. Но это не пятнадцать разрозненных вещей: каждый следующий проект собирается из деталей предыдущих. Считалка из главы 4 уйдёт внутрь лендинга из главы 6. Валидатор из главы 7 станет сердцем панели. Помощник из главы 9 перерастёт в консультанта из главы 18. К финалу у вас не «научились про ИИ» — у вас работающая система.

Что это не такое

Это не курс. Нет уроков, нет домашних заданий, нет «студентов». Это руководство, которое можно открывать с любой главы, если контекст предыдущих глав у вас уже есть.

Это не восторг. Там, где инструмент слаб, мы говорим прямо и показываем, как эту слабость проверить самому, а не верить на слово — ни обзору, ни нам.

Это не пошаговая инструкция по кнопкам. Кнопки меняются быстрее, чем книга выходит в свет. Поэтому в книге принципы, паттерны и, главное, критерии приёмки: то, что остаётся рабочим, когда меняется среда.

Для кого эта книга

Для тех, кто слышал про «вайб-кодинг» и агентную разработку, но не понимает, что из этого реально доступно, как это открывается и сколько стоит.

Для тех, кто уже обжигался: блокировка, оплата, «сегодня работает — завтра нет». Если вы из этой группы, первые главы будут про то, как перестать зависеть от одного инструмента.

Для тех, у кого день состоит из документов, таблиц и отчётов: аналитики, бухгалтеры, юристы, методисты, менеджеры, HR. Вам не нужно становиться разработчиком — нужно уметь ставить задачу, проверять результат и встраивать автоматизацию в свою рутину.

Для предпринимателей, экспертов, маркетологов, которым сайт или калькулятор нужны «на вчера», и фрилансеров, которым нужно портфолио из живых, работающих вещей, а не картинок в Figma.

Главная мысль

Доступный ИИ-стек — это не компромисс. Это полноценный инструмент, который на большинстве практических задач даёт сопоставимый результат при кратно меньших затратах.

Слово «сопоставимый» стоит разобрать, потому что на нём держится вся книга. На типовых задачах — калькулятор, страница, таблица, бот, сервис — доступные модели и среды справляются не хуже. На части задач — очень большие кодовые базы, экзотические интеграции, длинные цепочки рассуждений — разрыв есть, и он реальный. Книга не будет его отрицать. Но решение «справится или нет» должно приниматься на проверке вашей задачи, а не на предубеждении «всё, что не то, — хуже».

Именно этот навык — осознанного выбора инструмента под задачу — книга и собирает. Плюс второй навык: доводить проект до состояния «не стыдно показать заказчику или работодателю». Оба навыка держатся на одной вещи: критериях приёмки. Научиться формулировать результат так, чтобы его можно было принять или не принять, — это половина книги. Вторая половина — всё остальное.

Как читать

Первое правило: собирайте руками. Прочитать про калькулятор — полезно. Собрать калькулятор, который считает, — полезно вдвое больше. Каждая глава содержит практику, и пропускать практику — значит читать вдвое дольше.

Второе правило: сначала критерии, потом результат. В каждой практической главе мы сначала договоримся, как будем проверять, и только потом что-то делаем. «Вроде работает» — не приёмка.

Третье правило: читайте блоки «Где это ломается». Это самые ценные куски главы. Там — типичные ошибки, ограничения инструментов и то, как заметить проблему до того, как она стала чьей-то проблемой на проде. Инструмент, который никогда не ломается, не требует книги.

Четвёртое правило: не перескакивайте через раздел IV. Первые три раздела дают проекты, раздел IV — систему. Если вы остановитесь на разделе III, у вас останутся проекты. Если дойдёте до IV, у вас останется то, на что проекты собираются в следующий раз.

Слово о сроках

Версии моделей и интерфейсы меняются быстрее, чем книга выходит. Все конкретные названия, цены и лимиты в тексте помечены «на момент написания» и рассчитаны на актуаль-

ность примерно год. Через год некоторые цифры устареют — это нормально и заложено. Не устареют: методика постановки задачи, критерии приёмки, структура файла инструкций, паттерны навыков и подключений, схема субагентов. Это тот каркас, который работает с любой моделью и любой средой. Когда придёт новая модель, вы не начнёте с нуля — вы подключите её к своей системе.

Из чего состоит книга

Книга делится на шесть разделов, и каждый добавляет в ваш набор один новый элемент.



Путь по книге. Каждый раздел добавляет в набор новый элемент

Раздел I — разобраться: что это вообще, что есть на рынке и сколько стоит. Раздел II — первые шаги: считалка, ссылки, страница. Раздел III — данные, документы и обучение: таблицы, тренажёр, помощник. Раздел IV — свой набор: инструкции, запросы, навыки, подключения, субагенты. Раздел V — живой сервис: конвейер, заявки, публикация. Раздел VI — финал: консультант по вашим документам.

Что у вас будет в конце

Перечислим честно, без воды. К финалу у вас будет:

— настроенная среда, выбранная под ваши задачи, а не «как у всех»;
— файл инструкций, после которого агент ведёт себя одинаково предсказуемо;
— копилка из двадцати-трёхдесяти повторяемых запросов, каждый со своей ролью;
— три навыка: работа с документами, обработка данных, контент по шаблону;
— два подключения к внешним данным — с явными правами и журналом;
— схема распределения работы между несколькими агентами;
— набор опубликованных, живых проектов;
— консультант, который отвечает по вашей базе документов и показывает источник.

Система важнее проектов. Проекты устареют за год, система будет работать с новыми задачами без пересборки. И с новым инструментом — тоже, когда он появится.

Чек-лист перед стартом

Перед тем как открывать раздел I, проверьте четыре вещи:

— доступный ИИ-стек открыт и оплачен — как именно, будет в главе 2, но хотя бы одна карта/способ оплаты должен работать;

— у вас есть одна реальная задача, которую вы хотите довести до жизни в ближайшие недели, — она станет материалом для практики;

— на практику в каждой главе уходит около часа — закладывайте время;

— вы договорились с собой об одном правиле: нет проекта без критериев приёмки.

Всё, что дальше, — по порядку: сначала разобраться, потом делать, потом превращать сделанное в систему. Начинаем.

Глава 1. Почему это вообще работает — и почему не всегда

Две истории. Первая: человек говорит нейросети «сделай мне сайт» и через сорок минут получает красивую страницу. Он в восторге, показывает коллеге. Коллега пытается добавить форму — ломается. Хозяин просит поправить цены — слетает вёрстка. Третий запрос — и страница превращается в кашу, которую стыдно открывать. Вторая история: тот же человек, та же неделя, но другая постановка. «Калькулятор стоимости. Входы: площадь, цена за квадрат, число рабочих, число дней. Выходы: материалы, работа, итог с запасом десять процентов. Проверка: при площади 45 и цене 3500 итог по материалам — 157500; при двух рабочих, двенадцати днях и ставке 4000 работа — 96000; числа не превращаются в текст». Три итерации диалога — и через два часа у него вещь, которую не стыдно показать заказчику.

Разница между историями не в модели. Модель в обеих — одна и та же. Разница в том, как поставлена задача и как проверяется результат. Эта глава — про неё.

От синтаксиса к намерению

Ручная разработка устроена так: вы пишете код, компилятор проверяет синтаксис, вы ловите ошибки, тестируете, выпускаете. Ваш основной ресурс — время, которое вы тратите на написание и отладку строк.

Работа с агентом смещает этот процесс. Вы описываете намерение — что результат должен делать, на каких входах, что отдавать, — и агент генерирует код. Ваша работа становится другой: сформулировать, декомпозировать, проверить. Написание строк ушло к машине; осталось то, что раньше занимало меньше половины времени, а теперь занимает всё.



Два пути к результату. Время человека смещается от написания кода к постановке задачи и проверке

Из этого следует важный вывод: ценность разработчика в агентной модели определяется не скоростью печати, а качеством формулировок и строгой проверкой. Два разработчика с

одинаковым опытом дадут разный результат: один будет «просить красиво», второй — ставить задачу так, что результат можно принять или не принять по заранее оговорённым критериям.

Вайб-кодинг: что это и чего это не такое

Вайб-кодинг (от англ. vibe — «вайб», настроение) — способ разработки, при котором человек общается с ИИ на естественном языке и принимает результат, а не пишет код сам. Название говорит о главном: в чистом виде это работа «по ощущениям».

Но «по ощущениям» бывает двух сортов. В первом сорте — «чистый вайб» — вы принимаете вывод без проверки: показал демо, красиво, значит, работает. Во втором — структурированная агентная разработка — задача всегда имеет критерии приёмки, результат всегда проверяется по ним, а «красиво» считается только после проверки.

Книга работает со вторым сортом. И вот почему чистый вайб опасен не тем, что он «неправильный», а тем, что он производит хрупкие прототипы. Прототип, который принят без проверки, невозможно развивать: вы не знаете, что в нём работает на самом деле, и каждое изменение — лотерея. Через месяц такой «проект» становится тем самым наследством, которое нельзя трогать.

Для ориентира разложу смежные подходы. No-code и low-code — это конструкторы: вы собираете приложение из готовых блоков. Вы ограничены возможностями конструктора, но и ошибки делаете не вы. AI-assisted — ассистент рядом: вы пишете код, машина дополняет. Вайб-кодинг — третий вариант: вы не ограничены конструктором, но и ошибки уже не «машины» — они ваши, потому что принимаете вы. Отсюда и главная ответственность этого подхода: умение проверять.

Три оси: доступ, стоимость, качество

Когда выбирают инструмент, почти всегда спорят о качестве. Но в наших условиях качество — лишь одна из трёх осей, и не самая первая.

Первая ось — доступ. Открывается ли инструмент напрямую: можно ли зарегистрироваться, оплатить, получить доступ без танцев и ожиданий? Если инструмент недоступен, его качество не имеет значения — это ноль. В разных командах и регионах эта ось выглядит поразному: где-то ограничивают доступ, где-то — оплату, где-то — сам сервис. Поэтому вопрос «а у тебя открывается?» — не брезгливость, а первый технический вопрос.

Вторая ось — стоимость. Сколько стоит месяц реальной работы, а не «от» в прайсе. Расчёт будет в главе 3, здесь главное: стоимость складывается не только из подписки, но и из объёма диалога, и из переделок. Дорогая модель, которую переделывают трижды, дешевле обходится, чем дешёвая, которую переделывают семь раз.

Третья ось — качество. И вот тут главный навык: оценивать качество не по бенчмаркам и обзорам, а на своей задаче. Простейшая методика — три теста. Тест стандартной задачи: то, что инструмент делает «из коробки» — калькулятор, форма, таблица. Тест кривой задачи: данные с лишними строками, пробелами, кривыми форматами — те, что есть у вас в реальной работе. Тест-сюрприз: то, о чём вы не искали в описании — например, «а что будет, если ввести отрицательное число?». Инструмент, который проходит все три на ваших задачах, для вас качественный. Конец спора.

Китайская экосистема в этой системе координат выглядит так. Модели семейства GLM — линейка моделей общего назначения, в ней есть тяжёлые модели для сложных задач и лёгкие — для быстрых и дешёвых; облачные версии открываются напрямую, локальные — работают на вашем компьютере (на момент написания). Среда ZCode в экосистеме z.ai — редактор с агентом: терминал, файлы, управление контекстом; это «рабочее место», о котором будет глава 2. По первой оси у экосистемы преимущество: она открывается в тех условиях, где западные

сервисы уходят. По третьей — на типовых задачах разрыва с лидерами, как правило, нет; на части задач он есть, и дальше — честно о границах.

Мифы о «неполноценности»

Миф первый: «модели не из первой лиги плохо пишут код». Проверка простая: тип задачи. На генерации шаблонов, скриптов, интерфейсов, обработки данных — разрыва чаще всего нет. На поддержке кодовой базы в сотню тысяч строк — есть, и это нормальное свойство любых моделей, а не приговор конкретной экосистеме.

Миф второй: «дешевле — значит хуже». Цена определяется бизнес-моделью и объёмом спроса, а не качеством модели. На практике дешёвая модель, правильно подобранная под задачу, даёт те же результаты, что дорогая, — просто на другом классе задач. Глава 3 научит выбирать.

Миф третий: «контекст короче — всё пропало». Оконечный контекст — это память модели в рамках одного разговора. Да, он ограничен. Но решение — не в «дождаться длинного контекста», а в декомпозиции: разбить задачу на этапы, каждый этап — отдельный контекст с явной передачей (это будет раздел IV). Проблема контекста решается методикой, а не покупкой.

А теперь — честные границы, без мифов в другую сторону. Где доступные модели реально уступают: очень длинные проекты с большой кодовой базой (модель «забывает» удалённые решения), редкие интеграции (мало обучающих данных — больше галлюцинаций в API), задачи, требующие длинных цепочек рассуждений (математика, сложный анализ). В этих зонах работают обходные пути: меньший объём на шаг, явные файлы-памяти, внешние инструменты. Но обходные пути — это уже методика, и она впереди.

Где скорость, а где долг

Подход даёт скорость там, где задача шаблонна или ограничена: автоматизация рутины, одноразовые инструменты, прототипы на реальных данных, контент по шаблону. Здесь экономия времени измеряется не процентами, а порядком.

Долг накапливается там, где «вайб-код» живёт долго без порядка. У handwritten-кода есть один недостаток: вы знаете, что вы там сделали. У агентского — нет: код писала модель, и каждая итерация могла перекраивать то, что работало. Без инструкций, тестов и приёмки код, который «кажется рабочим», — это техдолг, который вы не видите, пока он не лопнет. Раздел IV целиком посвящён тому, как сделать долг управляемым: инструкции, копилка запросов, навыки, подключения.

Практика: переформулируйте свою задачу

Возьмите свою реальную задачу — ту, что вы отмечали в чек-листе предисловия. Не «сайт», а конкретное. И запишите её в три колонки.

Первая колонка — что строим: одно предложение, без «и прочее». Вторая — входы: что пользователь (или вы) вкладывает. Третья — проверки: два-четыре критерия, каждый наблюдаемый, с конкретным числом или поведением.

Пример для ориентира:

Задача: калькулятор стоимости работ.

Входы: площадь ($м^2$), цена материалов за $м^2$, число рабочих,

срок (дни), ставка за день.

Проверки: 1) $45 \times 3500 = 157500$ по материалам;

2) $2 \times 12 \times 4000 = 96000$ по работе;

3) итог включает запас 10% и показан отдельной строкой;

4) пустой ввод поля не даёт ошибку, а подсвечивает поле.

Теперь то же самое отправьте агенту. Вот как звучит задача, которую можно принять:
Сделай калькулятор стоимости работ.
Входы: площадь ($м^2$), цена материалов за $м^2$, число рабочих,
срок (дни), ставка за день.
Выходы: материалы, работа, запас 10%, итог.
Критерии приёмки:
1) при 45, 3500, 2, 12, 4000: материалы 157500, работа 96000,
итог 278850;
2) пустой ввод — подсвечивается поле, без сброса страницы;
3) итог — отдельной строкой, жирным.
Не делай лишнего: только форма и результат, без регистрации.

Обратите внимание на последнюю строку: «не делай лишнего». Это не придирка. Без неё агент с радостью добавит тёмную тему, сохранение в облако и «умные рекомендации» — и каждый лишний кусок станет тем, что потом придётся проверять и поддерживать.

Упражнение на десять минут: возьмите свою задачу, напишите три колонки и одну фразу «не делай лишнего». Если критерии не получаются конкретными — значит, задача ещё не сформулирована. Вернитесь к колонкам, это нормальная часть работы.

Где это ломается

Чистый вайб. Демо красиво — принято. Через неделю невозможно добавить ни форму, ни кнопку, потому что никто не знает, что реально работает. Лечится одним правилом: нет критериев — нет приёмки.

Путаница «доступно» и «качественно». Инструмент открылся и оплатился — значит, «всё в порядке». Доступность — это ось первая, а не оценка. Проверяйте три теста на своей задаче.

Слепая вера в чужие демо. Задача с лендинга и ваша задача — разные. Демо показывает стандартный случай; ваша работа живёт на кривых данных и граничных входах. Проверяйте тест-сюрприз.

Расплывчатая задача. «Сделай лучше», «сделай как у них», «сделай нормально» — агент выполнит что-то, но это будет не то, что вы имели в виду. Расплывчатость — не лень модели, это отсутствие критериев.

Итог главы

— Смена парадигмы: вы проектируете намерение и критерии, а не пишете строки.

— Чистый вайб (принятие без проверки) производит хрупкие прототипы; работа — со структурированным подходом: задача, критерии, проверка.

— Три оси выбора: доступ, стоимость, качество. Качество оценивается на вашей задаче тремя тестами: стандартная, кривая, сюрприз.

— Доступные модели не «неполноценные» — они имеют честные границы: большие кодовые базы, редкие интеграции, длинные цепочки рассуждений.

— Скорость — в шаблонах и ограниченном; долг — в долгоживущем коде без порядка.

— Любая задача формулируется в три колонки: что строим / входы / проверки, плюс «не делай лишнего».

Дальше — вопрос, который вы уже наверняка задаёте: а что конкретно есть, что открывается и как это соотносится между собой. Карта инструментов — в следующей главе.

Глава 2. Что к чему: карта инструментов

Вы слышали имена. Cursor, Claude Code, ZCode, GLM, Ollama — и кажется, что это всё одно и то же: «нейросеть, которая пишет код». Но это не так, и пока вы не видите разницу, вы выбираете инструмент наугад. Эта глава разложит имена по полкам: что есть что, как одно соотносится с другим и с чего именно начинать.

Три слоя — три вопроса

Вся экосистема складывается из трёх слоёв, и каждый отвечает на свой вопрос.

Модель отвечает на вопрос «кто думает». Это сам ИИ — язык, на котором он «понимает» задачу. Модели бывают облачные (вызов по сети, оплата за объём или подпиской) и локальные (работают на вашем компьютере). У моделей разные профили: скорость ответа, цена за объём, качество на разных типах задач, объём контекста — сколько текста модель держит в памяти за один разговор.

Среда отвечает на вопрос «где работает». Это редактор с агентом: окно, в котором вы открываете проект, пишете задачи, видите файлы и терминал, управляете контекстом. Среда не думает сама — она подключает модель к вашему проекту и организовывает процесс: какие файлы показать модели, что сохранить, что запустить.

Агентный слой отвечает на вопрос «как он себя ведёт». Это всё, что делает поведение предсказуемым: постоянные инструкции, повторяемые запросы, навыки, подключения к внешним данным, распределение работы между несколькими агентами. Этот слой — раздел IV книги, и именно он превращает «попросил — получилось» в повторяемый процесс.



Карта инструментов: три слоя и внешний мир. Сначала среда под задачу, затем модель

Из этой схемы следует правильный порядок выбора. Сначала — среда под вашу задачу (где вам удобно работать с файлами и терминалом). Потом — модель под тип задач (быстрая для черновиков, сильная для сложных). И только потом — настройка агентного слоя. Тот, кто начинает с «какая модель самая сильная», выбирает по каталогу, а не по задаче.

Карта экосистемы: что есть что

Платформа z.ai — международный сервис экосистемы китайских моделей: здесь открываются модели семейства GLM, сюда же подключаются среды и подписки (на момент написания). GLM — линейка моделей общего назначения: в ней есть тяжёлые модели для сложных задач и лёгкие — для быстрых и дешёвых; у каждой свой профиль по цене и контексту.

ZCode — среда разработки с агентом в этой экосистеме: редактор, терминал, работа с файлами проекта, управление контекстом. Для книги это «рабочее место», и в большинстве глав мы работаем именно в нём. Названия сменятся — функция останется: среда с редактором, терминалом и агентом.

Локальные модели — те, что работают на вашем компьютере. Их запускает отдельная утилита (по-английски обычно называется Ollama — «загрузил модель, она работает рядом»): вы скачиваете модель и вызываете её без сети. Локальные модели будут в главе 3 — там, где это выгодно по деньгам или по чувствительности данных.

Важно понимать соотношения. Модель — это «мозг», который среда подключает. Среда может работать с разными моделями, и одна и та же модель может работать в разных средах. Агентный слой (инструкции, навыки, подключения) — переносимый: он описывает правила и данные, а не кнопки конкретной среды. Поэтому ваш набор из раздела IV будет работать, даже если среда поменяется, — это и есть защита от «завтра не откроется».

Настройка рабочего места с нуля

Собираем рабочее место по шагам. Каждый шаг имеет наблюдаемый результат — если результат не появился, шаг не завершён.

Шаг 1. Открыть платформу и создать аккаунт. Результат: вы вошли, видите каталог моделей. Частый тупик — оплата: проверьте заранее, какой способ оплаты работает в вашем случае, и имейте запасной.

Шаг 2. Подключить модель к среде. В ZCode это выбор модели в настройках. Результат: среда показывает подключённую модель.

Шаг 3. Создать папку проекта. Отдельная папка для каждого проекта — это не формальность, а граница: агент видит проект, а не весь ваш диск. Результат: папка создана, среда открыта в ней.

Шаг 4. Первый запуск. Напишите самую простую задачу: «Создай файл hello. txt с текстом: рабочее место готово». Результат: файл появился в папке. Если появился — цепочка «вы → среда → модель → файлы» работает.

Шаг 5. Зафиксировать версии. Запишите в отдельный файл: какая модель подключена, какая версия среды, дата. Это ваш «паспорт» рабочего места — по нему вы поймёте, что изменилось, когда что-то перестанет работать. Этот файл станет основой файла инструкций из главы 10.

Если на шаге 4 файл не появился — не уходите дальше. Девять проблем из десяти на этом этапе — в подключении модели или в путях: среда не видит папку. Переподключите, проверьте путь, повторите.

Как выбирать модель: четыре критерия

Критерий первый — скорость. Сколько времени модель тратит на ответ. Для черновиков, переписки, простых скриптов скорость решает больше качества: вы будете ждать, и медленная модель превратит «быструю автоматизацию» в «то же самое, но через час».

Критерий второй — деньги. Цена за объём текста. На типовых задачах разница между «дешёвая» и «дорогая» внутри одной линейки часто меньше, чем разница между «одна итерация» и «три итерации». Считаем в главе 3.

Критерий третий — качество на вашем типе задач. Не «качество по обзорам», а по вашим трём тестам из главы 1: стандартная задача, кривая задача, сюрприз. Прогоните один и тот же тест на двух моделях линейки — и у вас будет ответ, а не мнение.

Критерий четвёртый — контекст. Сколько текста модель держит в памяти. Для коротких задач — не важно. Для «посмотри весь проект и предложи, где чинить» — критично: при коротком контексте приходится декомпозировать (раздел IV), при длинном — можно показать больше за раз.

Задача — инструмент: рабочий список

Вместо «одного правильного инструмента» у вас будет список пар «задача → инструмент». Начните с этого и дополняйте своей практикой (свои пары вы соберёте в упражнении):

— простой скрипт, автоматизация рутины → лёгкая быстрая модель;

— интерфейс, страница, калькулятор → средняя модель линейки;

— работа с таблицами и данными → средняя модель + явные правила в задаче;

— работа с длинными документами → модель с длинным контекстом или декомпозиция на фрагменты;

— закрытые данные, которые нельзя отправлять в облако → локальная модель (глава 3);

— черновик, а потом доработка → дешёвая модель для черновика, сильная — для финальной проверки.

Последняя пара — ключевая и самая недооценённая: черновик дешёвой моделью, проверку — сильной. Так вы платите «по качеству» только за то, что действительно требует качества.

Практика: ваше рабочее место и ваш список

Задание состоит из двух частей.

Первая часть — рабочее место. Пройдите пять шагов настройки. Критерии приёмки: (1) вы вошли в платформу и видите каталог моделей; (2) среда показывает подключённую модель; (3) создана папка проекта; (4) файл hello. txt появился; (5) записан «паспорт» — модель, версия среды, дата. Если все пять выполнились — рабочее место готово.

Вторая часть — список пар. Возьмите три реальные задачи из вашей работы (не «сайт вообще», а конкретные: «собрать отчёт из таблицы», «сделать страницу услуги», «проверить документы»). Для каждой запишите пару: задача → какая модель и почему (по какому из четырёх критериев). Если для какой-то задачи вы не можете ответить «почему» — это нормально, значит, вы ещё не прогнали тесты. Запишите вопрос и прогоните его в главе 4, на первой реальной задаче.

Где это ломается

Гонка за моделями. «Вышла новая, надо пересадиться». Пересадка — это время и риск, а польза — только если новая модель лучше на вашей задаче. Проверяйте три теста, а не читайте анонсы.

Игнорирование лимитов контекста. «Почему он забыл, что мы договорились?» — потому что договорённость вышла за пределы контекста. Это не баг, это свойство. Решение — декомпозиция и файлы-памяти (раздел IV), а не «перепросить».

Привязка к одному инструменту. Всё живёт в одной среде, и когда она «не открывается», всё падает. Защита — переносимый агентный слой (инструкции, навыки, данные в файлах) и запасная среда. Это не паранойя, это страховка.

«Новое = лучшее» и «дорогое = лучшее». Оба срабатывают на каталоге, а не на задаче. На типовых задачах разница часто нулевая, а на ваших — решает профиль, а не цена или дата выхода.

Итог главы

— Три слоя: модель (кто думает), среда (где работает), агентный слой (как ведёт себя).

— Порядок выбора: сначала среда под задачу, потом модель под тип задач, потом настройка поведения.

— Рабочее место собирается за пять шагов, каждый с наблюдаемым результатом; «паспорт» рабочего места — основа будущих инструкций.

— Четыре критерия выбора модели: скорость, деньги, качество на вашей задаче, контекст.

— Вместо одного инструмента — список пар «задача → инструмент», ключевая пара: черновик дешёвым, проверку сильным.

Теперь вы знаете, что есть что. Остался вопрос, который вы откладываете: а сколько это на самом деле стоит в месяц, когда считать не «от» из прайса, а реальные проекты? Считаем в следующей главе.

Глава 3. Сколько это стоит на самом деле

«А это дешевле, правда?» — этот вопрос звучит в каждом разговоре об доступных инструментах. И почти всегда за ним следует «ну, посмотрите, у них там цена за токены в разы меньше». Это правда. Но это не ответ на ваш вопрос. Ваш вопрос — «сколько я буду платить за месяц реальной работы», а не «сколько стоит токен». У этих двух чисел разница — как между ценой бензина и стоимостью содержания машины.

Разберём, из чего складывается настоящая стоимость, и научимся её считать до того, как начать, а не после первого счёта.

Из чего складывается стоимость

Первая и самая заметная статья — токены. Токен — это кусочек текста, на который модель «тратит» вычисление. Модели считают и вход (то, что вы им отправляете: задача, файлы, контекст), и выход (то, что они возвращают: ответ, код). Вход и выход часто стоят по-разному — выход обычно дороже, потому что его модель генерирует, а не просто читает. Отсюда первое правило: длинный контекст, который вы отправляете каждый раз, — это вход, и он тоже стоит денег.

Вторая статья — способ оплаты. Подписка: фиксированная сумма, но с лимитами на объём; если упрётесь в лимит, либо ждёте, либо доплачиваете. Оплата за расход: платите ровно за то, что использовали, без «фикса», но и без предсказуемости. У каждого способа свои ловушки, о них ниже.

Третья — скрытая статья, о которой все забывают. Хостинг, если вы публикуете. Внешние API, если проект что-то вызывает. И самое дорогое — переделки. Переделка — это ещё один проход по модели, то есть ещё токены. Проект, который переделывают семь раз, стоит в семь раз дороже, чем тот, который приняли с третьего. На этом построена половина экономии во всей главе.



Три источника расходов: токены, способ оплаты, скрытые статьи. Переделки — самая дорогая строка.

Как считать до старта

Методика простая, и она держится на трёх оценках.

Оценка первая — объём текста. Сколько слов вы отправляете (вход) и сколько получаете (выход) за один проход. Грубое правило для замера: одно слово ≈ полтора токена (это порядок, а не точность; точность не нужна, нужна оценка). Запишите: «вход на проход — около N слов, выход — около M слов».

Оценка вторая — число проходов. Сколько раз модель будет вызвана на задачу. Это и есть «число итераций» из главы 1. Хорошая постановка — три-пять проходов. Плохая — «посмотрим, сколько придётся».

Оценка третья — цена. Сколько стоит токен входа и выхода у выбранной модели (в каталоге платформы, на момент написания).

Складываем. Стоимость задачи ≈ (вход × цена входа + выход × цена выхода) × число проходов. На этом и строится расчёт. Покажу на двух проектах из книги.

Пример 1: считалка (проект из главы 4)

Задача — калькулятор. Вход на проход: задача плюс файлы проекта, грубо 300 слов. Выход: код и пояснения, грубо 500 слов. Итераций — четыре (первый раз, две правки, приёмка). Подставим условные цены «на момент написания» для средней модели: вход — 0,6 за миллион токенов, выход — 2,2.

Вход: 300 слов ≈ 450 токенов × 4 = 1800 токенов → доли цента. Выход: 500 слов ≈ 750 токенов × 4 = 3000 → около 0,007. Итого проект — копейки. Это и есть ощущение, почему «дешевле» правда: на проектах такого масштаба деньги не в токенах, а в вашей скорости. Но методика важна, потому что на других проектах она покажет другое.

Пример 2: консультант (проект из главы 18)

Задача — сервис, который отвечает по базе из десятков документов. Здесь вход большой: каждый вопрос тащит за собой фрагменты документов. Вход на проход — 2000 слов, выход — 300. Вопросов за месяц — сто. Выход и вход уже не копейки, и начинается игра: какую модель взять на каждый вопрос, можно ли сократить контекст, что отдать дешёвой, что — сильной.

Именно на таких проектах методика окупается. Если пересчитать «в лоб» сильной моделью, месяц выйдет ощутимо дороже, чем нужен. Если черновики и простые вопросы отдать лёгкой модели, а сложные — сильной, — стоимость падает в разы. Решение «какая модель» перестает быть мнением и становится расчётом.

Сравнение с западными сервисами

Здесь честно и с оговоркой: цены меняются, и конкретные цифры устареют первыми (на момент написания). Порядок такой: цена за токены у доступных моделей ниже, чем у лидеров рынка, на порядок — в разы. «В разы» — это и есть то, что вы слышали. Но обратите внимание, что это говорит и что нет.

Это говорит: на одном и том же объёме работы вы заплатите кратно меньше. Это не говорит: «значит, можно работать вдесятеро больше». Объём работы определяет задача, а не цена. Дешевле — значит, вы можете позволить себе больше итераций, больше экспериментов, больше «что если», не глядя на счёт. Это меняет сам способ работы: можно проверять гипотезы, которые при дорогой модели «не окупаются».

Облако или своя машина

Теперь второе решение, которое вы откладываете: платить за облако или держать модель на своём компьютере.



Облако vs локальная модель. Своё железо окупается при больших объёмах и закрытых данных.

Облако: вы не покупаете железо, платите за объём, модель обновляется без вас, качество — на уровне линейки. Минус: данные уходят в сервис, и объём не бесконечно дешёвый.

Локально: модель работает на вашем компьютере (через утилиту вроде Ollama). Плюса два. Первый — данные не покидают машину: это решение по чувствительности, а не по деньгам. Второй — при больших объёмах «цена за токен» уходит к нулю, остаются электричество и амортизация железа.

Когда локально выгодно: (1) данные закрытые и их нельзя отправлять в облако — тогда это вообще не вопрос цены; (2) объём большой и стабильный — тогда «за токен» накапливается в сумму, большую, чем железо. Когда облако выгоднее: объём маленький или прыгающий, нужен максимум качества линейки, нет желания следить за железом.

Своя машина — это ещё и стоимость входа: вам нужно железо с нужной памятью, время на настройку и факт, что локальные модели на части задач уступают облачным флагманам. Поэтому «пересесть на своё из принципа» — плохое решение. Хорошее решение: посчитать ваш объём, решить по чувствительности данных и только потом по деньгам.

Как не переплачивать, не теряя в качестве

Четыре приёма, которые держат стоимость низкой без потери результата.

Приём первый — черновик дешёвым, проверку сильным. Это главный приём. Черновик, переписку, простые правки отдаёте лёгкой модели; финальную проверку сложного — сильной. Вы платите за качество ровно там, где оно нужно.

Приём второй — не тащить весь контекст каждый раз. Длинный вход, повторяющийся в каждом проходе, — это деньги, которые вы платите многократно. Учите агента держать в контексте только нужное; остальное — в файлы (раздел IV).

Приём третий — бюджет на задачу. Прежде чем начать, задайте себе: «сколько проходов я готов потратить». Пять? Семь? Если задача требует больше — значит, она плохо поставлена, и дешевле переформулировать, чем продолжать качать.

Приём четвёртый — не покупать «максимум» для шаблона. На шаблонных задачах (генерация по шаблону, простые скрипты) сильная модель не даст заметно лучшего, а заплатите вы за неё заметно больше. Профили моделей — из главы 2, примените их к деньгам.

Практика: ваш расчёт

Возьмите две свои реальные задачи — одну маленькую (на вечер) и одну серьёзную (на месяц). Для каждой заполните текст расчёта (не таблицу — три строки):

Задача: _____

Вход на проход (слов): __ Выход (слов): __

Число проходов: __ Модель (профиль): __

Итого на задачу ≈ __ Итого на месяц ≈ __

Критерии приёмки упражнения: (1) у вас есть числа, а не «мало»/«много»; (2) для серьёзной задачи вы записали, что отдадите лёгкой модели, а что — сильной; (3) вы записали способ оплаты и лимит подписки, если он есть. Если после расчёта вы понимаете, где ваши деньги и где их можно не тратить, — упражнение сделано.

Где это ломается

Недооценка объёма токенов. «Да это же коротко». Коротко на один проход, но на семь проходов и с длинным контекстом — уже сумма. Считайте проходы, а не «впечатление».

«Дешёвая» модель, требующая больше итераций. Вы сэкономили на цене и потеряли в трёх лишних проходах. Если дешёвая модель постоянно переделывается — это не «она дешевле», это «она дороже». Сравнивайте итог, а не цену за токен.

Забытые хостинг и внешние API. Токены копейки, а домен, сервер и чужой сервис — реальные деньги в месяц, которые не вписали в расчёт. Считайте стоимость «живого» проекта, а не «сделанного».

Оплата «в лоб» сильной моделью всего подряд. На шаблонных задачах вы переплачиваете за качество, которое не используется. Профили моделей существуют для того, чтобы платить по профилю.

Итог главы

— Стоимость = токены (вход и выход) × число проходов + способ оплаты + скрытые статьи (хостинг, API, переделки).

— Переделки — самая дорогая строка; хорошая постановка задачи — это и есть экономия.

— Расчёт до старта: объём текста, число проходов, цена; две реальные задачи в тексте.

— «В разы дешевле» — правда про цену за токен, а не про объём работы; это позволяет больше итераций и экспериментов.

— Локально выгодно по чувствительности данных и при большом стабильном объёме; не «из принципа».

— Четыре приёма: черновик дешёвым / проверку сильным, короткий контекст, бюджет на задачу, профиль под задачу.

Теперь вы знаете, что это стоит. Осталось сделать первую вещь, чтобы всё это не было абстракцией: считалку, которая считает. Следующая глава — первый полный цикл, от формулировки до работающего результата.

Глава 4. Начинаем с малого: считалка, которая считает

Первая программа — это всегда обряд. «Hello, world», «сделай мне сайт», «напиши скрипт» — и где-то между ожиданием и результатом наступает момент, когда непонятно, что именно не так. Может, модель, может, вы, может, задача. Эта глава пройдёт первый полный цикл — от формулировки до работающего результата — на задаче, где нельзя обмануть ни себя, ни модель: калькулятор. У него есть числа, и числа не спорят.

Почему именно считалка

Потому что у неё проверяемый результат. У сайта «нормально или нет» — вопрос вкуса. У калькулятора «45 × 3500 = 157500» — вопрос факта. Пока вы учитесь ставить задачу и принимать результат, вам нужен именно такой объект: где «работает» не означает «красиво», а означает «считает правильно на этих входах».

Считалка — это ещё и кирпичик будущих глав. В главе 6 она уйдёт внутрь лендинга, в главе 7 рядом с ней заживут таблицы, а в главе 16 — заявки. То, что вы сейчас соберёте, будет переиспользовано.



Первый цикл: задача и критерии → генерация → проверка → итерация

Приёмка — по критериям, а не по ощущению.

Постановка: три колонки

Вспомним упражнение из главы 1. Три колонки: что строим, входы, проверки. Для считалки:

Что строим: калькулятор стоимости работ в одном файле.

Входы: площадь ($м^2$), цена материалов за $м^2$, число рабочих,

срок (дни), ставка за день.

Проверки:

1) при 45, 3500, 2, 12, 4000: материалы 157500,

работа 96000, итог 278850 (с запасом 10%);

2) пустой ввод — поле подсвечивается, страница не сбрасывается;

3) итог — отдельной строкой, заметно;

4) ничего лишнего: только форма и результат.

Четвёртый пункт — про «не делай лишнего», и он уже здесь. На первой задаче лишний код бьёт больнее всего: каждый добавленный блок — это ещё что-то, что надо проверять.

Первый запрос

Отправляем задачу агенту. Вот как она звучит целиком:

Сделай калькулятор стоимости работ одним файлом (HTML + CSS + JS).

Входы: площадь ($м^2$), цена материалов за $м^2$, число рабочих,

срок (дни), ставка за день.

Выходы: материалы, работа, запас 10%, итог.

Критерии приёмки:

1) при 45, 3500, 2, 12, 4000: материалы 157500, работа 96000,

итог 278850;

2) пустой ввод — подсвечивается поле, без сброса страницы;

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.