KHOLMOV_BLOG · WWCODING · ГАЙД №1

От нуля до своего сайта или приложения в Telegram

Один вечер, приложение на компьютере и два промпта — без терминала и команд. Скачали программу, вставили два промпта — и работаете.

12 глав | 23 промптов | Без терминала | Windows и macOS | Приложение в комплекте | +5 $ бонус

MX

Максим Холмов

WWCODING · 22 года в IT · 240+ проектов

Содержание

1. Что это за гайд
2. Шаг 0. Из чего состоит система
3. Первый результат за 5 минут
4. Терминал: 8 команд, которых достаточно
5. Скиллы: что это и почему это главный усилитель
6. Промпт-грамотность: 5 правил
7. Проект «Деньги»: что получится
8. 20 идей: как читать таблицу
9. Telegram Mini App: что это и почему это ваш шанс
10. Где брать заказы: 8 каналов
11. Топ-10 ошибок новичка
12. Шпаргалка: всё на одной странице

Что это за гайд

Это не статья «что такое нейросети». Это короткий маршрут: **за вечер вы ставите себе ИИ-агента и делаете первое приложение, которое живёт в Telegram.**

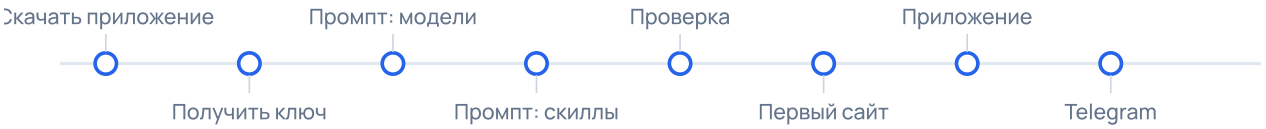
Вы не будете программировать в классическом смысле. Вы будете ставить задачи и проверять результат — как руководитель, у которого появился очень быстрый и очень дешёвый исполнитель, знающий 297 профессиональных методик.

ЧТО БУДЕТ К КОНЦУ ГАЙДА	КАК ПРОВЕРИТЬ
Работающий ИИ-агент на вашем компьютере	Пишете «скажи ок» — он отвечает
297 скиллов (профессиональных инструкций для агента)	Скилл включается сам по вашей задаче
Приложение «Деньги» — учёт доходов и расходов	Открывается в браузере, считает баланс
Это же приложение как Telegram Mini App	Открывается кнопкой в вашем боте на телефоне
Понимание, где брать заказы и сколько за них просить	8 каналов + вилки цен в главе 10

Сколько это стоит. Сам OpenCode бесплатный. Платите только за модель, и это копейки: пополнить баланс на **5 долларов** хватит на несколько недель работы. Оплата — через СБП из приложения банка, зарубежная карта не нужна.

Работаем мы на **DeepSeek V4.1 Flash** — дешёвой модели, которой хватает с запасом на всё в этом гайде. Весь маршрут от установки до первого проданного заказа обойдётся примерно в 30–60 рублей. Это не подписка: деньги списываются по факту использования, пока не работаете — не тратится.

Если с оплатой возникнут вопросы — их разберут в чате участников, ссылка в главе 1.



Кому подходит, а кому нет

Подходит, если вы:

- умеете устанавливать обычные программы (как Zoom или Telegram);
- готовы скопировать один промпт и вставить его в поле ввода;
- хотите делать продукты, а не «изучать программирование 6 месяцев»;
- готовы к тому, что первые 2–3 запуска что-то не сработает — и это нормально.

Не подходит, если вы:

- ждёте, что всё сделается «само» без ваших действий;
- не готовы платить за API даже 5 долларов;
- ищете способ обходить блокировки и нарушать чужие правила.

> Честно про ожидания: агент пишет код быстро, но **не думает за вас**. Ваша работа — точно объяснить, что нужно, и проверить, что получилось. Чем точнее формулировка, тем меньше правок. Этому и посвящена глава 6.

Маршрут: весь путь за вечер

Основной путь — без терминала. Всё делается в обычном приложении и в промптах.

ШАГ	ЧТО ДЕЛАЕМ	ВРЕМЯ
1	Скачиваем приложение OpenCode для своей ОС	5 мин
2	Регистрируемся в SpiderSense, получаем ключ и 5 $	5 мин
3	Вставляем один промпт — агент настраивает модели	5 мин

ШАГ	ЧТО ДЕЛАЕМ	ВРЕМЯ
4	Второй промпт — распаковка 297 скиллов	5 мин
5	Проверяем, что всё работает	2 мин
6	Первый результат: сайт одной фразой	5 мин
7	Терминал и git — по желанию, для надёжности	15 мин
8	Промпты: как формулировать задачи	15 мин
9	Делаем приложение «Деньги»	60–90 мин
10	Превращаем его в Telegram Mini App	40 мин
11	Разбираем, где брать заказы и сколько просить	20 мин

Как это работает — простыми словами

Если вам 13 лет и вы никогда не программировали — эта часть для вас. Если вы взрослый и «всё это не понимаю» — тоже.

Представьте, что у вас появился **робот-помощник**, который сидит внутри компьютера. Он умеет:

- читать любые файлы на вашем компьютере;
- создавать новые файлы;
- писать в них код;
- запускать команды и смотреть, что получилось.

Но сам по себе он **не думает**. Он просто руки. А «мозг» — это отдельная штука, которая находится далеко, в интернете. Она называется **модель**.

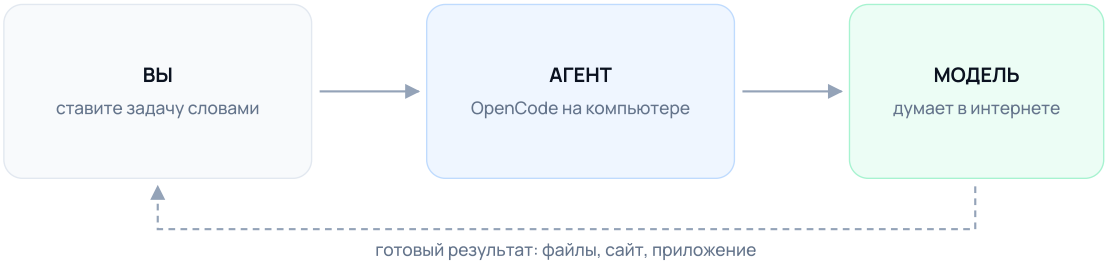


Схема простая:

1. **Вы** пишете в чат обычными словами: «сделай сайт-визитку».
2. **Агент** (OpenCode) читает вашу папку с файлами, понимает, что нужно, и отправляет вопрос модели.

3. **Модель** думает и отвечает: «вот код, который надо записать в такой-то файл».
4. **Агент** берёт этот код и **сам** создаёт файл на вашем компьютере.
5. **Вы** открываете файл и смотрите, что получилось.

Всё. Никакой магии. Вы говорите словами — робот делает руками, а думает за него модель в интернете.

Почему это стоит денег

Модель живёт на чужих мощных компьютерах, и за каждый вопрос ей платят. Считают не «за вопрос», а за **токены** — кусочки текста.

ЧТО	СКОЛЬКО ТОКЕНОВ	СКОЛЬКО ЭТО В ДЕНЬГАХ
Ваш запрос «сделай сайт»	6	почти ноль
Файлы проекта + инструкции, которые модель читает	≈ 20 000	≈ 3 копейки
Ответ модели с кодом	≈ 2 000	≈ 1 копейка
Один рабочий запрос целиком	**≈ 22 000**	**≈ 3–4 копейки**

схема tokens

Проще говоря: **одна рабочая просьба стоит меньше, чем жвачка**. Весь этот гайд — от установки до первого приложения — обойдётся рублей в 30–60. Поэтому мы и берём дешёвую модель: зачем платить в 20 раз больше за то же самое.

Что такое API-ключ

Ключ — это ваш **пропуск**. Как читательский билет в библиотеку: без него книгу не дадут, а с ним — пожалуйста, но за каждую взятую книгу списывают деньги.

```
CODE

sk-uno-4f2a...   ← это ключ. Он секретный, как пароль.
```

Ключ показывает, **с чьего счёта** списывать. Поэтому:

- никому не отправляйте ключ;
- не показывайте его на скриншотах;
- если случайно показали — удалите и создайте новый (30 секунд).

Что такое скилл

Скилл — это **инструкция**, которую агент подкладывает модели перед работой. Как шпаргалка для повара: «сначала разогрей масло, потом клади лук».

Например, приходит задача «сделай лендинг». Агент находит скилл `copywriting`, читает в нём правила сильных заголовков — и модель отвечает уже не «в среднем по интернету», а как копирайтер с опытом.

У нас таких инструкций **297 штук**: про дизайн, тексты, код, налоги, видео, документы. Агент сам выбирает нужные — вам не надо ничего запоминать.

Почему OpenCode, а не Claude Code

Это самый частый вопрос, поэтому отвечаю прямо, без маркетинга.

Claude Code — отличный инструмент. Но у него есть три особенности, из-за которых новичку на старте больно:

1. **Оплата.** Официальная подписка требует зарубежной карты. Российские карты не проходят, и «серые» варианты оплаты — это отдельный риск: подписку могут отозвать, а деньги не вернуть.
2. **Привязка к аккаунту и региону.** Anthropic активно блокирует аккаунты, которые заходят через VPN из неподдерживаемых регионов или делят подписку на несколько людей. Это не «миф про бан» — это реальные жалобы в сообществах. Аккаунт с историей, проектами и оплатой можно потерять в один день.
3. **Закрытость.** Claude Code работает только с моделями Anthropic. Если модель подорожала, стала недоступна или вы хотите попробовать другую — вы ничего не можете сделать.

OpenCode — открытый агент с обычным приложением для рабочего стола. Он:

- бесплатный и ставится как любая программа — двойным кликом;
- работает с **любым** провайдером моделей — сегодня Anthropic через посредника, завтра GPT, послезавтра локальная модель;
- не требует подписки: платите за токены напрямую, сколько потратили;
- не привязан к одному аккаунту, поэтому «потерять доступ» сложнее — вы просто меняете ключ.

Что важно понимать: мы не «обходим блокировку». Мы используем легальный инструмент и легальный API-доступ через провайдера. Никаких взломов, никаких чужих аккаунтов.

Риски, о которых нужно знать заранее:

- посредник-провайдер может поднять цены, изменить качество моделей или закрыться;
- ключ — это деньги: если он утечёт, потратят ваши;
- качество ответа зависит от модели, а не только от агента.

Поэтому в конфиг мы сразу закладываем **несколько моделей от одного провайдера**: основную дешёвую и запасные. Если одна подведёт — переключаетесь в списке моделей, а не ищете новый сервис в панике.

> Если у вас уже есть рабочая подписка Claude и она вас устраивает — работайте в Claude Code, этот гайд всё равно подойдёт: принципы те же. OpenCode — для тех, кто начинает с нуля и не хочет зависеть от одной компании.

5 правил безопасности перед стартом

Эти правила спасут вам деньги и нервы. Прочитайте до установки.

1. Ключ API — как банковская карта. Не отправляйте его в чаты, не показывайте на скриншотах, не вставляйте в публичные файлы. Если ключ утёк — сразу удаляйте его в кабинете и создавайте новый.

2. Поставьте лимит на ключ. В кабинете провайдера задайте месячный лимит расходов. Первый месяц — 5 долларов (этого достаточно). Так вы не узнаете о проблеме из списания на 300 долларов.

3. Агент может удалить файлы. Он выполняет команды, и если вы попросите «почисти проект» — он почистит. Поэтому: работаем только в отдельной папке и с первого дня делаем git-коммиты (глава 7). Это кнопка «отменить всё».

4. Не давайте агенту свои пароли и доступы. Ни к банку, ни к почте, ни к серверу. Всё, что ему нужно для учёбы, — локальная папка.

5. Резервная копия перед экспериментами. Копия папки проекта на флешку или в облако стоит 30 секунд. Восстановление проекта с нуля — вечер.

> Отдельно про ключи в этом гайде: **здесь нет рабочих ключей**. Везде стоит заглушка `sk-uno-ВАШ_КЛЮЧ`. Ключ вы создаёте сами в своём кабинете — и он принадлежит только вам.

Файлы, которые идут вместе с гайдом

К гайду приложен архив со скиллами. Больше ничего набирать руками не нужно — всё делается промптами.

ФАЙЛ	ДЛЯ ЧЕГО
`Скиллы публичный.zip`	297 скиллов, распаковываются вторым промптом
`bonus/money-app/index.html`	готовое приложение «Деньги» (открыть в браузере)
`bonus/money-app-tg/index.html`	та же версия с Telegram CloudStorage — её и публикуем

Файлы можно прочитать: внутри нет ничего скрытого. Всегда полезно понимать, что запускаешь на своём компьютере.

Где задавать вопросы

Пока читаете и делаете — почти наверняка что-то не сработает. Это нормально: у всех разный компьютер, разный интернет, разные версии программ. Для этого есть **чат участников**:

```
CODE
https://t.me/+WNNP-XISz8NjYTQ6
```

Туда можно:

- **задать любой вопрос** по установке — даже «глупый»;
- **показать ошибку** — скриншотом или текстом;
- **кинуть домашку и пруфы** — с хэштегом `#первыйминияпп`;
- посмотреть, что получилось у других — часто это отвечает на вопрос быстрее, чем сам вопрос.

Правило простое: **скриншот + текст ошибки + что вы делали**. Тогда ответ будет точным, а не «попробуйте перезапустить».

КУДА	ЗАЧЕМ
Чат `t.me/+WNNP-XISz8NjYTQ6`	вопросы, разбор ошибок, домашка, пруфы
Личный Telegram `@wroomwroomseo`	если вопрос личный или по оплате
Провайдер `spiderssense.com`	только про ключ, баланс, доступ к сайту

> Не присылайте в чат API-ключи и пароли. Если ключ случайно попал на скриншот — сразу перевыпустите его в кабинете, это 30 секунд.

Шаг 0. Из чего состоит система

Прежде чем что-то ставить, разложим по полочкам — это снимет половину страхов.

ЧАСТЬ	ЧТО ЭТО	АНАЛОГИЯ
OpenCode	программа-агент на вашем компьютере, которая читает файлы, пишет код и запускает команды	водитель
Модель (DeepSeek, Qwen, Claude, GPT, Gemini)	«мозг», который отвечает на запросы	двигатель

ЧАСТЬ	ЧТО ЭТО	АНАЛОГИЯ
API-ключ	ваш пропуск к модели, по нему списываются деньги	топливная карта
Провайдер (SpiderSense)	посредник, который продаёт доступ к моделям и принимает оплату	заправка
Скилл	текстовый файл с профессиональной инструкцией, который агент подгружает под задачу	методичка в бардачке



Всё это мы подключим **без терминала** — обычными окнами и одним текстом, который скопируете в поле ввода.

Шаг 1. Регистрация в SpiderSense и ключ

SpiderSense — провайдер доступа к моделям (DeepSeek, Qwen, Claude, GPT, Gemini и другие) с оплатой из России. Именно через него мы подключаем «мозг» к OpenCode.

> **Если сайт не открывается — это не поломка и не блокировка вас лично.** У SpiderSense стоит антибот-защита (сервис DiamWall): при первом заходе она показывает экран «Остался один клик» и проверяет браузер. Что делать:
>
> 1. Откройте ссылку в **обычном браузере** (Chrome, Safari, Firefox) — не во встроенном в Telegram или Instagram.
> 2. Подождите 5–10 секунд и нажмите на карточку с надписью «Остался один клик».
> 3. Если написано «Access Denied» — отключите VPN и попробуйте снова; иногда помогает мобильный интернет вместо Wi-Fi.
> 4. Не открывайте ссылку больше 3–4 раз подряд за минуту — защита воспринимает это как атаку и блокирует на время.
>
> **Важно:** API (то, чем работает агент) защитой не закрыт — он отвечает всегда. Так что даже если главная страница капризничает, работать вы сможете.

1.1. Откройте ссылку и зарегистрируйтесь:

```
CODE
https://spiderssense.com/register?ref=bef11400
```

1.2. Пополните баланс **на 10 долларов** — и получите **15 долларов на счёт**.'

БОНУС ПРИ ПЕРВОМ ПОПОЛНЕНИИ

10 $	+	5 $	=	15 $
вы пополняете		бонусом сверху		на вашем балансе

Бонус начисляется автоматически при **первом** пополнении от 10 долларов — ничего активировать не нужно. Это единоразовый подарок новому пользователю.

Чего хватит этих 15 долларов: на DeepSeek V4.1 Flash весь гайд — установка, приложение «Деньги», Telegram, все доработки и эксперименты — стоит порядка 30–60 рублей, то есть меньше одного доллара. Пятнадцати долларов хватит на **несколько месяцев** спокойной работы. Почему такой запас, а не «ровно на гайд»: вы захотите экспериментировать, а эксперименты — это и есть обучение.

Оплата проходит через СБП — обычным переводом из приложения банка. Никакой зарубежной карты, крипты и посредников не нужно.

> **Если оплата не проходит или что-то непонятно** — не пытайтесь угадать и не платите дважды. Задайте вопрос в нашем чате, там разберём по шагам:
>
> ```
> CODE
> https://t.me/+WNNP-XISz8NjYTQ6
> ```
>
> Приложите скриншот экрана оплаты (без данных карты) и напишите, на каком шаге остановились.

Честно про реферальную ссылку

Вы наверняка заметили хвост `?ref=bef11400` в адресе. Объясню прямо, без недомолвок, потому что скрывать тут нечего.

У SpiderSense есть партнёрская программа: *Invite & earn*. Условия такие:

1	**Пригласить** до 5 друзей по ссылке

2 **Друг пополняет**
от 10 долларов — первый раз

3 **Оба получают по 5 долларов**
друг — бонусом к балансу, автор гайда — вознаграждение

Это значит две вещи.

Вы получаете на 5 долларов больше. Пополняете 10 — на балансе 15. Бонус ваш, он никуда не денется и не зависит от меня.

Я получаю 5 долларов. Тоже один раз, только если вы зарегистрировались по этой ссылке и пополнили от 10 долларов. Пригласить можно до пяти человек — больше сервис не даёт никому.

На цену вашего пополнения это не влияет: 10 долларов есть 10 долларов, и тарифы на модели у всех одинаковые. Если бы ссылка делала вас дороже или урезала бонус, я бы её не ставил.

Можно ли обойтись без неё? Да: откройте `spiderssense.com` напрямую, зарегистрируйтесь — бонус, скорее всего, тоже дадут, но условия акции лучше уточнить в кабинете. Просто по ссылке из гайда его начисляют автоматически.

Почему я вообще её ставлю: гайд бесплатный, а работа над ним — нет. Реферальные 5 долларов — единственное, что я с него получаю. Если это для вас некомфортно, регистрируйтесь без ссылки, гайд всё равно работает полностью.

1.3. В личном кабинете найдите раздел с API-ключами и создайте новый ключ. Он выглядит так:

```
CODE

sk-uno-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

1.4. Сразу сделайте две вещи, о которых потом забывают:

- **поставьте лимит расходов** на этот ключ (месячный лимит, начните с 5 $ — этого достаточно);
- **скопируйте ключ в надёжное место** (менеджер паролей или закрытая заметка). Ключ показывается один раз — потом только перевыпуск.

> Ключ никому не пересылайте. Даже «в поддержку». Настоящая поддержка никогда не просит ключ целиком.

Шаг 2. Скачиваем приложение

Никакого терминала, никаких команд. Это обычная программа, как Zoom или Telegram: скачали, установили, открыли.

СИСТЕМА	ССЫЛКА
Windows	https://opencode.ai/update/api/latest/desktop/opencode/opencode-desktop-win-x64.exe
macOS (Apple Silicon — M1/M2/M3/M4)	https://opencode.ai/update/api/latest/desktop/opencode/opencode-desktop-mac-arm64.dmg
macOS (Intel)	https://opencode.ai/update/api/latest/desktop/opencode/opencode-desktop-mac-x64.dmg
Страница со всеми версиями	https://opencode.ai/download

Не знаете, какой у вас Mac? Нажмите яблоко в левом верхнем углу → «Об этом Mac». Строка «Чип»: если там `Apple M...` — берите arm64. Если `Intel Core` — берите x64.

Windows. Скачайте `.exe`, запустите, при предупреждении «Защитник Windows» нажмите «Подробнее» → «Всё равно запустить». Дальше везде «Далее».

macOS. Скачайте `.dmg`, откройте, перетащите значок в папку «Программы». Первый запуск: правый клик по значку → «Открыть» (иначе система скажет, что программа от неизвестного разработчика).

> **Приложению понадобится Node.js?** Нет. Версия для рабочего стола ничего не требует — всё уже внутри.

Шаг 3. Первый запуск и выбор бесплатной модели

3.1. Откройте приложение. Появится окно с полем ввода внизу — как в чате.

3.2. Модель сейчас никакая, но приложение обычно предлагает выбрать бесплатную — ищите внизу переключатель с названием модели или пункт «Choose model» / «Модель».

3.3. Выберите **любую бесплатную модель** — она нужна всего на один шаг, чтобы агент выполнил настройку. Не важно, какая именно: DeepSeek, GLM, что угодно со словом Free.

3.4. Напишите в поле:

```
CODE
скажи ок
```

Ответил — приложение работает. Теперь дадим агенту нормальную модель.

> **Если в списке нет бесплатных моделей** — пропустите этот шаг и сразу переходите к шагу 4. Агент настоится и без проверки.

Шаг 4. Один промпт — и модели настроены

Это главный шаг. Вы копируете текст, подставляете три значения и отправляете. Агент сам создаст конфиг со всеми моделями.

4.1. Подготовьте три значения (просто держите в блокноте):

ЧТО	ГДЕ ВЗЯТЬ
Твой **ключ**	кабинет SpiderSense, начинается с `sk-uno-`
Адрес API	`https://spiderssense.com/v1` — одинаковый для всех
Имя папки для проектов	`C:\projects` на Windows или `~/projects` на Mac

4.2. Скопируйте промпт целиком и замените три места, отмеченные `<>`. Угловые скобки тоже уберите.

ПРОМПТ 1 · НАСТРОЙКА МОДЕЛЕЙ — первый промпт после установки

```
Ты помогаешь настроить OpenCode. Создай конфиг с моделями SpiderSense.

ДАННЫЕ
Ключ: <КЛЮЧ>
Адрес API: https://spiderssense.com/v1
Папка для проектов: <ПАПКА>

1. Создай папку для проектов.
2. Создай конфиг OpenCode:
   Windows: %USERPROFILE%\.config\opencode\opencode.json
   macOS:   ~/.config/opencode/opencode.json
   Если файл есть — сделай копию .bak.
3. Конфиг ровно такой (подставь ключ):

{
  "$schema": "https://opencode.ai/config.json",
  "model": "ss/deepseek-v4.1-flash",
  "small_model": "ss/deepseek-v4.1-flash",
  "provider": {
    "ss": {
      "name": "SpidersSense",
      "npm": "@ai-sdk/openai-compatible",
      "options": {
        "baseURL": "https://spiderssense.com/v1",
        "apiKey": "МОЙ_КЛЮЧ",
        "timeout": 300000,
        "headerTimeout": 120000,
        "chunkTimeout": 180000
      },
      "models": {
        "deepseek-v4.1-flash": { "name": "DeepSeek V4.1 Flash", "id": "deepseek/deepseek-
v4.1-flash", "tool_call": true, "limit": { "context": 1048576, "output": 128000 } },
        "deepseek-v4-flash": { "name": "DeepSeek V4 Flash", "id": "deepseek/deepseek-v4-
```

```
flash", "tool_call": true, "limit": { "context": 1048576, "output": 128000 } },
        "claude-sonnet-4.6": { "name": "Claude Sonnet 4.6", "id": "anthropic/claude-sonnet-
4.6", "tool_call": true, "limit": { "context": 1000000, "output": 128000 } },
        "claude-opus-5.5": { "name": "Claude Opus 5.5", "id": "anthropic/claude-opus-5.5",
"tool_call": true, "limit": { "context": 1000000, "output": 128000 } },
        "gpt-6-sol": { "name": "GPT-6 Sol", "id": "openai/gpt-6-sol", "tool_call": true,
"limit": { "context": 1000000, "output": 128000 } },
        "gemini-4-argon": { "name": "Gemini 4 Argon", "id": "google/gemini-4-argon",
"tool_call": true, "limit": { "context": 1000000, "output": 64000 } }
      }
    }
  }
}

4. Создай рядом AGENTS.md с правилами:

   # Правила для всех проектов

   ## Язык
   - Отвечай ТОЛЬКО по-русски, даже если вопрос по-английски.
   - Простой человеческий язык, без канцелярита.
   - Объясняй так, чтобы понял тот, кто никогда не программировал.

   ## Как отвечать
   - Коротко. Сначала результат, потом объяснение.
   - Списки и таблицы вместо сплошных абзацев.
   - Если не знаешь — скажи прямо, не выдумывай.

   ## Как работать с кодом
   - Сначала план, потом код, если задача больше одного шага.
   - Не переписывай файл целиком без просьбы.
   - Не добавляй библиотеки без просьбы.

5. Проверь JSON и выведи: путь к конфигу, список моделей, модель по умолчанию.
6. В конце: «Перезапустите приложение, чтобы оно перечитало конфиг».
```

4.3. Вставьте в поле ввода приложения и нажмите Enter. Агент начнёт писать файлы — на это уйдёт полминуты.

4.4. В конце он ответит примерно так:

```
CODE

Готово. Конфиг создан, 6 моделей подключены.
По умолчанию: DeepSeek V4.1 Flash
Файл: C:\Users\Вы\.config\opencode\opencode.json
```

4.5. Закройте приложение **полностью** и откройте заново — оно перечитает конфиг.

4.6. Внизу появится выбор моделей. Там должно быть **DeepSeek V4.1 Flash** и ещё несколько. Выберите DeepSeek, если он не выбран автоматически.

> **Если в списке появились модели — но не DeepSeek** — просто выберите ту, что с пометкой SpiderSense или `ss/`. Остальные никуда не денутся.

Шаг 5. Второй промпт — распаковка 297 скиллов

Скиллы — это 297 готовых инструкций: как писать тексты, как делать дизайн, как считать налоги. Агент будет подгружать их сам, когда задача совпадёт.

5.1. Распакуйте архив `Скиллы публичный.zip` куда-нибудь, например на рабочий стол. Внутри будет папка `skills-pack`, а в ней — `skills`.

5.2. Скопируйте промпт, заменив два пути на свои:

ПРОМПТ 2 · СКИЛЛЫ — второй промпт, после перезапуска

```
Распакуй и установи скиллы.

ОТКУДА
Папка со скиллами: <ПУТЬ_К_ПАПКЕ_skills>
Внутри 297 папок, в каждой файл SKILL.md.

КУДА
Windows: %USERPROFILE%\.config\opencode\skills
macOS:   ~/.config/opencode/skills

1. Создай папку назначения, если её нет.
2. Скопируй все папки, внутри которых есть SKILL.md. Существующие — перезапиши.
3. Посчитай, сколько скопировалось.
4. Выведи: сколько скопировано (должно быть 297), путь, 5 примеров названий.
5. Если не 297 — скажи сколько и в чём может быть причина.
6. В конце: «Перезапустите приложение, чтобы скиллы подхватились».
```

5.3. Отправьте. Агент скопирует 297 папок в нужное место.

5.4. В ответе должно быть:

```
CODE

Установлено скиллов: 297
Папка: C:\Users\Вы\.config\opencode\skills
```

5.5. Перезапустите приложение ещё раз.

5.6. Проверьте: напишите агенту

```
CODE

какие скиллы тебе доступны? покажи 5 штук с коротким описанием
```

Если он перечислил — всё работает.

Шаг 6. Проверка: всё ли на месте

Один промпт, который проверит всю настройку и скажет, что не так.

ПРОМПТ 3 · ПРОВЕРКА — убедиться, что всё настроено

```
Проверь мою настройку OpenCode и скажи, всё ли в порядке.

1. Найди конфиг (Windows: %USERPROFILE%\.config\opencode\opencode.json,
   macOS: ~/.config/opencode/opencode.json). Проверь:
   - файл существует и читается; JSON валидный
   - provider.ss.options.apiKey начинается с sk-
   - provider.ss.options.baseURL = https://spiderssense.com/v1
   - model = ss/deepseek-v4.1-flash
2. Посчитай папки в папке skills и скажи число.
3. Сделай тестовый запрос:
   POST https://spiderssense.com/v1/chat/completions
   модель deepseek/deepseek-v4.1-flash, короткий промпт «ответь: работает»,
   заголовок Authorization: Bearer <ключ из конфига>.
4. Выведи таблицу: ПРОВЕРКА | РЕЗУЛЬТАТ | ОК или ПРОБЛЕМА.
5. Если что-то не так — скажи, что исправить, по шагам.
   Если всё в порядке — «Всё в порядке, можно работать».
```

Если агент ответил «всё в порядке» — переходите к главе 3.

> **Если что-то не так** — не пытайтесь чинить сами. Пришлите ответ агента в чат: по нему видно всё нужное для точного ответа.

Если что-то пошло не так

ЧТО ВИДИТЕ	ЧТО ДЕЛАТЬ
Приложение не запускается на Mac: «не удалось проверить разработчика»	Правый клик по значку → «Открыть» → «Открыть»
Windows: «Защитник заблокировал»	«Подробнее» → «Всё равно запустить»
В списке моделей нет DeepSeek	Выберите любую с пометкой `ss/` или SpiderSense
Агент отвечает «не могу создать файл»	Он не имеет прав в папке. Создайте папку `projects` в домашней и откройте её через «File → Open Folder»
Ответы на английском	Напишите «отвечай по-русски»
«бесплатный лимит закончился»	Вы работаете на бесплатной модели. Выберите модель SpiderSense внизу окна

ЧТО ВИДИТЕ	ЧТО ДЕЛАТЬ
Ничего не понятно	Скриншот в чат → https://t.me/+WNNP-XISz8NjYTQ6

Модели: что выбирать и когда

Основная модель — DeepSeek V4.1 Flash. Именно она стоит в конфиге по умолчанию, и для этого гайда её хватает с запасом: 1 миллион токенов контекста (это примерно 700 000 слов — влезает весь ваш проект целиком), поддержка инструментов, скорость ответа 3–6 секунд и цена, которая не ощущается.

Вот реальные цифры от провайдера:

МОДЕЛЬ	ВХОД (ЗА 1 МЛН ТОКЕНОВ)	ВЫХОД (ЗА 1 МЛН ТОКЕНОВ)	ДЛЯ ЧЕГО
DeepSeek V4.1 Flash	**$0,014**	**$0,028**	по умолчанию: код, тексты, правки, весь гайд
DeepSeek V4 Flash	$0,014	$0,028	альтернатива, чуть другой характер
Claude Sonnet 4.6	$0,30	$1,50	в 21 раз дороже; нужен редко
Claude Opus 5.5	дороже	дороже	сложная архитектура, крайний случай
GPT-6 Sol	средняя	средняя	альтернативный взгляд, таблицы
Gemini 4 Argon	средняя	средняя	огромные файлы, скриншоты

Практическое правило:

1. **Работайте на Flash.** 95% задач гайда — его уровень.
2. **Не решается с третьей попытки?** Переключитесь на Claude Sonnet в списке моделей.
3. **Совсем сложная архитектура** — Claude Opus, но это уже не про первый проект.
4. Начинайте новый чат при переходе к другой задаче. Старый контекст тянет деньги и путает модель.

Приложение или терминал?

Приложение — для всех, кто только начинает. Терминал — для тех, кто хочет автоматизировать рутину (например, генерировать 100 статей скриптом).

ЧТО	ПРИЛОЖЕНИЕ	ТЕРМИНАЛ
Установка	двойной клик	команда
Работа	окно с полем ввода	окно с полем ввода
Скрипты и циклы	сложнее	удобнее
Подходит новичку	**да**	не сразу

Если терминал всё-таки захочется — в главе 7 есть минимум из восьми команд, и его хватит.

Первый результат за 5 минут

Не переходите к приложению, пока не получите «вау». Это важно психологически: вы должны своими глазами увидеть, что машина делает работу.

Откройте своё приложение, откройте папку с проектами (File → Open Folder → ваша папка `projects`) и вставьте:

ПРОМПТ 4 · ПЕРВЫЙ САЙТ — первый результат за 5 минут

```
Создай в папке site файл index.html — одностраничный сайт-визитку.
Тёмный премиальный дизайн, шрифт Manrope, акцентный цвет #2563EB.
Заголовок «Максим Холмов», подзаголовок «Делаю сайты и ИИ-автоматизацию».
Три блока услуг, кнопка «Написать в Telegram» со ссылкой https://t.me/wroomwroomseo.
Адаптивно под телефон. Без внешних библиотек.
```

Через 20–40 секунд агент создаст файл. Откройте его двойным кликом в проводнике — сайт откроется в браузере.

Вуаля. Вы только что сделали сайт одной фразой. Дальше можно просить правки — и он будет менять именно этот файл.

3 промпта, которые сразу показывают силу

1. Личный сайт — уже сделали выше.

2. Разбор своих денег (это подготовка к проекту из главы 9). Скопируйте свои траты за месяц из банковского приложения и вставьте:

ПРОМПТ · РАЗБОР ДЕНЕГ — вставьте свои траты

```
Вот мои расходы за месяц списком. Разбери их:
1. Сгруппируй по категориям, посчитай сумму и долю каждой.
2. Выведи таблицей: категория, сумма, процент.
```

```
3. Найди 3 самые неочевидные утечки денег.
4. Предложи, где сэкономить без потери качества жизни.
5. Выводы — 5 пунктов, коротко.

Мои траты:
- Такси 1200
- Продукты 8400
- Подписка на кино 799
```

3. «Что ты вообще можешь» — полезно один раз в начале:

ПРОМПТ · ЧТО ТЫ МОЖЕШЬ полезно один раз в начале

```
Ты работаешь в папке проекта и можешь создавать файлы и запускать команды.
Составь список из 10 задач, которые ты реально сделаешь за 5 минут
и которые принесут мне пользу уже сегодня. Для каждой — одна строка
промпта, которую мне нужно тебе отправить. Я выберу одну.
```

Дальше — по накатанной: выбираете задачу, отправляете, смотрите результат.

> Правило первого дня: **не просите сразу «сделай мне приложение с оплатой и базой данных»**. Начните с одного файла. Успех на маленьком шаге важнее, чем провал на большом.

Терминал: 8 команд, которых достаточно

Терминал — это просто окно, куда пишут команды текстом. Если вы работаете в приложении, он не обязателен. Но иногда он нужен: чтобы запустить скрипт, проверить, что всё на месте, или откатить проект. Здесь — минимум, чтобы не бояться.

схема terminal

ЧТО НУЖНО	WINDOWS (POWERSHELL)	MACOS (TERMINAL)
Где я сейчас	`pwd`	`pwd`
Что в папке	`dir`	`ls`
Перейти в папку	`cd C:\projects`	`cd ~/projects`
Шаг назад	`cd ..`	`cd ..`
Создать папку	`mkdir site`	`mkdir site`

ЧТО НУЖНО	WINDOWS (POWERSHELL)	MACOS (TERMINAL)
Открыть в проводнике	`explorer .`	`open .`
Открыть файл программой	`start site\index.html`	`open site/index.html`
Прервать выполнение	`Ctrl + C`	`Ctrl + C`

Три вещи, которые снимают 90% паники:

1. **Терминал ничего не «ломает» сам по себе.** Он выполняет ровно то, что написано.
2. **Текущая папка — это контекст.** Если агент «не видит файл», скорее всего вы не в той папке.
3. `Ctrl + C` **останавливает что угодно.** Зависло, печатает бесконечно, страшно — жмите.

> **Но в приложении почти всё это не нужно.** Папку можно открыть через File → Open Folder, файл — двойным кликом. Терминал нужен, только когда об этом просит агент.

Git-спасатель: 3 команды и кнопка «отменить»

Git — это история изменений. Для нас он нужен ровно для одного: **вернуть проект к состоянию, когда всё работало**.

Если вы работаете в приложении, можно попросить агента включить git в один промпт:

● ПРОМПТ · ВКЛЮЧИТЬ GIT — страховка от поломок

```
Включи в этой папке git и запомни правило работы.
1. Выполни git init, если репозитория ещё нет.
2. Добавь в AGENTS.md правило: после каждого законченного шага делай
   git commit с осмысленным сообщением. Если что-то ломается —
   откатывайся к последнему рабочему коммиту и говори мне.
3. Сделай первый коммит текущего состояния.
4. Выведи: где репозиторий, сколько коммитов, что попало в первый.
```

А если хотите руками — три команды на всё. Один раз на проект:

BASH

```
git init
git add -A
git commit -m "старт: рабочая версия"
```

Дальше — после каждого удачного шага:

```bash
git add -A && git commit -m "работает ввод расходов"
```

Если агент всё сломал — откат к последнему коммиту:

```bash
git restore .
```

Если сломано так, что восстановление не помогает:

```bash
git reset --hard HEAD
```

Проверить, что происходит:

```bash
git status
```

> **Лайфхак:** в самом проекте однажды напишите: «После каждого законченного шага делай git commit с осмысленным сообщением. Если что-то ломается — откатывайся к последнему рабочему коммиту и говори мне». Это одна фраза, которая спасает вечера.

Правило папок

Порядок в файлах — это не педантизм, а скорость работы агента: он читает папку и ему нужно понимать, где что.

```
C:\projects\              ~/projects/
├── site\                 ├── site\              ← первый сайт-визитка
├── money-app\            ├── money-app\         ← приложение «Деньги»
│   ├── index.html        │   ├── index.html
│   └── README.md         │   └── README.md
└── experiments\          └── experiments\       ← всё, что «просто попробовать»
```

- **Один проект = одна папка.** Не работайте в «Загрузках» и на рабочем столе.
- `AGENTS.md` — файл с правилами проекта. Агент читает его автоматически. Например: «Отвечай по-русски. Не добавляй библиотеки без спроса. Каждый шаг — коммит».

- `experiments` — свалка для экспериментов. Там можно ломать всё, что угодно.

Скиллы: что это и почему это главный усилитель

Скилл — это папка с файлом `SKILL.md`. Внутри — инструкция на 100–500 строк: как профессионально делать конкретную вещь. Например, скилл `copywriting` содержит правила сильных заголовков и структуру лендинга, `ui-design` — сетки, отступы и типографику, `tax-prep` — расчёт авансовых платежей по УСН.

Работает так: вы пишете «сделай лендинг для барбершопа» → агент видит, что задача совпадает с описанием скилла `copywriting` и `design-taste-frontend` → подгружает их → отвечает **не как обычная нейросеть, а как копирайтер с 15-летним опытом и арт-директор**.

Разница на практике огромная. Без скиллов вы получите «шаблонный сайт от нейросети». Со скиллами — работу, которую не стыдно показать клиенту.

В комплекте, который идёт с этим гайдом, — **297 скиллов**: маркетинг, дизайн, архитектура, безопасность, документы, видео, презентации, малый бизнес по РФ, память и процессы.

Установка 297 скиллов — одним промптом

Вы уже сделали это в главе 2, шаг 5. Если пропустили — вернитесь и сделайте: без скиллов качество ответов будет заметно ниже.

Проверить, что скиллы на месте, можно так — напишите агенту:

ПРОМПТ · ПРОВЕРКА СКИЛЛОВ посчитать установленные скиллы

```
Посчитай, сколько скиллов установлено в папке skills.
Windows: %USERPROFILE%\.config\opencode\skills
macOS:   ~/.config/opencode/skills
Выведи: путь, количество папок с файлом SKILL.md, 5 примеров названий.
Если меньше 297 — скажи сколько и в чём может быть причина.
```

Агент посчитает папки и скажет точное число. Должно быть **297**.

Уже всё настроено, нужны только скиллы?

Такое бывает: приложение стоит, ключ работает, модели настроены — а скиллов нет. Заново всё проходить не нужно.

1. Скачайте архив кнопкой **Скиллы** в шапке этой страницы. Он же лежит рядом с гайдом: `Скиллы публичный.zip`.

2. Распакуйте его куда-нибудь, например на рабочий стол.

3. Отправьте агенту второй промпт из главы 2 (шаг 5) — он разложит все 297 штук по местам.

4. Перезапустите приложение.

5. Проверьте промптом из этого раздела — агент посчитает папки.

> Застряли на любом шаге — напишите в чат участников. Отвечаем быстро, вопрос «а куда кидать папку» — нормальный.

Стартовый набор: 12 скиллов, которые нужны каждый день

297 скиллов — это библиотека. Как в библиотеке, вы не читаете всё сразу. Вот 12, которые реально работают в первый месяц:

СКИЛЛ	ЧТО ДЕЛАЕТ	КОГДА ВКЛЮЧАЕТСЯ
`brainstorming`	задаёт вам правильные вопросы до начала работы	«хочу сделать приложение»
`make-plan`	разбивает задачу на этапы с проверками	«составь план проекта»
`do`	выполняет план по шагам	«выполняй план»
`verification-before-completion`	не даёт агенту сказать «готово», если не проверено	перед сдачей работы
`ui-design`	сетки, отступы, типографика	любой интерфейс
`design-taste-frontend`	убирает «шаблонный вид нейросети»	лендинги, дашборды
`impeccable`	полирует интерфейс: состояния, края, доступность	«сделай красиво»
`frontend-api-integration-patterns`	правильная работа с запросами и ошибками	приложения с данными
`copywriting`	заголовки, офферы, тексты	сайт, презентация, лендинг
`bug-hunter`	ищет причины багов, а не симптомы	«не работает»
`webapp-testing`	тестирует приложение в браузере	проверка перед сдачей
`skill-creator`	помогает сделать свой скилл	«хочу свой скилл»

Принудительно включить скилл — просто назовите его:

```
CODE
используй скилл copywriting и перепиши заголовок этого лендинга
```

Сколько это стоит по токенам

Каждый скилл добавляет в начало запроса своё описание — чтобы модель понимала, что у неё есть. 297 описаний ≈ **77 КБ ≈ 20 000 токенов**. Это накладные расходы **в каждом** запросе.

Простыми словами:

- ответы становятся чуть дороже (порядка 5–8 копеек на запрос вместо 3–5);
- старт сессии чуть медленнее;
- при очень длинных диалогах модель иногда «отвлекается» на лишние инструкции.

Для обучения это нормально и даже полезно. Экономить начнёте, когда поймёте, какими скиллами реально пользуетесь — тогда лишние можно просто отложить в другую папку.

> **Простое правило:** пока учитесь — держите все 297. Скиллы дешевле, чем время на их отбор.

Как проверить, что скилл реально сработал

Спросите агента напрямую:

```
CODE
Какие скиллы тебе доступны? Покажи 10 штук с коротким описанием.
```

Затем дайте задачу, которая точно попадает в скилл, и попросите отчитаться:

```
CODE
Используй скилл ui-design. Скажи, какой скилл ты применил и какие 3 правила
из него повлияли на решение.
```

Если агент называет правила из файла — скилл работает. Если отвечает общими словами — не подхватился (проверьте путь и перезапустите приложение).

Свой скилл за 10 минут

Скиллы — это ваше конкурентное преимущество: вы кодифицируете свой опыт и агент работает по вашим правилам, а не «в среднем по интернету».

Проще всего — попросить агента:

ПРОМПТ · СВОЙ СКИЛЛ закодифицировать свой стиль

```
Создай в папке skills новый скилл my-style со своим стилем оформления.

Спроси меня 5 вопросов: какие цвета, какой шрифт, какая плотность отступов,
какие радиусы, какие запреты (например «никаких градиентов»).
Дождись ответов и только потом создай папку my-style с файлом SKILL.md:

---
name: my-style
description: Мой стиль оформления — цвета, шрифты, отступы
---

# Правила

(дальше — мои ответы, оформленные как чёткие правила)

Выведи путь к созданному файлу.
```

А структура такая: папка + `SKILL.md` с заголовком файла.

```markdown
---
name: my-style
description: Мой стиль лендингов — тёмный премиум, Manrope, акцент #2563EB
---

# Правила

## Цвета
- Фон #080D19, карточки #0F172A, акцент #2563EB
- Никаких градиентов, кроме одного акцентного

## Типографика
- Заголовки Manrope 700, letter-spacing -0.03em
- Текст 17px / 1.65
```

Дальше он будет делать все ваши лендинги в вашем стиле.

> Скиллы `skill-creator` и `writing-skills` помогают собрать такой файл правильно — вплоть до тестов, что скилл вообще включается.

Промпт-грамотность: 5 правил

Умение формулировать задачу — это 80% результата. Хорошая новость: здесь нет магии, есть 5 правил.

Правило 1. Одна задача — один запрос. Не «сделай приложение с графиками, экспортом и тёмной темой». А: «сделай экран ввода расходов». Потом отдельно графики. Так вы видите, что сломалось, и откатываетесь на один шаг, а не на десять.

Правило 2. Дайте контекст и роль. «Ты senior frontend-разработчик. Проект — одностраничное приложение на чистом HTML/CSS/JS без библиотек». Одно предложение экономит три круга правок.

Правило 3. Опишите результат, а не процесс. Не «создай div и повесь обработчик», а «после нажатия «Добавить» расход появляется в списке сверху, баланс пересчитывается».

Правило 4. Дайте критерий готовности. «Готово, когда: данные сохраняются после перезагрузки; на телефоне ничего не уезжает за экран; в консоли нет ошибок». Тогда агент проверяет себя сам.

Правило 5. Просите проверку и откат. «Проверь результат, покажи, что изменилось, и если не работает — откати последнее изменение».

6 шаблонов промптов

1. Сделать новое — блок выше `setup-format`.

2. Починить

ШАБЛОН · ПОЧИНИТЬ — когда что-то сломалось

```
Не работает: <что именно происходит>
Ожидалось: <как должно быть>
Шаги: 1) открыл 2) нажал 3) получил ...
Текст ошибки: <вставить целиком>

Сначала объясни причину, потом чини.
Не меняй ничего, кроме причины.
```

3. Объяснить

ШАБЛОН · ОБЪЯСНИТЬ — разобраться в непонятном

```
Объясни <файл / функцию / ошибку> простыми словами,
как человеку без опыта в программировании.
Затем — что будет, если это убрать.
```

4. Улучшить (ревизия)

ШАБЛОН · ПРОВЕРКА ревизия перед сдачей

```
Проверь <файл / проект> по критериям: безопасность, читаемость,
адаптивность, обработка ошибок, доступность.
Дай таблицу: проблема / риск / как исправить.
Ничего не меняй, пока я не выберу.
```

5. План перед большой задачей

ШАБЛОН · ПЛАН перед большой задачей

```
Разбей задачу «<что нужно>» на шаги по 15-20 минут.
Для каждого: что делаем, как проверить, что готово.
Не пиши код, пока я не скажу «поехали».
```

6. Откат

ШАБЛОН · ОТКАТ если сломалось

```
Последнее изменение сломало <что>.
Откати его к рабочему состоянию (git restore), объясни,
что именно было не так, и предложи другой подход.
```

Плохо и хорошо: 5 примеров

ПЛОХО	ПОЧЕМУ	ХОРОШО
«сделай сайт»	неизвестно что, для кого и как проверять	«сделай одностраничный сайт для мастера маникюра: услуги с ценами, отзывы, кнопка записи в WhatsApp, тёмно-розовая палитра, адаптив под телефон»
«не работает»	нет данных для диагностики	«при нажатии «Добавить» сумма не появляется. В консоли: `TypeError: Cannot read properties of null`»
«сделай красиво»	у каждого своё «красиво»	«убери лишние рамки, увеличь отступы между блоками до 64px, заголовки крупнее в 1.5 раза»
«добавь всё как у банков»	невыполнимо за один шаг	«добавь только фильтр по месяцу — выпадающий список над списком расходов»
«перепиши весь код»	ломает рабочее	«перепиши только функцию подсчёта баланса, остальное не трогай»

Лайфхаки, которые экономят часы

Начните сессию с правил проекта. Создайте `AGENTS.md` в папке проекта — или попросите агента:

ШАБЛОН · ПРАВИЛА ПРОЕКТА один раз на проект

```
Создай в этой папке файл AGENTS.md с правилами проекта:

# Правила проекта

- Отвечай по-русски, кратко.
- Стек: один файл index.html, чистый HTML/CSS/JS. Без библиотек и сборщиков.
- Не переписывай файл целиком без просьбы — делай точечные правки.
- После каждого законченного шага: git commit с осмысленным сообщением.
- Перед «готово» проверь: нет ошибок в консоли, работает на мобильном.
- Если задача непонятна — задай вопрос, не угадывай.
```

Агент читает этот файл автоматически в каждом запросе. Один раз написали — работает всегда.

Просите показать план. На сложной задаче: «сначала план на 5 шагов, потом жди моего «да»». Так вы не получите 400 строк кода, который делает не то.

Новый чат при смене темы. Начните новый диалог — контекст чистый. Старая переписка стоит денег и сбивает модель.

Скриншот вместо описания. Если интерфейс «не такой», приложите картинку: «сделай как здесь» работает лучше трёх абзацев текста.

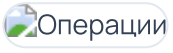
Не спорьте больше двух раз. Если третья попытка не дала результата — меняйте модель в списке или формулировку. Упорство с одной моделью — самая дорогая привычка.

Проект «Деньги»: что получится

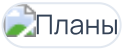
Делаем полноценное приложение для учёта личных денег. Не «калькулятор из туториала», а продукт: четыре экрана, бюджеты с предупреждениями, цели-накопления, постоянные платежи, долги, графики, поиск, экспорт.



Обзор
баланс, бюджеты, графики



Операции
поиск и фильтры



Планы
цели, платежи, долги

Почему именно это приложение:

- **понятно без объяснений** — за 5 секунд ясно, что нажимать;
- **решает вашу задачу** — вы сами им пользуетесь и сразу видите, что неудобно;
- **есть что показать** — живое приложение в телефоне, а не картинка;
- **легко превращается в Telegram Mini App** (глава 9) и в товар (глава 10).

Четыре экрана: что где лежит

ЭКРАН	ЧТО НА НЁМ	ЗАЧЕМ НУЖЕН
Обзор	остаток за месяц, доходы, расходы, бюджеты с полосками, график по категориям, тренд за 6 месяцев, последние операции	открыл — и сразу понял, как дела с деньгами
Операции	весь список по дням, поиск, фильтры-чипы, редактирование, удаление	найти любую трату за 3 секунды
Планы	лимиты по категориям, цели с прогрессом, постоянные платежи, долги	увидеть, куда движешься
Ещё	экспорт CSV, резервная копия, итоги, удаление	не потерять данные

схема app

Шаг 1. Папка и правила проекта

Создайте папку `money-app` внутри вашей папки проектов и откройте её в приложении (File → Open Folder).

Затем отправьте промпт с правилами — агент сам создаст `AGENTS.md`:

ПРОМПТ · ПРАВИЛА ПРОЕКТА — шаг 1 проекта

```
Создай в этой папке файл AGENTS.md с правилами проекта.

# Проект: Деньги

Одностраничное приложение учёта доходов и расходов.
Стек: один файл index.html, чистый HTML/CSS/JS, без библиотек.
Дизайн: тёмный, фон #080D19, карточки #0F172A, акцент #2563EB,
доходы #10B981, расходы #F87171, шрифт Manrope.
Язык интерфейса — русский. Суммы в рублях, формат 1 234 ₽.
Данные в localStorage, ключ money.v2.

Правила:
- Отвечай кратко, по-русски.
- Не переписывай файл целиком — делай точечные правки.
- После каждого законченного шага делай git commit.
- Перед словом «готово» проверь: нет ошибок в консоли, работает на 375px.
```

Шаг 2. Какие скиллы включить

Перед началом работы попросите агента опереться на конкретные скиллы. Это разница между «шаблон от нейросети» и «работа дизайнера»:

ПРОМПТ · СКИЛЛЫ В РАБОТЕ перед началом проекта

```
Для этого проекта используй скиллы:
ui-design (сетки, отступы, типографика),
design-taste-frontend (чтобы не выглядело шаблонно),
impeccable (состояния, края, доступность),
frontend-api-integration-patterns (работа с данными и ошибками).
Перед кодом перечисли 5 правил из этих скиллов, которые применишь.
```

СКИЛЛ	ЧТО ДАЁТ В ЭТОМ ПРОЕКТЕ
`ui-design`	ритм отступов, сетка, размеры шрифтов, иерархия
`design-taste-frontend`	убирает «дефолтный вид нейросети»
`impeccable`	пустые состояния, границы значений, доступность
`frontend-api-integration-patterns`	работа с сохранением, ошибки localStorage
`verification-before-completion`	агент проверяет себя до слова «готово»

Шаг 3. Промпт-бриф (отправьте целиком)

Не просите «сделай приложение для учёта денег» — получите заготовку. Дайте структуру: экраны, данные, поведение, критерии готовности.

ПРОМПТ · ТЗ НА ПРИЛОЖЕНИЕ главный промпт проекта

```
Ты senior frontend-разработчик и продуктовый дизайнер.
Создай index.html — приложение «Деньги» для учёта личных финансов.
Один файл, чистый HTML/CSS/JS, без библиотек.

ЧЕТЫРЕ ЭКРАНА (нижняя навигация, переключение без перезагрузки)

1. ОБЗОР
   - карточка баланса: остаток за месяц крупно, доходы и расходы двумя
     блоками, переключатель месяца (назад/вперёд) и свайп влево-вправо
   - бюджеты: для категорий с лимитом — полоска прогресса, сумма
     «потрачено из лимита»; при 80% — жёлтый, при 100% — красный
   - график расходов по категориям: полоски, сумма и процент
   - тренд за 6 месяцев: столбики, текущий месяц выделен цветом
   - последние 6 операций

2. ОПЕРАЦИИ
   - поиск по комментарию и названию категории
   - чипы-фильтры: Все / Расходы / Доходы / по категориям
   - список по дням: заголовок дня («Сегодня», «Вчера») и сумма за день
   - тап по строке — редактирование, крестик — удаление с подтверждением

3. ПЛАНЫ
   - бюджеты на месяц: список лимитов, «изменить», «убрать»
```

```
   - цели: название, эмодзи, прогресс в %, «из 150 000 ₽», кнопка «Отложить»
   - постоянные платежи: название, день месяца, сумма, «через N дней»
   - долги: «мне должны» и «я должен», имя, сумма, дата возврата

4. ЕЩЁ
   - экспорт CSV (разделитель «;», BOM для Excel)
   - скачать копию JSON, загрузить копию
   - итоги за всё время, удалить все данные

ДАННЫЕ
state = { tx: [], budgets: {}, goals: [], recurring: [], debts: [] }
tx: {id, date:"YYYY-MM-DD", type:"expense"|"income", cat, note, amount}
Хранение: localStorage, ключ money.v2.

ДИЗАЙН
Тёмная тема: фон #080D19, карточки #0F172A, рамки #1E2A44,
акцент #2563EB, доход #10B981, расход #F87171, текст #E8EEF8.
Радиусы 14-18px, отступы 16-20px. Максимальная ширина 520px.

ГОТОВО, КОГДА
- добавил операцию — она сразу в списке, балансе, графике и бюджете;
- перезагрузил страницу — всё на месте;
- на 375px ничего не выезжает;
- в консоли нет ошибок.

НАЧНИ с плана на 6 шагов и жди моего «поехали».
```

Последняя строка — важная. Вы получите план, поправите его словами (дешевле, чем переделывать код), и только потом агент начнёт писать.

Шаг 4. Как это устроено внутри

Пока агент пишет, разберитесь в устройстве. Понимая эти четыре вещи, вы сможете менять приложение сами.

Одно состояние — один источник правды. Все данные лежат в объекте `state`, а экраны лишь перерисовываются из него.

```JS
const state = {
  tx: [],            // операции
  budgets: {},       // {food: 20000, transp: 5000}
  goals: [],         // цели
  recurring: [],     // постоянные платежи
  debts: [],         // долги
  month: "2026-11",  // какой месяц смотрим
  view: "overview"   // какой экран открыт
};
```

Одна функция сохранения. Любое изменение заканчивается вызовом `save()`.

```js
function save(){
  try {
    localStorage.setItem("money.v2", JSON.stringify({
      tx: state.tx, budgets: state.budgets, goals: state.goals,
      recurring: state.recurring, debts: state.debts
    }));
  } catch(e){
    toast("Не удалось сохранить: возможно, мало места");
  }
}
```

Расчёты — отдельно от отрисовки. Сначала считаем цифры, потом рисуем.

```js
function totals(ym){
  const list = state.tx.filter(t => t.date.slice(0,7) === ym);
  const inc = list.filter(t => t.type === "income").reduce((s,t) => s + t.amount, 0);
  const exp = list.filter(t => t.type === "expense").reduce((s,t) => s + t.amount, 0);
  return { list, inc, exp, bal: inc - exp };
}
```

Перерисовка по текущему экрану.

```js
function render(){
  if (state.view === "overview") renderOverview();
  else if (state.view === "tx") { renderFilterChips(); renderTxList(); }
  else if (state.view === "plans") {
    renderPlanBudgets(); renderGoals(); renderRecurring(); renderDebts();
  }
  else if (state.view === "more") renderMore();
}
```

Три цвета статуса. В приложении цвет — не украшение, а сигнал:

ЦВЕТ	ЗНАЧЕНИЕ	ГДЕ ВСТРЕЧАЕТСЯ
Синий `#2563EB`	норма	полоски бюджета до 80%, кнопки
Зелёный `#10B981`	доход, плюс	суммы доходов
Красный `#F87171`	расход, проблема	расходы, превышенный лимит
Жёлтый `#F59E0B`	осторожно	бюджет на 80%

Шаг 5. Логика бюджетов

Это самая полезная функция и самая частая причина ошибок.

```JS
function budgetRow(cat, list){
  const c = catOf("expense", cat);
  const limit = state.budgets[cat] || 0;
  const spent = list.filter(t ⇒ t.type ⩵ "expense" && t.cat ⩵ cat)
                    .reduce((s,t) ⇒ s + t.amount, 0);

  const pct = Math.min(100, Math.round(spent / limit * 100));
  const cls = pct ⩾ 100 ? "over" : pct ⩾ 80 ? "warn" : "";
  const left = limit - spent;

  // 100% — красным «Лимит превышен на N ₽»
  // 80%  — жёлтым «Осталось N ₽»
  return { c, limit, spent, pct, cls, left };
}
```

Три правила:

1. **Процент обрезаем до 100** для ширины полоски, но сумму превышения считаем по-настоящему.
2. **Цвет — не украшение, а сигнал.** Синий = норма, жёлтый = осторожно, красный = стоп.
3. **Предупреждение появляется заранее** — на 80%, а не когда деньги кончились.

Шаг 6. Мобильная вёрстка: 6 правил

1. Нижняя навигация — святое. Фиксирована, не скроллится, 5 элементов максимум. Центральная кнопка «+» приподнята.

2. Safe-area для iPhone. Без этого на iPhone с «монобровью» кнопки упираются в системную полоску.

3. Тап-зона минимум 44px. Крестик удаления нарисован 26px, но зона нажатия вокруг больше.

4. Ничего не выезжает за экран. Проверяется одной строкой в консоли браузера:

```JS
document.documentElement.scrollWidth ⩵ document.documentElement.clientWidth
```

5. Цифры — табличными цифрами. `font-variant-numeric: tabular-nums`. Без этого суммы «прыгают».

6. Контент не прячется под навигацией. Нижний отступ больше высоты панели.

Шаг 7. Полировка: промпт для финальной проверки

Когда всё работает, отправьте агенту этот запрос. Он найдёт то, что вы не заметили.

ПРОМПТ · ПОЛИРОВКА финальная проверка

```
Проверь приложение «Деньги» по списку и исправь найденное:

1. Все пустые состояния: нет операций, нет бюджета, нет целей,
   нет платежей, нет долгов.
2. Экраны 375px и 1440px: ничего не выезжает, навигация
   не перекрывает последний элемент списка.
3. Тап-зоны не меньше 44px, особенно крестик удаления.
4. Форматирование сумм: 1 234 ₽ с неразрывным пробелом,
   tabular-nums для всех цифр.
5. Ошибки localStorage: приватный режим, переполнение —
   приложение не должно падать.
6. Двойные нажатия на «Добавить» не создают две операции.
7. Длинные названия обрезаются с многоточием, а не ломают строку.
8. Контраст текста: второстепенный текст читается на тёмном фоне.

Выведи таблицу: что нашёл / что исправил / что оставил на будущее.
```

Готовое приложение

Полный рабочий код лежит в папке гайда — `bonus/money-app/index.html`. Откройте двойным кликом: работает без установки и без интернета (кроме загрузки шрифта).

Что с ним делать:

- **сравнить** с тем, что написал ваш агент — увидите разницу в деталях;
- **открыть как есть**, если агент застрял — и двигаться дальше;
- **разобрать** по частям: структура данных, расчёты, отрисовка, сохранение.

Код специально не вставлен в книгу целиком: 1 400 строк в PDF — это 30 страниц, которые невозможно читать.

Версия с синхронизацией через Telegram — `bonus/money-app-tg/index.html`. Её и публикуем в главе 9.

Как проверять каждый шаг

#	ДЕЙСТВИЕ	ЧТО ДОЛЖНО БЫТЬ
1	Добавить расход 500 ₽	строка в списке, баланс −500
2	Перезагрузить страницу	всё на месте
3	Переключить месяц назад	цифры и список за другой месяц

#	ДЕЙСТВИЕ	ЧТО ДОЛЖНО БЫТЬ
4	Вернуть месяц	данные вернулись
5	Поставить лимит 1000 ₽ на «Еду»	полоска бюджета появилась на обзоре
6	Добавить расход 850 ₽ на еду	полоска жёлтая, «Осталось 150 ₽»
7	Добавить ещё 500 ₽ на еду	полоска красная, «Лимит превышен на 350 ₽»
8	Создать цель 100 000, отложить 10 000	10% на прогресс-баре
9	Добавить платёж 5-го числа	подпись «через N дней»
10	Добавить долг «Аня 3000»	«должен мне · 3 000 ₽»
11	Поиск «обед»	остались только совпадения
12	Экспорт CSV	открывается в Excel, русские буквы читаются
13	Скачать копию и загрузить её	данные восстановились
14	Открыть на телефоне	ничего не выезжает

Если пункт не проходит — не объясняйте словами. Пришлите агенту номер пункта, скриншот и текст ошибки из консоли.

> Перед переходом к Telegram сделайте коммит: `git add -A && git commit -m "деньги: полная версия работает"`. Или попросите агента: «сделай коммит с текущим состоянием».

20 идей: как читать таблицу

Приложение «Деньги» — это каркас. Дальше вы решаете, каким оно станет. Идеи разбиты на три уровня:

УРОВЕНЬ	ЧТО ЗНАЧИТ	ВРЕМЯ	ЗАЧЕМ
S	маленькая фича, которую видно сразу	вечер	учиться и получать быстрый результат
M	полноценная функция, меняет продукт	выходные	портфолио и первое «вау» у клиента

УРОВЕНЬ	ЧТО ЗНАЧИТ	ВРЕМЯ	ЗАЧЕМ
L	требует сервера или внешнего API	неделя+	то, за что платят деньги

Правило: **не берите L, пока не сделали три S и одну M**. Иначе утонете в архитектуре, не получив ни одного законченного продукта.

Уровень S — 5 идей на вечер

#	ИДЕЯ	ЧТО ДАЁТ	ПРОМПТ В ОДНУ СТРОКУ
1	**Быстрый ввод строкой**	ввод «такси 450» одной строкой, без формы	«добавь поле: ввёл текст вида "такси 450" — создалась операция с категорией по ключевому слову»
2	**Категории с иконками и цветами**	приятнее пользоваться, легче читать	«сделай каждой категории свой цвет и эмодзи, покажи их в графике»
3	**Лимит на месяц**	прогресс-бар «потрачено из 50 000»	«добавь поле "лимит на месяц", полоску прогресса и предупреждение при 80% и 100%»
4	**Тема из Telegram**	интерфейс подстраивается под тему пользователя	«возьми themeParams из Telegram.WebApp и перекрась интерфейс под них»
5	**Импорт из CSV**	перенести данные из другого приложения	«добавь загрузку CSV и создание операций из него»

Уровень M — 10 идей на выходные

#	ИДЕЯ	ЧТО ДАЁТ	СЛОЖНОСТЬ
6	**Регулярные платежи**	подписки, аренда, кредиты — напоминание и автозапись	средняя
7	**Бюджеты по категориям**	«на еду 20 000, осталось 7 400»	средняя
8	**График по месяцам**	тренд: трачу больше или меньше, чем в прошлом	средняя
9	**Долги «кто кому»**	«дал в долг 5 000», «вернули»	средняя
10	**Цели-накопления**	«на отпуск 150 000», прогресс	средняя

#	ИДЕЯ	ЧТО ДАЁТ	СЛОЖНОСТЬ
11	**Мультивалютность**	поездки, доходы в долларах	средняя
12	**Голосовой ввод**	«потратил пятьсот на такси»	средняя. Web Speech API
13	**Скан чеков**	фото чека → сумма и категория	высокая. Нужен внешний vision-API
14	**Отчёт за месяц**	«куда ушли деньги» одной страницей + PDF	средняя. Печать через `window.print()`
15	**Поиск и фильтры**	найти «все траты на Wildberries»	низкая

Уровень L — 5 идей, за которые платят

#	ИДЕЯ	КОМУ ПРОДАВАТЬ	ПОЧЕМУ ПЛАТЯТ
16	**Семейный / командный бюджет**	семьи, пары, небольшие бригады	нужен общий доступ → сервер, аккаунты, роли. От 15 000 ₽
17	**Учёт для самозанятых и ИП**	мастера, репетиторы, курьеры	нужны налоги, чеки, отчёты для банка. От 25 000 ₽
18	**ИИ-советник по тратам**	те, кто «не понимает, куда уходит»	анализ + рекомендации через LLM. Только с сервером, ключ нельзя в клиент! от 40 000 ₽
19	**Белый лейбл под нишу**	автосервисы, барбершопы, клиники	тот же движок, но с брендом клиента. От 50 000 ₽
20	**PWA + офлайн + синк с банком**	предприниматели	открытый банковский API, ежедневная сверка. От 100 000 ₽

> **Важно про идею 18.** Ключ API нельзя держать в коде мини-аппа: любой человек откроет исходники и заберёт его. Всё, что обращается к платной модели, живёт на сервере. Это тема второго гайда.

Какие 3 идеи продавать первыми

Если ваша цель — деньги, а не учёба, начинайте с этих трёх.

1. Учёт для самозанятых (идея 17). Аудитория: мастера маникюра, репетиторы, фотографы, курьеры. Боль: «сколько я реально заработал и сколько отложить на налог». Цена: 15 000–30 000 ₽.

2. Семейный бюджет (идея 16). Аудитория: пары, где деньги «уходят непонятно куда». Цена: 20 000–40 000 ₽. Требует сервера.

3. Белый лейбл для ниши (идея 19). Аудитория: малые бизнесы, у которых нет денег на CRM. Вы продаёте не «приложение», а «личный кабинет клиента с вашим логотипом». Цена: 50 000 ₽ + поддержка 10 000 ₽/мес.

Порядок действий: сделайте демо на 3 идеи → запишите экран телефона на 40 секунд → покажите трём знакомым предпринимателям → спросите «сколько бы вы за это отдали». Дальше решайте не по фантазиям, а по их ответам.

Telegram Mini App: что это и почему это ваш шанс

Mini App — это обычное веб-приложение, которое открывается внутри Telegram по кнопке в боте. Не надо ставить приложение из App Store, не надо проходить модерацию, не надо платить за магазины.

Почему это выгодно вам прямо сейчас:

- **аудитория уже есть** — 900+ млн человек в Telegram, и они привыкли открывать мини-аппы;
- **не нужен сервер для старта** — данные хранит Telegram (`CloudStorage`), бесплатно;
- **не нужен домен** — GitHub Pages или Netlify дают бесплатный HTTPS;
- **легко продать** — предприниматели понимают «личный кабинет в Telegram» лучше, чем «сайт».

Что важно технически: Telegram открывает мини-апп по **HTTPS-ссылке**. Локальный файл (`file://`) и `http://` не подойдут — поэтому мы сначала публикуем, потом привязываем.

Шаг 1. Создаём бота в BotFather

1.1. В Telegram найдите `@BotFather` (галочка верификации). Нажмите «Start».

1.2. Отправьте:

```
CODE
/newbot
```

1.3. BotFather спросит имя — то, что видно в списке чатов. Например: `Мои деньги`.

1.4. Потом спросит username — он должен заканчиваться на `bot`. Например: `my_money_2026_bot`. Если занято — придумайте другой.

1.5. В ответ придёт токен вида:

```
CODE

7123456789:AAH8xQ2m-пример_токена_не_показывайте_никому
```

Токен = полный доступ к боту. Не публикуйте его, не показывайте на видео, не вставляйте в код мини-аппа. Если утёк — в BotFather: `/mybots` → ваш бот → `API Token` → `Revoke`.

1.6. Там же настройте внешний вид (по желанию):

```
CODE

/setdescription
/setabouttext
/setuserpic
```

Шаг 2. Что меняем в коде под Telegram

Самый быстрый путь: у вас уже есть готовый файл — `bonus/money-app-tg/index.html` из папки гайда. В нём всё подключено: SDK Telegram, готовность, `expand()`, синхронизация через CloudStorage кусками по 3500 символов, разрешение конфликтов по времени. Его можно сразу публиковать (шаг 3) и привязывать к боту (шаг 5).

Если хотите повторить руками или применить эти изменения к своему приложению — просто попросите агента:

ПРОМПТ · АДАПТАЦИЯ ПОД TELEGRAM перед публикацией

```
Адаптируй приложение под Telegram Mini App:
1. Подключи https://telegram.org/js/telegram-web-app.js с атрибутом async
   и дождись загрузки SDK (до 2,5 секунд), чтобы приложение работало и в браузере.
2. После загрузки: ready(), expand(), setHeaderColor/setBackgroundColor #080D19.
3. Синхронизация через CloudStorage: при сохранении выгружать JSON кусками
   по 3500 символов в ключи m0, m1, ..., плюс ключ mc с количеством частей
   и ключ mt с меткой времени (писать последним). При запуске — загружать
   из облака и заменять локальные данные, только если облако свежее.
Остальной код не трогай.
```

Почему CloudStorage решает главную проблему

Обычное приложение хранит данные в браузере — и они привязаны к одному устройству. Человек поменял телефон — всё потерял. Для учёта денег это неприемлемо.

Обычный выход — свой сервер с базой данных. Это деньги (от 300 ₽/мес), настройка, безопасность и целая глава работы. Для первого проекта это лишнее.

Telegram даёт третий путь: **облачное хранилище внутри самого Telegram**. Бесплатно, без сервера, и данные привязаны к аккаунту — не к устройству.

схема cloud

Ограничения: до **1024 ключей** и **4096 байт на значение**. Поэтому мы режем JSON на куски по 3500 символов и складываем в `m0`, `m1`, `m2`... Получается до ~3,5 МБ — десятки тысяч операций.

Шаг 3. Публикуем на GitHub Pages

3.1. Зарегистрируйтесь на https://github.com (бесплатно).

3.2. Создайте репозиторий: `+` → `New repository`. Имя: `money-app`. Тип: **Public**.

3.3. Загрузите файл: на странице репозитория → `uploading an existing file` → перетащите `index.html` → `Commit changes`.

3.4. Включите Pages: `Settings` → `Pages` → `Source: Deploy from a branch` → `Branch: main`, папка `/ (root)` → `Save`.

3.5. Через 1–2 минуты появится адрес:

```
CODE
https://ваш-логин.github.io/money-app/
```

Откройте — приложение работает. **Скопируйте ссылку: она нужна для бота.**

> Если вместо приложения видите README — значит `index.html` лежит не в корне или назван иначе. Имя строго `index.html`.

Шаг 4. Альтернатива без git: Netlify Drop

Если GitHub кажется сложным:

1. Откройте https://app.netlify.com/drop
2. Перетащите **папку** `money-app` (не файл) в окно.
3. Получите ссылку вида `https://random-name.netlify.app`.

Ссылка сразу с HTTPS и работает в Telegram. Минус: чтобы обновить — надо зарегистрироваться. Для первого теста идеально.

Шаг 5. Привязываем мини-апп к боту

Вариант А — кнопка в меню (универсальный):

1. `@BotFather` → `/mybots` → выберите бота.
2. `Bot Settings` → `Menu Button` → `Configure Menu Button`.
3. Пришлите ссылку на приложение.
4. Пришлите название кнопки, например `Мои деньги`.

Готово: у бота внизу появится кнопка, которая открывает мини-апп.

Вариант Б — полноценный Mini App с короткой ссылкой:

1. `@BotFather` → `/newapp` → выберите бота.
2. Введите название, описание, загрузите картинку 640×360.
3. Укажите `Web App URL` — ссылку из шага 3 или 4.

BotFather выдаст короткий адрес вида `t.me/ваш_бот/имя_приложения`.

Шаг 6. Тест на телефоне: чеклист

- [ ] Кнопка меню видна и открывает приложение
- [ ] Приложение открылось на весь экран, без рамки браузера
- [ ] Шапка тёмная, не «прыгает» при открытии
- [ ] Можно добавить расход — цифры обновились
- [ ] Закрыл мини-апп, открыл снова — данные на месте
- [ ] Открыл с другого устройства под тем же аккаунтом — данные те же
- [ ] Кнопка «назад» в Telegram не закрывает приложение неожиданно
- [ ] На маленьком экране ничего не обрезано
- [ ] Нет горизонтальной прокрутки

Ошибки мини-аппов, из-за которых их не ставят

СИМПТОМ	ПРИЧИНА	РЕШЕНИЕ
«Не удалось открыть» / пустой экран	ссылка `http://` или `file://`	нужен **HTTPS**-адрес
Открылось, но пусто	`index.html` не в корне или 404	проверьте путь и имя файла
Данные не сохраняются	CloudStorage не подключён или данные > 4096 байт	режем на куски
Вёрстка «съехала» внизу	нет учёта безопасной зоны	`padding-bottom: env(safe-area-inset-bottom)`
Кнопка меню не появилась	не сохранена настройка в BotFather	`/mybots` → `Bot Settings` → `Menu Button`
Старая версия после обновления	кэш WebView	в BotFather пересохраните URL (добавьте `?v=2`)

СИМПТОМ	ПРИЧИНА	РЕШЕНИЕ
Приложение «дёргается» при скролле	`expand()` не вызван	вызовите `tg.expand()`

> **Правило публикации:** сначала тест на своём телефоне, потом отправка клиенту. Ссылка, которая не открывается у заказчика, стоит вам заказа.

Где брать заказы: 8 каналов

Навык «сделать мини-апп» и навык «продать мини-апп» — разные. Второй важнее. Вот каналы, которые работают без бюджета на рекламу, от простого к сложному.

схема money

1. Своё окружение (первые 2 заказа — почти всегда здесь). Друзья, знакомые предприниматели, чаты вашего города. Формулировка: «Я учусь делать приложения в Telegram. Сделаю тебе учёт клиентов за отзыв». Бесплатно — только первые 2–3, дальше это обесценивает работу.

2. Instagram / свой блог. Самый быстрый канал, если есть аудитория. Не «я разработчик», а «показываю, как за вечер собрать приложение». Формат: 30-секундное видео экрана телефона + «напиши «ХОЧУ» в комментариях».

3. Telegram-чаты и каналы по нишам. Ищите не «чат фрилансеров», а чаты предпринимателей: барбершопы, мастера маникюра, автосервисы, кофейни. Правило: сначала 2 недели просто помогаете, потом предлагаете. Спам = бан.

4. Kwork и аналоги. Заказы «бот для записи», «мини-апп для магазина» есть каждый день. Начинайте с чека 5 000–10 000 ₽, чтобы набрать 5 отзывов, дальше поднимайте цену.

5. Авито и Профи. Работает для локального бизнеса: «сделаю мини-приложение для записи клиентов». Объявление с 3 скриншотами и ценой «от 15 000 ₽».

6. Холодные сообщения в Instagram бизнесам. Самый тяжёлый, но самый масштабируемый. Найдите 20 салонов без записи в Telegram → напишите персонально, с конкретикой про их бизнес. Конверсия 5–10% — это нормально.

7. Партнёры. Маркетологи, таргетологи, дизайнеры — у них есть клиенты, которым нужен «личный кабинет». Предложите 10–15% за приведённого клиента.

8. Сарафан. Приходит сам после 3–5 сделанных проектов. Условие: каждый проект заканчивается просьбой «порекомендуйте меня двум знакомым» и ссылкой на кейс.

Вилки цен 2026

ЧТО ДЕЛАЕТЕ	ЦЕНА	СРОК
Мини-апп «калькулятор / квиз / визитка»	10 000 – 25 000 ₽	1–3 дня
Учёт/трекер (как наше «Деньги») без сервера	15 000 – 35 000 ₽	2–5 дней
Мини-апп с базой данных и админкой	60 000 – 150 000 ₽	2–4 недели
Магазин внутри Telegram (каталог + заказы)	80 000 – 200 000 ₽	3–6 недель
Бот + мини-апп + оплата	100 000 – 250 000 ₽	3–6 недель
Поддержка и доработки	10 000 – 30 000 ₽/мес	постоянно

Три правила ценообразования для новичка:

1. **Не называйте цену первым**, пока не спросили «а сколько вы планировали?». Часто бюджет больше, чем вы думаете.
2. **Считайте не часы, а результат.** Клиенту всё равно, что вы сделали это за вечер.
3. **Никогда не работайте без предоплаты 50%.** Это не жадность, а фильтр.

Скрипт первого сообщения

ШАБЛОН · ПЕРВОЕ СООБЩЕНИЕ — для поиска клиентов

```
Здравствуйте! Смотрел ваш профиль — вы записываете клиентов через
директ, и, судя по сторис, иногда отвечаете с задержкой.

Я делаю мини-приложения в Telegram: клиент открывает бота, выбирает
услугу и время — запись приходит вам сообщением. Ничего скачивать
не надо, всё внутри Telegram.

Сделал демо на примере <ниша> — 40 секунд видео: <ссылка>
Если актуально, покажу, как это будет выглядеть с вашими услугами
и ценами. Это бесплатно и без обязательств.
```

Почему работает: сначала наблюдение о его проблеме, потом решение, потом демо, потом «бесплатно показать». Ни одного слова «купить».

Что отвечать на «сколько стоит»:

ШАБЛОН · ОТВЕТ НА «СКОЛЬКО СТОИТ» — когда спросили цену

```
Зависит от объёма. Базовый вариант с записью и уведомлениями — от 25 000 ₽,
делаю 3-4 дня. Если нужна оплата и личный кабинет — от 60 000 ₽.
```

```
Давайте я задам 5 вопросов и скажу точную цену за 10 минут?
```

Как вести первый заказ, чтобы не пожалеть

1. Пять вопросов до цены. Кто пользователь? Что он делает в приложении? Какие данные храним? Нужна ли оплата? Что считаем «готово»?

2. Короткое ТЗ на одну страницу. Не юридический договор, а список экранов и функций + что НЕ входит. Отправьте клиенту и попросите написать «согласен».

3. Предоплата 50%. Вторые 50% — после показа рабочей версии, до передачи исходников.

4. Демо каждые 2–3 дня. Скринкаст на 30 секунд лучше, чем «всё готово, покажу в конце».

5. Две итерации правок включены. Третья и дальше — по 2 000–5 000 ₽ за круг. Пропишите это заранее.

6. Передача. Ссылка + инструкция на одну страницу + исходники в архиве + 15 минут созвона. После передачи — предложите поддержку за 10 000 ₽/мес. Треть клиентов соглашается.

7. Отзыв и кейс. Скриншот, 2–3 предложения результата, разрешение показать.

Что говорить, когда «дорого»

Не снижайте цену — уменьшайте объём:

ШАБЛОН · КОГДА «ДОРОГО» вместо скидки

```
Понимаю. Давайте сделаем первый этап: только запись и уведомления,
без оплаты и кабинета, — 15 000 ₽. Запустим, посмотрим, как пользуются,
и добавим остальное позже.
```

Так вы получаете заказ, клиент — результат, а не «скидку 50%». Скидка обесценивает работу и притягивает самых сложных клиентов.

> Деньги в этом гайде — не главная цель. Главная — **первый законченный продукт и первый человек, который им пользуется**. После этого заказы становятся вопросом времени и настойчивости, а не удачи.

Топ-10 ошибок новичка

#	ОШИБКА	ЧЕМ ЗАКАНЧИВАЕТСЯ	КАК ИЗБЕЖАТЬ
1	Начать с «супер-приложения» на 20 экранов	через неделю брошено	один экран, одна функция, один вечер
2	Не делать git-коммиты	агент сломал — откатить нечего	коммит после каждого рабочего шага
3	Ставить 297 скиллов и не понимать, как они работают	«ничего не изменилось»	проверять скилл на конкретной задаче (глава 5)
4	Просить «сделай красиво»	5 кругов правок и всё равно не то	конкретика: отступы, цвета, размеры
5	Работать в одной папке на все проекты	файлы путаются, агент правит не то	один проект = одна папка
6	Держать ключ API в чате и на скриншотах	чужие траты с вашего баланса	ключ только в конфиге, лимит на ключ
7	Отправлять клиенту ссылку без теста на телефоне	«у меня не открывается»	чеклист из главы 9
8	Работать без предоплаты	сделали — клиент исчез	50% вперёд, всегда
9	Спорить с моделью 10 раз подряд	сожжённые деньги и время	на третьей неудаче — меняем модель
10	Учиться «вообще», без цели	через месяц всё забыто	каждый день — один маленький готовый результат

Чеклист: я всё сделал

- [ ] Приложение OpenCode установлено и открывается
- [ ] Ключ SpiderSense создан, лимит расходов выставлен
- [ ] Промпт настройки выполнен, модели появились в списке
- [ ] DeepSeek V4.1 Flash выбран по умолчанию
- [ ] 297 скиллов установлены, агент их видит
- [ ] Первый сайт-визитка сделан одной фразой

- [ ] Git инициализирован, первый коммит есть
- [ ] `AGENTS.md` с правилами проекта создан
- [ ] Приложение «Деньги» работает локально
- [ ] Приложение опубликовано и открывается по HTTPS-ссылке
- [ ] Бот создан, мини-апп привязан, кнопка в меню работает
- [ ] Тест на телефоне пройден по всем пунктам
- [ ] Данные синхронизируются между устройствами
- [ ] Есть 3 идеи из главы 8, которые вы хотите продать
- [ ] Написан первый вариант сообщения клиенту

Если отмечено 12 из 15 — вы уже впереди 95% тех, кто «интересуется нейросетями».

Домашка с пруфом

Это не формальность. Домашка нужна, чтобы вы не просто прочитали, а **получили результат**.

Задание: сделать свой вариант приложения «Деньги» (не копию — добавьте минимум две идеи из главы 8), опубликовать и привязать к боту.

Пруф: скриншот или экранная запись на 20–30 секунд, где видно, как вы добавляете операцию и открываете приложение из Telegram.

Куда: в наш чат с хэштегом `#первыйминиапп`:

```
CODE
https://t.me/+WNNP-XISz8NjYTQ6
```

Что вы получите: разбор от меня, обратную связь по интерфейсу и — самое ценное — увидите 20–30 работ других участников.

> Правило чата: скриншот + вопрос. «Не работает» без скриншота и текста ошибки разбирать нечем. И никогда не присылайте API-ключ.

Что дальше: гайд №2

Первое приложение живёт без сервера — и это правильно. Но как только появляются **деньги, клиенты и общий доступ**, начинается второй уровень. Ему посвящён платный гайд:

«Мини-апп на сервере: база данных, оплата, админка, продажа»

Что будет внутри:

- сервер за 300 ₽/мес с HTTPS и доменом;
- база данных: клиенты, заказы, роли;
- админка, в которой владелец бизнеса видит заявки;
- приём оплаты внутри Telegram;
- уведомления в чат, напоминания, отчёты;
- как упаковать это в продукт с ценой 100 000+ ₽;
- шаблон ТЗ, договора и инструкции для клиента.

Смысл разделения простой: **сначала вы делаете бесплатно то, что уже приносит пользу, — потом продаёте то, что уже умеете**.

Поддержка и связь

Автор: Максим Холмов, WWCODING — цифровая архитектура для бизнеса.

КУДА	АДРЕС
Чат участников (вопросы, домашка, пруфы)	https://t.me/+WNNP-XISz8NjYTQ6
Личный Telegram	`@wroomwroomseo`
Сайт	wwcoding.ru
Почта	wroomwroomsup@yandex.ru
Провайдер моделей	https://spiderssense.com/register?ref=bef11400

Что спрашивать можно: ошибки в установке, непонятные сообщения агента, проверка идеи, разбор первого заказа.

Что не надо присылать: свои API-ключи, пароли, доступы к серверам. Никогда.

> Один финальный совет, который стоит дороже остальных глав: **не учитесь «нейросетям»**. Учитесь делать конкретные вещи — сайт, приложение, бота. Инструменты меняются каждые полгода, а умение доводить до результата остаётся навсегда.

Шпаргалка: всё на одной странице

Скачать приложение

СИСТЕМА	ССЫЛКА
Windows	opencode-desktop-win-x64.exe
macOS Apple Silicon	opencode-desktop-mac-arm64.dmg
macOS Intel	opencode-desktop-mac-x64.dmg
Все версии	https://opencode.ai/download

Ключ и оплата

```text
Регистрация: https://spiderssense.com/register?ref=bef11400
Пополнить:   5 $ (проходит через СБП из приложения банка)
Адрес API:   https://spiderssense.com/v1
Вопросы по оплате и установке: https://t.me/+WNNP-XISz8NjYTQ6
```

Где лежат файлы

```text
Конфиг:  Windows  %USERPROFILE%\.config\opencode\opencode.json
         macOS    ~/.config/opencode/opencode.json

Скиллы:  Windows  %USERPROFILE%\.config\opencode\skills
         macOS    ~/.config/opencode/skills
```

Git-спасатель

```bash
git init
git add -A && git commit -m "рабочая версия"
git restore .            # откатить всё к последнему коммиту
git reset --hard HEAD    # жёсткий откат
git status               # что происходит
```

Публикация

СПОСОБ	ССЫЛКА	ПЛЮС
GitHub Pages	github.com → Settings → Pages	бесплатно, обновляется коммитом
Netlify Drop	app.netlify.com/drop	30 секунд, без регистрации

СПОСОБ	ССЫЛКА	ПЛЮС
Свой хостинг	залить `index.html` в корень	если домен уже есть

Бот и мини-апп

```text
@BotFather → /newbot                 создать бота
@BotFather → /mybots → Menu Button   кнопка запуска мини-аппа
@BotFather → /newapp                 полноценный Mini App
```

Словарь

ТЕРМИН	ПРОСТЫМИ СЛОВАМИ
Агент	программа, которая читает файлы, пишет код и запускает команды по вашей просьбе
Модель	«мозг», который придумывает ответ
API	способ обращаться к модели по интернету, оплата по количеству текста
Токен	кусочек текста (~0,7 слова). Токенами меряют объём и цену
Провайдер	посредник, который продаёт доступ к моделям
Скилл	файл с инструкцией, который агент подгружает под задачу
Промпт	ваша задача, сформулированная текстом
Git	история изменений проекта, «машина времени»
Коммит	сохранённая точка в истории
localStorage	память браузера, данные на одном устройстве
CloudStorage	память Telegram, данные на всех устройствах
Mini App	веб-приложение, открывающееся внутри Telegram
HTTPS	защищённое соединение; обязательно для мини-аппа
Деплой	публикация проекта в интернете

Что читать дальше

- OpenCode — https://opencode.ai/docs

- Telegram Mini Apps — https://core.telegram.org/bots/webapps
- Telegram Bot API — https://core.telegram.org/bots/api

Что попросить у агента вместо чтения документации:

```
CODE

Объясни простыми словами, как работает [тема] в моём проекте,
и покажи на примере моего файла. Без общих слов.
```

Финальная мысль

Вы прочитали гайд, где нужно было несколько раз что-то сделать руками. Если сделали — у вас теперь есть то, чего нет у большинства «интересующихся ИИ»: **работающее приложение, опубликованное, с пользователями и понятным путём к деньгам**.

Дальше всё просто: делайте по одному маленькому продукту в неделю. Через месяц у вас будет портфолио. Через три — первые деньги. Через год — репутация человека, который «делает», а не «разбирается».

Инструменты будут меняться. Этот навык — нет.

WWCODING — цифровая архитектура для бизнеса · Максим Холмов

Telegram @wroomwroomseo · чат участников t.me/+WNNP-XlSz8NjYTQ6 · wwcoding.ru