

# Codex



## В деле

Практическое  
руководство  
для разработчиков  
по работе с ИИ-агентом  
OpenAI Codex

-  14 готовых рецептов
-  Шпаргалки команд
-  Шаблоны
-  Библиотека запросов
-  Чек-листы
-  План внедрения  
в команду

От простых  
задач до CI  
и облачных запусков



Милана Панова

**Codeх в деле. Практическое  
руководство для разработчиков по  
работе с ИИ-агентом OpenAI Codeх**

«Автор»

2026

**Панова М.**

Codex в деле. Практическое руководство для разработчиков по работе с ИИ-агентом OpenAI Codex / М. Панова — «Автор», 2026

Codex от OpenAI — уже не подсказчик строк кода, а полноценный ИИ-агент: он читает репозиторий, правит файлы, запускает тесты, работает в облаке и встраивается в CI. Но без правильных навыков агент быстро превращается в генератор технического долга. Эта книга — практическое руководство для разработчиков, тимлидов и DevOps-инженеров. Вы научитесь ставить задачи так, чтобы агент с первого раза делал то, что нужно, настраивать права доступа без риска для кода и данных, вести AGENTS.md и создавать скиллы, подключать внешние сервисы через MCP, автоматизировать ревью и рутину через codex exec и CI, запускать задачи в Codex Cloud и проверять работу агента. Внутри — 14 готовых рецептов, шпаргалки команд, шаблоны, библиотека запросов, чек-листы и план внедрения Codex в команду. Материал актуален на осень 2026 года.

© Панова М., 2026

© Автор, 2026

# Содержание

|                                                             |    |
|-------------------------------------------------------------|----|
| Предисловие                                                 | 6  |
| Для кого эта книга                                          | 7  |
| Как читать книгу                                            | 8  |
| Важная оговорка о версиях                                   | 9  |
| Часть I. Основы                                             | 10 |
| Глава 1. Что такое Codex и чем агент отличается от чат-бота | 11 |
| От автодополнения к агенту                                  | 11 |
| Цикл работы агента                                          | 11 |
| Где живёт Codex: поверхности продукта                       | 11 |
| Модели под капотом                                          | 12 |
| Доступ и тарифы                                             | 12 |
| Чем Codex не является                                       | 12 |
| Итоги главы                                                 | 12 |
| Глава 2. Установка и первый запуск                          | 13 |
| Установка CLI                                               | 13 |
| Вход в аккаунт                                              | 13 |
| Подготовка учебного проекта                                 | 14 |
| Первая сессия                                               | 14 |
| Три специальных символа в поле ввода                        | 14 |
| Первое изменение                                            | 14 |
| Итоги главы                                                 | 15 |
| Глава 3. Модели и уровень рассуждений                       | 16 |
| Линейка моделей                                             | 16 |
| Как переключать модель                                      | 16 |
| Уровень рассуждений                                         | 16 |
| Локальные модели                                            | 17 |
| Итоги главы                                                 | 17 |
| Глава 4. Безопасность: песочница и подтверждения            | 18 |
| Два механизма контроля                                      | 18 |
| Готовые режимы прав                                         | 18 |
| Защищённые пути                                             | 18 |
| Флаг, который нельзя использовать на рабочей машине         | 18 |
| Что всегда читать перед подтверждением                      | 19 |
| Prompt injection и веб-поиск                                | 19 |
| Секреты                                                     | 19 |
| Итоги главы                                                 | 19 |
| Часть II. Ежедневная работа                                 | 20 |
| Глава 5. Искусство постановки задачи                        | 21 |
| Агент — это новый коллега, у которого открыт ваш проект     | 21 |
| Четыре элемента хорошей задачи                              | 21 |
| Плохо и хорошо                                              | 21 |
| Примеры лучше прилагательных                                | 21 |
| Давайте настоящие улики                                     | 22 |
| Не диктуйте реализацию без необходимости                    | 22 |
| Одна задача — один результат                                | 22 |
| Шаблоны запросов                                            | 22 |

Конец ознакомительного фрагмента.

23

# **Милана Панова**

## **Codex в деле. Практическое руководство для разработчиков по работе с ИИ-агентом OpenAI Codex**

### **Предисловие**

Эта книга — о том, как перестать «спрашивать нейросеть» и начать работать с ИИ-агентом как с членом команды. Codex от OpenAI умеет гораздо больше, чем подсказывать строку кода: он читает репозиторий, правит десятки файлов, запускает тесты, сам находит свои ошибки, работает в облаке, пока ваш ноутбук закрыт, и встраивается в CI. Но чтобы всё это приносило пользу, а не хаос в истории Git, нужны навыки: правильно ставить задачу, ограничивать права, проверять результат и выстраивать повторяемые процессы.

## Для кого эта книга

- Для backend-, frontend- и fullstack-разработчиков, которые хотят ускорить рутину: баг-фиксы, тесты, рефакторинг, миграции.
- Для тимлидов и техлидов, которым нужно внедрить агентов в команду так, чтобы не пострадали качество кода и безопасность.
- Для DevOps- и SRE-инженеров, которым интересна автоматизация через codex exes, CI и работу на серверах.
- Для QA-инженеров и аналитиков, которые пишут скрипты и хотят доверить агенту черновую работу.
- Для студентов и начинающих — при условии, что вы готовы читать код, который пишет агент, а не просто копировать его.

## Как читать книгу

Книга разбита на пять частей. Первая объясняет, что такое Codex, как его установить и как устроена безопасность. Вторая — ежедневная работа: постановка задач, файл AGENTS.md, планирование, ревью, управление сессиями. Третья — расширение возможностей: скиллы, MCP, плагины, хуки, субагенты и конфигурация. Четвёртая — автоматизация: скрипты, CI, Codex Cloud и десктоп-приложение. Пятая — практика: готовые рецепты, типичные ошибки, экономика лимитов и внедрение в команду. В приложениях собраны шпаргалки, шаблоны и чек-листы, к которым удобно возвращаться.

Если вы уже пользуетесь Codex, начните с глав 5, 6 и 8 — они дают наибольший прирост качества. Если только начинаете — читайте по порядку и повторяйте примеры на учебном репозитории.

## Важная оговорка о версиях

Codex развивается очень быстро: новые модели выходят раз в несколько месяцев, команды и флаги CLI меняются от релиза к релизу. Материал книги соответствует состоянию продукта на октябрь 2026 года (после конференции OpenAI DevDay 2026). Названия моделей, тарифы и лимиты обязательно сверяйте с актуальной официальной документацией, а для своей установленной версии используйте `codex --help` и `codex <команда> --help`. Принципы работы — постановка задач, контроль прав, ревью, автоматизация — меняются гораздо медленнее, и именно на них сделан главный упор.

**Обратите внимание.** Книга не является официальным изданием OpenAI. Все упомянутые названия продуктов и товарные знаки принадлежат их правообладателям.

## **Часть I. Основы**

Первая часть закладывает фундамент: что такое Codex, как он устроен, как его установить, какую модель выбрать и как настроить безопасность так, чтобы агент помогал, а не создавал риски.

## Глава 1. Что такое Codex и чем агент отличается от чат-бота

### От автодополнения к агенту

За последние годы инструменты ИИ для программистов прошли три стадии. Сначала было автодополнение: модель предлагала продолжение строки. Затем — чат: вы копировали кусок кода в окно, получали ответ и вставляли его обратно. Третья стадия — агенты. Агент получает цель, сам решает, какие файлы прочитать, какие команды выполнить и что изменить, проверяет результат и сообщает о проделанной работе.

Codex — агент OpenAI для разработки. Его ключевое отличие от чата в том, что у него есть «руки»: доступ к файловой системе проекта, терминалу, Git, браузеру и внешним сервисам через подключаемые инструменты. Поэтому меняется и роль разработчика: вы меньше пишете код руками и больше ставите задачи, задаёте рамки и проверяете результат.

### Цикл работы агента

Почти любая задача в Codex проходит один и тот же цикл:

1. Вы описываете желаемый результат и критерий готовности.
2. Codex изучает проект: читает файлы, ищет по коду, смотрит историю Git, инструкции в AGENTS.md.
3. Агент вносит изменения и запускает команды: сборку, тесты, линтеры.
4. Агент проверяет результат и, если что-то сломалось, исправляет.
5. Вы просматриваете diff, принимаете, просите доработать или откатываете.

Хорошая новость: каждый шаг этого цикла можно настроить. Плохая: если пропустить первый (нечёткая постановка) или последний (отсутствие ревью), агент быстро превратится в генератор технического долга.

### Где живёт Codex: поверхности продукта

Один и тот же агент доступен в нескольких интерфейсах. Они используют общую конфигурацию (~/.codex/config.toml), поэтому настройки, сделанные в одном месте, работают и в другом.

- **Codex CLI** — Работа в терминале, на удалённых серверах по SSH, скрипты и CI через `codex exec`
- **Десктоп-приложение** — Режим Codex внутри приложения ChatGPT для macOS, Windows и Linux: панели diff, терминала, файлов, встроенный браузер, worktrees, задачи по расписанию
- **Расширение для IDE** — Работа от открытого в редакторе кода и выделенного фрагмента
- **Codex Cloud** — Длинные задачи на виртуальных машинах OpenAI, пока ваш компьютер выключен; несколько попыток решения одной задачи
- **Интеграции** — Ревью пулл-реквестов на GitHub, запуск задач из Slack и Microsoft Teams в корпоративных тарифах
- **SDK и CI** — Встраивание агента в собственные инструменты и пайплайны

CLI — проект с открытым исходным кодом, написанный на Rust (репозиторий `openai/codex` на GitHub). Это удобно: можно читать исходники, следить за изменениями и сообщать об ошибках.

## Модели под капотом

Codex — это «обвязка» (harness), а интеллект ему дают модели OpenAI. На момент написания книги основной моделью для Codex является GPT-6.1 Sol, представленная на DevDay 29 сентября 2026 года. Для самых сложных задач есть более мощная и дорогая GPT-6 Astra, для простых повторяющихся задач — быстрая и экономная GPT-6 Luna. Подробно о выборе модели и уровня рассуждений — в главе 3.

## Доступ и тарифы

Codex входит в планы ChatGPT. Бесплатные и самые дешёвые тарифы дают доступ к Codex в десктоп-приложении с облегчённой моделью; планы уровня Plus и выше добавляют CLI, расширение IDE, облачные функции и основные модели; дорогие планы Pro, а также Business и Enterprise расширяют лимиты и открывают самые мощные режимы. Лимиты Codex общие с остальной работой в ChatGPT и сбрасываются по скользящим окнам.

Альтернатива — вход по ключу OpenAI API: тогда вы платите за токены по тарифам API, а не из лимитов подписки, но часть облачных функций, привязанных к аккаунту ChatGPT, недоступна. Для CI и автоматизации ключ API обычно удобнее.

## Чем Codex не является

Codex не заменяет инженерное мышление. Агент не знает бизнес-контекста, который не записан в коде или документации; не несёт ответственности за продакшен; может уверенно ошибаться. Относитесь к нему как к очень быстрому и начитанному, но новому в проекте коллеге: ему нужно объяснять контекст, его работу нужно проверять, а доступы выдавать по принципу минимально необходимых.

## Итоги главы

- Codex — агент: он сам читает код, меняет файлы и запускает команды.
- Один агент доступен в CLI, десктоп-приложении, IDE, облаке и CI с общей конфигурацией.
- Качество результата определяется постановкой задачи и проверкой, а не только моделью.

## Глава 2. Установка и первый запуск

### Установка CLI

Самый быстрый способ на macOS и Linux — официальный скрипт установки. Эта же команда обновляет уже установленную версию:

```
curl -fsSL https://chatgpt.com/codex/install.sh | sh
```

Альтернативы — Homebrew или npm (если у вас уже стоит Node.js):

```
brew install --cask codex  
npm install -g @openai/codex
```

На Windows установка выполняется из PowerShell:

```
powershell -ExecutionPolicy ByPass -c "irm https://chatgpt.com/codex/install.ps1 | iex"
```

Проверьте результат и запомните две полезные команды:

```
codex --version # какая версия установлена  
codex update # обновиться тем же способом, каким установлено  
codex doctor # диагностика установки, конфига, входа и окружения
```

Песочница на Linux опирается на bubblewrap; если ваш пакетный менеджер не поставил его автоматически, установите отдельно. На macOS песочница использует встроенный механизм Seatbelt и ничего дополнительно не требует.

### Вход в аккаунт

При первом запуске codex предложит выбрать способ входа:

- **Вход через ChatGPT** — используются лимиты вашей подписки. Вариант по умолчанию для большинства.

- **Вход по коду устройства** — для машин без браузера, например сервера, к которому вы подключились по SSH. Код вводится на другом устройстве.

- **Собственный ключ API** — оплата за токены по тарифам OpenAI API.

Войти можно и без запуска сессии:

```
codex login # обычный вход через браузер  
codex login --device-auth # вход по коду на сервере  
printenv OPENAI_API_KEY | codex login --with-api-key  
codex login status # каким способом вы сейчас вошли
```

**Совет.** Передавайте ключ API через переменную окружения и конвейер, как в примере выше, а не прямо в командной строке. Так он не попадёт в историю оболочки.

## Подготовка учебного проекта

Для экспериментов заведите маленький репозиторий, который не жалко сломать. Обязательно инициализируйте Git и сделайте первый коммит: Git — ваша кнопка «отменить» при работе с агентом, а многие функции Codex (просмотр diff, ревью, codex exec) опираются на него.

```
mkdir codex-lab && cd codex-lab
git init
# ... создайте пару файлов с кодом и тестами ...
git add . && git commit -m "Initial commit"
```

## Первая сессия

Запустите codex в папке проекта. Первый вопрос, который вы увидите, — «доверять ли этой папке». Доверие разрешает Codex загружать проектные настройки из .codex/, хуки и правила, которые могут автоматически выполнять код. Для своего проекта отвечайте «да». Для только что клонированного чужого репозитория — сначала посмотрите, что лежит в .codex/.

Первый запрос делайте «только для чтения» — так вы увидите, как агент понимает проект, ничего не рискуя:

Объясни, что делает этот проект и как он устроен. Ничего не меняй.

В ленте над полем ввода вы увидите каждый прочитанный файл и каждую выполненную команду. Это важная привычка: смотрите не только на итоговый ответ, но и на ход работы.

## Три специальных символа в поле ввода

- @ — упомянуть файл: начните вводить имя, выберите из списка, и путь попадёт в запрос.
- ! — выполнить команду оболочки самому, не выходя из Codex, например !npm test. Команда исполняется с теми же ограничениями песочницы.
- \$ — вызвать скилл по имени (подробнее в главе 10).

Картинки тоже работают: вставьте скриншот ошибки или макет в поле ввода или передайте файл при запуске:

```
codex -i error.png "Объясни ошибку на скриншоте и предложи минимальное исправление"
```

## Первое изменение

Теперь попросите агента что-то изменить и проверить:

Добавь в функцию parseDate обработку пустой строки: она должна возвращать null. Допиши тест на этот случай и запусти тесты.

Codex откроет нужные файлы, внесёт правку, добавит тест и запустит тестовый набор. Каждое изменение отображается как diff. С настройками по умолчанию агент не спрашивает

разрешения на редактирование файлов внутри проекта и запуск тестов, но остановится и спросит, если ему понадобится сеть (например, чтобы установить пакет).

## **Итоги главы**

- Установка — одна команда; codex doctor помогает при проблемах.
- Работайте только в Git-репозитории и коммитьте перед крупными задачами.
- Начинайте знакомство с проектом с запросов «только для чтения».

## Глава 3. Модели и уровень рассуждений

### Линейка моделей

Выбор модели — это компромисс между качеством, скоростью и расходом лимитов. На октябрь 2026 года в Codex доступны три основные модели семейства GPT-6:

- **GPT-6.1 Sol (gpt-6.1-sol)** — Модель по умолчанию. Сложная разработка, отладка, многофайловые изменения. По данным OpenAI, приближается к Astra по качеству при заметно меньшей стоимости

- **GPT-6 Astra** — Самые трудные задачи: архитектурные изменения, запутанные баги, крупные миграции. Дороже и медленнее

- **GPT-6 Luna (gpt-6-luna)** — Сфокусированные, повторяемые задачи: форматирование, простые тесты, массовые однотипные правки, скрипты в CI

Для ориентира: в API GPT-6.1 Sol стоит 2 доллара за миллион входных токенов, 0,10 доллара за миллион заэкшированных входных и 10 долларов за миллион выходных. В подписке вы платите не деньгами, а лимитами, но пропорции расхода похожи.

**Совет.** Названия моделей меняются. Открывайте /model, чтобы увидеть список, доступный именно вашему плану, и не зашивайте устаревшие имена в скрипты без необходимости.

### Как переключать модель

- /model внутри сессии открывает выбор модели, а затем уровня рассуждений. Выбор сохраняется в ~/.codex/config.toml и применяется к новым сессиям.

- codex -m gpt-6-luna запускает сессию с конкретной моделью.

- codex -c 'model\_reasoning\_effort="high"' разово меняет уровень рассуждений.

- /fast переключает быстрый режим: ответы приходят быстрее, но расходуют лимиты в 2,5 раза интенсивнее.

- /status показывает текущую модель, права и остаток контекстного окна; /usage — расход лимитов плана.

Учтите: если строка model записана в вашем конфиге, она побеждает встроенное значение по умолчанию. Когда OpenAI сменит модель по умолчанию, вы останетесь на той, что выбрали сами, — иногда это полезно, иногда нет.

### Уровень рассуждений

Уровень рассуждений (reasoning effort) определяет, сколько модель «думает» перед действием. Высокий уровень помогает на сложных задачах, но работает дольше и тратит больше лимитов. Практическое правило:

- **Опечатка, переименование, правка конфига** — низкий

- **Типовая фича, тест, небольшой рефакторинг** — средний

- **Отладка нестабильного бага, миграция, изменения в нескольких модулях** — высокий

- **Архитектурное решение, конкурентность, безопасность** — максимальный

Не ставьте максимальный уровень «на всякий случай»: на простых задачах он не улучшает результат, а лишь замедляет работу.

## **Локальные модели**

Codex умеет работать с локальными моделями через Ollama или LM Studio. Это бесплатно и полезно, когда код нельзя отправлять во внешний сервис, но такие модели заметно слабее. Давайте им маленькие, чётко определённые задачи.

```
codex --oss --local-provider ollama -m qwen3-coder
```

## **Итоги главы**

- Начинайте с модели по умолчанию; переходите на Astra для сложного, на Luna — для рутинного.
- Подбирайте уровень рассуждений под задачу, а не под тревожность.
- Следите за расходом через `/status` и `/usage`.

## Глава 4. Безопасность: песочница и подтверждения

### Два механизма контроля

Агент, который может выполнять команды на вашей машине, требует продуманной защиты. В Codex за неё отвечают два независимых механизма:

- **Песочница (sandbox)** определяет, что команды агента могут технически сделать: в какие папки писать и есть ли доступ к сети.
- **Политика подтверждений (approval policy)** определяет, когда агент должен остановиться и спросить вас, прежде чем выйти за пределы песочницы.

### Готовые режимы прав

Быстрее всего настроить оба механизма командой `/permissions`. Она предлагает четыре пресета:

- **Read Only** — Только чтение файлов проекта. Перед редактированием и выходом в сеть агент спрашивает
- **Default** — Чтение и редактирование файлов проекта, запуск команд. Спрашивает перед доступом к сети и правкой файлов вне проекта
- **Approve for me** — Те же границы, что у Default, но на запросы подтверждения отвечает отдельный агент-проверяющий. Расширить песочницу он не может
- **Full Access** — Правка любых файлов и доступ в интернет без вопросов. Включается только после предупреждения

Режим Default подходит для большинства задач. Read Only удобен, когда вы изучаете чужой код или хотите получить только ответы.

Те же режимы задаются при запуске:

```
codex --sandbox read-only
codex --approve-for-me
codex --add-dir ../shared-lib # дать запись во вторую папку вместо Full Access
```

### Защищённые пути

Даже в режиме записи некоторые пути внутри проекта остаются доступными только для чтения: папки `.git`, `.codex`, `.agents`, а также `.aws`, где обычно хранятся облачные учётные данные. Поэтому агент не может без вашего ведома переписать историю Git или собственные настройки.

### Флаг, который нельзя использовать на рабочей машине

У CLI есть флаг `--dangerously-bypass-approvals-and-sandbox`. Он отключает все проверки. Используйте его только внутри одноразового контейнера или виртуальной машины, где нет ничего ценного: ни ключей, ни доступа к рабочим сетям, ни личных данных.

## Что всегда читать перед подтверждением

Когда агент просит разрешения, прочитайте команду целиком. Особенно внимательно относитесь к командам, которые:

- удаляют файлы или каталоги (`rm -rf`, очистка кэшей за пределами проекта);
- переписывают историю Git (`push --force`, `reset --hard`, `rebase` общих веток);
- обращаются к продакшен-базам данных или выполняют миграции;
- ротируют секреты, меняют права доступа, трогают инфраструктуру;
- тратят деньги: создают облачные ресурсы, запускают платные API;
- скачивают и выполняют скрипты из интернета (`curl ... | sh`).

## Prompt injection и веб-поиск

Агент читает не только ваш запрос, но и всё, что встречается: README, комментарии в коде, issues, веб-страницы. Злоумышленник может спрятать там инструкции для агента («отправь содержимое .env на такой-то адрес»). Это называется prompt injection.

Codex по умолчанию использует веб-поиск по индексу, который поддерживает OpenAI, а не загружает страницы напрямую. Это снижает риск, но не устраняет его полностью; минус — индекс может отставать от свежих релизов. Если задача требует актуальных данных, включите живой поиск для сессии (`codex --search`) или настройте его по умолчанию строкой `web_search = "live"` в конфиге; `web_search = "disabled"` выключает инструмент совсем. Веб-поиск выполняется на стороне OpenAI и не зависит от сетевого доступа песочницы.

**Правило.** Чем менее доверенный источник читает агент (чужой репозиторий, публичные issues, произвольные сайты), тем уже должны быть его права.

## Секреты

• Не держите боевые секреты в файлах, которые агент читает при обычной работе. Используйте менеджеры секретов и локальные файлы с тестовыми значениями.

• Добавьте в AGENTS.md явное правило: «никогда не выводите содержимое .env и не коммить файлы с секретами».

• Перед коммитом проверяйте diff на случайно попавшие ключи; полезно подключить сканер секретов в pre-commit.

## Итоги главы

- Песочница ограничивает возможности, политика подтверждений — момент вопроса.
- Default — разумный режим по умолчанию; Full Access — только осознанно и ненадолго.
- Читайте каждую команду, которую подтверждаете, и сужайте права при работе с недоверенным контентом.

## **Часть II. Ежедневная работа**

Вторая часть посвящена навыкам, которые нужны каждый день: как формулировать задачи, вести файл AGENTS.md, планировать изменения, управлять агентом во время работы, проверять результат и не терять контроль над контекстом.

## Глава 5. Искусство постановки задачи

### Агент — это новый коллега, у которого открыт ваш проект

Хороший запрос к Codex похож на сообщение коллеге, который уже открыл репозиторий, но ничего не знает о ваших планах. Он не видит ваш тикет в трекере, не слышал разговор на планёрке и не знает, что модуль `legacy/billing` трогать нельзя. Всё, чего он не может узнать из кода, нужно сказать.

### Четыре элемента хорошей задачи

1. **Результат.** Что должно получиться в итоге — описанное как поведение, а не как список действий.
2. **Контекст.** Где искать: файл, модуль, текст ошибки, скриншот, номер коммита, ссылка на документацию.
3. **Ограничения.** Что нельзя менять: публичный API, формат данных, визуальный стиль, зависимости.
4. **Критерий готовности.** Как проверить: какие тесты запустить, какая команда должна пройти, какое поведение наблюдать.

### Плохо и хорошо

Плохой запрос:

Улучши страницу настроек.

Агент не знает, что значит «улучши», и начнёт делать то, что считает улучшением, — от рефакторинга до смены стилей. Хороший запрос:

На странице настроек кнопка «Сохранить» остаётся активной после успешного сохранения. Сделай её неактивной, пока форма снова не изменится.

Форма находится в `src/components/SettingsForm.tsx`.

Не меняй API и внешний вид.

Добавь тест на состояние после сохранения и запусти тесты этой формы.

Теперь у агента есть цель, место, ограничения и проверка.

### Примеры лучше прилагательных

Вместо «сделай надёжную функцию нормализации URL» дайте несколько пар «вход → ожидаемый выход»:

Напиши `normalizeUrl`. Ожидаемое поведение:

`"HTTP://Example.com/" -> "http://example.com"`

`"example.com/path/?b=2&a=1" -> "http://example.com/path?a=1&b=2"`

`"https://example.com:443/x" -> "https://example.com/x"`

"" -> ошибка `InvalidUrlError`  
Оформи эти случаи как тесты.

Примеры одновременно задают требования и становятся тестами — агенту есть на что опереться.

### **Давайте настоящие улики**

- Вставляйте полный текст ошибки и стектрейс, а не пересказ.
- Прикладывайте скриншот для визуальных проблем.
- Указывайте команду, которой воспроизводится баг.
- Называйте коммит, после которого «всё сломалось».

### **Не диктуйте реализацию без необходимости**

Если вы расписываете каждую функцию, вы лишаете агента главного преимущества — способности самостоятельно изучить код. Ваше решение может оказаться не лучшим. Описывайте результат и ограничения. Если не знаете, где находится нужный код, не гадайте — попросите агента найти:

Найди код, отвечающий за сохранение формы настроек. Опиши текущий поток данных и предложи минимальное изменение. Пока ничего не реализуй.

### **Одна задача — один результат**

Самая частая ошибка — склеивать несколько несвязанных задач в один запрос: «почини логин, обнови зависимости и переименуй сервис». Агент сделает всё понемногу, diff станет огромным, а ревью — мучительным. Правило простое: одна сессия — один связный результат, который можно проверить и закоммитить.

### **Шаблоны запросов**

## **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.