Claude Code по максимуму

Без терминала и команд.

Что такое Claude Code и где его использовать

Claude Desktop (вкладка Code)

Упрощённый вариант. Отсутствие терминала и файловой системы делает его идеальным для **простых задач** и быстрого прототипирования. Не требует сложной настройки.

VS Code + расширение Claude Code

Настоятельно рекомендую. Это полноценная среда разработки, где агент Claude Code **управляет файлами и директориями** напрямую. Вы общаетесь с ним как с чатом прямо внутри проекта, получая интерактивную помощь.

Терминал

Не рекомендую. Использование через терминал неудобно и часто отпугивает новичков, создавая ненужные барьеры в работе с AI. Лучше избегать, если есть другие опции.

> Ключевой момент: Claude Code — это не просто инструмент для написания кода. Это полноценный **агент**, который сам управляет файлами и директориями вашего проекта, а не просто генерирует код для ручного копирования.

Настройка VS Code + Claude Code — пошагово

01

Скачать VS Code

Получите бесплатную версию **vsCode.com** для вашей операционной системы.

02

Создать папку проекта

Создайте новую папку для вашего проекта, используя **только латинские буквы** в названии и во всем пути к ней.

03

Открыть папку в VS Code

В VS Code выберите «File» → «Open Folder...» (или «Open...» на macOS) и укажите только что созданную папку.

04

Установить расширение "Claude Code"

Перейдите в раздел расширений (значок квадратов слева или `Ctrl/Cmd+Shift+X`), найдите и установите "Claude Code".

05

Открыть Claude Code

Вызовите палитру команд (`Ctrl/Cmd+Shift+P`) и введите "Claude Open Tab", затем выберите соответствующую опцию.

06

Авторизоваться

Войдите в систему, используя те же учетные данные и подписку, что и для доступа к `Claude.ai`.

Важный совет: Названия папок и весь путь к проекту должны быть на английском языке — использование русских символов может привести к ошибкам в работе Claude Code.

Режимы работы Claude Code

Auto-edit (автоматические изменения)

Агент сразу изменяет файлы без подтверждения. Идеально для рутинных задач, где вы доверяете Claude Code.

Before-edits (с подтверждением)

Агент изучает проект, предлагает план изменений и ожидает вашего подтверждения. Отлично для контроля и сложных задач.

Plan mode (режим планирования)

Для больших и многоэтапных задач. Claude сначала составляет детальный план (до 10 страниц), а затем реализует его шаг за шагом.

Effort (сила раздумий)

Регулятор глубины мышления AI. Для простых задач можно снизить "усилие", для архитектурных решений или отладки сложных багов — ставьте на максимум.

Выбор модели

Claude Opus 4/5 — для всех задач

Рекомендуется при подписке Max. То, что Opus делает за 1 промпт, Sonnet потребует 3–4. Экономит время, а не токены.

Claude Sonnet — для простых задач

Подходит для мелких правок: поменять кнопку, исправить текст, небольшие изменения. Расходует меньше токенов.

Thinking mode — всегда включён

Режим глубокого мышления. Оставляйте включённым всегда, особенно при архитектурных задачах и создании нового функционала.

Вы можете открывать несколько параллельных сессий Claude Code в разных окнах, например, для аудита кода в одном и разработки новой фичи в другом.

Система двух мозгов — как правильно работать

Эффективность Claude Code раскрывается при правильном распределении задач между двумя "мозгами" — чатом Claude.ai для стратегического планирования и Claude Code для технической реализации.

Мозг 1: Claude.ai (обычный чат)

Используется для продумывания идей, формулирования вопросов, создания и уточнения промптов. Важно: **не имеет доступа к коду** вашего проекта.

Мозг 2: Claude Code в VS Code

Отвечает за непосредственную реализацию кода, управление файлами и директориями проекта. Получает чёткие технические задания (ТЗ) от "Мозга 1".

Представьте этот процесс как взаимодействие между командами: Вы — менеджер/CEO, Claude.ai — продакт-менеджер, а Claude Code — разработчик.

Как создать проект: от идеи до первого промпта

Доступ к улучшателю промптов

Начните с бесплатного инструмента, который превратит вашу сырую идею в структурированный запрос для AI.

Скилл улучшателя для Claude

Хаотичное описание идеи

Не беспокойтесь о формулировках. Выложите все свои мысли о проекте, даже если они кажутся сумбурными и неорганизованными.

Генерация начального промпта

Улучшатель проанализирует ваши записи и создаст первый, оптимизированный промпт для Claude.ai, готовый к диалогу.

Диалог с Claude.ai: уточняющие вопросы

Claude.ai задаст детализирующие вопросы по ключевым аспектам проекта: клиенты, задачи, дизайн, технические ограничения.

Ввод ответов (Голос/Текст)

Предоставьте исчерпывающие ответы на вопросы Claude.ai, используя голосовой ввод для скорости или текстовый для точности.

Системный дизайн и последовательные промпты

Claude.ai сформирует комплексный системный дизайн и сгенерирует набор последовательных промптов, адаптированных для Claude Code.

Реализация в Claude Code

Копируйте и вставляйте сгенерированные промпты один за другим в Claude Code, чтобы начать непосредственную разработку проекта.

Пример: **CRM для фрилансера** — создана за **1 час 32 минуты** с нуля, демонстрируя невероятную скорость работы с AI.

Совет: Для дизайна ищите вдохновение на **Dribbble** (по запросам 'CRM', 'task management' и т.д.). Рекомендуемый стек для большинства проектов: `Next.js + SQLite`.

Постановка задач в терминале

Попросите агента запустить сервер

Вместо того чтобы вводить команды вручную, просто напишите в чате: «запусти dev server». Claude Code сам выполнит нужную команду в терминале.

Исправление ошибок

Когда в Next.js появляются ошибки, скопируйте их из браузера и напишите: «исправь ошибки» + вставьте текст ошибки. Агент сам разберётся и починит.

Никаких команд вручную

За всё время разработки двух сервисов ни одна команда не была введена вручную. Агент знает все нужные команды и выполняет их самостоятельно.

Какой язык программирования выбрать?

Next.js — лучший выбор для большинства проектов

Понимание AI

Нейросети лучше всего разбираются в Next.js, что значительно снижает количество ошибок и ускоряет разработку.

Отличная масштабируемость

Проекты на Next.js легко масштабируются, и вам не придется переделывать их с нуля при росте требований.

Универсальность

Подходит как для личных пет-проектов, так и для крупных коммерческих сервисов с подпиской.

Проверенное решение

Next.js используется для всех проектов автора, включая Ulib.ai и Connectio, что подтверждает его надежность.

Выбор базы данных

- **SQLite:** Идеально подходит для локальной разработки и небольших проектов благодаря простоте и возможности работать без интернета.
- **PostgreSQL / другие:** Рекомендуются при масштабировании проекта и деплое на сервер для обеспечения высокой производительности и надежности.

Сравнение с альтернативами

Next.js: Надежность и перспектива

Хотя на старте Next.js может показаться более сложным, его стабильность и скорость разработки в долгосрочной перспективе окупаются.

- Снижение багов с помощью AI
- Готовность к росту и развитию

Другие фреймворки и языки: Потенциальные сложности

Python: Популярен и часто рекламируется, но для разработки веб-приложений Next.js обеспечивает более удобный и эффективный процесс.

Иные фреймворки: Могут быть использованы, но часто приводят к большим проблемам и необходимости доработок в будущем.

Важные файлы в проекте

Для эффективной работы с Claude Code ваш проект нуждается в нескольких ключевых файлах, которые определяют его поведение и безопасность.

.env

Этот файл хранит все секретные переменные вашего проекта, такие как API-ключи и токены. Агент Claude Code может создать его первую версию, но вы должны самостоятельно добавить свои конфиденциальные данные. **Никогда не допускайте попадания .env в публичный репозиторий!**

.gitignore

Файл .gitignore содержит список файлов и папок, которые система контроля версий (например, Git) должна игнорировать и не загружать в репозиторий. Всегда просите Claude Code проверить и дополнить этот файл для обеспечения максимальной безопасности.

CLAUDE.md

Это главный контекстный файл для вашего AI-агента. Он предоставляет Claude всю необходимую информацию для работы, включая:

- Правила дизайна (цвета, шрифты, UI-кит)
- Технический стек и архитектуру проекта
- Инструкции по запуску dev-сервера
- Четкие указания "Что делать" и "Чего не делать"

Рекомендуемый размер этого файла — до 300 строк.

> **Ключевое правило:** Никогда не редактируйте эти файлы вручную — всегда используйте для этого AI-агента, чтобы избежать ошибок и несоответствий.

Документация проекта

Проектная документация — это краеугольный камень любого успешного продукта, обеспечивающий его развитие и поддержку.

Почему документация критически важна?

- Позволяет AI-агенту постоянно быть в контексте и понимать актуальное состояние всех частей проекта.
- Является незаменимым ресурсом при масштабировании команды и онбординге новых разработчиков, сокращая время на погружение.
- Централизованное хранение, например, в папке `/docs` (как `project.md`), упрощает доступ и управление информацией.

Процесс создания и поддержки

- Используйте Claude.ai для генерации начального промпта, который поможет структурировать основные разделы документации.
- Включите в файл `CLAUDE.md` правило: AI-агент должен автоматически обновлять документацию после любых изменений в коде или архитектуре.
- Не беспокойтесь о размере: объем документации в 1000+ строк — это нормальная практика для детализированных проектов.

Что должно быть включено в документацию?

- Высокоуровневая архитектура системы и ее компонентов.
- Подробное описание моделей данных, схем баз данных и связей между ними.
- Функциональное описание всех ключевых модулей и их взаимодействия.
- Список принятых технических решений, ограничений и исключений.

> 💡 **Совет:** Скопируйте готовую документацию в любую нейросеть — она мгновенно усвоит весь проектный контекст, что значительно упростит дальнейшее взаимодействие и запросы.

GitHub и деплой проекта

GitHub — это не только хранилище кода, но и критически важный инструмент для управления версиями и автоматизации развертывания ваших приложений.

Контроль версий

Возможность отслеживать все изменения и откатываться на любую предыдущую версию проекта.

Платформа для деплоя

Служит ключевым связующим звеном между вашей локальной разработкой и продакшен-серверами.

Безопасное хранение

Предоставляет бесплатные приватные репозитории для безопасного хранения всего вашего кода.

Процесс деплоя (Workflow)

1

Dev-сервер

Ваша локальная среда разработки.

2

GitHub (Push)

Загрузка изменений в репозиторий.

3

Сервер (Деплой)

Развертывание приложения для пользователей.

Сервисы для деплоя

Railway

- **Стоимость:** ~$20/мес.
- **Преимущества:** Крайне удобен и выгоден, позволяет размещать как приложение, так и базу данных.
- **Требование:** Необходима иностранная банковская карта.

TimeWeb

- **Назначение:** Оптимален для проектов, ориентированных на российских клиентов.
- **Преимущества:** Соответствует законодательству РФ, предоставляет российский хостинг.
- **Тип:** Российский облачный сервис.

Важно: Перед финальным развертыванием всегда просите Claude Code **оптимизировать проект для продакшена** — это повысит производительность и безопасность вашего приложения.

Для доступа к большинству современных зарубежных сервисов и комфортного **«вайб-кодинга»** рекомендуется иметь иностранную карту (стоимость получения ~10 000 руб.).

Подключение GitHub к проекту — пошаговая инструкция

01

Установи Git и GitHub CLI

```
# Проверь что git установлен
git --version

# Установи GitHub CLI (Mac)
brew install gh

# Установи GitHub CLI (Windows)
winget install GitHub.cli

# Установи GitHub CLI (Ubuntu/Debian)
sudo apt install gh
```

02

Авторизуйся в GitHub

```
gh auth login
```

Выбери: GitHub.com → HTTPS → Yes → Login with a web browser

03

Инициализируй Git в проекте

```
cd /путь/к/твоему/проекту

git init
git add .
git commit -m "first commit"
```

04

Создай репозиторий и подключи

```
# Создаёт реп на GitHub и сразу подключает — одна команда
gh repo create название-проекта --public --source=. --remote=origin --push
```

- --public — публичный репозиторий (замени на --private если нужно)
- --source=. — текущая папка
- --push — сразу пушит код

05

Проверь

```
git remote -v
# Должно показать: origin https://github.com/твой-юзер/название-проекта
```

Если репозиторий уже существует на GitHub

```
git remote add origin https://github.com/твой-юзер/название-проекта.git
git branch -M main
git push -u origin main
```

Дальнейшая работа

```
git add .
git commit -m "описание изменений"
git push
```

Промпт для создания CLAUDE.md

CLAUDE.md — это специальный файл, который Claude Code читает автоматически при каждом запуске. Это даёт ИИ полный контекст о проекте без повторных объяснений. **Промпт лучше прогнать через улучшатель.**

Проанализируй весь проект и создай файл CLAUDE.md в корне проекта.

Включи в файл:

1. Обзор проекта
- Название, назначение, что делает приложение
- Основной стек технологий

2. Архитектура
- Структура папок с описанием каждой директории
- Основные модули и их назначение

3. Команды разработки
- Установка зависимостей
- Запуск в dev режиме
- Сборка для продакшена
- Запуск тестов
- Линтинг

4. Переменные окружения
- Список всех .env переменных с описанием
- Пример .env.example

5. API / Основные модули
- Ключевые endpoints или функции
- Как они работают

6. База данных (если есть)
- Схема / модели
- Как запустить миграции

7. Важные решения и соглашения
- Код-стайл и соглашения принятые в проекте
- Что нельзя менять и почему
- Известные проблемы и ограничения

8. Как контрибьютить
- Workflow работы с ветками
- Как делать коммиты

Пиши кратко и по делу. Файл должен помочь
новому разработчику или AI быстро войти в проект.

Совет: после создания файла прочитай его и попроси добавить всё что ИИ мог пропустить — бизнес-логику, нюансы архитектуры, договорённости команды.

Подписки и токены — что выбрать?

Оптимизируйте работу с Claude.ai, выбрав подходящую подписку и модель, чтобы максимально эффективно использовать свои ресурсы и время.

Подписки Claude.ai для "Вайб-кодинга"

Базовая ($20/мес)

Токены быстро заканчиваются при активной работе. Подходит для эпизодического использования или небольших задач.

Расширенная ($100/мес)

Лучше, но при интенсивной работе токены могут закончиться. Хороший выбор для умеренной разработки.

Максимальная ($200/мес)

Рекомендуется для профессиональной разработки. При 8-9 часах кодинга в день лимиты заканчивались лишь однажды за 3 месяца.

Рекомендации по выбору модели

- **Claude Opus (4/5):** Используйте для всех задач, если позволяет подписка. То, что Opus делает за 1 промпт, Sonnet часто требует 3-4.
- **Claude Sonnet:** Экономный выбор для простых задач, помогает сберечь токены.
- **Режим "Thinking":** Всегда держите его включенным для более глубокого анализа и качественных результатов.

> **Экономика:** Инвестиция в $200/мес в Claude Max против 300 000 руб/мес на команду разработчиков обеспечивает колоссальную экономию в 10 раз при значительном повышении продуктивности.

Промт-исследование для Claude:

Prompt for Claude — Personal CRM & Task Manager Design (Solo Freelancer, Local Dev Setup)

1) Role & Objective

You are a senior product manager, UX designer, and systems architect combined.Your task is to help me design a personal CRM + task management system tailored specifically for a solo freelance marketer working across multiple niches.

Your goal is NOT to jump into building immediately.Instead, you must:

Conduct a structured discovery interview with me

Fully understand my workflows, preferences, and constraints

Then design:

Ideal UX/UI

System architecture (optimized for local/dev use first)

Data structure

Feature set (lean but powerful)

You are proactive, structured, and think in systems.

2) Context

I am a freelance marketer

I work with multiple clients across different niches

I need:

CRM (clients, deals, interactions)

Task manager (simple but effective)

This system is for personal use only (not SaaS)

I will initially run it:

locally (dev server, possibly on my home machine)

WITHOUT deployment to external infrastructure (at least at first)

UI must be:

clean

modern

visually pleasing

fast and not overloaded

3) Your Process (STRICT)

Phase 1 — Deep Discovery (MANDATORY)

You MUST begin by asking me questions.DO NOT propose solutions yet.

Ask questions in logical blocks:

Block A — Work & Clients

What types of clients do you work with?

How many active clients at a time?

Do you manage leads or only current clients?

What stages does a client go through? (lead → deal → work → retention, etc.)

Do you need revenue tracking?

Block B — Tasks & Workflow

How do you currently manage tasks?

Types of tasks (one-time, recurring, campaigns, etc.)

Do tasks belong to:

clients?

projects?

both?

Do you need deadlines, priorities, reminders?

Block C — Daily Usage

How often will you open the system?

What should the main dashboard show?

What actions should be 1-click?

Block D — CRM Logic

What info do you store per client?

Do you track:

communication history?

notes?

files?

Do you need tagging or categorization?

Block E — UI/UX Preferences

Minimalist vs detailed?

Dark/light mode?

Inspirations (Notion, Linear, etc.)

Mobile usage or desktop-only?

Block F — Technical Constraints

Preferred stack (if any)

Are you comfortable with:

local databases (SQLite, etc.)

Docker?

Do you want offline-first behavior?

Block G — Future Vision

Will this stay personal or grow?

Any automation ideas?

Integrations (Telegram, email, etc.)