ДОГОВОРИСЬ С ИИ

ЧАСТЬ 2

РЕАЛЬНЫЕ ПРОЕКТЫ, ПРОФЕССИОНАЛЬНЫЕ ПРАКТИКИ И ПРОДВИНУТЬЕ ВОЗМОЖНОСТИ CLAUDE CODE

Даниил Глушко

Договорись с ИИ. Часть 2 Реальные проекты, профессиональные практики и продвинутые возможности Claude Code

«Автор»
2026

Глушко Д. А.

Договорись с ИИ. Часть 2 Реальные проекты, профессиональные практики и продвинутые возможности Claude Code /
Д. А. Глушко — «Автор», 2026

Первая книга «Договорись с ИИ» научила разговаривать с Claude Code и создавать сайты. Эта учит доводить дело до конца.Между «сделал сайт для себя» и «им пользуются люди» — пропасть вопросов: пароли, оплата без риска, сбои сервера.На трёх проектах — магазине, боте и системе учёта — путь от идеи до публикации: тесты, деплой, базы данных, платежи, CI/CD. Плюс возможности Claude Code: CLAUDE.md, хуки, субагенты, MCP.В финале — три провальных сценария для учёбы на чужих ошибках.Первая книга научила создавать сайт. Эта — защищать его и доводить до пользователей.

Содержание

Оглавление 7
Предисловие 8
Что изменилось в подходе этой книги 9
Насколько глубоко мы пойдём 10
Часть 1. От новичка к уверенному пользователю 11
Глава 1. Что вы уже умеете и куда двигаться дальше 11
Короткая инвентаризация навыков 11
Что изменится в этой книге 11
Как выбрать свой маршрут по книге 11
Небольшое резюме главы 12
Глава 2. Как устроена эта книга и как с ней работать 13
Сквозные учебные проекты 13
Структура каждой главы 13
Уровень технических подробностей 13
Небольшое резюме главы 13
Глава 3. Расширяем словарь: термины, которые понадобятся дальше 15
Часть 2. Реальные проекты от и до 17
Глава 4. Интернет-магазин: от каталога до оформления заказа 17
Знакомство с проектом «ВелоМаркет» 17
Шаг 1. Планируем структуру заранее 17
Шаг 2. Каталог и карточка товара 17
Шаг 3. Корзина 18
Шаг 4. Оформление заказа 18
Шаг 5. Проверяем весь путь целиком 19
Шаг 6. Административная часть (кратко) 19
Небольшое резюме главы 19
Глава 5. Telegram-бот: от идеи до первого пользователя 20
Чем бот принципиально отличается от сайта 20
Шаг 1. Регистрация бота у BotFather 20
Шаг 2. Формулируем задачу для Claude Code 20
Шаг 3. Запускаем бота локально и тестируем 21
Шаг 4. Логика напоминаний — что происходит «под капотом» 21
Шаг 5. От локального запуска — к боту, который работает всегда 21
Расширяем функциональность бота 22
Небольшое резюме главы 22
Глава 6. Личный кабинет с авторизацией и базой данных 23
Зачем нужен личный кабинет 23
Шаг 1. Формулируем задачу с учётом ролей 23
Шаг 2. Как устроена регистрация и вход технически 23
Шаг 3. Структура данных 24
Шаг 4. Реализация авторизации 24
Шаг 5. Разграничение доступа 24

Шаг 6. Восстановление пароля и другие практические мелочи 25
Небольшое резюме главы 25
Глава 7. Автоматизация для небольшого бизнеса: кейс от заявки до отчёта 26
От разрозненных задач — к цельному процессу 26
Сценарий: от заявки клиента до ежемесячного отчёта 26
Шаг 1. Автоматическое распределение заявок 26
Шаг 2. Уведомления на ключевых этапах 26
Шаг 3. Автоматическое формирование отчёта 27
Шаг 4. Взгляд на процесс целиком 27
Пределы автоматизации: когда стоит остановиться 27
Небольшое резюме главы 28
Часть 3. Профессиональные практики 29
Глава 8. Тестирование: как убедиться, что всё работает, не проверяя вручную 29
Проблема ручной проверки 29
Конец ознакомительного фрагмента. 30

Даниил Глушко
Договорись с ИИ. Часть 2 Реальные проекты, профессиональные практики и продвинутые возможности Claude Code

Договорись с ИИ. Часть 2

Реальные проекты, профессиональные практики и продвинутые возможности Claude Code

2026

Оглавление

Часть 1. От новичка к уверенному пользователю
Глава 1. Что вы уже умеете и куда двигаться дальше
Глава 2. Как устроена эта книга и как с ней работать
Глава 3. Расширяем словарь: термины, которые понадобятся дальше
Часть 2. Реальные проекты от и до
Глава 4. Интернет-магазин: от каталога до оформления заказа
Глава 5. Telegram-бот: от идеи до первого пользователя
Глава 6. Личный кабинет с авторизацией и базой данных
Глава 7. Автоматизация для небольшого бизнеса: кейс от заявки до отчёта
Часть 3. Профессиональные практики
Глава 8. Тестирование: как убедиться, что всё работает, не проверяя вручную
Глава 9. Публикация в интернете: как разместить сайт или приложение так, чтобы его видели другие
Глава 10. Git глубже: ветки, слияния, откаты, совместная работа
Глава 11. Работа в команде: как совместно с другими людьми использовать ИИ в одном проекте
Глава 12. Основы CI/CD: автоматическая проверка и публикация изменений
Часть 4. Продвинутые возможности Claude Code
Глава 13. Файл CLAUDE.md: тонкая настройка поведения ИИ под ваш проект
Глава 14. Кастомные команды и повторяемые сценарии
Глава 15. Хуки (hooks): как автоматизировать реакции на действия ИИ
Глава 16. Субагенты: когда одна задача — это на самом деле несколько
Глава 17. MCP: как подключить Claude Code к внешним сервисам
Глава 18. Работа в фоновом и «безголовом» режиме, скрипты и автоматизация
Часть 5. Данные, интеграции и безопасность
Глава 19. Выбор базы данных под задачу: обзор вариантов простыми словами
Глава 20. Аутентификация и работа с пользователями
Глава 21. Приём платежей: как это устроено и на что обратить внимание
Глава 22. Работа с внешними API на практике: три примера
Глава 23. Безопасность данных: чек-лист для непрофессионала
Часть 6. Качество и рост проекта
Глава 24. Рефакторинг: как улучшать код, не ломая рабочую версию
Глава 25. Документация проекта: зачем она нужна, даже если вы работаете один
Глава 26. Производительность: когда сайт или приложение начинает тормозить
Глава 27. Code review глазами ИИ: как попросить критику собственного проекта
Часть 7. Кейсы и что дальше
Глава 28. Разбор трёх реальных провальных сценариев и как их избежать
Глава 29. Как читать техническую документацию с помощью ИИ
Глава 30. Куда двигаться после этой книги: сообщества, ресурсы, следующие шаги

Предисловие

Если вы читаете эту книгу, значит, первая часть — «Договорись с ИИ» — сделала своё дело: вы уже установили Claude Code, создали свой первый сайт, разобрались с ошибками, git и базовым промптингом. Возможно, вы уже сделали что-то своё, не из примеров книги — сайт для хобби, небольшой трекер, что-то для работы.

Эта книга — не повторение пройденного, а следующий шаг. Мы больше не будем объяснять, что такое терминал или зачем нужен git — это уже позади. Вместо этого мы возьмёмся за то, с чем сталкивается каждый, кто продолжает работать над реальными проектами: как довести проект от идеи до состояния, которым реально пользуются другие люди; как не бояться более сложных технических тем — тестирования, деплоя, баз данных, платежей; и как использовать более продвинутые возможности самого Claude Code, которые в первой книге мы сознательно оставили за кадром, чтобы не перегружать новичка.

Что изменилось в подходе этой книги

Первая книга была построена вокруг одной сквозной метафоры — превращения полного новичка в человека, способного создать простой сайт. Эта книга устроена иначе: это скорее сборник практических маршрутов. Каждая часть посвящена отдельной большой теме — реальным проектам, профессиональным практикам, продвинутым возможностям инструмента, работе с данными, качеству кода — и внутри каждой темы мы идём от объяснения к практике с конкретными примерами команд и кода.

Это значит, что книгу необязательно читать строго по порядку от первой до последней страницы. Если вас в первую очередь интересует, как разместить сайт в интернете — можно сразу перейти к главе 9. Если хочется разобраться с Telegram-ботами — глава 5 ждёт вас. Тем не менее, для первого чтения мы рекомендуем идти по порядку: главы построены так, что более поздние иногда опираются на понятия, введённые раньше.

Насколько глубоко мы пойдём

Философия книги остаётся прежней: мы не готовим вас к карьере профессионального программиста и не будем заставлять запоминать синтаксис языков программирования. Но мы пойдём заметно глубже, чем в первой части, — в частности, потому что на данном этапе вы уже умеете формулировать задачи и не боитесь диалога с ИИ, а значит, готовы понимать смысл более сложных тем даже без намерения самостоятельно писать код.

В каждой главе, где это уместно, вы найдёте разобранные примеры — не абстрактные, а привязанные к сквозным учебным проектам (интернет-магазин, Telegram-бот, личный кабинет), которые мы будем развивать на протяжении нескольких глав подряд, как это часто происходит и в реальной работе.

Итак, продолжаем путь.

Часть 1. От новичка к уверенному пользователю

Глава 1. Что вы уже умеете и куда двигаться дальше

Короткая инвентаризация навыков

Прежде чем двигаться вперёд, полезно честно зафиксировать, что у вас уже есть в активе после первой книги. Скорее всего, это:

• умение установить и запустить Claude Code, ориентироваться в его текстовом интерфейсе;

• навык формулировать задачи достаточно подробно, чтобы получать предсказуемый результат;

• понимание базового цикла работы: задача → результат → проверка → уточнение;

• знакомство с git на уровне «сделать коммит» и «откатить изменения»;

• умение читать и передавать модели сообщения об ошибках;

• общее представление о том, что такое база данных и API, пусть пока без практического опыта.

Если что-то из этого списка вызывает сомнение — не страшно, вернуться к соответствующей главе первой книги можно в любой момент, эта книга не проверяет вас на входе строгим экзаменом.

Что изменится в этой книге

Три вещи будут отличать материал этой книги от предыдущей.

Во-первых, масштаб задач. Вместо сайта-визитки из одной страницы мы будем строить проекты с несколькими связанными частями — например, интернет-магазин, где нужно согласованно работать с каталогом товаров, корзиной, оформлением заказа и уведомлениями.

Во-вторых, ответственность. Проект, которым реально пользуются другие люди, требует внимания к вещам, которые можно было игнорировать в учебном проекте для себя: что произойдёт, если два человека одновременно нажмут кнопку заказа; что если сервер временно недоступен; как защитить личные данные пользователей.

В-третьих, инструментарий. Мы познакомимся с более продвинутыми возможностями самого Claude Code — субагентами, хуками, MCP-подключениями, — которые в первой книге были бы преждевременны, но теперь, когда основы усвоены, органично расширяют ваши возможности.

Как выбрать свой маршрут по книге

Если у вас уже есть конкретная цель — скажем, вы хотите сделать интернет-магазин для небольшого семейного бизнеса, — имеет смысл читать части 2 и 5 (реальные проекты и данные/интеграции) внимательнее, а часть 4 (продвинутые возможности) можно пролистать по диагонали и вернуться к ней позже, когда возникнет конкретная потребность.

Если вам интереснее сам инструмент и его возможности — тогда часть 4 станет для вас основной, а сквозные проекты из части 2 можно использовать просто как учебный полигон для практики.

СОВЕТ: Не пытайтесь усвоить книгу за один присест. Каждая глава рассчитана на то, чтобы её можно было прочитать, попробовать на практике за один-два присеста, и вернуться к следующей уже после того, как знания улеглись на практическом опыте.

Небольшое резюме главы

• К этому моменту у вас уже есть работающая база навыков: установка, промптинг, git, работа с ошибками.

• Эта книга увеличивает масштаб задач, добавляет ответственность за качество и знакомит с продвинутыми возможностями инструмента.

• Книгу можно читать не строго по порядку, ориентируясь на собственные практические цели.

В следующей главе — краткая карта самой книги и рекомендации, как с ней работать эффективнее всего.

Глава 2. Как устроена эта книга и как с ней работать

Сквозные учебные проекты

На протяжении книги мы будем регулярно возвращаться к трём сквозным проектам, каждый из которых развивается от главы к главе:

«ВелоМаркет» — небольшой интернет-магазин запчастей для велосипедов. На нём мы разберём каталог товаров, корзину, оформление заказа, платежи, личный кабинет покупателя и администратора.

«Напоминалка» — Telegram-бот для личных напоминаний и небольших списков дел. На нём мы разберём работу с внешними API мессенджеров, хранение состояния пользователя, простую логику диалога.

«Учёт клиента» — небольшая внутренняя система для условной мастерской по ремонту техники: список клиентов, заявок, простая отчётность. На этом проекте удобно разбирать темы личного кабинета, ролей пользователей и автоматизации отчётов.

Не обязательно повторять именно эти три проекта дословно — гораздо полезнее выбрать для себя один реальный проект, важный лично для вас, и по ходу чтения применять описываемые техники именно к нему, ориентируясь на примеры книги как на шаблон подхода, а не точную инструкцию.

Структура каждой главы

Как и в первой книге, главы построены по единому принципу: объяснение понятия простыми словами → пример подробной формулировки задачи для Claude Code → разбор того, что происходит и почему → практические рекомендации и предостережения → краткое резюме.

В этой книге добавляется новый повторяющийся элемент — врезки «Разбор кейса», где мы подробно показываем реальный диалог с ИИ от формулировки задачи до финального результата, включая типичные повороты сюжета вроде неожиданных ошибок и уточняющих вопросов.

Уровень технических подробностей

Важная оговорка: там, где в главе приводится код (например, фрагмент JavaScript или структура таблицы базы данных), цель — не в том, чтобы вы вручную вводили и отлаживали этот код, а в том, чтобы вы понимали, что примерно происходит внутри, когда Claude Code выполняет вашу задачу, и могли осмысленно читать и оценивать предлагаемые решения. Реальная работа по написанию кода, как и в первой книге, остаётся за ИИ — ваша роль — постановка задачи, проверка результата и принятие решений.

Небольшое резюме главы

• Книга построена вокруг трёх сквозных учебных проектов: интернет-магазин, Telegram-бот и внутренняя система учёта.

• Рекомендуется параллельно применять изучаемые техники к собственному реальному проекту.

• Новый элемент этой книги — врезки «Разбор кейса» с подробным диалогом от задачи до результата.

• Приводимый код нужен для понимания происходящего, а не для ручного набора и отладки.

В следующей главе — расширенный словарь терминов, которые встретятся на страницах этой книги и которых не было в первой части.

Глава 3. Расширяем словарь: термины, которые понадобятся дальше

Прежде чем погружаться в практику, полезно бегло познакомиться с несколькими понятиями, которые часто будут встречаться в последующих главах. Не нужно запоминать эти определения наизусть — достаточно один раз прочитать, чтобы позже, встретив термин в контексте, не спотыкаться о него.

Сервер — компьютер (обычно удалённый, расположенный в дата-центре), который постоянно включён, подключён к интернету и обрабатывает запросы от пользователей вашего сайта или приложения. В первой книге мы работали в основном с файлами на собственном компьютере — теперь речь пойдёт о том, как код начинает жить на сервере, доступном всем в интернете.

Бэкенд и фронтенд — условное разделение частей приложения. Фронтенд — то, что видит и с чем напрямую взаимодействует пользователь (страницы, кнопки, формы). Бэкенд — невидимая пользователю часть, работающая на сервере: обработка данных, взаимодействие с базой данных, бизнес-логика (например, расчёт скидки или проверка остатков на складе).

Деплой (развёртывание) — процесс размещения готового проекта на сервере так, чтобы им могли пользоваться реальные люди через интернет, а не только вы на своём компьютере.

Окружение (environment) — набор условий, в которых работает программа: локальное окружение (ваш компьютер), тестовое окружение (копия проекта для проверок) и продакшен (боевая версия, которой пользуются реальные люди).

Переменные окружения — способ хранить настройки и секретные данные (например, ключи доступа к внешним сервисам) отдельно от основного кода, чтобы их не пришлось «зашивать» прямо в файлы проекта.

Аутентификация и авторизация — два близких, но разных понятия. Аутентификация — процесс подтверждения личности («вы действительно тот, за кого себя выдаёте»), обычно через логин и пароль. Авторизация — определение того, какие действия разрешены уже подтверждённому пользователю (например, обычный покупатель не может зайти в панель администратора магазина).

Тестирование — процесс автоматической или ручной проверки того, что программа работает правильно, в том числе после внесения новых изменений.

CI/CD (непрерывная интеграция и доставка) — практика, при которой изменения в проекте автоматически проверяются и разворачиваются на сервере без ручного повторения одних и тех же шагов каждый раз.

Рефакторинг — улучшение внутреннего устройства кода без изменения его внешнего поведения: код продолжает работать точно так же, но становится понятнее, надёжнее или быстрее.

API-эндпоинт — конкретный «адрес» внутри API, отвечающий за одно определённое действие (например, «получить список товаров» или «оформить заказ»).

Токен доступа — временный или постоянный секретный код, подтверждающий право пользователя или программы на выполнение определённых действий, часто используется вместо постоянного ввода пароля.

Хук (hook) — механизм, который автоматически запускает определённое действие в ответ на какое-либо событие (например, автоматическая проверка кода перед каждым коммитом).

Субагент — вспомогательный, специализированный процесс внутри Claude Code, которому основной диалог может делегировать отдельную часть большой задачи.

MCP (Model Context Protocol) — протокол, позволяющий Claude Code подключаться к внешним сервисам и инструментам (например, к таск-трекеру или базе данных) стандартизированным способом.

СОВЕТ: Если по ходу дальнейшего чтения какой-то термин из этого списка забудется — не нужно листать книгу назад в поисках этой главы. Проще всего в любой момент спросить у самого Claude Code: «Напомни простыми словами, что такое [термин]» — и получить объяснение, адаптированное именно под контекст вашего текущего проекта.

Вооружившись этим словарём, переходим к главному — реальным проектам от идеи до результата.

Часть 2. Реальные проекты от и до

Глава 4. Интернет-магазин: от каталога до оформления заказа

Знакомство с проектом «ВелоМаркет»

В этой главе мы пройдём путь создания небольшого интернет-магазина полностью — от первой страницы каталога до момента, когда покупатель может оформить заказ. Это самая длинная практическая глава книги, и это не случайно: интернет-магазин — один из самых частых запросов к ИИ-инструментам, и на его примере удобно показать, как согласованно работать сразу с несколькими связанными частями проекта.

Шаг 1. Планируем структуру заранее

В отличие от простого сайта-визитки, для проекта с несколькими связанными разделами полезно начать не с прямого запроса «сделай магазин», а с просьбы о плане:

Хочу сделать небольшой интернет-магазин запчастей для

велосипедов "ВелоМаркет". Нужны: каталог товаров с

категориями (тормоза, цепи, колёса, аксессуары), карточка

товара, корзина, оформление заказа. Прежде чем начинать,

предложи структуру проекта — какие страницы и файлы

понадобятся и как они будут связаны между собой

Такой запрос экономит время в дальнейшем: вы получаете общую карту проекта, можете скорректировать её на раннем этапе, прежде чем будет написана хоть одна строка кода.

Шаг 2. Каталог и карточка товара

Начинаем с самого фундамента — списка товаров:

Создай страницу каталога товаров. Пока используй тестовые

данные прямо в коде — 12 товаров в 4 категориях, у каждого

название, цена, короткое описание и заглушка вместо фото.

Добавь фильтр по категориям и сортировку по цене

После получения результата — переходим к карточке отдельного товара:

Сделай, чтобы при клике на товар в каталоге открывалась

отдельная карточка товара с более подробным описанием,

крупным изображением-заглушкой и кнопкой "Добавить в корзину"

РАЗБОР КЕЙСА. На этом этапе частая заминка — Claude Code может создать переход на карточку товара, который “теряет” информацию о том, какой именно товар был выбран (например, всегда показывает первый товар из списка независимо от клика). Правильная формулировка для исправления: «При клике на разные товары в каталоге всегда открывается карточка первого товара вместо выбранного. Похоже, не передаётся идентификатор товара при переходе — проверь эту логику». Такая точная формулировка, ссылающаяся на конкретное наблюдаемое поведение, обычно приводит к быстрому и точному исправлению.

Шаг 3. Корзина

Корзина — это первое место в проекте, где нужно согласованное поведение сразу нескольких страниц: добавление товара на странице каталога должно отражаться в отдельном разделе корзины.

Добавь функциональность корзины:

1. Кнопка "Добавить в корзину" сохраняет товар в корзину

2. Иконка корзины в шапке сайта показывает количество товаров

3. Отдельная страница корзины со списком добавленных товаров,

количеством каждого, возможностью изменить количество или

удалить товар, и итоговой суммой

4. Данные корзины должны сохраняться при обновлении страницы

Обратите внимание на последний пункт — мы уже сталкивались с похожим требованием в первой книге (глава 15, приложение-трекер привычек). Механизм тот же самый — сохранение данных в браузере, — но здесь он становится частью более крупной, взаимосвязанной системы.

Шаг 4. Оформление заказа

Добавь страницу оформления заказа, доступную из корзины по

кнопке "Оформить заказ". Нужна форма: имя, телефон, адрес

доставки, способ оплаты (пока просто выбор "Наличными при

получении" или "Картой онлайн" — саму оплату подключим позже).

После отправки формы — показать страницу подтверждения

с номером заказа и очистить корзину

Намеренно откладываем реальную обработку платежей — эта тема достаточно объёмна и весома, чтобы посвятить ей отдельную главу (глава 21). На данном этапе цель — собрать

весь путь покупателя от каталога до подтверждения заказа как цельный, логически замкнутый сценарий.

Шаг 5. Проверяем весь путь целиком

После того как отдельные части готовы, важно проверить не каждую по отдельности, а весь путь пользователя целиком, от начала до конца, как это сделал бы реальный покупатель:

Открыть каталог, отфильтровать по категории.

Открыть карточку конкретного товара.

Добавить его в корзину, вернуться в каталог, добавить ещё один товар.

Перейти в корзину, изменить количество одного из товаров.

Оформить заказ, проверить, что показывается корректное подтверждение.

Обновить страницу и убедиться, что корзина после оформления заказа действительно пуста.

ВНИМАНИЕ: Тестирование одного пути целиком, от начала до конца, — гораздо более надёжный способ найти реальные проблемы, чем проверка каждого отдельного элемента изолированно. Мы вернёмся к этой мысли подробнее в главе 8, посвящённой тестированию.

Шаг 6. Административная часть (кратко)

Реальному интернет-магазину, помимо интерфейса покупателя, нужна и возможность для владельца управлять товарами — добавлять, редактировать, убирать из продажи. Мы подробнее разберём тему разграничения ролей (покупатель / администратор) в главе 20, посвящённой аутентификации, но уже сейчас можно поставить простую задачу:

Добавь простую административную страницу /admin со списком

товаров и возможностью добавить новый товар через форму

(название, цена, категория, описание). Пока не нужна защита

паролем — просто отдельная страница для управления каталогом,

безопасность добавим позже

Небольшое резюме главы

• Крупный проект стоит начинать с запроса общего плана структуры, а не сразу с реализации.

• Работу удобно вести последовательными, логически завершёнными этапами: каталог → карточка товара → корзина → оформление заказа.

• Корзина — первый элемент, требующий согласованного поведения нескольких частей сайта одновременно.

• Тестировать нужно не только отдельные элементы, но и весь путь пользователя целиком.

• Административная часть и полноценная защита паролем — отдельные темы, которые мы развернём в последующих главах.

В следующей главе мы возьмёмся за принципиально другой тип проекта — Telegram-бота, который живёт не на веб-странице, а внутри мессенджера.

Глава 5. Telegram-бот: от идеи до первого пользователя

Чем бот принципиально отличается от сайта

До сих пор все наши проекты были веб-страницами, которые пользователь открывает в браузере. Бот устроен иначе: он живёт внутри мессенджера (в нашем примере — Telegram) и общается с пользователем через отправку и получение сообщений, а не через клики по элементам страницы. Идея разговора с ИИ на естественном языке, которую мы применяли для постановки задач Claude Code, здесь оказывается вдвойне уместной: сам бот, которого мы создаём, тоже будет «разговаривать» с конечным пользователем.

Шаг 1. Регистрация бота у BotFather

Прежде чем писать какой-либо код, нужно зарегистрировать самого бота в Telegram — это делается через специального служебного бота с именем @BotFather.

Откройте Telegram, найдите в поиске @BotFather.

Отправьте команду /newbot.

Следуйте подсказкам: укажите отображаемое имя бота и его техническое имя пользователя (должно заканчиваться на bot, например napominalka_test_bot).

В ответ BotFather выдаст токен — длинную строку символов вида 123456789:AAHdqTcvCH1vGWJxfSeofSAs0K5PALDsaw.

ВНИМАНИЕ: Этот токен — секретный ключ доступа к вашему боту, по сути пароль. Никому не показывайте его и не публикуйте в открытом доступе. Если токен всё же случайно попал в публичное место — в том же диалоге с BotFather есть команда /revoke, которая аннулирует старый токен и выдаёт новый.

Шаг 2. Формулируем задачу для Claude Code

Хочу создать простого Telegram-бота "Напоминалка" для личных

напоминаний. Функциональность:

1. Команда /start — приветствие и краткое объяснение, что

умеет бот

2. Команда /remind [текст] [время] — создать напоминание,

например "/remind Позвонить маме через 2 часа"

3. В указанное время бот присылает пользователю сообщение

с текстом напоминания

4. Команда /list — показать все активные напоминания

пользователя

5. Возможность удалить напоминание из списка

Токен бота: [вставьте сюда ваш токен]

Claude Code самостоятельно выберет подходящий технический подход для реализации бота (обычно используется готовая библиотека для работы с Telegram API, о котором мы говорили в первой книге, глава 16) и создаст структуру проекта.

СОВЕТ: Если не хотите вставлять токен прямо в текст сообщения (например, если сессия синхронизируется в облаке или её видит кто-то ещё), попросите: «Создай файл для хранения токена отдельно от основного кода, а токен я впишу туда сам» — это относится к теме переменных окружения, о которой шла речь в главе 3, и мы вернёмся к ней подробнее в главе 21.

Шаг 3. Запускаем бота локально и тестируем

После создания кода Claude Code, скорее всего, предложит запустить бота прямо на вашем компьютере командой вроде:

python bot.py

Пока эта команда выполняется (и терминал остаётся «занятым» — то есть строка ввода не появляется снова, потому что программа продолжает работать и ждать сообщений), откройте Telegram, найдите своего бота по имени пользователя и отправьте /start.

РАЗБОР КЕЙСА. Частый первый опыт новичка с ботом: команда запущена, но бот не отвечает. Прежде чем паниковать, проверьте простое: не закрылось ли случайно окно терминала (тогда программа бота остановилась), и действительно ли вы написали боту в правильном чате. Если проблема не в этом — самое время вернуться к главе 12 первой книги: скопируйте всё, что terminal вывел после отправки сообщения боту, и покажите Claude Code.

Шаг 4. Логика напоминаний — что происходит «под капотом»

Полезно понимать общий принцип, даже не вникая в детали кода: бот должен где-то хранить список всех активных напоминаний (обычно — в небольшой базе данных, о которой мы подробнее поговорим в главе 19) и постоянно, в фоновом режиме, проверять, не наступило ли время для отправки одного из них. Это отличается от привычной для веб-страниц модели «запрос — ответ»: бот должен «жить» непрерывно, даже когда никто с ним не общается в данный момент, чтобы вовремя напомнить о том, что было запланировано заранее.

Шаг 5. От локального запуска — к боту, который работает всегда

Пока бот запущен только на вашем компьютере и работает лишь до тех пор, пока открыт терминал и компьютер включён — это удобно для тестирования, но не годится для реального использования: как только вы закроете терминал или выключите компьютер, бот перестанет отвечать.

Чтобы бот работал постоянно, его нужно разместить (задеплоить, вспоминая термин из главы 3) на сервере, который всегда включён. Подробно процесс размещения проектов в интернете мы разберём в главе 9 — принципы, изложенные там, применимы и к ботам, и к обычным сайтам.

Расширяем функциональность бота

После того как базовая версия работает, можно постепенно добавлять более сложные сценарии:

Добавь возможность создавать повторяющиеся напоминания —

например, "каждый день в 9 утра" или "каждый понедельник"

Добавь inline-кнопки под сообщением с напоминанием:

"Отложить на час" и "Отметить выполненным"

Обратите внимание, что второй запрос вводит понятие, специфичное именно для интерфейса ботов, — кнопки, встроенные прямо в сообщение (в отличие от обычных кнопок на веб-странице). Если формулировка непонятна — как всегда, можно и нужно спросить: «Что такое inline-кнопки в Telegram и как они выглядят для пользователя?»

Небольшое резюме главы

• Бот в мессенджере работает принципиально иначе, чем сайт: общение происходит через сообщения, а не через клики на странице.

• Перед созданием бота его нужно зарегистрировать через @BotFather и получить секретный токен доступа.

• Токен бота — это пароль, который нельзя публиковать в открытом доступе.

• Бот должен «жить» непрерывно, что требует размещения на постоянно включённом сервере для реального использования, а не только локального запуска для тестирования.

• Функциональность бота, как и в случае с сайтом, лучше наращивать постепенно, отдельными проверяемыми шагами.

В следующей главе разберём третий сквозной проект книги — личный кабинет с авторизацией пользователей и базой данных.

Глава 6. Личный кабинет с авторизацией и базой данных

Зачем нужен личный кабинет

Многие проекты, в отличие от простого сайта-визитки, подразумевают, что у каждого пользователя есть своя, персональная область — история заказов в интернет-магазине, личные напоминания в боте, заявки клиента в системе учёта. Это требует двух вещей, которые мы пока рассматривали только теоретически (в главе 3 и в первой книге): базы данных для хранения информации разных пользователей и системы авторизации, чтобы каждый видел только своё.

В этой главе мы создадим личный кабинет для нашего третьего сквозного проекта — системы «Учёт клиента» для условной мастерской по ремонту техники, но описанный подход одинаково применим и к личному кабинету покупателя интернет-магазина.

Шаг 1. Формулируем задачу с учётом ролей

Хочу сделать систему учёта для мастерской по ремонту техники.

Нужны две роли:

1. Администратор — видит всех клиентов, все заявки, может

создавать и редактировать записи

2. Мастер — видит только заявки, назначенные лично ему,

может менять статус заявки (в работе / готово)

Начни с регистрации и входа: страница логина с email и

паролем, после входа — перенаправление в зависимости от роли

Обратите внимание на явное разделение ролей прямо в формулировке задачи — это гораздо надёжнее, чем добавлять разграничение прав задним числом, когда система уже частично реализована без учёта этой структуры.

Шаг 2. Как устроена регистрация и вход технически

Полезно понимать общий принцип происходящего, даже не вникая в код. При регистрации пользователь вводит пароль, но сохранять его в базе данных «как есть», обычным текстом, — серьёзная и распространённая ошибка (если база данных когда-либо окажется скомпрометирована, все пароли пользователей будут немедленно раскрыты). Вместо этого пароль преобразуется необратимым математическим способом (это называется **хешированием**) в бессмысленный на вид набор символов, и именно этот набор сохраняется в базе. При каждой попытке входа введённый пароль точно так же преобразуется и сравнивается с сохранённым хешем — если совпадают, значит, пароль верный, при этом сам исходный пароль нигде, кроме момента ввода пользователем, не хранится.

СОВЕТ: Не нужно самостоятельно разбираться в деталях алгоритмов хеширования — это ровно тот случай, когда достаточно знать общий принцип и доверить реализацию Claude

Code, который по умолчанию использует общепринятые надёжные практики. Но полезно явно спросить: «Пароли пользователей точно хранятся хешированными, а не в открытом виде?» — простая проверка, которая не помешает.

Шаг 3. Структура данных

Прежде чем создавать заявки клиентов, полезно попросить Claude Code показать, как будут устроены данные:

Прежде чем реализовывать, покажи структуру таблиц базы данных:

какие поля будут у пользователя, у клиента мастерской

и у заявки на ремонт

Ответ, скорее всего, будет выглядеть примерно так (мы приводим упрощённый пример, чтобы показать сам принцип):

Таблица "Пользователи": id, имя, email, пароль (хеш), роль

Таблица "Клиенты": id, имя, телефон, дата первого обращения

Таблица "Заявки": id, клиент_id, мастер_id, описание проблемы,

статус, дата создания, дата завершения

Обратите внимание на поля клиент_id и мастер_id — они не хранят полную информацию о клиенте или мастере повторно в каждой заявке, а лишь ссылаются на соответствующую запись в других таблицах. Это фундаментальный и очень распространённый принцип устройства баз данных — избегать дублирования одной и той же информации в разных местах.

Шаг 4. Реализация авторизации

Реализуй регистрацию и вход по описанной структуре. После

успешного входа сохраняй сессию пользователя, чтобы не

приходилось вводить пароль заново при каждом переходе между

страницами. Добавь также кнопку "Выйти"

Шаг 5. Разграничение доступа

После базовой авторизации переходим к самому важному — тому, что администратор и мастер должны видеть разные наборы данных:

Убедись, что страница со списком всех заявок доступна только

администратору. Мастер при попытке напрямую открыть эту

страницу должен видеть сообщение "Доступ запрещён" и

перенаправляться на свою страницу с назначенными ему заявками

ВНИМАНИЕ: Разграничение доступа — одна из тех областей, где стоит проявлять особую тщательность при проверке (вспоминая материал главы 18 первой книги о том, чему стоит доверять безоговорочно, а что перепроверять). Обязательно протестируйте сценарий «что если мастер попробует напрямую открыть адрес страницы администратора, введя его в браузере вручную» — а не только «не видна ли ссылка на эту страницу в интерфейсе». Скрытая, но фактически доступная страница — распространённая и серьёзная уязвимость.

Шаг 6. Восстановление пароля и другие практические мелочи

Реальная система авторизации обычно требует ещё нескольких вещей, о которых легко забыть на этапе первоначального планирования:

Добавь функцию "Забыли пароль?" — отправку ссылки для

сброса пароля на email пользователя

Добавь ограничение: после 5 неудачных попыток входа подряд

временно блокировать возможность входа для этого email на

15 минут, чтобы защититься от подбора пароля

Последний пример — практика, защищающая от так называемого **перебора** (brute force) — автоматизированных попыток угадать пароль путём быстрого перебора множества вариантов подряд.

Небольшое резюме главы

• Личный кабинет требует двух взаимосвязанных компонентов: базы данных для хранения информации разных пользователей и системы авторизации.

• Пароли должны храниться в захешированном, а не открытом виде — это стандартная и обязательная практика.

• Полезно заранее продумать структуру данных, прежде чем переходить к реализации, и явно закладывать разграничение ролей в исходную формулировку задачи.

• Разграничение доступа нужно тестировать не только через видимость элементов интерфейса, но и через прямые попытки обратиться к защищённым страницам.

• Полноценная система авторизации включает такие практические детали, как восстановление пароля и защиту от подбора пароля.

В следующей главе — четвёртый и последний сквозной проект книги: автоматизация процессов небольшого бизнеса от заявки до отчёта.

Глава 7. Автоматизация для небольшого бизнеса: кейс от заявки до отчёта

От разрозненных задач — к цельному процессу

В первой книге (глава 17) мы говорили об автоматизации отдельных, разовых задач — переименовании файлов, объединении таблиц. В этой главе мы соберём похожие элементы в единый, работающий процесс, что гораздо ближе к тому, как автоматизация выглядит в реальной небольшой компании: не разовый скрипт, а связанная цепочка шагов, где результат одного этапа становится исходными данными для следующего.

Сценарий: от заявки клиента до ежемесячного отчёта

Продолжим развивать проект «Учёт клиента» из прошлой главы и построим вокруг него полноценный процесс: клиент оставляет заявку → заявка автоматически распределяется мастеру → по завершении работы формируется запись для бухгалтерии → в конце месяца автоматически собирается отчёт по всем заявкам.

Шаг 1. Автоматическое распределение заявок

Когда поступает новая заявка без указанного мастера,

автоматически назначай её тому мастеру, у которого сейчас

меньше всего активных (незавершённых) заявок. Если у

нескольких мастеров одинаковое количество — выбирай того,

кто дольше не получал новую заявку

Обратите внимание, насколько эта задача отличается от простого CRUD-интерфейса (создание, чтение, обновление, удаление записей), которым мы занимались в предыдущей главе, — здесь появляется настоящая бизнес-логика, отражающая реальное правило распределения нагрузки в мастерской. Именно такие правила стоит формулировать предельно точно, поскольку неоднозначность здесь напрямую ведёт к неправильному, но незаметному на первый взгляд поведению системы.

Шаг 2. Уведомления на ключевых этапах

Добавь уведомления:

1. Когда заявка назначена мастеру — отправить email мастеру

с описанием заявки

2. Когда мастер отмечает заявку выполненной — отправить email

клиенту с уведомлением, что ремонт завершён

3. Если заявка находится в статусе "в работе" дольше 3 дней

без изменений — отправить email администратору с

напоминанием проверить эту заявку

Третий пункт — хороший пример автоматизации, которая не просто реагирует на действие пользователя, а сама отслеживает время и инициирует действие при определённых условиях. Такая логика обычно реализуется через периодическую фоновую проверку (например, раз в час) — стоит явно спросить у Claude Code, как именно организована эта проверка в вашем случае, чтобы понимать общий принцип.

Шаг 3. Автоматическое формирование отчёта

Добавь автоматическое формирование ежемесячного отчёта в

формате Excel: первого числа каждого месяца собирать все

заявки за прошедший месяц со статусами, суммами и мастерами,

и отправлять готовый файл на email администратора

РАЗБОР КЕЙСА. При реализации подобной задачи полезно сразу же попросить возможность запустить формирование отчёта вручную, не дожидаясь начала следующего месяца: «Добавь также кнопку в административной панели “Сформировать отчёт за текущий период сейчас”, чтобы я мог проверить, что всё работает правильно, не дожидаясь календарной даты». Это общий, крайне полезный приём при работе с любой логикой, привязанной к дате или времени, — тестирование по «настоящему» календарю занимает недели, тестирование по требованию — секунды.

Шаг 4. Взгляд на процесс целиком

После того как отдельные части реализованы, полезно попросить общий обзор всей цепочки автоматизации, чтобы убедиться, что все части согласованы между собой:

Опиши весь процесс от момента поступления новой заявки до

момента, когда информация о ней попадает в месячный отчёт —

по шагам, включая все автоматические уведомления

Такой запрос выполняет ту же функцию, что и просьба «объяснить, что было сделано» из первой книги (глава 11), но применительно уже не к отдельному изменению, а ко всему выстроенному процессу целиком — это хороший способ убедиться, что общая картина в голове совпадает с тем, что реально реализовано.

Пределы автоматизации: когда стоит остановиться

Соблазн автоматизировать абсолютно всё довольно велик, но не каждая задача выигрывает от полной автоматизации. Если какой-то шаг процесса требует человеческого суждения, редко повторяется или его последствия при ошибке серьёзны (например, окончательное реше-

ние об увольнении сотрудника или крупный финансовый платёж), разумнее оставить финальное решение за человеком, автоматизировав лишь подготовку информации для этого решения.

СОВЕТ: Хороший ориентир — автоматизировать сбор информации и рутинные, часто повторяющиеся действия с понятными и предсказуемыми правилами, но оставлять точку принятия решения за человеком там, где цена ошибки высока или правило сложно сформулировать однозначно.

Небольшое резюме главы

• Реальная автоматизация в бизнесе — это обычно не разовый скрипт, а цепочка связанных этапов, где результат одного шага становится входом для следующего.

• Бизнес-правила (например, порядок распределения заявок) стоит формулировать предельно точно, поскольку неоднозначность приводит к незаметным, но реальным ошибкам поведения.

• Для логики, привязанной к дате или времени, полезно сразу предусмотреть возможность запуска вручную для тестирования.

• Не всё стоит автоматизировать полностью — там, где цена ошибки высока, разумно оставлять финальное решение за человеком.

Мы завершили часть, посвящённую сквозным реальным проектам. В следующей части книги переходим к профессиональным практикам — начиная с тестирования.

Часть 3. Профессиональные практики

Глава 8. Тестирование: как убедиться, что всё работает, не проверяя вручную

Проблема ручной проверки

В первой книге и в предыдущих главах этой мы регулярно проверяли результат вручную — открывали страницу в браузере, кликали по кнопкам, смотрели, что происходит. Это прекрасно работает для небольшого проекта, но с ростом числа функций возникает практическая проблема: каждое новое изменение теоретически может незаметно сломать что-то из уже сделанного раньше, а вручную перепроверять весь функционал заново после каждой правки — крайне утомительно и медленно.

Конец ознакомительного фрагмента.