Telegram-бот с монетизацией

практическое руководство

Максим Поднебесный

Максим Поднебесный

Telegram-бот с монетизацией: практическое руководство для тех, кто не умеет программировать

«Автор»
2026

Поднебесный М.

Telegram-бот с монетизацией: практическое руководство для тех, кто не умеет программировать / М. Поднебесный — «Автор», 2026

Практическое руководство для тех, кто хочет запустить своего Telegram-бота,но не умеет программировать.Большинство инструкций заканчиваются на «бот ответил Привет». Здесь пройденвесь путь: рабочий код, база данных, напоминания, приём платежей черезTelegram Stars и размещение на сервере, где бот работает круглосуточно.Внутри: полный код бота, который можно скопировать и запустить; регистрацияи безопасное хранение токена; меню с кнопками вместо команд; напоминания порасписанию; приём оплаты без ИП и эквайринга; бесплатный хостинг; отдельнаяглава с реальными ошибками и их причинами; честный разбор вывода денег —пороги, сроки и комиссии.Книга не обещает лёгких денег. В ней прямо сказано, сколько времени занимаеткаждый этап и почему найти читателей сложнее, чем написать код.Данные о тарифах актуальны на июль 2026 года.

Содержание

Максим Поднебесный Telegram-бот с монетизацией: практическое руководство для тех, кто не умеет программировать

Глава 1. Введение: чего ждать и чего не ждать

Эта глава короткая, но пропускать её не стоит. Она про ожидания: что вы получите к концу книги, чего не получите ни при каких условиях и сколько времени всё это займёт на самом деле.

Зачем эта книга

В интернете полно статей «Создай Telegram-бота за 10 минут». Почти все они заканчиваются на моменте, когда бот отвечает «Привет!» на команду /start. Дальше начинается то, о чём не пишут: бот падает при первом же обрыве сети, засыпает на бесплатном хостинге, теряет базу данных при перезапуске, а когда вы наконец подключаете платежи — выясняется, что вывести деньги не так просто, как казалось.

Эта книга описывает **весь путь целиком**: от регистрации бота до момента, когда он крутится на сервере круглосуточно и умеет принимать деньги. Включая ошибки, на которых я реально терял время.

Весь код в книге — не учебные примеры. Это исходники работающего бота, который был написан, задеплоен и запущен. Ошибки в главе про грабли — не выдуманные «типичные проблемы», а конкретные сообщения, которые я видел в логах.

Для кого

Для тех, кто **не умеет программировать** или умеет очень поверхностно. Если вы никогда не писали код — это нормально, весь нужный код есть в книге, его можно скопировать. Понимать, что происходит внутри, полезно, но не обязательно на старте.

Отдельно эта книга будет полезна тем, кто занимается маркетингом или продуктом и хочет проверить идею, не нанимая разработчика.

Чего ждать

К концу книги у вас будет:

— работающий бот, доступный любому пользователю Telegram;

— база данных, которая помнит пользователей и их данные;

— система напоминаний, которая работает по расписанию сама;

— приём платежей через Telegram Stars — без ИП, эквайринга и юридического лица;

— бот, развёрнутый на сервере и работающий, когда ваш компьютер выключен;

— понимание, где искать первых пользователей.

Технически это достижимо за несколько дней. Не за 10 минут, но и не за месяцы.

Чего не ждать

Здесь придётся быть неприятно честным, потому что на этом обжигаются чаще всего.

Бот не начнёт приносить деньги сам по себе. Самая большая проблема Telegram-ботов — не разработка, а дистрибуция. У ботов **нет каталога с поиском**. Ни App Store, ни Google Play, ни Chrome Web Store, где люди сами ищут «трекер привычек» и находят вас. Единственный способ получить пользователей — приводить их вручную: посты в тематических чатах, каналы, рекомендации, реклама.

Это ключевое отличие от мобильных приложений, и его нужно осознать до того, как вы вложите неделю в разработку. Хороший бот без дистрибуции не заработает ничего. Средний бот с хорошей дистрибуцией заработает.

Первый доход — это не «через неделю после запуска». Реалистичная последовательность: сначала бот работает, потом приходят первые десятки пользователей, потом становится ясно, какая часть из них вообще возвращается на второй день, и только потом кто-то платит. Каждый шаг может занять недели.

Эта книга не сделает вас программистом. Она даёт рабочий шаблон и объясняет, что в нём происходит. Чтобы писать своё с нуля, нужно учиться отдельно.

Что понадобится

Стоимость — Аккаунт Telegram: бесплатно; Компьютер с Python: бесплатно; Аккаунт GitHub: бесплатно; Хостинг (Render, бесплатный тариф): бесплатно; Сервис пингов (UptimeRobot): бесплатно; Постоянный диск на хостинге: ~$7/мес.

Обязательно? — Аккаунт Telegram: да; Компьютер с Python: да; Аккаунт GitHub: да; Хостинг (Render, бесплатный тариф): да; Сервис пингов (UptimeRobot): да; Постоянный диск на хостинге: нет, но нужен при реальных пользователях.

На старте можно уложиться в ноль рублей. Платить придётся позже — и в книге честно сказано, в какой момент бесплатный вариант перестаёт быть достаточным и почему.

Актуальность данных

Книга написана в июле 2026 года. Это важно, потому что часть её содержания привязана к условиям сторонних сервисов, а они меняются.

Что может устареть быстрее всего:

Что: Порог вывода Telegram Stars — Значение на июль 2026: 1000 звёзд

Что: Заморозка новых звёзд — Значение на июль 2026: 21 день

Что: Выплата разработчику за звезду — Значение на июль 2026: около $0,013

Что: Бесплатный тариф Render — Значение на июль 2026: без постоянного диска, засыпает без запросов

Что: Версия библиотеки aiogram — Значение на июль 2026: 3.x

Всё это — цифры на момент написания, а не обещание. **Перед тем как рассчитывать на конкретные суммы и сроки, проверьте текущие условия на самих площадках**: в @BotFather, на fragment.com и в панели вашего хостинга. В соответствующих главах указано, где именно смотреть.

Что устареть практически не может: логика работы бота, структура кода, принципы удержания пользователей, подход к выбору платных функций и разбор типичных ошибок. Это основная часть книги, и она переживёт смену тарифов.

Как читать

Главы идут в порядке реальных действий. Если пройти их подряд, на выходе получится работающий бот. Пропускать главы 3, 5 и 10 не стоит — без них ничего не заработает. Главу 11 («Грабли») имеет смысл прочитать заранее, ещё до того как что-то сломается: она сэкономит больше времени, чем все остальные.

Глава 2. Почему бот, а не мобильное приложение

Прежде чем писать код, стоит потратить час на выбор формата. Ошибка здесь стоит недель.

Сравнение форматов

Telegram-бот — Взнос за публикацию: нет; Модерация перед запуском: нет; Каталог с поиском: нет; Приём платежей: Stars из коробки; Срок до первой версии: дни; Установка для пользователя: не нужна.

Мобильное приложение — Взнос за публикацию: $25 Google Play, $99/год Apple; Модерация перед запуском: 1–14 дней; Каталог с поиском: да, сильный; Приём платежей: комиссия магазина до 30%; Срок до первой версии: недели; Установка для пользователя: нужна.

Chrome-расширение — Взнос за публикацию: $5 разово; Модерация перед запуском: 1–7 дней; Каталог с поиском: да; Приём платежей: своя платёжка; Срок до первой версии: дни–недели; Установка для пользователя: нужна.

Сайт/веб-сервис — Взнос за публикацию: нет; Модерация перед запуском: нет; Каталог с поиском: нет (только SEO); Приём платежей: эквайринг; Срок до первой версии: недели; Установка для пользователя: не нужна.

Главный плюс бота: скорость и отсутствие барьеров

Бот запускается сегодня. Не нужно регистрироваться разработчиком, платить взнос, проходить модерацию, ждать ревью. Написали код — бот работает. Ошиблись — исправили и залили заново через минуту, без переподачи на проверку.

Для пользователя барьер тоже минимальный: не нужно ничего скачивать и устанавливать, не нужно занимать место на телефоне. Переход по ссылке — и человек уже внутри. Для проверки сырой идеи это огромное преимущество: вы узнаете, нужен ли продукт людям, до того как вложите месяц.

Отдельно стоит приём платежей. Telegram Stars работают **без ИП, без юридического лица и без подключения эквайринга** — это редкость. В мобильных магазинах вы отдадите комиссию до 30%, на своём сайте придётся возиться с платёжным провайдером и юридическим оформлением.

Главный минус бота: нет каталога

Это то, что стоит понять до начала работы.

У Google Play и App Store есть поиск. Человек вводит «трекер привычек» — и видит список приложений, включая ваше. Это бесплатный поток пользователей, который идёт сам, круглосуточно, без вашего участия. То же самое у Chrome Web Store.

У Telegram-ботов такого нет. Внутренний поиск Telegram ищет по точному названию — то есть находит вас только тот, кто уже знает, что вы существуете. Каталогов ботов формально много, но реального трафика они почти не дают.

Практический вывод: закладывайте на дистрибуцию **столько же времени, сколько на разработку, если не больше**. Продумайте канал заранее — есть ли у вас доступ к тематическому чату, каналу, сообществу. Если ответа нет, идея бота может быть отличной, а результат — нулевым.

Когда бот — правильный выбор

— Нужно быстро проверить идею с минимальными вложениями.

— Продукт по сути диалоговый: напоминания, вопросы-ответы, короткие действия.

— У вас уже есть аудитория или доступ к сообществу, куда можно принести продукт.

— Важен приём денег без юридического оформления.

Когда лучше выбрать другое

— Нужен именно органический поток незнакомых пользователей — тогда мобильное приложение или расширение, где есть каталог.

— Продукту нужен сложный интерфейс: графики, перетаскивание, много экранов. В боте это неудобно.

— Нужна работа офлайн.

Промежуточный вариант: Telegram Mini App

Существует формат посередине — Mini App. Это полноценное веб-приложение с обычным интерфейсом, которое открывается внутри Telegram. Платежи Stars там работают так же, как в боте.

Mini App решает проблему интерфейса, но **не решает проблему дистрибуции** — каталога с поиском у него тоже нет. Выбирать его стоит тогда, когда продукту тесно в формате «команды и кнопки», а не тогда, когда хочется больше пользователей.

Итог

Бот — это инструмент быстрой проверки гипотезы и монетизации без бюрократии. Его слабое место одно, но серьёзное: пользователей придётся приводить руками. Всё остальное в этой книге исходит из того, что выбор сделан осознанно.

Глава 3. Создание бота: BotFather и токен

Первый практический шаг. Занимает три минуты.

Регистрация бота

Все боты в Telegram создаются через один официальный бот — **@BotFather**.

— Откройте Telegram и найдите **@BotFather** (проверьте галочку верификации — фейков с похожими именами хватает).

— Нажмите Start.

— Отправьте команду /newbot.

— BotFather спросит **имя** — это то, что видят люди в заголовке чата. Может быть на русском, с пробелами: Трекер привычек.

— Затем спросит **username** — уникальный адрес бота. Только латиница, и обязан заканчиваться на bot: например habit_tracker_bot.

Если username занят — BotFather скажет об этом, придумывайте следующий. Хорошие короткие имена разобраны, готовьтесь перебирать варианты.

После этого BotFather пришлёт сообщение с **токеном** — строкой вида:

1234567890:AAHrZmPlXqK3vN8dWfYtBcE5gJ7nQsL2xVw

Токен — это пароль от бота

Здесь важно остановиться, потому что ошибка стоит дорого.

Токен даёт **полный контроль** над ботом. Любой, у кого он есть, может читать все сообщения пользователей, писать от имени бота, менять его поведение. Восстановить контроль после утечки можно только отзывом токена — а это значит, что бот перестанет работать до обновления настроек.

Правила простые:

— **Никогда не публикуйте токен** в открытом репозитории, скриншоте, посте на форуме или в чате поддержки.

— **Не храните токен прямо в коде.** Об этом ниже.

— Если токен всё-таки утёк — немедленно откройте BotFather, /mybots ваш бот **API Token Revoke current token**. Старый перестанет работать, вы получите новый.

Как хранить токен правильно

Стандартный способ — файл .env рядом с кодом. Внутри одна строка:

BOT_TOKEN=1234567890:AAHrZmPlXqK3vN8dWfYtBcE5gJ7nQsL2xVw

Код читает токен из этого файла, а сам файл **никогда не попадает в репозиторий**. Для этого рядом создаётся файл .gitignore со строкой:

.env

Git будет игнорировать .env — вы физически не сможете случайно его закоммитить.

Чтобы другие люди (и вы сами через полгода) понимали, какие переменные нужны, рядом кладут файл-образец .env.example — с теми же именами, но без реальных значений:

BOT_TOKEN=paste_your_token_from_botfather_here

Вот этот файл в репозиторий попадает — он безопасен, секретов в нём нет.

Базовая настройка бота

Пока вы в BotFather, стоит сразу заполнить то, что видят пользователи. Команда /mybots выберите бота **Edit Bot**:

— **Edit About** — короткий текст в профиле бота, до 120 символов.

— **Edit Description** — текст, который человек видит **до** нажатия Start, до 512 символов. Это ваша витрина: объясните, что бот делает и зачем он нужен.

— **Edit Botpic** — аватар. Квадратная картинка, которая должна читаться в маленьком круге.

— **Edit Commands** — список команд, который выпадает по кнопке «/» в поле ввода.

Список команд можно задавать и программно, из кода — так удобнее, потому что он обновляется вместе с ботом. Как это делается, показано в главе 5.

Проверка токена

Прежде чем писать код, полезно убедиться, что токен рабочий. Откройте в браузере адрес, подставив свой токен:

https://api.telegram.org/bot<ВАШ_ТОКЕН>/getMe

Если токен верный, в ответ придёт JSON с данными бота:

{"ok": true,
"result": {
"id": 1234567890,
"is_bot": true,
"first_name": "Habit Tracker",
"username": "my_habit_tracker_bot"
}}

Если видите {"ok":false,"error_code":401} — токен неверный или отозван.

Это простейший способ отделить «у меня сломался код» от «у меня неправильный токен» — пригодится позже при отладке.

Итог главы

На этом этапе у вас есть: зарегистрированный бот, токен, понимание того, что токен нельзя публиковать, и заполненный профиль. Бот пока ничего не умеет — он не запущен. Этим займёмся в главе 5, а перед этим определимся, что именно он должен делать.

Глава 4. Что должен уметь MVP

MVP — минимальная версия продукта, которую уже можно дать людям. Главный навык здесь не «что добавить», а **что не добавлять**.

Правило отсечения

Каждая функция стоит времени на разработку, увеличивает шанс поломки и усложняет интерфейс. Пока у продукта ноль пользователей, вы не знаете, что им нужно — значит, любая сложная функция может оказаться выброшенной работой.

Проверочный вопрос для каждой идеи: **«Может ли продукт выполнить своё главное обещание без этого?»** Если да — функция ждёт до второй версии.

Пример: трекер привычек

Возьмём продукт, на котором построена эта книга. Главное обещание: *помочь человеку не бросить привычку*.

Вошло в MVP:

Функция: Добавить привычку — Почему обязательна: без этого продукта нет

Функция: Отметить выполнение — Почему обязательна: без этого продукта нет

Функция: Стрик (счётчик дней подряд) — Почему обязательна: главный механизм удержания

Функция: Ежедневные напоминания — Почему обязательна: без них человек забывает и уходит

Функция: Список привычек — Почему обязательна: нужен, чтобы отмечать

Не вошло в MVP (и правильно):

— статистика по месяцам и графики;

— категории и теги привычек;

— совместные привычки с друзьями;

— экспорт данных;

— настройка часового пояса;

— достижения и уровни.

Каждый пункт из второго списка выглядит разумно. Но ни один не нужен, чтобы человек мог завести привычку и не бросить её. Всё это можно добавить потом — если пользователи попросят.

Что почти всегда нужно, даже в MVP

Есть вещи, которые кажутся необязательными, но на практике решают судьбу продукта.

Онбординг за десять секунд. Первое, что видит человек после Start, определяет, останется ли он. Пустое поле ввода с просьбой «напишите название привычки» — плохо. Кнопки с готовыми вариантами («Пить воду», «Читать 10 минут», «Спорт») — хорошо: одно нажатие, и человек уже пользуется продуктом.

Понятный способ вернуться. Человек нажал Start, посмотрел, закрыл. Через день он не помнит ни одной команды. Постоянное меню с кнопками внизу экрана решает это полностью — ничего не нужно запоминать и печатать.

Причина вернуться завтра. Это самое важное и самое часто забываемое. Продукт, который ничего не напоминает о себе, теряет почти всех пользователей после первого дня. Напоминания — не «фича», а условие выживания.

Как решить, что делать платным

Типичная ошибка — сделать платным только *количество*: «бесплатно три привычки, платно безлимит». Само по себе это работает слабо: большинству людей трёх хватает, и они просто никогда не заплатят.

Платить люди готовы за то, что **снимает боль**, а не за то, что снимает ограничение. В трекере привычек такой функцией оказалась **заморозка стрика** — возможность пропустить день, не потеряв накопленный прогресс. Человек, у которого 40 дней подряд, реально боится их потерять — и это ощутимая ценность, за которую есть смысл платить.

В коде из приложения есть и то, и другое: лимит на одну привычку и заморозка. Но роли у них разные. Лимит — это мягкий сигнал «здесь есть платная версия», он сам по себе никого не убедит. Продаёт заморозка. Если убрать её и оставить только лимит, платить перестанут почти все.

Хорошие кандидаты в платные функции:

— защита накопленного результата (заморозка, восстановление);

— то, что экономит время при регулярном использовании;

— аналитика для тех, кто пользуется продуктом всерьёз.

Плохие кандидаты:

— базовая функциональность, без которой продукт бессмысленен (оттолкнёт всех на входе);

— то, чем пользуются раз в жизни.

Пробный период

Практичная схема для старта — **бесплатный пробный период на несколько дней с полным доступом**, без привязки карты. Человек успевает накопить стрик и привыкнуть, а потом ему есть что терять.

Длина периода — компромисс. Слишком короткий (1–2 дня) не даёт привыкнуть. Слишком длинный (30 дней) отодвигает решение так далеко, что человек успевает забыть о продукте. Для трекера привычек разумно 5–7 дней.

Тарифных уровней на старте делать не нужно. Одна подписка, одна цена. Усложнить успеете, когда появятся платящие пользователи и станет понятно, за что они готовы доплачивать.

Формулировка перед началом работы

Перед тем как открывать редактор кода, запишите одним предложением:

*Мой бот помогает **[кому]** сделать **[что]**, и главное действие пользователя — **[одно конкретное действие]**.*

Для нашего трекера привычек получается так:

Мой бот помогает человеку не бросить привычку, и главное действие пользователя — нажать кнопку «выполнено» раз в день.

Всё, что не обслуживает это одно действие, в MVP не входит.

Глава 5. Код бота: aiogram и база данных

В этой главе появляется работающий бот. Весь код можно копировать как есть.

Если разбираться в коде самому не хочется совсем — загляните в главу 13: там описано, как получить рабочий код с помощью языковой модели и что при этом обязательно проверять руками.

Что понадобится установить

Python — язык, на котором написан бот. Скачайте с python.org, версия 3.10 или новее. При установке на Windows обязательно поставьте галочку **«Add Python to PATH»**, иначе команды не будут работать.

Проверка, что установилось:

python --version

Должно вывести номер версии.

Библиотеки

Боту нужны три сторонние библиотеки. Создайте в папке проекта файл requirements.txt:

aiogram>=3.4,<4.0

APScheduler>=3.10,<4.0

python-dotenv>=1.0

aiohttp>=3.9

Что это такое:

— **aiogram** — библиотека для работы с Telegram. Берёт на себя всё общение с серверами Telegram.

— **APScheduler** — планировщик, запускает задачи по расписанию (для напоминаний).

— **python-dotenv** — читает токен из файла .env.

— **aiohttp** — веб-сервер, понадобится при деплое (глава 10).

Установка одной командой:

pip install -r requirements.txt

Структура проекта

Весь бот разложен на четыре файла. Это не обязательное требование, но с ростом проекта один файл на тысячу строк становится невыносимым.

main.py точка входа: запуск бота

db.py база данных: хранение и вся бизнес-логика

handlers.py обработчики: что бот отвечает на действия пользователя

scheduler.py напоминания по расписанию

Принцип разделения простой: handlers.py знает, **как разговаривать** с пользователем, db.py знает, **как считать и хранить**. Смешивать не стоит — иначе невозможно проверить логику отдельно от Telegram.

База данных

Для бота с тысячами пользователей достаточно **SQLite** — базы, которая живёт в одном файле и не требует установки сервера. Она встроена в Python, ставить ничего не нужно.

Таблиц две: пользователи и привычки.

```
SCHEMA = """
CREATE TABLE IF NOT EXISTS users (
user_id INTEGER PRIMARY KEY,
created_at TEXT NOT NULL,
trial_end TEXT NOT NULL,
subscribed_until TEXT,
reminder_time TEXT NOT NULL DEFAULT '09:00'
```

```
);
CREATE TABLE IF NOT EXISTS habits (
id INTEGER PRIMARY KEY AUTOINCREMENT,
user_id INTEGER NOT NULL,
name TEXT NOT NULL,
streak INTEGER NOT NULL DEFAULT 0,
last_done TEXT,
created_at TEXT NOT NULL,
FOREIGN KEY (user_id) REFERENCES users(user_id)
);
"""
```

user_id — это идентификатор пользователя в Telegram. Он приходит с каждым сообщением, уникален и не меняется, поэтому отлично подходит на роль первичного ключа.

Подключение к базе удобно обернуть в контекстный менеджер, чтобы не забывать закрывать соединение и сохранять изменения:

```
import sqlite3
from contextlib import contextmanager
DB_PATH = "bot.db"
@contextmanager
def get_conn():
conn = sqlite3.connect(DB_PATH)
conn.row_factory = sqlite3.Row
conn.execute("PRAGMA foreign_keys = ON")
try:
yield conn
conn.commit()
finally:
conn.close()
```

row_factory = sqlite3.Row позволяет обращаться к полям по имени (row["streak"]) вместо номера — код становится читаемым.

Регистрация пользователя

Пользователь должен появиться в базе при первом обращении. Одна функция делает и то, и другое: возвращает существующего или создаёт нового.

```
from datetime import datetime, timedelta
TRIAL_DAYS = 5
def get_or_create_user(user_id: int):
with get_conn() as conn:
row = conn.execute(
"SELECT * FROM users WHERE user_id = ?", (user_id,)
).fetchone()
if row:
return row
now = datetime.utcnow()
trial_end = now + timedelta(days=TRIAL_DAYS)
conn.execute(
"INSERT INTO users (user_id, created_at, trial_end) VALUES (?, ?, ?)",
(user_id, now.isoformat(), trial_end.isoformat()),
)
return conn.execute(
```

```
"SELECT * FROM users WHERE user_id = ?", (user_id,)
).fetchone()
```

Обратите внимание на ? в запросах вместо подстановки значений в строку. Это не стилистика, а защита: подстановка пользовательских данных напрямую в SQL открывает дыру, через которую можно испортить базу.

Логика стрика

Здесь находится главная бизнес-логика продукта, и её стоит разобрать внимательно.

```
def mark_habit_done(habit_id: int, user_id: int):
"""Засчитывает выполнение один раз в сутки.
Возвращает (стрик, было_ли_уже_отмечено_сегодня)."""
today = datetime.utcnow().date().isoformat()
with get_conn() as conn:
row = conn.execute(
"SELECT * FROM habits WHERE id = ? AND user_id = ?",
(habit_id, user_id),
).fetchone()
if row is None:
return None, False
if row["last_done"] == today:
return row["streak"], True
yesterday = (datetime.utcnow().date() - timedelta(days=1)).isoformat()
new_streak = row["streak"] + 1 if row["last_done"] == yesterday else 1
conn.execute(
"UPDATE habits SET streak = ?, last_done = ? WHERE id = ?",
(new_streak, today, habit_id),
)
return new_streak, False
```

Три случая, которые обрабатывает эта функция:

— **Уже отмечено сегодня** — ничего не меняем, сообщаем об этом. Без этой проверки пользователь накрутит стрик до сотни за минуту, нажав кнопку сто раз.

— **Отмечено вчера** — стрик продолжается, увеличиваем на единицу.

— **Отмечено давно или никогда** — стрик прерван, начинаем заново с единицы.

Заметьте: в запросах везде проверяется не только id, но и user_id. Это защита от ситуации, когда один пользователь пытается изменить чужие данные, подставив чужой идентификатор.

Обработчики: что бот отвечает

Файл handlers.py. Каждая функция обрабатывает одно действие пользователя.

```
from aiogram import Router
from aiogram.filters import Command
from aiogram.types import Message
import db
router = Router()
@router.message(Command("start"))
async def cmd_start(message: Message):
```

Конец ознакомительного фрагмента.