

Telegram-бот

записи клиентов

один файл, ноль зависимостей

Салону, барберу, репетитору: приём записи,
напоминание и выгрузка в CSV —
без администратора в мессенджере

Записаться

Сегодня 20.30 — 1 слот

10:00

13:30

18:00

20:30

```
$ python bot_booking.py --check
```

OK: 13 проверок — слоты, отмена, CSV

Шадрин Данил

**Telegram-бот записи клиентов:
один файл, ноль зависимостей**

«Автор»

2026

Данил Ш. Б.

Telegram-бот записи клиентов: один файл, ноль зависимостей /
Ш. Б. Данил — «Автор», 2026

Один файл `bot_booking.py` — около 300 строк на стандартной библиотеке Python: Telegram-бот, который принимает запись на услугу, не даёт забронировать пересекающееся время, помнит клиента, отменяет свою запись, напоминает о визите и выгружает журнал в CSV для Excel. Без frameworks и внешних зависимостей: SQLite и urllib, long-polling через Bot API. В книге: схема базы, функция `book()` с честной проверкой пересечений интервалов, напоминания, экспорт и 13 самопроверок, которые запускаются одной командой `--check`. Подходит салону, барберу, автомасляному сервису, репетитору, стоматологу — кому нужна запись без администратора в мессенджере. Пошагово показано, как получить токен у @BotFather, запустить бота и что именно проверяет самоконтроль.

© Данил Ш. Б., 2026

© Автор, 2026

Содержание

Telegram-бот записи клиентов: один файл, ноль зависимостей	5
1. Для кого такая запись	6
2. Хранение: одна база, два файла-таблицы	7
Конец ознакомительного фрагмента.	8

Шадрин Данил

Telegram-бот записи клиентов: один файл, ноль зависимостей

Telegram-бот записи клиентов: один файл, ноль зависимостей

Практическая работа с исходным кодом и самопроверкой. Тема: как принять заказ на «бота, который записывает клиентов», не посадив проект на тридцать библиотек, и как показать покупателю, что код действительно работает, — без токена и без интернета.

Всё, о чём здесь речь, лежит в одном файле `bot_booking.py` (~300 строк) и проверяется одной командой.

1. Для кого такая запись

Сценарий повторяется у любой точки, где визит идёт по времени: салон, барбер, косметолог, репетитор, стоматолог, автомасляный сервис, мастер по ногтям на дому. Сейчас это выглядит так: клиент пишет в мессенджер, владелец отвечает вручную, ведёт список в заметках и хотя бы раз в неделю сажает двух клиентов на одно время.

Бот закрывает три вещи:

клиент сам выбирает услугу и видит только свободное время;

два клиента не могут занять пересекающееся окно — это проверяется кодом, а не вниманием человека;

владелец одной командой получает журнал записей в CSV, который открывается в Excel без настройки кодировки.

Ключевое для покупателя: **ни одной внешней библиотеки**. На сервере или рабочем компьютере уже есть Python — значит «поставь и работает»: без pip, без виртуального окружения, без конфликтов версий. У этого есть и учебная ценность: видно, как устроен бот изнутри, а не как обёртка над чужим фреймворком.

2. Хранение: одна база, два файла-таблицы

SQLite — один файл на диске, никаких серверов. Обёртка `db()` создаёт таблицы при первом запуске и сразу наполняет услуги, если база пустая.

```
def db():
    con = sqlite3.connect(DB)
    con.row_factory = sqlite3.Row
    with con:
        con.executescript("""
        CREATE TABLE IF NOT EXISTS services(
            id INTEGER PRIMARY KEY AUTOINCREMENT, title TEXT NOT NULL, price
INTEGER DEFAULT 0);
        CREATE TABLE IF NOT EXISTS bookings(
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            chat_id INTEGER NOT NULL,
            service_id INTEGER NOT NULL REFERENCES services(id),
            starts TEXT NOT NULL, -- ISO datetime
            minutes INTEGER NOT NULL,
            status TEXT NOT NULL DEFAULT 'active', -- active | canceled | done
            reminded INTEGER NOT NULL DEFAULT 0,
            created TEXT NOT NULL
        );
        CREATE INDEX IF NOT EXISTS ix_book_starts ON bookings(starts, status);
        """)
    seed_defaults(con)
    return con
```

Три решения, которые здесь важны.

`row_factory = sqlite3.Row` — строки доступны по именам колонок (`r["starts"]`),

код не сползает в `row[3]` и не ломается, когда кто-то добавит колонку.

Отдельная колонка `minutes`, а не жёстко «час»: услуга длиной 90 минут ложится в ту же схему, и проверка пересечений остаётся честной.

Флаг `reminded` — чтобы напоминание отправлялось ровно один раз. Поиску «что напомнить» не нужны часы в памяти процесса, состояние живёт в базе: бота можно перезапустить, и он никого не побеспокоит дважды.

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.