# README + код бота в одном файле

# ---------------------------

# Этот файл содержит:

# 1) Подробную пошаговую инструкцию (ниже) — делай по пунктам.

# 2) Полный рабочий код Telegram-бота (ниже инструкции).

#

# Формат: сохраните этот файл как bot.py в проекте Cursor, поместите рядом google.json

# и .env с нужными переменными — затем запустите `python bot.py`.

#

# ===========================

# Пошаговая инструкция (делай ПО ОЧЕРЕДИ):

# ===========================

# 0) Требования

# - У тебя есть аккаунт Google.

# - У тебя есть доступ к Cursor (где будешь запускать бот).

#

# 1) Создать бота в Telegram (BotFather)

# - Открой Telegram -> @BotFather -> /start -> /newbot

# - Придумай имя и username (должен заканчиваться на "bot").

# - Сохрани токен (формат: 1234567890:ABC...)

#

# 2) Создать Google Sheets (таблицу)

# - Создай новую таблицу в Google Drive, назови, например: "iPhone17\_Orders"

# - В первой строке создай заголовки в таком порядке (важно для скрипта):

# Telegram | Phone | Date | FIO | Model | Memory | Color

# - Запомни точное название таблицы или укажи в переменной SPREADSHEET\_NAME

#

# 3) Создать Service Account и JSON-ключ для доступа к Google Sheets

# - Открой https://console.cloud.google.com

# - Создай новый проект (или используй существующий).

# - Перейди в «APIs & Services» -> "Library" -> включи Google Sheets API (и Google Drive API, если попросит).

# - Перейди в "IAM & Admin" -> "Service Accounts" -> "Create Service Account".

# - Создай аккаунт, потом внутри сервисного аккаунта -> "Keys" -> "Create Key" -> JSON -> скачать файл.

# - Файл положи в проект Cursor и назови, например, google.json

# - В файле JSON найди поле client\_email (что-то@...iam.gserviceaccount.com) — открой Google Sheets -> Share -> добавь этот email с правами "Editor".

#

# 4) Подготовка окружения в Cursor (файлы и зависимости)

# - Создай в проекте файлы: bot.py (этот файл), .env, google.json (скачанный ключ), requirements.txt (опционально).

# - В .env положи:

# BOT\_TOKEN=твой\_токен\_от\_BotFather

# MANAGER\_CHAT\_ID=ID\_чата\_где\_сидят\_менеджеры (число) # см. как получить ниже

# SPREADSHEET\_NAME=iPhone17\_Orders

# GOOGLE\_JSON=google.json

#

# - Установи зависимости в терминале Cursor:

# pip install python-telegram-bot==20.3 gspread google-auth python-dotenv

#

# 5) Как получить MANAGER\_CHAT\_ID (групповой чат с менеджерами)

# Вариант A (рекомендуемый):

# 1. Создай групповой чат с менеджерами в Telegram.

# 2. Добавь туда бота (через "Add member").

# 3. В BotFather отключи приватность, чтобы бот мог получать сообщения в группах: /setprivacy -> выбери бота -> Disable

# 4. В группе напиши любое сообщение. Запусти бот (python bot.py). В боте реализована команда /getchatid — просто в группе вызови эту команду или напиши любой текст, бот ответит в логах/консоли и пришлёт ID (см. шаги по тестированию ниже).

# Вариант B (если не хочешь отключать приватность):

# - Попроси менеджера написать приватно боту команду /getchatid — бот вернёт personal chat id; но тогда рассылка будет в приватный чат, не в группу.

#

# 6) Запуск бота

# - Убедись, что google.json и .env находятся в корне проекта и .env заполнен.

# - Запусти в терминале Cursor:

# python bot.py

# - В логах должно появиться "Bot started".

#

# 7) Тестирование сценария (пошагово):

# - В Telegram открой бота -> /start -> нажми "🚀 Запустить".

# - Выбери модель (нажатием на кнопку).

# - Выбирай одну или несколько опций памяти (нажимаешь, опции отмечаются галочкой). Нажми "Готово".

# - Выбирай одну или несколько опций цвета (аналогично). Нажми "Готово".

# - Подтверди — нажми "Оставить предзаказ" -> бот попросит ФИО и телефон в одной строке.

# - Отправь, например: "Иванов Иван Иванович +79001234567".

# - Проверь Google Sheets — должна появиться новая строка в порядке: Telegram, Phone, Date, FIO, Model, Memory, Color.

# - Проверь чат менеджеров — должно прийти сообщение с деталями заказа.

#

# 8) Отладка (частые ошибки)

# - Ошибка авторизации Google: проверь, что ты поделился таблицей с client\_email сервисного аккаунта.

# - Ошибка "file not found google.json": проверь имя файла и переменную GOOGLE\_JSON в .env.

# - Бот не видит сообщения в группе: проверь "Privacy" (BotFather /setprivacy -> Disable) и что бот добавлен в группу.

# - MANAGER\_CHAT\_ID пуст: установи в .env полученный ID (без кавычек).

#

# 9) Что дальше (опционально)

# - Можно добавить проверку номера телефона (валидация), автозаполнение времени доставки, интеграцию с amoCRM, webhook и хостинг (Railway/Render).

#

# ===========================

# Код бота (ниже):

*import* os

*import* re

*import* logging

*from* datetime *import* datetime

*from* dotenv *import* load\_dotenv

*from* telegram *import* Update, InlineKeyboardButton, InlineKeyboardMarkup, ReplyKeyboardRemove

*from* telegram.ext *import* (

Application, CommandHandler, CallbackQueryHandler,

MessageHandler, ConversationHandler, ContextTypes, filters

)

*import* gspread

*# ------------------ НАСТРОЙКИ ------------------*

load\_dotenv()

TOKEN = os.getenv("BOT\_TOKEN")

MANAGER\_CHAT\_ID = os.getenv("MANAGER\_CHAT\_ID")

SPREADSHEET\_NAME = os.getenv("SPREADSHEET\_NAME", "iPhone17\_Orders")

GOOGLE\_JSON = os.getenv("GOOGLE\_JSON", "google.json")

*if* not TOKEN:

*raise* RuntimeError("Не найден BOT\_TOKEN в .env")

*if* MANAGER\_CHAT\_ID:

*try*:

MANAGER\_CHAT\_ID = int(MANAGER\_CHAT\_ID)

*except* Exception:

logging.warning("MANAGER\_CHAT\_ID некорректен")

MANAGER\_CHAT\_ID = None

*# Google Sheets*

*try*:

gc = gspread.service\_account(*filename*=GOOGLE\_JSON)

sheet = gc.open(SPREADSHEET\_NAME).sheet1

print("✅ Google Sheets подключена")

*except* Exception *as* e:

logging.warning("Ошибка подключения к Google Sheets: %s", e)

sheet = None

*# ------------------ Конфиг сцен ------------------*

CHOOSING, MODEL, MEMORY, COLOR, CONFIRM, CONTACT = range(6)

MODELS = ["iPhone 17", "iPhone 17 Pro", "iPhone 17 Pro Max", "iPhone 17 Air"]

*# Память и цвета по моделям*

MEMORY\_BY\_MODEL = {

"iPhone 17": ["128", "256", "512"],

"iPhone 17 Air": ["128", "256", "512"],

"iPhone 17 Pro": ["256", "512", "1 TB"],

"iPhone 17 Pro Max": ["256", "512", "1 TB"],

}

COLOR\_BY\_MODEL = {

"iPhone 17": ["black", "white", "green", "purple", "light blue"],

"iPhone 17 Air": ["black", "white", "light gold", "light blue"],

"iPhone 17 Pro": ["black", "white", "grey", "dark blue", "orange"],

"iPhone 17 Pro Max": ["black", "white", "grey", "dark blue", "orange"],

}

logging.basicConfig(*level*=logging.INFO)

*# ------------------ Клавиатуры ------------------*

def build\_model\_kb():

kb = [[InlineKeyboardButton(m, *callback\_data*=f"model\_{i}")] *for* i, m *in* enumerate(MODELS)]

*return* InlineKeyboardMarkup(kb)

def build\_multi\_kb(*options*, *selected\_set*, *prefix*="opt"):

kb = []

*for* opt *in* *options*:

text = ("✅ " *if* opt in *selected\_set* *else* "") + opt

*# Используем callback\_data безопасно — заменяем пробелы в opt, но сохраняем отображение*

cb = f"{*prefix*}\_{opt}".replace(" ", "\_")

kb.append([InlineKeyboardButton(text, *callback\_data*=cb)])

kb.append([InlineKeyboardButton("Готово", *callback\_data*=f"{*prefix*}\_done")])

*return* InlineKeyboardMarkup(kb)

def post\_order\_menu():

kb = [

[InlineKeyboardButton("📱 Новый предзаказ", *callback\_data*="new\_order")],

[InlineKeyboardButton("📞 Связаться с менеджером", *callback\_data*="contact\_manager")],

[InlineKeyboardButton("🔥 Акции", *callback\_data*="show\_sales")],

[InlineKeyboardButton("ℹ️ О магазине", *callback\_data*="about\_shop")]

]

*return* InlineKeyboardMarkup(kb)

*# ------------------ Хендлеры Conversation ------------------*

async def cmd\_start(*update*: Update, *context*: ContextTypes.DEFAULT\_TYPE):

kb = InlineKeyboardMarkup([[InlineKeyboardButton("🚀 Запустить", *callback\_data*="start\_order")]])

*await* *update*.message.reply\_text(

"Здравствуйте! Вас приветствует магазин \"TechStore\" — бот для предзаказа iPhone 17.\n\nНажмите кнопку и начнём оформление предзаказа.",

*reply\_markup*=kb

)

*return* CHOOSING

async def cb\_start\_order(*update*: Update, *context*: ContextTypes.DEFAULT\_TYPE):

"""

Обрабатывает нажатие '🚀 Запустить' или 'Новый предзаказ' (если он зарегистрирован как entry point).

Переходит к выбору модели.

"""

query = *update*.callback\_query

*await* query.answer()

*await* query.edit\_message\_text(*text*="Выберите модель:", *reply\_markup*=build\_model\_kb())

*return* MODEL

async def cb\_choose\_model(*update*: Update, *context*: ContextTypes.DEFAULT\_TYPE):

query = *update*.callback\_query

*await* query.answer()

*# callback\_data = model\_{index}*

idx = int(query.data.split("\_")[1])

model = MODELS[idx]

*context*.user\_data["model"] = model

*# сбрасываем прошлые выборы*

*context*.user\_data["memory"] = set()

*context*.user\_data["colors"] = set()

*# Подставляем опции для этой модели*

*context*.user\_data["memory\_options"] = MEMORY\_BY\_MODEL.get(model, [])

*context*.user\_data["color\_options"] = COLOR\_BY\_MODEL.get(model, [])

*# Отправляем клавиатуру памяти (мультивыбор)*

*await* query.edit\_message\_text(

*text*=f"Модель: {model}\n\nВыберите объём памяти (можно выбрать несколько):",

*reply\_markup*=build\_multi\_kb(*context*.user\_data["memory\_options"], *context*.user\_data["memory"], *prefix*="mem")

)

*return* MEMORY

async def cb\_memory(*update*: Update, *context*: ContextTypes.DEFAULT\_TYPE):

query = *update*.callback\_query

*await* query.answer()

data = query.data *# mem\_128 or mem\_256 or mem\_done*

memory\_options = *context*.user\_data.get("memory\_options", [])

*if* data.endswith("\_done"):

*# перейти к выбору цвета*

*await* query.edit\_message\_text(

*text*="Выберите цвет (можно несколько):",

*reply\_markup*=build\_multi\_kb(*context*.user\_data.get("color\_options", []), *context*.user\_data["colors"], *prefix*="col")

)

*return* COLOR

*else*:

*# извлекаем опцию — восстановим пробелы если были заменены*

opt = data.split("\_", 1)[1].replace("\_", " ")

sel = *context*.user\_data.setdefault("memory", set())

*if* opt in sel:

sel.remove(opt)

*else*:

sel.add(opt)

*# обновляем клавиатуру с выделением выбранных*

*await* query.edit\_message\_reply\_markup(*reply\_markup*=build\_multi\_kb(memory\_options, sel, *prefix*="mem"))

*return* MEMORY

async def cb\_color(*update*: Update, *context*: ContextTypes.DEFAULT\_TYPE):

query = *update*.callback\_query

*await* query.answer()

data = query.data

color\_options = *context*.user\_data.get("color\_options", [])

*if* data.endswith("\_done"):

model = *context*.user\_data.get("model")

mem = ", ".join(sorted(*context*.user\_data.get("memory", []))) or "(не выбрано)"

cols = ", ".join(sorted(*context*.user\_data.get("colors", []))) or "(не выбрано)"

summary = f"Ваш выбор:\nМодель: {model}\nПамять: {mem}\nЦвет: {cols}"

kb = InlineKeyboardMarkup([[InlineKeyboardButton("Оставить предзаказ", *callback\_data*="confirm\_order")]])

*await* query.edit\_message\_text(*text*=summary, *reply\_markup*=kb)

*return* CONFIRM

*else*:

opt = data.split("\_", 1)[1].replace("\_", " ")

sel = *context*.user\_data.setdefault("colors", set())

*if* opt in sel:

sel.remove(opt)

*else*:

sel.add(opt)

*await* query.edit\_message\_reply\_markup(*reply\_markup*=build\_multi\_kb(color\_options, sel, *prefix*="col"))

*return* COLOR

async def cb\_confirm\_order(*update*: Update, *context*: ContextTypes.DEFAULT\_TYPE):

query = *update*.callback\_query

*await* query.answer()

*await* query.edit\_message\_text(

*text*="Пожалуйста, введите ваше ФИО и номер телефона в одной строке, например:\nИванов Иван +79001234567",

*reply\_markup*=None

)

*return* CONTACT

PHONE\_RE = re.compile(r"(\+?\d[\d\s\-()]{5,}\d)")

async def contact\_handler(*update*: Update, *context*: ContextTypes.DEFAULT\_TYPE):

text = *update*.message.text.strip()

m = PHONE\_RE.search(text)

phone = m.group(1).strip() *if* m *else* ""

fio = text.replace(m.group(1), "").strip() *if* m *else* text

user = *update*.message.from\_user

nick = f"@{user.username}" *if* user.username *else* f"{user.first\_name or ''} {user.last\_name or ''}".strip() or str(user.id)

now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

model = *context*.user\_data.get("model", "")

memory = ", ".join(sorted(*context*.user\_data.get("memory", [])))

colors = ", ".join(sorted(*context*.user\_data.get("colors", [])))

*# Запись в Google Sheets*

*if* sheet:

*try*:

sheet.append\_row([nick, phone, now, fio, model, memory, colors])

logging.info("Запись добавлена в Google Sheets: %s", [nick, phone, now, fio, model, memory, colors])

*except* Exception *as* e:

logging.warning("Ошибка при записи в Google Sheets: %s", e)

*# Уведомление менеджеру*

msg = (

f"📦 \*Новый предзаказ!\*\n\n"

f"👤 Пользователь: {nick}\n"

f"📇 ФИО: {fio}\n"

f"📞 Телефон: {phone}\n"

f"📱 Модель: {model}\n"

f"💾 Память: {memory}\n"

f"🎨 Цвет: {colors}\n"

f"🕒 Дата: {now}"

)

*if* MANAGER\_CHAT\_ID:

*try*:

*await* *context*.bot.send\_message(*chat\_id*=MANAGER\_CHAT\_ID, *text*=msg)

*except* Exception *as* e:

logging.warning("Не удалось отправить сообщение менеджеру: %s", e)

*# Меню после заказа (кнопки остаются видимыми)*

*await* *update*.message.reply\_text(

"✅ Спасибо! Ваша заявка принята. Менеджеры свяжутся с вами.",

*reply\_markup*=post\_order\_menu()

)

*context*.user\_data.clear()

*return* ConversationHandler.END

*# ------------------ Кнопки после заказа (вне ConversationHandler) ------------------*

async def cb\_contact\_manager(*update*: Update, *context*: ContextTypes.DEFAULT\_TYPE):

query = *update*.callback\_query

*await* query.answer()

text = "📞 Связаться с менеджером:\nТелефон: +7 (900) 123-45-67\nTelegram: @manager\nEmail: support@techstore.com"

*# отправляем ответ и оставляем меню видимым*

*await* query.message.reply\_text(text, *reply\_markup*=post\_order\_menu())

async def cb\_show\_sales(*update*: Update, *context*: ContextTypes.DEFAULT\_TYPE):

query = *update*.callback\_query

*await* query.answer()

text = "🔥 Текущие акции:\n- Скидка 20% на аксессуары при предзаказе\n- Рассрочка 0% на 6 месяцев"

*await* query.message.reply\_text(text, *reply\_markup*=post\_order\_menu())

async def cb\_about\_shop(*update*: Update, *context*: ContextTypes.DEFAULT\_TYPE):

query = *update*.callback\_query

*await* query.answer()

text = (

"ℹ️ О магазине TechStore:\n"

"Адрес: ул. Примерная, 10, Москва\n"

"Телефон: +7 (900) 123-45-67\n"

"Соц.сети: @techstore\_vk, @techstore\_telegram\n"

"Гарантия 1 год на всю технику.\n"

"Режим работы: Пн–Пт 10:00–20:00, Сб–Вс 11:00–18:00"

)

*await* query.message.reply\_text(text, *reply\_markup*=post\_order\_menu())

*# ------------------ Прочие команды ------------------*

async def cmd\_info(*update*: Update, *context*: ContextTypes.DEFAULT\_TYPE):

*await* *update*.message.reply\_text("ℹ️ Наш магазин TechStore — официальные продажи, гарантия 1 год. Адрес магазина: ул. Примерная, 10, Москва")

async def cmd\_sales(*update*: Update, *context*: ContextTypes.DEFAULT\_TYPE):

*await* *update*.message.reply\_text("🔥 Текущие акции:\n- Скидка 20% на аксессуары при предзаказе\n- Рассрочка 0% на 6 месяцев")

async def cmd\_cancel(*update*: Update, *context*: ContextTypes.DEFAULT\_TYPE):

*await* *update*.message.reply\_text("Операция отменена.", *reply\_markup*=ReplyKeyboardRemove())

*context*.user\_data.clear()

*return* ConversationHandler.END

*# ------------------ Запуск бота ------------------*

def main():

app = Application.builder().token(TOKEN).build()

conv = ConversationHandler(

*entry\_points*=[

CommandHandler("start", cmd\_start),

CallbackQueryHandler(cb\_start\_order, *pattern*="^start\_order$"), *# кнопка на стартовом сообщении*

CallbackQueryHandler(cb\_start\_order, *pattern*="^new\_order$"), *# теперь new\_order — entry point*

],

*states*={

CHOOSING: [CallbackQueryHandler(cb\_start\_order, *pattern*="^start\_order$")],

MODEL: [CallbackQueryHandler(cb\_choose\_model, *pattern*="^model\_")],

MEMORY: [CallbackQueryHandler(cb\_memory, *pattern*="^mem\_")],

COLOR: [CallbackQueryHandler(cb\_color, *pattern*="^col\_")],

CONFIRM: [CallbackQueryHandler(cb\_confirm\_order, *pattern*="^confirm\_order$")],

CONTACT: [MessageHandler(filters.TEXT & ~filters.COMMAND, contact\_handler)],

},

*fallbacks*=[CommandHandler("cancel", cmd\_cancel)],

*allow\_reentry*=True,

)

app.add\_handler(conv)

*# Обработчики кнопок, которые должны работать вне ConversationHandler*

app.add\_handler(CallbackQueryHandler(cb\_contact\_manager, *pattern*="^contact\_manager$"))

app.add\_handler(CallbackQueryHandler(cb\_show\_sales, *pattern*="^show\_sales$"))

app.add\_handler(CallbackQueryHandler(cb\_about\_shop, *pattern*="^about\_shop$"))

*# Прочие команды*

app.add\_handler(CommandHandler("info", cmd\_info))

app.add\_handler(CommandHandler("sales", cmd\_sales))

app.add\_handler(CommandHandler("cancel", cmd\_cancel))

print("Bot started — запустите в терминале. Ctrl+C для остановки.")

app.run\_polling()

*if* \_\_name\_\_ == "\_\_main\_\_":

main()