**README — Бот Волонтёрского центра (Aiogram v3)**

**Общее описание**

Телеграм-бот для Волонтёрского центра РЭУ, помогающий:

* собирать **фото-отчёты** о мероприятиях;
* принимать **отзывы** в удобном формате;
* записывать волонтёров на **интервью** по свободным слотам;
* предоставлять **админ-инструменты** для управления слотами и выгрузки данных.

Бот использует **FSM** для диалоговых сценариев, **SQLite** (через aiosqlite) для хранения, а также простую систему **прав администраторов** по списку admin\_ids.

**Технологии и библиотеки**

* **Python 3.10+**
* **aiogram v3** — маршрутизация, FSM, фильтры, разметка.
* **asyncio** — асинхронные обработчики.
* **aiosqlite** — асинхронная работа с SQLite.
* **logging** — базовое логирование (уровень INFO).
* **FSM** из aiogram.fsm — сценарии /feedback, /interviews, админ-флоу.
* **datetime** — форматирование дат/времени.
* **pathlib / re / collections** — утилиты.
* **Конфиг**: from conf\_reader import config — токен читается из внешней конфигурации.

По умолчанию бот отправляет сообщения в HTML-разметке (ParseMode.HTML) и сохраняет фотографии в папку ./photos рядом с bot.py.

**Быстрый старт**

1. **Клонирование и окружение**

python -m venv .venv

. .venv/bin/activate # Windows: .venv\Scripts\activate

pip install -U aiogram aiosqlite pydantic python-dotenv

1. **Конфигурация**  
   Создайте conf\_reader.py (если его нет) со считыванием токена из переменных окружения/.env:

# conf\_reader.py (пример)

from pydantic import BaseSettings, SecretStr

class Settings(BaseSettings):

bot\_token: SecretStr

class Config:

env\_file = ".env"

env\_file\_encoding = "utf-8"

config = Settings()

.env:

BOT\_TOKEN=123456:ABC-DEF...

1. **Запуск**

python bot.py

При первом запуске создастся база data.db и каталог photos.

**Структура БД**

SQLite-файл: data.db.

**Таблица feedbacks — отзывы**

| **Поле** | **Тип** | **Назначение** |
| --- | --- | --- |
| id\_feedbacks | INTEGER PK AUTOINCREMENT | Идентификатор отзыва |
| full\_name\_feedbacks | TEXT | ФИО автора |
| group\_name\_feedbacks | TEXT | Учебная группа (если применимо) |
| event\_name\_feedbacks | TEXT | Название/дата мероприятия |
| feedback\_text\_feedbacks | TEXT | Полный текст отзыва |
| tg\_id\_feedbacks | TEXT | TG ID / @username отправителя |

**Таблица interviews — слоты интервью**

| **Поле** | **Тип** | **Назначение** |
| --- | --- | --- |
| id\_interviews | INTEGER PK AUTOINCREMENT | Идентификатор слота |
| day\_interviews | TEXT | Дата (произвольная строка, обычно DD.MM.YYYY) |
| time\_interviews | TEXT | Время (например, 12:30-12:50) |
| tg\_id\_volunteer\_interviews | TEXT | TG ID волонтёра (после брони) |
| tg\_id\_recruiter\_interviews | TEXT | TG ID тим-лидера/рекрутера |
| full\_name\_interviews | TEXT | ФИО волонтёра (после брони) |
| is\_busy\_interviews | INTEGER | Занят ли слот: 0 — свободен, 1 — занят |

Инициализация схемы выполняется в init\_db() при старте.

**Состояния FSM**

**FeedbackStates — сбор отзыва**

1. full\_name\_feedbacks — запрос ФИО.
2. group\_name\_feedbacks — запрос номера группы.
3. event\_name\_feedbacks — запрос названия/даты мероприятия.
4. feedback\_text\_feedbacks — запрос структурированного текста отзыва.

**InterviewStates — запись на интервью**

1. full\_name\_interviews — запрос ФИО.
2. user\_time\_slot — выбор слота (ID из списка свободных).

**AdminAddSlotStates — добавление слота (админ)**

1. day\_interviews — дата.
2. time\_interviews — интервал времени.
3. tg\_id\_recruiter\_interviews — TG ID тим-лидера.

**AdminDeleteSlotStates — удаление слота (админ)**

1. slot\_id — ID слота.
2. confirm — подтверждение строкой подтверждаю.

**Команды: администраторы**

Доступ по списку admin\_ids = { ... }. Измените список перед деплоем.

* **/admin\_add\_slot**  
  Диалог: дата → время (HH:MM-HH:MM) → TG ID рекрутера.  
  В БД добавляется запись в interviews с is\_busy\_interviews=0.  
  В конце бот отправляет резюме добавленного слота и дублирует в служебный чат.
* **/admin\_delete\_slot**  
  Диалог: ID слота → подтверждение подтверждаю.  
  При подтверждении удаляет строку из interviews и присылает отчёт (также — в служебный чат).
* **/admin\_get\_info\_interviews**  
  Выводит список свободных слотов (форматированный список id/date/time).
* **/admin\_get\_info\_feedbacks**  
  Вытягивает все отзывы (ФИО, группа, событие, текст, TG ID) и отправляет их администратору.
* **Служебные**:
  + /info — время запуска бота (берётся из dp["started\_at"]).
  + /chat\_id — показать chat.id текущего чата (удобно для настройки служебного канала).
  + /dice — отправка 🎲 (тест).
  + /test — demo-сообщения с HTML/без разметки.

В коде используется служебный канал/чат для дублирования сообщений: -4822221970. Замените на Ваш ADMIN\_CHAT\_ID.

**Команды: пользователи**

* **/start**  
  Короткая справка по функциям бота.
* **/images** — отправка фото-отчёта  
  Пошаговый сценарий:
  1. ввести **дату** мероприятия (YYYY-MM-DD пример в подсказке);
  2. ввести **название** мероприятия;
  3. отправлять **фотографии по одной** (файлом или фото).  
     Файлы сохраняются в ./photos с именем:
* {дата}\_{название}\_{tgid\_или\_userid}\_{счётчик}.{расширение}

Завершить сценарий — командой **/done** (очистит временные данные в памяти).

* **/feedback** — отправка отзыва  
  Пошагово собираются: ФИО → группа → мероприятие → текст отзыва по предложенному шаблону.  
  После сохранения бот отправляет итоговый текст пользователю и дублирует в служебный чат.
* **/interviews** — запись на интервью  
  Пользователь вводит ФИО, бот показывает сгруппированные по дате **свободные слоты** (is\_busy\_interviews=0) и просит прислать **ID** подходящего.  
  После выбора слот помечается занятым (is\_busy\_interviews=1), в строку записываются TG/ФИО волонтёра.  
  Итог с деталями (дата, время, место встречи) отправляется пользователю и дублируется в служебный чат.

**Механика и поток данных**

* **Маршрутизация** — декораторы @dp.message(Command("...")) и фильтры F.photo | F.document.
* **Проверка админ-прав** — в админ-командах ранняя проверка message.from\_user.id in admin\_ids.
* **Хранилище фото** — папка PHOTOS\_PATH = Path(\_\_file\_\_).parent / "photos".  
  Имена нормализуются через \_sanitize (замена пробелов/небезопасных символов).  
  Скачивание файлов: await bot.download(file\_id, destination=file\_path) с try/except.
* **База данных** — все операции через aiosqlite в async with блоках, коммиты после INSERT/DELETE/UPDATE.
* **Группировка слотов** — выборка свободных слотов и сортировка (даты/время) с нормализацией строк и безопасным парсингом.
* **Дублирование в служебный чат** — итоговые сообщения (feedback, бронь интервью, сохранённые имена файлов) также отправляются в заданный chat\_id.

**Логирование и обработка ошибок**

* **Логирование**: logging.basicConfig(level=logging.INFO).
* **Декоратор @safe\_handler** оборачивает обработчики и:
  + перехватывает любые исключения,
  + пытается ответить пользователю сообщением Произошла непредвиденная ошибка: {e},
  + логирует стек через logging.exception("Unhandled exception in handler").
* **Сохранение файлов** окружено try/except — при ошибке скачивания фото пользователь получает понятное сообщение и сценарий не падает.

Рекомендации:  
• Добавьте файловый хендлер (RotatingFileHandler) для логов в проде.  
• Включите уровень WARNING/ERROR для шумных окружений.  
• Валидация пользовательского ввода (дат/времени) уже есть базовая; при необходимости усилите регулярки/парсинг.

**Конфигурация и параметры**

* **Токен бота** — через conf\_reader.config.bot\_token.
* **Список админов** — admin\_ids = {...} (обновите под Ваши ID).
* **Служебный канал/чат** — замените -4822221970 на свой.
* **Каталог фото** — ./photos (создаётся автоматически).
* **БД** — ./data.db (создаётся/мигрирует автоматически в init\_db()).

**Известные ограничения**

* **Память сценария /images** хранится в оперативной памяти (user\_photo\_info); при рестарте бота активные сессии забудутся.
* **Формат дат/времени** в слотах не нормализован в единую ISO-схему (хранится строкой) — так проще оперировать, но сложнее агрегировать.
* **Валидация расширений** при загрузке фото минимальная; для строгих политик безопасности добавьте белый список.

**Полезные подсказки для админов**

* Узнать ID текущего чата для служебных уведомлений: запустите команду **/chat\_id** в нужном чате/канале.
* Проверить, что бот жив: **/info**.
* Экспорт отзывов: **/admin\_get\_info\_feedbacks** и сохраните ответ.

**Структура проекта (минимум)**

project/

├─ bot.py

├─ conf\_reader.py # пример см. выше

├─ .env # BOT\_TOKEN=...

├─ data.db # создаётся автоматически

└─ photos/ # создаётся автоматически