

# Документация к проекту: Telegram-бот "TelePost Manager"

## 1. Введение

### 1.1. Описание проекта

Проект "TelePost Manager" представляет собой Telegram-бота, разработанного на языке Python с использованием библиотеки Aiogram 3.x. Основная цель бота — предоставление пользователям удобного инструмента для автоматизации публикаций и управления контентом в их Telegram-каналах. Бот поддерживает многопользовательский режим, позволяя каждому пользователю управлять своим набором каналов и шаблонов.

### 1.2. Актуальность и цели

В условиях роста популярности Telegram как платформы для распространения информации, эффективное управление контентом в каналах становится все более актуальной задачей. "TelePost Manager" нацелен на:

- Упрощение процесса создания и публикации постов.
- Автоматизацию отложенного постинга.
- Обеспечение консистентности оформления публикаций через систему шаблонов.
- Предоставление базовых инструментов контроля контента.
- Реализацию многопользовательской архитектуры с разделением данных.

## 2. Функциональные возможности бота

### 2.1. Управление пользователями

\* Автоматическая регистрация пользователя при первом взаимодействии с ботом (команда /start).

\* Определение роли "супер-администратора" по уникальному Telegram ID, указанному в конфигурации. Супер-администратор получает доступ к расширенным функциям управления ботом.


### 2.2. Управление каналами (индивидуально для каждого пользователя)

\* **Подключение каналов:** Пользователь может подключить свои Telegram-каналы к боту. Для этого необходимо назначить бота администратором канала с правом на публикацию сообщений и затем переслать любое сообщение из канала в чат с ботом.


\* **Просмотр списка каналов:** Команда /my\_channels (или кнопка "🔔 Мои каналы") отображает список подключенных пользователем каналов.

\* **Удаление канала:** Пользователь может удалить канал из своего списка управления через меню "Мои каналы".


## 2.3. Создание и публикация постов

- \* **Ручное создание:** Мастер создания поста (команда `/new_post` или кнопка  "Создать пост") ведет пользователя через шаги выбора канала, ввода текста и прикрепления медиа.
- \* **Запланированные публикации:** Возможность указать точное время и дату для автоматической публикации поста.
- \* **Немедленная публикация:** Опция "сейчас" для мгновенной отправки поста.
- \* **Поддержка медиа:** Возможность прикрепления одного фото или видео к посту.
- \* **Предпросмотр:** Перед финальной публикацией/планированием пользователь видит предпросмотр поста.
- \* **Система шаблонов:**
  - \* **Личные шаблоны:** Каждый пользователь может создавать, просматривать и удалять свои личные шаблоны постов (текст + опционально медиа).
  - \* **Общие шаблоны:** Супер-администратор может создавать и управлять общими шаблонами, доступными для использования всем пользователям бота.
  - \* **Пользовательские переменные в шаблонах:** Пользователи могут определять в тексте шаблона собственные переменные в формате `{[название_переменной]}`. При создании поста по такому шаблону бот запросит значения для каждой переменной.
  - \* **Информация о переменных:** В меню шаблонов доступна справка о том, как создавать и использовать пользовательские переменные.

## 2.4. Управление запланированными постами

- \* **Просмотр:** Команда `/my_scheduled` (или кнопка  "Запланированные") отображает список запланированных постов текущего пользователя.
- \* **Отмена:** Пользователь может отменить любой из своих запланированных постов.

## 2.5. История публикаций

- \* **Просмотр:** Команда `/history` (или кнопка  "История") отображает историю опубликованных, отмененных или неудачно отправленных постов для текущего пользователя с пагинацией.
- \* **Статус и ссылка:** Для опубликованных постов отображается ссылка на сообщение в канале.

## 2.6. Контроль контента

- \* **Фильтр запрещенных слов:** Глобальный список запрещенных слов проверяется при создании поста или заполнении переменных шаблона. При обнаружении таких слов бот предупреждает пользователя и не позволяет продолжить без исправления.
- \* **Администрирование черного списка (супер-админ):**

- \* `/add_banned_word <слово>`: Добавить слово в черный список.
- \* `/remove_banned_word <слово>`: Удалить слово из черного списка.
- \* `/list_banned_words`: Показать текущий черный список.

## 2.7. Администрирование бота (супер-админ)

- \* `/admin_stats`: Показать статистику по боту (количество пользователей, каналов, постов по статусам, шаблонов).
- \* `/list_users [N]`: Показать список последних N зарегистрированных пользователей.

## 2.8. Взаимодействие с пользователем

- \* **Команды**: Поддержка основных команд (`/start`, `/help`, `/cancel` и др.).
- \* **Интерфейс**: Использование Reply-клавиатур для основного меню и Inline-клавиатур для контекстных действий.
- \* **Обработка ошибок**: Сообщения об ошибках и инструкции для пользователя.

## 3. Архитектура и технологический стек

### 3.1. Структура проекта

Проект имеет модульную структуру для лучшей организации кода:

- \* `main.py`: Точка входа приложения, инициализация бота и диспетчера, запуск поллинга.
- \* `loader.py`: Инициализация глобальных объектов (Bot, Dispatcher, DBManager, Scheduler, ContentFilter).
- \* `config.py`: Загрузка конфигурационных данных (токен, имя БД, ID администратора) из переменных окружения (файл `.env`).
- \* `handlers/`: Пакет с обработчиками команд и сообщений, разделенными по функциональным блокам (`common.py`, `channels.py`, `posts.py`, `templates.py`, `history.py`, `admin_features.py`, `scheduled_posts.py`).
- \* `models/`: Пакет с моделями данных и логикой работы с ними (`database.py` для SQLite, `content_filter.py` для фильтра слов).
- \* `services/`: Пакет для вспомогательных сервисов (`scheduler.py` для APScheduler).
- \* `filters/`: Пакет для пользовательских фильтров (например, `admin.py` для `IsAdmin`).
- \* `utils/`: (Если бы был, содержал бы общие утилиты. В данном проекте `bot_utils.py` находится в корне).
- \* `bot_utils.py`: Вспомогательные функции (генерация клавиатур, уведомления, экранирование HTML).
- \* `post_states.py`: Определения состояний для FSM (Finite State Machine).

### 3.2. Технологический стек

- \* **Язык программирования:** Python 3.x
- \* **Фреймворк для Telegram-бота:** Aiogram 3.x
- \* **База данных:** SQLite (через модуль `sqlite3`)
- \* **Планировщик задач:** APScheduler (для отложенных публикаций)
- \* **Управление конфигурацией:** `python-dotenv` (для загрузки из `.env`)
- \* **Логирование:** Стандартный модуль `logging` Python.

### 4. База данных

Используется SQLite. Схема БД включает следующие таблицы:

- \* `bot_users`: Информация о пользователях бота (`user_id`, `username`, `first_name`, `last_name`, `is_admin`, `created_at`, `last_seen_at`).
- \* `channels`: Информация о подключенных пользователями каналах (`id`, `user_id`, `channel_id`, `title`).
- \* `posts`: Информация о созданных постах (`id`, `user_id`, `channel_id`, `content`, `media`, `media_type`, `publish_time`, `status`, `message_id`, `created_at`).
- \* `templates`: Информация о шаблонах постов (`id`, `user_id` (0 для общих), `name`, `content`, `media`, `media_type`).

### 5. Инструкция по развертыванию и запуску

#### 5.1. Предварительные требования

- \* Python 3.9 или выше.
- \* Менеджер пакетов `pip`.

#### 5.2. Установка зависимостей

1. Склонировать репозиторий проекта (если он есть) или скопировать файлы проекта.
2. Создать и активировать виртуальное окружение (рекомендуется):  
`bash python -m venv .venv source .venv/bin/activate # для Linux/macOS`  
`.venv\Scripts\activate # для Windows`
3. Установить необходимые библиотеки из файла `requirements.txt`:  
`bash pip install -r requirements.txt`

#### 5.3. Конфигурация

1. Создать файл `.env` в корневой директории проекта.
  2. Заполнить файл `.env` необходимыми данными:  
`env BOT_TOKEN=ВАШ_ТЕЛЕГРАМ_БОТ_TOKEN`  
`DATABASE_NAME=telepost_manager.db # Можно изменить имя файла БД`  
`BANNED_WORDS_FILE=banned_words.txt # Имя файла для запрещенных слов`  
`SUPER_ADMIN_ID=ВАШ_ТЕЛЕГРАМ_ID # Числовой ID супер-администратора`
- \* `BOT_TOKEN`: Получить у `@BotFather` в Telegram.

\* `SUPER_ADMIN_ID`: Ваш числовой Telegram ID (можно узнать у @userinfobot).

#### **5.4. Запуск бота**

Выполнить команду в терминале из корневой папки проекта:

```
bash python main.py
```

Бот начнет работу и будет готов принимать команды.

#### **6. Заключение**

Telegram-бот "TelePost Manager" предоставляет комплексное решение для управления публикациями в каналах. Реализованный функционал покрывает основные потребности пользователей по созданию, планированию постов и управлению шаблонами в многопользовательском режиме. Архитектура проекта обеспечивает возможность дальнейшего расширения и добавления новых функций.