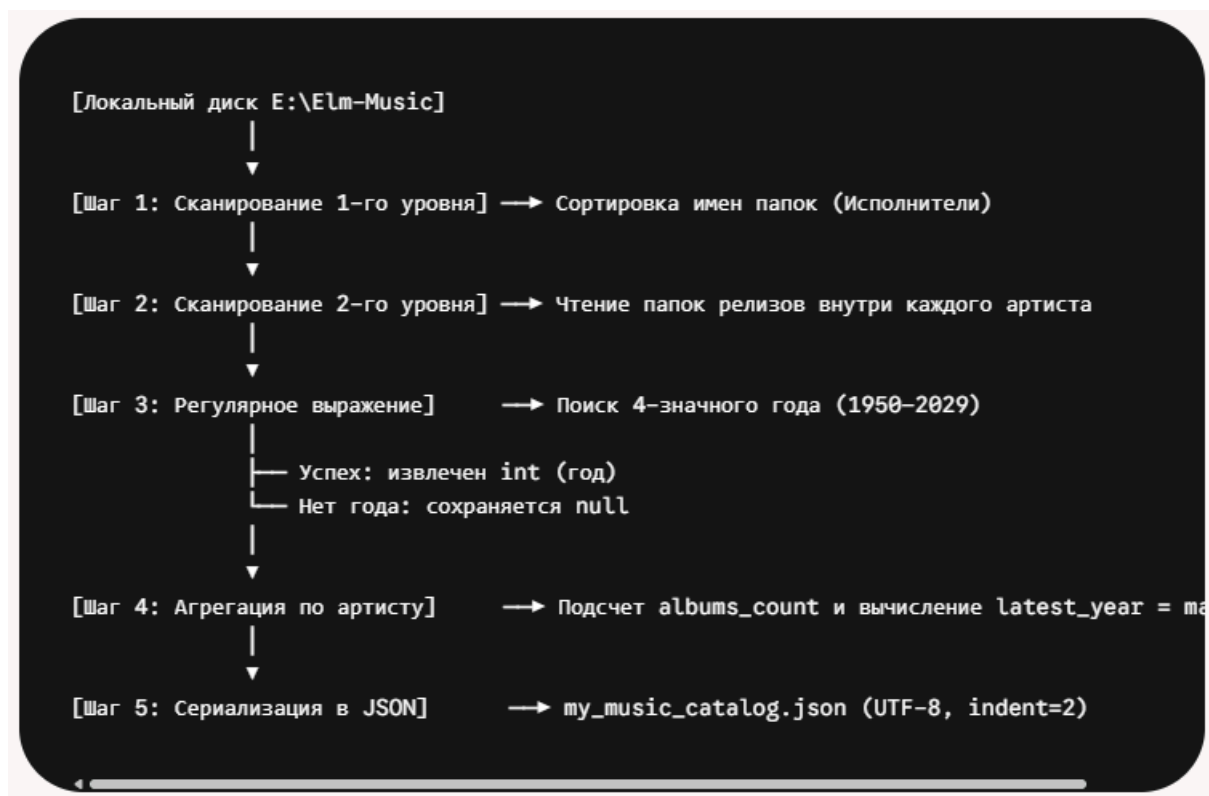


# Итоговый отчет: архитектура, функционал и эксплуатация системы мониторинга музыкальной коллекции

## 1. Принципы сборки каталога коллекции

Формирование файла `my_music_catalog.json` выполняется локальным скриптом `scan_catalog.py` непосредственно на вашем компьютере с физического накопителя `E:\Elm-Music`. Каталог служит эталонной базой («слепком») для удаленного сервера.



### 1. Последовательность работы сканера

#### 1. Сканирование верхнего уровня директорий:

Скрипт читает корневой каталог `E:\Elm-Music` и берет только папки первого уровня, считая имя каждой папки именем артиста или проекта (`artist`). Список исполнителей сортируется по алфавиту.

#### 2. Обход вложенных папок альбомов:

Внутри папки каждого артиста скрипт сканирует поддиректории следующего уровня, воспринимая каждую из них как отдельный релиз (`album_folder`). Файлы вне подпапок игнорируются.

### 3. Парсинг года выпуска релиза:

Имя папки альбома прогоняется через регулярное выражение `\b(19[5-9]\d|20[0-2]\d)\b`. Паттерн ищет отдельно стоящее четырехзначное число в диапазоне от **1950 по 2029** год:

- Если совпадение найдено — берется первое вхождение и приводится к целому числу (`int`).
- Если числа в диапазоне нет — в поле `year` записывается `null`.

### 4. Агрегация и вычисление контрольного года:

- Считается общее число распознанных папок (`albums_count`).
- Из всех распознанных годов альбомов выбирается наибольший: `latest_year = max(years)`. Если ни в одной папке год не найден, `latest_year` получает значение `null`.

### 5. Экспорт данных:

Итоговая структура записывается в файл `my_music_catalog.json` в кодировке UTF-8 с сохранением кириллицы (`ensure_ascii=False`).

## Что скрипт УЧИТЫВАЕТ:

- **Имена папок верхнего уровня** как точные идентификаторы артистов для запросов к API.
- **Любые форматы названий папок**, если в них присутствует год: `1985 - Hunting High And Low`, `2024 - Humanoid`, `Blues Power (2016)`, `1979-Accept (320)`.
- **Несколько релизов одного года**: все они фиксируются в массиве `albums`.

## Что скрипт НЕ УЧИТЫВАЕТ (и отбрасывает):

- **ID3-теги и метаданные треков**: парсинг идет строго по именам директорий, сами аудиофайлы (`.mp3`, `.flac`) не открываются, что обеспечивает сборку большого объема дискографий за пару секунд.
- **Глубокую вложенность (3-й уровень и глубже)**: диски внутри альбома (например, папки `CD1/`, `CD2/`) сканером не разбираются, если лежат внутри папки альбома.
- **Папки без года в имени**: папки вида `VOL 1`, `GOLD Complete Edition 2CD`, `Covers` или `Greatest Hits` сохраняются с `year: null` и не участвуют в вычислении `latest_year`.
- **Года до 1950 года без специального шаблона**: например, в архивных записях Эдит Пиаф папки с именами `1935–1949` скрипт оставил с `year: null`, так как регулярное выражение настроено на диапазон от 1950 года. Но такая музыка — это редкое исключение.
- **Тип релиза**: скрипт не различает, является ли папка студийным альбомом, синглом, сборником или бутлегом — фильтрация этого возложена на второй этап ([checker.py](#)).

## 2. Детальный разбор функционала скрипта `checker.py`

Скрипт `checker.py` — это серверный бэкенд на Python 3, выполняющий еженедельный аналитический опрос баз данных и доставку отчетов в Telegram.



## Пошаговая логика работы модулей

### 1. Чтение и валидация структуры

- Скрипт находит `my_music_catalog.json` в своей папке. При отсутствии файла отправляется экстренное сообщение в Telegram.
- Перебираются все артисты каталога.

- **Фильтрация заглушек:** если `latest_year` равен `null` (нет распознанных релизов, компиляции, лекции), артист пропускается без отправки запросов в сеть.

## 2. Поиск в MusicBrainz (`check_musicbrainz`)

- **Точечный запрос:** к <https://musicbrainz.org/ws/2/release-group> уходит поисковый запрос `artist:"<Имя>" AND type:album AND status:official`.
- **Отсечение мусора:** берутся только релизы, где `primary-type == "Album"`, а массив `secondary-types` пуст. Это автоматически отбрасывает:
  - Концертники (Live);
  - Сборники (Compilation, Best Of);
  - Синглы, макси-синглы и EP;
  - Ремиксы и трибьюты.
- **Сравнение годов:** из `first-release-date` берется год. Если он больше локального `latest_year`, альбом добавляется в список с пометкой `[MusicBrainz]`.
- **Rate Limit:** пауза `time.sleep(1.1)` соблюдает лимит сервиса (не чаще 1 запроса в секунду).

## 3. Поиск в Discogs (`check_discogs`)

- **Авторизация:** запрос идет к <https://api.discogs.com/database/search> с токеном `DISCOGS_TOKEN`.
- **Фильтрация мастер-релизов:** `type=master` схлопывает сотни переизданий альбома (на кассетах, CD, виниле в разных странах) в один корневой релиз, а `format=album` убирает синглы.
- **Очистка названия:** Discogs часто возвращает строку вида `Группа - Альбом`. Скрипт отрезает имя исполнителя, оставляя только название релиза.
- **Покрытие русскоязычного сегмента:** компенсирует слабую базу MusicBrainz по советскому року, бардам и российским изданиям. Добавляет релизы с пометкой `[Discogs]`.
- **Rate Limit:** пауза 1.1 с укладывается в лимит 60 запросов в минуту.

## 4. Дедупликация и сведение

- Списки объединяются через `list(dict.fromkeys(...))` — если альбом найден в обеих базах, дубликат исключается.

## 5. Отказоустойчивая отправка в Telegram (`send_telegram_message`)

- **Нарезка на фрагменты (Chunking):** отчет объемом более 4096 символов Telegram блокирует. Алгоритм делит длинный текст построчно на блоки до 3800 символов.
- **Защита от сетевых сбоев (Retry):** при временных разрывах связи на хостинге (`[Errno 101] Network is unreachable`) скрипт делает до 4 попыток отправки каждого блока с паузой в 3 секунды, исключая потерю страниц отчета.

## 3. Итоговая сводка выполненной работы

### 1. Серверная инфраструктура на хостинге

В директории `/home/.../username/music_bot/` развернуты два ключевых файла:

- `my_music_catalog.json` — реестр локальной фонотеки с диска `E:\Elm-Music` (с посчитанным числом дисков и предельным годом у каждого артиста).
- `checker.py` — управляющий скрипт еженедельной сверки.

### 2. Источники данных

- **MusicBrainz API** — основной открытый каталог для зарубежных исполнителей с фильтрацией официальных номерных LP.
- **Discogs API** — подключен по токenu разработчика для поиска отечественных и русскоязычных групп по базе мастер-релизов.
- Данные из обеих систем объединяются без дублирования.

### 3. Telegram-бот

- Задействован бот `@..._bot` для отправки отчетов в мой чат по личному идентификатору.
- Реализована верстка в HTML, нарезка сообщений на части и защита от кратковременных падений сети.

### 4. Автоматизация (Cron)

В планировщике Beget активно задание:

Скрипт стартует автономно в последнее воскресенье каждого месяца в 12:00 дня.

### 5. Результаты анализа 88 нестандартных папок

Все 88 папок без зафиксированных годов были разобраны вручную:

- **Сборники разных артистов (VA):** компиляции (*Cafe Enigma*, *Blues Guitar Duels*, *Crucial Blues*, *The Blues Collection* и др.) исключены из поиска персональных релизов.
- **Немузыкальные записи:** подкасты, лекции и радиопередачи (*Арзамас*, радиовыпуски *Аэроstat*, лекции Н. Басовской) выведены за рамки музыкального мониторинга.
- **Скрытые дискографии и опечатки:** найдены и зафиксированы пропущенные номерные релизы реальных артистов.

## 4. Регламент актуализации каталога

Для поддержания системы в актуальном состоянии требуется только одно действие при добавлении новой музыки на ПК:

1. Скачав новые альбомы на диск `E:\Elm-Music`, запускаем локально `scan_catalog.py`.
2. Загружаем обновленный `my_music_catalog.json` по FTP на хостинг в папку `/music_bot/`.

В последнее воскресенье следующего месяца бот учтет обновленные даты и продолжит искать только последующие новинки.