1. Сбор подробных требований

Текстовое описание

Функциональные требования (подробнее)

1. **Сбор и отслеживание данных**
    - **Основной источник**: ВКонтакте — сбор постов, видео и клипов.
    - **Возможность расширения**: подключение Instagram, Facebook и других соцсетей.
    - **Метрики**: количество постов, видео, клипов, лайков, репостов, комментариев (опционально).
    - **Частота сбора**: автоматическая или ручная с заданными интервалами.
2. **Управление источниками (пабликами)**
    - **Добавление/удаление** пабликов: пользователь вводит ID или ссылку на паблик.
    - **Проверка уникальности**: чтобы исключить дублирование.
    - **Настройки фильтрации**:
        - по типам контента (посты, видео, клипы и т.д.),
        - по дате или временным интервалам (день, неделя, месяц).
3. **Автоматизация и планирование**
    - **Планировщик задач** (например, APScheduler или Celery beat):
        - обновление данных раз в 24 часа,
        - автоматическое формирование отчётов в 10:00 по МСК,
        - ручной запуск в любой момент.
    - **Фоновые задачи**: для сборки статистики, отправки уведомлений, рассылки отчётов.
4. **Личные кабинеты пользователей и управление ролями**
    - **Регистрация и авторизация**:
        - подтверждение по email или через соцсети,
        - возможность включения двухфакторной аутентификации.
    - **Личный кабинет**:
        - список отслеживаемых пабликов,
        - настройки сбора (выбор типов контента, фильтров),
        - экспорт и просмотр отчётов.
    - **Роли**:
        - *Пользователь (SMM-аналитик)*: видит только свои данные,
        - *SMM-директор*: сводка по нескольким пользователям,

        - *Администратор*: полное управление системой (пользователи, настройки, логи и т.д.).

5. **Формирование и экспорт отчётов**
    - **Автоматическая генерация**:
        - ежедневные отчёты (каждое утро в 10:00 по МСК),
        - опционально — еженедельные и месячные.
    - **Содержание**:
        - детальная статистика по каждому паблику,
        - сводные данные по всем источникам,
        - наглядные графики и диаграммы.
    - **Экспорт**:
        - Excel, CSV,
        - автоматическая рассылка на email.
6. **Уведомления и оповещения**
    - **Система уведомлений**:
        - email, SMS или внутренняя система,
        - триггеры при превышении порогов (например, 100 постов в день),
        - индивидуальные настройки уведомлений.

Нефункциональные требования (подробнее)

1. **Технологический стек**
    - Python (актуальные версии, поддержка async при необходимости).
    - Фреймворки для веб-сервиса: FastAPI или Django (на твой выбор).
    - Планировщики: Celery (с Redis или RabbitMQ) или APScheduler.
    - Клиентская часть:
        - мобильное приложение на Flutter,
        - веб-интерфейс (например, на React/Vue или на базе Django/FastAPI-шаблонов).
2. **База данных**
    - PostgreSQL:
        - хранение данных о пользователях, ролях, пабликах, статистике, логах и т.д.
        - важна стабильность и масштабируемость.
3. **Масштабируемость и надежность**
    - **Docker** для контейнеризации приложения.

- **Горизонтальное масштабирование**: разделение по микросервисам (по желанию), клонирование инстансов.
- **Резервное копирование** БД на регулярной основе.
- **Мониторинг** работы (например, Prometheus + Grafana).

4. **Безопасность**
   - **Шифрование** паролей (bcrypt или argon2).
   - **Защита** API-ключей.
   - **Логирование действий** (аудит).
   - **Регулярный аудит** уязвимостей (например, проверка зависимостей).
5. **Удобство использования и документация**
   - **UI/UX**:
     - адаптивный дизайн (Flutter, веб),
     - интуитивная навигация.
   - **Документация** для разработчиков (техническая) и для пользователей (руководство).
   - **Комментарии в коде** (на русском языке) при необходимости.

Таблица с требованиями

Ниже сводная таблица, где кратко указано каждое требование, его тип и основные детали.

Требование	**Тип**	**Детали**	**Польза**
Сбор данных из ВК	Функциональное	Использовать официальное API ВКонтакте, собирать посты, видео, клипы	Автоматизация работы SMM-специалистов
Расширение на другие соцсети	Функциональное	Возможность подключить Instagram, Facebook и т.д.	Гибкость и масштабируемость
Метрики вовлеченности	Функциональное	Лайки, репосты, комментарии (по желанию)	Глубокий анализ охвата
Добавление/удаление пабликов	Функциональное	Проверка уникальности, ввод ID/ссылки	Удобство управления источниками
Фильтрация по типу контента и дате	Функциональное	Настройки (посты, видео, период: день/неделя/месяц)	Точность и релевантность данных

Планировщик задач	Функциональное	Автоматическое обновление, формирование отчётов, ручной запуск	Сокращение ручного труда
Личные кабинеты с разными ролями	Функциональное	Роли: Аналитик, Директор, Админ; личный кабинет с настройками	Гибкое разграничение прав, персонализация
Формирование отчётов (ежедневно, еженедельно, ежемесячно)	Функциональное	Детальные и сводные отчёты, графики, таблицы, экспорт в Excel/CSV	Удобство аналитики и командной работы
Уведомления и оповещения	Функциональное	Email, SMS, внутренняя система, настройка пороговых значений	Оперативное реагирование
Технологический стек (Python, Docker, PostgreSQL)	Нефункциональное	Разработка backend на Python, хранение в PostgreSQL, деплой в Docker	Стабильность, масштабируемость
Масштабируемость, надёжность	Нефункциональное	Горизонтальное масштабирование, бэкапы БД, логирование, мониторинг	Высокая доступность и защита от сбоев
Безопасность	Нефункциональное	Шифрование паролей, защита ключей, аудит	Сохранность данных и доверие пользователей
Интуитивный интерфейс	Нефункциональное	Простая и понятная навигация, адаптивность (Flutter, веб)	Удобство использования
Документация	Нефункциональное	Руководство для пользователей, техническая документация для разработчиков	Облегчение внедрения и сопровождения проекта

Вывод по сбору требований

1. Мы определили основные функциональные и нефункциональные требования, которые помогут создать удобную и полезную систему для анализа активности в соцсетях.
2. Главные акценты: гибкость настройки источников, автоматизация сборки данных, удобная визуализация и экспорт отчётов.

3. Масштабируемость и безопасность обеспечиваются современными технологиями (Docker, PostgreSQL, шифрование), что позволит системе работать с ростом числа пользователей и объёмом данных.

---

2. Анализ целевой аудитории

Текстовое описание

Аудитория системы состоит из нескольких категорий, каждая со своими задачами и приоритетами:

1. **SMM-аналитики и маркетологи**
    - Основная цель: получать детальные отчёты и статистику по активности в пабликах.
    - Важны автоматические отчёты, чтобы экономить время на рутинных сборках данных.
    - Фильтры и метрики нужны для глубокой аналитики и понимания вовлечённости.
2. **SMM-директора и руководители**
    - Им важны сводные отчёты по нескольким аналитикам или по нескольким проектам одновременно.
    - Нужно быстро оценивать общую картину, смотреть тренды и динамику.
    - Возможность управлять ролями и иметь общий контроль.
3. **Администраторы системы**
    - Отвечают за стабильность и безопасность.
    - Управляют доступами, резервными копиями и настройками системы.
    - Важно иметь удобные инструменты мониторинга и логи, чтобы оперативно реагировать на сбои.
4. **Владельцы бизнеса (опционально)**
    - Могут просматривать только верхнеуровневую аналитику.
    - Нужны наглядные графики и простые отчёты, понятные без технических знаний.

Таблица с анализом целевой аудитории

Группа пользователей	**Цели/задачи**	**Что важно**	**Как решается в системе**
SMM-аналитики, маркетологи	Собирают статистику, анализируют эффективность публикаций	Глубокие метрики, графики, удобные фильтры	Персональный кабинет, фильтры, ежедневные отчёты
SMM-директора, руководители	Сводные отчёты, мониторинг нескольких проектов и команд	Высокоуровневые отчёты, контроль показателей	Ролевая система, сводные отчёты, удобная панель управления
Администраторы системы	Техническое сопровождение,	Управление пользователями,	Отдельный интерфейс, логи,

	безопасность, стабильность	логирование, мониторинг	система уведомлений, бэкапы
Владельцы бизнеса	Общая картина продвижения, бюджетирование	Наглядность, простота отчётов	Связанные отчёты с ключевыми показателями

Вывод по анализу аудитории

1. У каждой группы есть свои приоритеты и задачи, однако SMM-аналитики и руководители — ключевые пользователи, для которых создаётся основной функционал (сбор статистики, отчёты, управление пабликами).
2. Администраторам важны стабильность и безопасность, поэтому обязательно нужны инструменты для резервного копирования, логирования и быстрого реагирования на ошибки.
3. Возможность подключать разных пользователей с разными правами доступа делает систему универсальной для компаний любого масштаба.

---

3. Архитектура системы

Выбор технологического стека

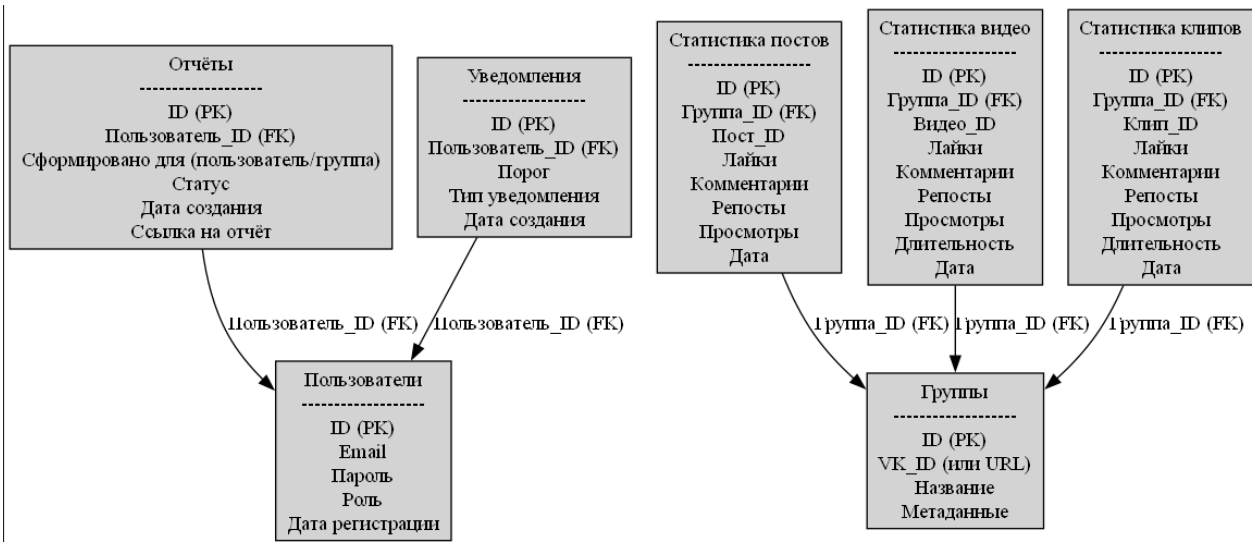
1. **Язык программирования**: Python
    - Быстро развивающаяся экосистема, множество библиотек для работы с API и статистикой.
    - Удобен для написания микросервисов и REST API.
2. **Веб-фреймворк**: FastAPI или Django
    - *FastAPI* — современный, лёгкий, быстрая разработка, хорошая производительность (async).
    - *Django* — более «тяжёлый», но из коробки предлагает админку, ORM, работу со схемами.
3. **База данных**: PostgreSQL
    - Надёжная реляционная СУБД, отлично подходит для хранения структурированных данных и сложных запросов.
    - Хорошо масштабируется и работает в Docker-среде.
4. **Планировщики и фоновые задачи**:
    - **Celery** с брокером (RabbitMQ или Redis), либо **APScheduler**.
    - Позволяют запускать задачи (сбор статистики, отправку писем, генерацию отчётов) по расписанию и в фоне.
5. **Контейнеризация**: Docker
    - Упрощает деплой и масштабирование.
    - Можно быстро разворачивать идентичные окружения.
6. **Фронтенд**:
    - **Flutter** для мобильных приложений.
    - **Веб-часть** может быть на React/Vue или встроенных шаблонах (в зависимости от предпочтений и возможностей команды).

Структура базы данных (общие моменты)

- **Таблица users**: хранит данные пользователей (email, пароль, роль, дата регистрации и т.д.).
- **Таблица groups** (или pages): хранит список отслеживаемых пабликов (ID или URL из ВК, название, метаданные).
- **Таблица statistics**: данные о постах, видео, клипах, лайках, комментариях и т.д. Возможно, дробим её на несколько таблиц для разных типов контента.
- **Таблица reports**: ссылки на сформированные отчёты, статусы генерации, время, пользователь, для кого сформирован.

- **Таблица notifications**: информация об уведомлениях (порог, тип уведомления, пользователь).

Для крупных систем можно разбивать статистику на отдельные таблицы, например stats_posts, stats_videos, stats_clips и т.д. — это упростит работу с большими данными.



Примерная иллюстрация структуры базы данных (не строгая схема)

Архитектура серверного приложения

1. **Сервис аутентификации**: отдельная часть приложения, отвечающая за регистрацию, логин, токены (JWT), двухфакторную аутентификацию (при необходимости).
2. **Сервис сбора данных**: модуль/микросервис, который обращается к API ВКонтакте, сохраняет данные о постах, видео и клипах. Работает в фоне по расписанию.
3. **Сервис формирования отчётов**:
    - Генерирует отчёты на основе собранных данных,
    - Раз в сутки формирует ежедневные отчёты (в 10:00),
    - По запросу формирует недельные или месячные отчёты,
    - Формирует файлы Excel/CSV и отправляет по email (или доступ через личный кабинет).
4. **Сервис уведомлений**:
    - Отслеживает пороговые значения публикаций/вовлеченности,
    - Шлёт уведомления (email/SMS/Push) при достижении порога,
    - Индивидуальные настройки для каждого пользователя.
5. **Административная панель**:
    - Управление пользователями, ролями, логирование действий, просмотр системных метрик,

- Настройка расписаний, обновлений, резервных копий.

Эти сервисы могут быть объединены в одном монолитном приложении (с разными модулями) или разделены на микросервисы (например, при большой нагрузке).

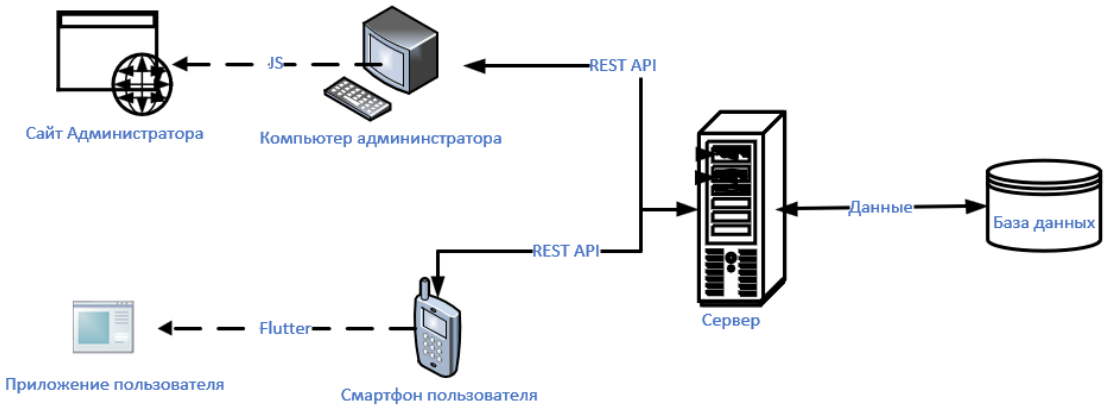


Рисунок модель архитектуры системы

Общая схема взаимодействия

1. **Пользователь (Аналитик)** через мобильное приложение (Flutter) или веб-интерфейс заходит в систему → Авторизуется → Настраивает паблики, типы контента.
2. **Фоновый сервис (Сбор данных)** регулярно (или при ручном запуске) обращается к API ВКонтакте → Сохраняет статистику в PostgreSQL.
3. **Сервис формирования отчётов** в 10:00 (или по требованию) берёт нужные данные из БД → Генерирует отчёт → Сохраняет результат (файл) → Отправляет уведомление пользователю.
4. **Пользователь** получает уведомления, может зайти в личный кабинет и скачать Excel/CSV-отчёт или просмотреть графики в интерфейсе.
5. **Администратор** имеет доступ к управлению пользователями, логам, системе уведомлений. При сбоях может просмотреть логи и перезапустить сбор данных.

Вывод по архитектуре

1. Архитектура предполагает использование современных инструментов (Docker, Celery/APScheduler, PostgreSQL, Python-фреймворки), что позволит системе обрабатывать большой объём данных и легко интегрироваться с другими сервисами.
2. Разделение сервисов по модулям (аутентификация, сбор данных, отчёты, уведомления) обеспечивает гибкость и упрощает дальнейшее развитие.
3. Контейнеризация даёт возможность быстро масштабировать систему на разных серверах или облачных платформах.
