Техническое задание на разработку MVP Telegram-бота

Influence-платформа для TikTok-авторов

Тип проекта	MVP
Основная платформа	Telegram
Основная соцсеть MVP	TikTok
Backend	Python + aiogram 3.x + PostgreSQL
Референс	@playerokclips_bot

1. Цель проекта

Разработать Telegram-бота для авторов TikTok-контента, которые участвуют в рекламных промоакциях, публикуют видео и получают вознаграждение в зависимости от фактического количества просмотров.

Основной пользовательский сценарий:

> Регистрация - привязка TikTok - выбор промо-акции - публикация видео - отправка на модерацию - одобрение - автоматический мониторинг просмотров - CPM-начисления - баланс - заявка на вывод USDT.

Администрирование первой версии предполагается выполнить внутри Telegram-бота. Но если рациональнее и удобнее выполнить администрирование через web-admin в MVP - это будет большим плюсом.

2. Технологический стек

- Python 3.12+
- Flask or Django для админ - панели управления настройками системы
- aiogram 3.x
- PostgreSQL
- SQLAlchemy 2.x / async ORM
- Alembic для миграций
- Redis - желательно для background jobs, locks и cache
- APScheduler / Celery / ARQ или аналогичный background worker
- Docker / Docker Compose
- Linux VPS

Допускается изменение отдельных библиотек после согласования, если сохраняется масштабируемая архитектура и возможность дальнейшего расширения.

3. Архитектурные требования

Проект должен быть разделён минимум на следующие логические модули:

- Telegram handlers
- application/service layer
- database layer
- TikTok integration service
- statistics/background workers
- financial/accrual service
- admin module через - веб-панель
- configuration
- logging
- migrations

Бизнес-логика начислений не должна находиться непосредственно внутри Telegram handlers. Интеграция TikTok должна быть изолирована за provider/adapter интерфейсом.

4. Этап 1 - TikTok Statistics PoC

Первый этап **обязателен** до реализации полной системы. Готового оплаченного сервиса/API у нас на данный момент нет. Исполнитель самостоятельно подбирает **стабильный** способ получения статистики TikTok; стоимость внешних API, proxy, VPS и других сервисов оплачивается заказчиком отдельно после предварительного согласования.

4.1. Задачи PoC

- Исследовать 2-3 варианта получения статистики: официальный API (если применим), сторонний TikTok API/data provider, внешний scraping provider или собственный парсинг при техническом обосновании.
- На нескольких реальных публичных TikTok URL получить video ID, URL, username автора, количество просмотров и доступные metadata.
- Повторно запросить те же видео и подтвердить корректное обновление view_count.
- Проверить rate limits, стоимость, стабильность, необходимость proxy, региональные ограничения, обработку private/deleted video и ошибочных URL.
- Подготовить краткую рекомендацию по production-варианту.

4.2. Adapter

```
TikTokProvider
    get_profile(...)
    verify_account(...)
    get_video(...)
    get_video_stats(...)
```

Основная бизнес-логика не должна зависеть от конкретного провайдера. Замена TikTok provider не должна требовать переписывания CPM (системы начисления за просмотры), финансовой логики или админ-модуля.

4.3. Критерий успешности PoC

- По реальному TikTok URL стабильно возвращается view_count.
- Повторный запрос возвращает актуальное значение.
- Определены стоимость и ограничения production-варианта.
- Выбран и согласован production-подход.

5. Роли пользователей

5.1. Автор

- Просмотр профиля
- Привязка TikTok
- Просмотр и участие в доступных промоакциях
- Отправка видео на модерацию
- Просмотр статуса видео, просмотров и заработка
- Добавление/изменение USDT TRC20 кошелька
- Создание заявки на вывод

5.2. Администратор

- Управление пользователями

- Ban/unban
- Просмотр TikTok-аккаунтов
- Создание и редактирование промоакций
- Модерация видео
- Остановка/возобновление начислений
- Просмотр статистики и начислений
- Обработка заявок на вывод

6. Регистрация и профиль

При первом /start создаётся пользователь. Повторный /start не должен создавать дубликат.

```
User:
id
telegram_id
telegram_username
first_name
last_name
registered_at
status
is_banned
```

В интерфейсе отображаются current balance, frozen balance, total earned и total paid. Источником истины по финансовым операциям должен быть ledger, а не только изменяемое поле balance.

7. Привязка TikTok

1. Пользователь вводит username или URL TikTok-профиля.
2. Система генерирует уникальный verification code, например WS-7F4K92.
3. Пользователь временно размещает код в bio TikTok.
4. После нажатия «Проверить» backend проверяет bio через TikTok provider.
5. При успешной проверке аккаунт получает статус verified и сохраняется verified_at.
6. После подтверждения пользователь может удалить код из bio.

```
TikTokAccount:
id
user_id
username
profile_url
platform_user_id
verification_code
verification_status
verified_at
last_checked_at
```

Не допускать привязку одного и того же TikTok-аккаунта к нескольким пользователям без отдельного административного решения.

8. USDT-кошелёк

В MVP поддерживается USDT TRC20. Пользователь может добавить и изменить кошелёк. Выполняется базовая валидация формата Tron address.

При создании withdrawal request адрес кошелька копируется в заявку, чтобы последующая смена кошелька пользователя не изменила уже созданную заявку.

9. Промоакции

```
Campaign:
id
name
description
status
start_at
end_at
current_cpm
campaign_budget
per_video_payout_cap (nullable)
currency
terms
materials
created_at
updated_at
```

Минимальные статусы: DRAFT, ACTIVE, PAUSED, COMPLETED.

10. CPM (система начисления) и ее правила

CPM - это определенный показатель, а точнее стоимость за 1000 новых просмотров.

```
earning = new_views / 1000 * current_cpm
```

Согласованное правило: администратор может изменить CPM активной кампании. Новая ставка применяется только к новым просмотрам после изменения. Уже начисленные средства не пересчитываются.

```
Пример:
0 - 10 000 views при CPM $2.00 = $20.00
CPM изменён на $1.50
10 000 - 14 000 views = 4 000 новых views
4 000 / 1000 * $1.50 = $6.00
Итого заработано: $26.00
```

11. Immutable accruals и защита от повторного начисления

Каждое начисление создается отдельной неизменяемой записью. Нельзя рассчитывать общий заработок как current_views * current_CPM.

```
Accrual:
id
user_id
video_id
campaign_id
views_from
views_to
views_delta
cpm
amount
created_at
```

После создания Accrual не пересчитывается.

Для каждого видео хранится last_processed_views. При новом snapshot:

```
new_views = current_views - last_processed_views
if new_views <= 0:
    accrual не создаётся
```

После успешной транзакции last_processed_views обновляется до current_views. Необходимо исключить двойное начисление при параллельной работе workers через DB row locking, advisory lock или Redis lock.

12. Лимиты кампании и видео

12.1. Campaign Budget Cap

Для каждой промоакции задаётся общий максимальный бюджет. Суммарные начисления всех авторов не могут превысить campaign_budget.

Если остаток бюджета меньше рассчитанного нового начисления, создаётся partial final accrual только на доступный остаток, после чего дальнейшие начисления кампании прекращаются.

12.2. Per-video payout cap

Для кампании может быть задан опциональный максимальный payout на одно видео. После достижения лимита статистика просмотров продолжает обновляться, но новые финансовые начисления по этому видео не создаются.

Если одновременно действует общий campaign budget и per-video cap, начисление ограничивается первым достигнутым лимитом.

12.3. Actual views и paid views

Система должна различать фактические просмотры и просмотры/сумму, по которым было произведено начисление. Достижение payout cap не должно останавливать сбор фактической статистики.

13. Видео и модерация

```
Video:
id
user_id
campaign_id
tiktok_url
platform_video_id
status
submitted_at
approved_at
rejected_at
rejection_reason
baseline_views
current_views
last_processed_views
accrual_enabled
total_earned
```

Статусы минимум: PENDING, APPROVED, REJECTED, STOPPED.

Администратор видит ссылку, автора, TikTok account, кампанию и доступные metadata. Действия: «Одобрить» / «Отклонить». При отклонении причина обязательна, она отправляется через бота пользователю в аккаунт Telegram.

До статуса APPROVED финансовые начисления не выполняются.

14. Baseline views

Рекомендуемое правило MVP: оплачиваемыми считаются только просмотры после момента APPROVED. При одобрении фиксируется baseline_views и last_processed_views равный текущему количеству просмотров. Просмотры, полученные до одобрения, не оплачиваются.

Этот пункт должен быть окончательно подтверждён заказчиком перед разработкой.

15. Background statistics worker

Approved-видео периодически обрабатываются worker-ом:

7. Получить активные видео.
8. Запросить TikTok provider.
9. Получить current views.
10. Сохранить snapshot.
11. Рассчитать delta относительно last_processed_views.
12. Проверить ban пользователя, статус кампании, accrual_enabled, campaign budget и video cap.
13. Создать Accrual и ledger transaction.
14. Транзакционно обновить last_processed_views.

Частота polling должна задаваться конфигурацией из .env, например TIKTOK_STATS_INTERVAL_MINUTES=30, а не быть зашита в код.

16. История статистики

```
VideoStat:
id
video_id
views
fetched_at
provider
```

Минимальная история snapshots обязательна для аудита и разбора спорных начислений.

17. Финансовый Ledger

Рекомендуемые типы операций:

- ACCRUAL
- WITHDRAWAL_RESERVE
- WITHDRAWAL_RELEASE
- PAYOUT
- ADMIN_ADJUSTMENT

```
LedgerTransaction:
id
user_id
type
amount
reference_type
reference_id
created_at
comment
```

Денежные расчёты выполняются только через Python Decimal и PostgreSQL NUMERIC. Использование float для денег запрещено.

18. Балансы

Разделить available_balance, frozen_balance, total_earned и total_paid.

```
Создание withdrawal:
available -= amount
frozen += amount

Подтверждение выплаты:
frozen -= amount
total_paid += amount

Отказ:
frozen -= amount
available += amount
```

19. Вывод средств

```
Withdrawal:
id
user_id
amount
wallet
network = TRC20
status
created_at
processed_at
admin_comment
```

Статусы: PENDING, PAID, REJECTED.

Минимальная сумма вывода должна быть настраиваемой. В MVP автоматическая отправка криптовалюты не реализуется: администратор выполняет перевод вручную и отмечает заявку как PAID либо REJECTED с причиной.

20. Admin Dashboard (web - panel)

20.1. Пользователи

- Список/поиск
- Профиль
- TikTok account
- Баланс и total earned
- Withdrawal history
- Ban/unban

20.2. Видео

- Фильтрация по статусам
- Просмотры
- Заработок
- Кампания
- Пользователь
- Stop accrual / Resume accrual

20.3. Кампании

- Создать/изменить
- Pause/Complete
- Изменить CPM
- Изменить общий budget
- Изменить per-video cap
- Описание/условия/материалы/сроки

20.4. Withdrawals

- Очередь PENDING
- Просмотр пользователя, суммы, TRC20 wallet и даты
- PAID
- REJECTED с причиной

21. Ban и остановка начислений

При ban пользователь не может пользоваться ботом, его история и баланс не удаляются, новые начисления должны быть остановлены. Возможность выплаты существующего баланса banned-пользователю остаётся административным решением.

22. Ошибки TikTok API

При временной ошибке provider запрещено обнулять views. Сохраняется последнее успешное значение, ошибка логируется, запрос повторяется позже с controlled retry/backoff.

При private/deleted/not found видео запись не удаляется автоматически. Видео получает технический статус/ошибку needs_review, администратор уведомляется.

23. Безопасность

- Secrets только в .env; .env не хранится в Git.
- Admin access через web - интерфейс по логину и паролю.
- ORM/input validation для защиты от некорректных данных.
- Проверка TikTok URL.
- Rate limiting критичных действий.
- Webhook secret при использовании Telegram webhook.
- Не логировать API keys, bot token и другие secrets.
- Не выводить полный USDT wallet в технические логи без необходимости.

24. Нагрузка и фоновые задачи

Ориентир исходного ТЗ - до 500 одновременных запросов. Это не означает 500 одновременных запросов к TikTok API.

Statistics worker должен учитывать rate limits провайдера и использовать queue, concurrency limits и batching. Telegram interactions не должны блокироваться TikTok polling.

25. Docker и конфигурация

Проект должен запускаться через Docker Compose.

```
Минимальные сервисы:
bot
worker
```

```
postgres
redis
```

Пример .env.example:

```
BOT_TOKEN=
ADMIN_IDS=
DATABASE_URL=
REDIS_URL=
TIKTOK_PROVIDER=
TIKTOK_API_KEY=
TIKTOK_STATS_INTERVAL_MINUTES=
MIN_WITHDRAWAL=
LOG_LEVEL=
```

Передаётся .env.example без секретов и инструкция запуска/обновления.

26. Логирование

- bot/worker start
- TikTok fetch errors
- campaign processing errors
- financial processing errors
- withdrawals
- administrative actions

Логи и структура данных должны позволять восстановить причину конкретного финансового начисления.

27. Основные сущности БД

- users
- tiktok_accounts
- campaigns
- videos
- video_stats
- accruals
- ledger_transactions
- withdrawals
- admin_actions

При необходимости допускаются campaign_materials, provider_errors и дополнительные служебные таблицы.

28. Тестирование

28.1. Пользователь

- /start
- повторный /start
- TikTok verification success/failure
- добавление/изменение wallet

28.2. Видео и статистика

- valid/invalid TikTok URL
- approval/rejection
- views increased/unchanged

- API timeout/error
- deleted/private video

28.3. Финансы

- CPM $2 для первых 10 000 views и $1.50 для следующих 4 000 - итог $26.
- Повторная обработка одного snapshot не создаёт второго начисления.
- Campaign budget cap.
- Per-video cap.
- Withdrawal: enough balance / insufficient / below minimum / reserve / reject / payout.

29. Acceptance Criteria MVP

15. Пользователь регистрируется через Telegram.
16. Пользователь может подтвердить TikTok account.
17. Пользователь может указать USDT TRC20 wallet.
18. Пользователь видит активные промоакции.
19. Пользователь может отправить TikTok video.
20. Администратор через веб-интерфейс может approve/reject видео.
21. Approved video автоматически отслеживается.
22. Views периодически обновляются и история сохраняется.
23. Начисляется только delta новых views.
24. CPM (система начисления) сохраняется у каждого начисления.
25. Старые начисления не пересчитываются после изменения CPM.
26. Campaign budget cap работает.
27. Per-video cap работает.
28. Double accrual исключён.
29. Пользователь видит balance и earnings в своем ТГ-аккаунте.
30. Withdrawal резервирует/замораживает сумму.
31. Администратор вручную подтверждает/отклоняет withdrawal.
32. Финансовая история сохраняется.
33. Администрирование MVP выполняется внутри веб-интерфейса.
34. Проект развёрнут на VPS через Docker.
35. Передана инструкция запуска и конфигурации.

30. Передача результата

- Полный исходный код в Git.
- Docker/Docker Compose конфигурация.
- Миграции БД.
- .env.example без секретов.
- Инструкция по запуску и обновлению.
- Перечень внешних сервисов/API и их назначение.
- Развёрнутый рабочий MVP на согласованном VPS.