Тестовое задание: Middle+/Senior Backend Developer (Python + FastAPI + Streaming)

Описание задания

Разработать API для мини-сервиса по управлению видеопотоками. Основные задачи включают загрузку информации о видео, генерацию HLS-плейлиста, кеширование данных в Redis и асинхронную обработку событий через background-задачи.

Стек технологий

- Python 3.10+
- FastAPI
- PostgreSQL
- Redis
- Docker
- Async queue (например, Celery, RQ, или встроенный BackgroundTasks)

Функциональные требования

1. API для загрузки информации о видео

Создать эндпоинт для добавления нового видео.

Запрос:

```
POST /api/videos
Content-Type: application/json
```

```
{
  "title": "Football Match",
  "url": "http://example.com/video.mp4"
}
```

Ответ:

```
{
  "id": 1,
  "title": "Football Match",
  "playlist_url": "http://example.com/hls/1.m3u8"
}
```

2. API для получения информации о видео

Создать эндпоинт для получения данных о видео. Данные должны кешироваться в Redis.

Запрос:

```
GET /api/videos/{id}
```

Ответ:

```
{
  "id": 1,
  "title": "Football Match",
  "playlist_url": "http://example.com/hls/1.m3u8"
}
```

3. Генерация HLS-плейлиста

При загрузке нового видео необходимо сгенерировать HLS-плейлист (`.m3u8`) с несколькими фейковыми сегментами.

4. Асинхронная обработка

Событие о загрузке видео должно обрабатываться в фоновом процессе: генерация HLS, запись в кеш, логгирование.

Нефункциональные требования

- Чистый, читаемый и структурированный код.
- Использование Docker для локального запуска (обязательно).
- Асинхронная реализация FastAPI.
- Наличие README с инструкциями по запуску.
- Пример `.env` файла и миграции базы данных (например, Alembic).
- Unit-тесты приветствуются.

Формат сдачи

- Репозиторий на GitHub/GitLab с README, содержащим инструкции по запуску.
- Код должен быть легко запускаемым на локальной машине через Docker Compose.

Время на выполнение: до 4 часов.