Техническая спецификация

Сервис lipsync-видеоаватара для ИИ-ассистентов (MuseTalk 1.5)

Версия: 1.0

Дата: 12.12.2025

Основание: АВР за периоды 01.10.2025 - 30.11.2025 (состояние на 01.12.2025).

1. Назначение и область применения

Сервис предназначен для генерации видеоаватар-ответов с синхронизацией губ и мимики (lip-sync) для ИИ-ассистентов Заказчика. Сервис принимает текстовый запрос, синтезирует речь через внешний TTS и формирует видеопоток в формате HLS (fMP4), пригодный для встраивания в веб-клиент.

1.1. Входные данные

- Текст запроса (строка).
- Параметры языка (русский/казахский/английский).

1.2. Выходные данные

- HLS-плейлист playlist.m3u8 и набор файлов потока (init.mp4 + сегменты segment_XXX.m4s).
- URL плейлиста, связанный с уникальной сессией генерации.

1.3. Границы текущего объёма

Глубокая интеграция с фронтендом (двойной прокси, кеширование, Range-запросы и т. д.) относится к работам после 01.12.2025 и не включена в данную спецификацию.

2. Термины и сокращения

Термин	Описание
HLS (fMP4)	HTTP Live Streaming с сегментами в формате fragmented MP4 (init.mp4 + *.m4s).
Сессия	Единица обработки одного запроса генерации, идентифицируется session_id.
TTS	Text-to-Speech, внешний сервис синтеза речи по тексту.
MuseTalk 1.5	Движок lipsync-инференса (UNet + VAE).
Whisper/AudioProcessor	Компонент извлечения аудио-признаков для инференса.

3. Стек и зависимости

- Python (виртуальная среда).

- Flask (HTTP API, app.py).
- PyTorch с поддержкой CUDA (инференс на GPU).
- MuseTalk 1.5 (конфигурация и веса модели).
- Whisper/AudioProcessor (извлечение аудио-признаков).
- FFmpeg (сборка и кодирование итогового видеопотока).
- Кодеки: видео H.264, аудио AAC.

При наличии GPU используется кодирование h264_nvenc; при отсутствии или перегрузке GPU используется fallback на libx264.

4. Функциональные требования

4.1. Генерация lipsync-видео

Сервис должен:

- Принимать текст запроса и параметры языка через HTTP API.
- Вызывать внешний TTS-сервис, сохранять результат как tts.mp3 в директорию data/audio/.
- Запускать аудиопайплайн (Whisper/AudioProcessor) для получения аудио-признаков.
- Выполнять инференс MuseTalk 1.5 и формировать последовательность кадров лица.
- Накладывать сгенерированное лицо на базовые кадры аватара с использованием маски и bounding box.
- Собирать видеоряд и аудио в HLS (fMP4) и сохранять в static/hls/{session}/.

4.2. Управление сессиями и статусами

- При старте генерации создавать новую сессию и возвращать session_id и playlist_url.
- Обеспечивать эндпоинт статуса сессии: created / processing / ready / error.
- Возвращать процент прогресса и информацию о готовности плейлиста.

5. Нефункциональные требования и совместимость

5.1. Совместимость HLS с веб-клиентом

- HLS-плейлист должен содержать теги #EXT-X-MAP, #EXT-X-TARGETDURATION, #EXT-X-ENDLIST.
- Сегментация: init.mp4 и сегменты segment_XXX.m4s длительностью около 1 секунды.
- Кодеки: видео H.264, аудио AAC.
- Должны быть настроены CORS-заголовки для работы из браузера.

5.2. Надёжность и обработка ошибок

- Ошибки TTS (таймауты, пустой ответ) должны приводить к завершению сессии со статусом error и понятным текстом ошибки.
- Ошибки инференса (несоответствие размеров тензоров, нехватка VRAM) должны логироваться и завершать сессию безопасно.

6. Архитектура и поток обработки

Логический пайплайн сервиса:

- TTS -> извлечение аудио-признаков (Whisper/AudioProcessor) -> MuseTalk (UNet + VAE) -> HLS-стриминг (FFmpeg).
- Роли/подсистемы: препроцессинг видео, инференс, сборка потока, HTTP API.

6.1. Разделение ответственности в приложении

- Синхронная часть API: приём запроса, валидация, создание сессии, возврат session_id.
- Асинхронные задачи: TTS, инференс, кодирование и сборка HLS.

7. Препроцессинг видеоаватара и кеширование

Используется базовое видеоаватара: data/video/IMG_0006_30s.MOV.

- Нарезка видео на кадры с заданным FPS.
- Детекция лица, построение маски и bounding box области лица/губ.
- Вычисление и кеширование VAE-латентов базового видео для ускорения инференса.

7.1. Директории и артефакты

- results/webapp/avatar/ - подготовленные кадры аватара.
- results/webapp/prep_cache/ - кеш VAE-латентов (*.pt).
- data/audio/ - аудио TTS и тестовые аудио.
- static/hls/{session}/ - результаты HLS по каждой сессии.

8. Инференс и композитинг

- Получение аудио-признаков из tts.mp3 (Whisper/AudioProcessor).
- Подача аудио-признаков и закешированных латентов в UNet MuseTalk.
- Декодирование выходных латентов через VAE в кадры лица.
- Наложение лица на исходные кадры с учётом маски и bbox.

9. Сборка потока и параметры кодирования

- Передача кадров и аудио в FFmpeg через stdin.
- Видео-кодек: h264_nvenc (предпочтительно) или libx264 (fallback).
- Аудио-кодек: AAC.
- Выходной формат: HLS (fMP4) с init.mp4 и сегментами segment_XXX.m4s.

Значение #EXT-X-TARGETDURATION должно соответствовать фактической длине сегментов; после завершения генерации в плейлист добавляется #EXT-X-ENDLIST.

10. HTTP API

Сервис предоставляет следующие эндпоинты:

Метод	Путь	Назначение	Ответ (кратко)
POST	/api/stream/start	Создание сессии и запуск фоновой генерации	session_id, playlist_url
GET	/api/stream/status/{id}	Статус сессии и прогресс	status, progress, error
GET	/stream/{id}/playlist.m3u8	Выдача HLS-плейлиста	playlist.m3u8
GET	/stream/{id}/{filename}	Выдача init.mp4 и сегментов	файл

10.1. POST /api/stream/start

Назначение: создать новую сессию и запустить фоновую задачу генерации.

Пример тела запроса (JSON):

```
{ "text": "Привет!", "lang": "ru" }
```

Пример ответа (JSON):

```
{ "session_id": "abc123", "playlist_url": "/stream/abc123/playlist.m3u8" }
```

10.2. GET /api/stream/status/{id}

Назначение: получить статус сессии (created / processing / ready / error), прогресс и ошибки.

Пример ответа (JSON):

```
{ "status": "processing", "progress": 45, "playlist_ready": false, "error": null }
```

10.3. GET /stream/{id}/playlist.m3u8

Назначение: получить HLS-плейлист с обязательными тегами #EXT-X-MAP, #EXT-X-TARGETDURATION, #EXT-X-ENDLIST.

10.4. GET /stream/{id}/{filename}

Назначение: получить init.mp4 или отдельный сегмент segment_XXX.m4s, относящийся к сессии.

11. Трекинг сессий

Для трекинга сессий используется in-memory структура SESSIONS, которая хранит:

- Статус обработки.
- Пути к файлам HLS и внутренним артефактам.
- Текст/код ошибки при неуспехе.
- Промежуточный прогресс по этапам TTS / инференс / кодирование.

12. Логирование и диагностика

- Логирование ключевых этапов: старт сессии, TTS, извлечение признаков, инференс, сборка HLS, финализация.
- Логирование ошибок TTS и инференса с привязкой к session_id.

13. Тестирование и критерии приемки

13.1. Базовые тесты API

- Проверка последовательности вызовов: start -> status -> playlist -> segments.
- Проверка таймаутов и реакции на невалидные session_id.

13.2. End-to-end тестирование

- Несколько десятков прогонов с разной длиной текста и языком.
- Проверка стабильности работы при последовательных запросах (один за другим).
- Проверка воспроизведения HLS в тестовых плеерах без критичных ошибок.

13.3. Критерии приемки (на 01.12.2025)

- Сервис стабильно генерирует lipsync-видео на базе MuseTalk 1.5 по текстовому запросу.
- Результат доступен по API в виде HLS (init.mp4 + сегменты) и воспроизводится в HLS-плеере.
- Плейлист соответствует требованиям совместимости (теги, TARGETDURATION, ENDLIST) и использует H.264/AAC.
- Сессии отражают статусы created/processing/ready/error, ошибки возвращаются понятным образом.

14. Развертывание (в рамках подготовленного окружения)

- Развернуть Python виртуальную среду и зависимости, включая PyTorch с поддержкой CUDA.
- Установить FFmpeg и проверить поддержку NVENC; обеспечить fallback на libx264.
- Разместить веса MuseTalk 1.5 и Whisper-модель на сервере.
- Подготовить базовое видеоаватара и выполнить препроцессинг (кадры, маски, bbox, кеш латентов).
- Запустить Flask-приложение app.py и обеспечить доступ к эндпоинтам.