



Санкт-Петербургский государственный университет
Кафедра системного программирования

Разработка голосового помощника врача, интегрированного с медицинской информационной системой "СТОММИС", с использованием больших языковых моделей

Альшаеб Басель, группа 24.М71-мм

Санкт-Петербургский государственный университет
Кафедра системного программирования

Научный руководитель: ст. преподаватель каф. системного программирования, к.ф.-м.н. С. С. Сысоев.

Площадка аprobации: СПб ГБУЗ «СП № 8»

Проблема и мотивация

- В стоматологии значимая часть данных должна быть занесена в электронную карту: жалобы, диагнозы, процедуры, материалы, рекомендации.
- В МИС «СТОММИС» исходный прототип голосового режима активировался **кнопкой** и записывал короткий фрагмент речи.
- Во время лечения врачу **неудобно взаимодействовать с компьютером**: заняты руки, перчатки, внимание на пациенте.
- Цель проекта: обеспечить **управление картой голосом прямо в процессе лечения** без ручного запуска записи.

Постановка задачи

Цель: разработать ресурсоэффективный инструмент для анализа аудиопотока из браузера и выделения команд для МИС «СТОММИС».

Задачи:

- ① Детекция участков с речью и «тишиной» в аудиопотоке (VAD) в браузере.
- ② Определение начала/конца командного сегмента (окончание команды — пауза ≈ 2 с).
- ③ Экспериментировать с активацией голосового режима:
 - ▶ KWS/wake word на клиенте;
 - ▶ активация на сервере после ASR (ограниченный список команд и/или LLM-проверка).
- ④ Интеграция с серверной обработкой: транскодирование аудио, ASR, интерпретация команд, формирование действий для МИС.
- ⑤ Первичная апробация и сравнение с базовым режимом (непрерывная обработка потока).

Ключевое противоречие

- Для надежного распознавания команд хочется анализировать весь аудиопоток.
- Но непрерывная отправка/распознавание аудио:
 - ▶ увеличивает нагрузку на сеть и сервер;
 - ▶ увеличивает стоимость использования облачного ASR;
 - ▶ повышает риск обработки нерелевантной речи (комментарии, шум).
- Нужен механизм, который **минимизирует ресурсы, но не пропускает команды**.

Обзор существующих решений

- Диктовка в медицинских системах: полный протокол приёма переводится в текст и сохраняется.
- Голосовые ассистенты общего назначения: хорошо понимают речь, но не привязаны к структуре медкарты и справочникам.
- KWS (wake word) в клиенте: локальная активация, но чувствительна к микрофону/шуму и часто требует обучения под пользователя.

Недостатки в нашем контексте: нет гарантии структурированного заполнения карты и сложно обеспечить устойчивую активацию при низком потреблении ресурсов.

Отличие от типовых решений

- Мы не просто сохраняем речь: результатом являются **структурированные действия** в МИС.
- Команды интерпретируются и сопоставляются с **деревом сущностей** (диагнозы, процедуры, материалы) и полями карточки пациента.
- Предусмотрена фильтрация:
 - ▶ **вход:** VAD + активация, чтобы отсекать «тишину» и нерелевантный поток;
 - ▶ **выход:** ограничение допустимых команд и параметров (контроль формата ответа).

Архитектура решения

- Клиент (браузер):

- ▶ захват микрофона (WebAudio);
- ▶ VAD (@ricky0123/vad-web) выделяет сегменты речи;
- ▶ Сегмент речи отправляется на сервер, где проверяется слово активации; при успехе запускается захват и обработка команды.

- Сервер (поликлиника):

- ▶ приём аудио (`multipart/form-data`, поле `file`);
- ▶ транскодирование в Ogg/Opus (FFmpeg);
- ▶ ASR (Yandex Speech) → текст;
- ▶ Проверка слова активации;
- ▶ интерпретация (LLM) → структурированная команда → действие в МИС.

Активация голосового режима: что пробовали

- **Вариант 1 (браузер, KWS):** локальное распознавание «start»/«старт».
 - ▶ Плюс: не отправляет аудио на сервер до активации.
 - ▶ Минусы: много **FN** (пропусков) в шумной среде, зависит от микрофона, требует **обучения/калибровки на каждом враче**.
- **Вариант 2 (сервер, после ASR):** активация по распознанному тексту + ограниченный список команд / LLM-проверка.
 - ▶ Устойчивее, не требует обучения, но зависит от качества ASR.

В итоговой схеме основной режим — **серверная проверка активации**; клиентская KWS оставлена как опциональная.

Технические детали: сервер пайплайн

- **Вход:** аудиофайл сегмента команды из браузера (multipart/form-data, поле file).

```
if "file" not in request.files: ... raw = request.files["file"]
```
- Приём аудио в Flask:

```
ffmpeg -i pipe:0 -c:a libopus -b:a 48k
```
- **Нормализация формата:** транскодирование в Ogg/Opus (FFmpeg) для стабильного ASR.
- **Распознавание речи (ASR):** Yandex Speech → транскрипция команды.
- **Интерпретация:** LLM преобразует текст в *структурированное действие*.
- **Применение:** внесение изменений в карту пациента посредством **REST API МИС «СТОММИС»**.
- ASR → текст → фильтрация/интерпретация:
 - ▶ ограничение допустимых команд (whitelist);
 - ▶ LLM (например, qwen3-235b-a22b-fp8) формирует JSON-результат с типом команды и аргументами.

Устойчивость к ошибкам ASR/LLM

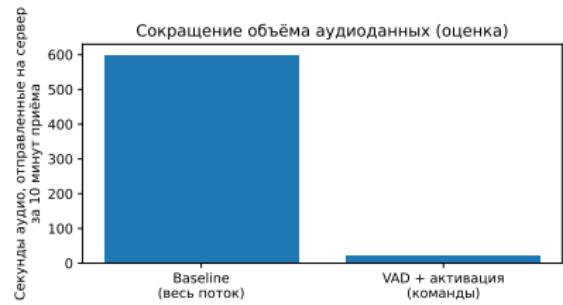
- Ограничение допустимых действий: **белый список интентов** и допустимых параметров.
- Жёсткий формат ответа LLM: **только JSON** по контракту «intent + slots».
- Валидация перед выполнением: синтаксис JSON → схема → проверка слотов по справочникам.
- Fail-safe: при низкой уверенности или ошибке валидации возвращается **уточняющий вопрос**, действие не применяется.

Экспериментальное исследование

- Площадка: тестовая среда и СПб ГБУЗ «СП № 8» (предварительная апробация).
- Сравнение двух режимов:
 - ① **Baseline:** непрерывная обработка аудиопотока/чанков.
 - ② **Предложенный:** VAD в браузере + активация + отправка сегмента команды.
- Метрики:
 - ▶ объём отправленного аудио;
 - ▶ задержка «конец команды → действие в МИС»;
 - ▶ качество активации (FP/FN), качество сегментации.

Результаты (оценка на типовом сценарии)

- За 10 минут приёма объём аудио на сервер существенно снижается: вместо всего потока отправляются только команды.
- Задержка выполнения команды остаётся на уровне $\approx 2\text{--}3$ с, но возникает **только по событию команды**.
- Снижается риск обработки нерелевантной речи.



Секунды аудио, отправленные на сервер (оценка).

Результаты работы

- Реализован клиентский модуль VAD в браузере для выделения речевых сегментов и окончания команды по паузе.
- Проработаны два варианта активации голосового режима (клиентский KWS и серверный контроль после ASR); выбран устойчивый режим без обучения на каждом враче.
- Реализован серверный конвейер: приём аудио, транскодирование, ASR (Yandex Speech), интерпретация команд LLM и формирование действий для МИС «СТОММИС».
- Проведено первичное сравнение с baseline и выполнена апробация в реальной поликлинике (результаты требуют дальнейшей стабилизации).

Ограничения и планы

- Чувствительность к шуму и микрофону остаётся важным фактором (особенно для клиентского KWS).
- Зависимость от сети/задержек при обращении к облачному ASR и LLM.
- Следующие шаги:
 - ▶ расширение и формализация набора команд;
 - ▶ стабилизация в полевых условиях и сбор метрик на большем объёме данных;
 - ▶ оптимизация времени ответа (кэширование, локальные модели на сервере).

Дополнительно: таблица сравнения режимов

Показатель	Baseline	VAD + активация
Длительность аудио за 10 минут	≈600 с	≈18–30 с
Длительность распознаваемого фрагмента	5–15 с	3–5 с
Задержка «конец → действие»	распознавание идёт постоянно	распознавание запускается только после активации
Риск нерелевантной речи	высокий	сниженный
Требования к взаимодействию	нужен ручной контроль	управление голосом без рук