

Санкт-Петербургский государственный университет

Программная инженерия

Группа 24.М71-мм

# Разработка и оптимизация системы голосового управления для автоматизированного заполнения медицинской документации

*Альшаеб Басель*

Отчёт по производственной практике  
в форме «Решение»

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

Санкт-Петербург  
2026

# Оглавление

<b>Введение</b>	<b>3</b>
<b>1. Постановка задачи</b>	<b>5</b>
<b>2. Обзор существующих решений</b>	<b>7</b>
2.1. Выделение участков речи: Voice Activity Detection . . . . .	7
2.2. Активация голосового режима: wake word / keyword spotting . . . . .	8
2.3. Распознавание речи (ASR) и выбор сервиса . . . . .	9
2.4. Интерпретация команд и ограничение допустимых действий . . . . .	9
2.5. Обзор используемых технологий . . . . .	10
2.6. Выводы . . . . .	10
<b>3. Описание решения</b>	<b>12</b>
3.1. Архитектура и поток данных . . . . .	12
3.2. Клиентский модуль: захват аудио и VAD . . . . .	13
3.3. Клиентский модуль: активация и выделение команды . .	13
3.4. Серверный модуль: приём аудио, ASR и интерпретация .	14
3.5. Интеграция со STOMMIS и особенности предметной области . . . . .	15
3.6. Нефункциональные требования: ресурсы и приватность .	16
<b>4. Алгоритмы обработки речевого сигнала</b>	<b>17</b>
4.1. Захват и предварительная обработка аудиосигнала . . .	17
4.2. Детекция речевой активности . . . . .	17
4.3. Выделение участков тишины . . . . .	18
4.4. Активация системы по ключевому слову . . . . .	18
4.5. Оптимизация вычислительных ресурсов . . . . .	19
4.6. Выводы по главе . . . . .	19
<b>5. Алгоритмы обработки команд и автоматического заполнения медицинской карты</b>	<b>20</b>

5.1. Разделение речевых фрагментов и управляющих команд	20
5.2. Формирование структурного представления команды . .	21
5.3. Использование языковой модели для интерпретации команд	21
5.4. Ограничение пространства решений и фильтрация команд	21
5.5. Особенности заполнения стоматологической карты . . .	22
5.6. Интеграция с медицинской информационной системой .	22
5.7. Сравнение с традиционным подходом . . . . .	23
5.8. Выводы по главе . . . . .	23
<b>6. Эксперимент</b>	<b>24</b>
6.1. Условия эксперимента . . . . .	24
6.2. Исследовательские вопросы . . . . .	25
6.3. Метрики . . . . .	25
6.4. Результаты . . . . .	26
6.5. Обсуждение результатов . . . . .	27
6.6. Угрозы нарушения корректности . . . . .	27
6.7. Воспроизводимость . . . . .	27
<b>7. Заключение</b>	<b>29</b>

# Введение

Цифровизация медицинских учреждений приводит к тому, что часть клинической работы переносится в информационные системы: врач фиксирует жалобы, анамнез, результаты осмотра и назначений в электронных формах. В стоматологии это особенно заметно: итогом приёма является структурированная «карта пациента» (диагнозы, манипуляции, зубная формула, рекомендации), которая должна быть заполнена быстро и без потери качества. В рамках эксплуатации медицинской информационной системы (МИС) «СТОММИС» было выявлено, что взаимодействие врача с компьютером непосредственно *в процессе лечения* крайне неудобно: руки заняты инструментами, врач вынужден постоянно переключать внимание между пациентом и интерфейсом. На практике это приводит к тому, что значимая часть заполнения карты откладывается на конец приёма, что увеличивает суммарное время приёма и повышает риск потери деталей из-за человеческого фактора.

Существующие решения на рынке нередко сводятся к диктовке «сплошного текста» в аудио/текстовый протокол или к полной записи консультации с последующей расшифровкой. Подходы такого типа уменьшают объём ручного ввода, но не решают задачу *структурирования* данных (заполнение конкретных полей карты) и создают дополнительную нагрузку: постоянное распознавание речи требует вычислительных ресурсов и, при использовании облачных сервисов, приводит к росту стоимости обработки и сетевого трафика. Кроме того, при непрерывной обработке возрастают риски ложных срабатываний на разговоры пациента и фоновые звуки.

Целью данной работы является разработка и экспериментальная оценка прототипа подсистемы голосового управления для МИС «СТОММИС», обеспечивающей выделение участков речи и «тишины» в аудиопотоке, захватываемом в браузере, и последующую обработку голосовых команд из короткого списка. При этом система должна быть ресурсоэффективной и не выполнять постоянное распознавание всего потока. Ключевые подзадачи, сформулированные руководителем, вклю-

чают:

1. Определение участков с речью и без речи (условной «тишиной»)
2. Выделение в речи команд из короткого списка (при наличии технической возможности и приемлемого качества)

Предлагаемый подход основан на многоэтапной обработке. На стороне клиента в браузере непрерывно работает лёгкий модуль VAD, который определяет начало и конец речевого сегмента и завершает его по правилу «конец команды — 2 секунды тишины». На сервер передаются только релевантные сегменты (после активации ключевым словом), что уменьшает сетевую нагрузку и стоимость распознавания. На сервере аудиофрагмент при необходимости транскодируется, распознаётся с использованием сервиса Yandex SpeechKit (ASR), а затем интерпретируется как структурированная команда для заполнения полей карты пациента с применением правил и языковой модели (LLM) при строгом ограничении допустимых действий. Прототип апробировался в том числе в условиях реальной стоматологической поликлиники СПб ГБУЗ «СП №8»; результаты пока нестабильны, однако подтверждают реализуемость выбранного подхода.

# 1. Постановка задачи

Целью работы является разработка прототипа подсистемы голосового управления для МИС «СТОММИС», позволяющей врачу выполнять операции с медицинской картой *в процессе лечения* с помощью голоса, без необходимости использовать клавиатуру/мышь и без сценария обязательного нажатия кнопки *push-to-talk*. При этом решение должно быть ресурсоэффективным: не выполнять постоянное распознавание всего аудиопотока и минимизировать сетевой трафик.

Для достижения цели были поставлены следующие задачи:

1. Разработать клиентский модуль захвата аудио в браузере и определения участков с речью/без речи (VAD) в реальном времени (раздел 3.2);
2. Реализовать механизм начала и окончания команды без участия рук: исследовать варианты активации голосового режима (кнопка и ключевое слово), а также завершение команды по паузе заданной длительности; обеспечить формирование аудиосегмента и отправку на сервер только релевантных фрагментов (раздел 3.2);
3. Интегрировать обработку аудиосегментов с серверным контуром: приём аудиофайла, транскодирование при необходимости, распознавание речи (ASR) с использованием Yandex SpeechKit и интерпретация распознанного текста в структурированное действие для заполнения полей карты пациента в МИС «СТОММИС» (разделы 3.4).
4. Провести экспериментальную валидацию прототипа, включая апробацию в реальных условиях (СПб ГБУЗ «СП №8»): оценить задержку, ресурсопотребление, частоту ложных срабатываний/-пропусков и потенциальное снижение затрат по сравнению с вариантом постоянного распознавания аудиопотока (раздел 6).

Ограничения и допущения работы:

- Решение должно исполняться либо в браузере (JavaScript), либо на сервере в поликлинике (Python), без необходимости в специализированном оборудовании;
- Обработка должна быть устойчивой к типовым помехам кабинета (разговоры, шум оборудования), а также не должна существенно увеличивать задержку взаимодействия врача с системой.

## 2. Обзор существующих решений

В данном разделе приведён обзор подходов и инструментов, применимых для построения голосового управления в медицинских информационных системах. Основной акцент сделан на практических аспектах: как выделять речевые сегменты в аудиопотоке, как запускать обработку только в нужный момент (wake word), и как преобразовывать распознанный текст в контролируемые действия внутри МИС.

### 2.1. Выделение участков речи: Voice Activity Detection

Задача детекции речевой активности (VAD) заключается в определении моментов начала и окончания речи в непрерывном аудиопотоке. Простейший подход основан на пороговой оценке энергии сигнала (например, RMS) и может быть реализован непосредственно в браузере. Преимущество такого решения — минимальная вычислительная сложность. Недостатки проявляются в реальных условиях кабинета: фоновые шумы, переменная громкость голоса и короткие паузы приводят к нестабильным границам сегментов и к росту числа ложных срабатываний.

Более устойчивый класс решений — классический VAD, применяемый в системах связи (например, VAD из стека WebRTC) [?]. Он широко применяется в VoIP и видеоконференциях и рассчитан на работу в реальном времени.

В последние годы получили распространение нейросетевые VAD-модели, обученные на больших корпусах аудио. К популярным открытым решениям относится *Silero VAD* [?], обеспечивающий высокую точность и работающий на CPU. Недостатком является необходимость серверного выполнения (Python) или интеграции через ONNX/WebAssembly, что усложняет развёртывание на клиентской стороне.

В рамках дипломной работы для браузера был выбран open-source

инструмент `@ricky0123/vad-web` [?], который применяет нейросетевую модель (в формате ONNX) и исполняет её через ONNX Runtime Web [?]. Данный вариант обеспечивает более устойчивое выделение речи по сравнению с пороговой энергией и допускает работу «на месте» (в браузере) без постоянных серверных вычислений.

## 2.2. Активация голосового режима: wake word / keyword spotting

Чтобы врач мог использовать голосовые команды *в процессе лечения* без нажатия кнопок, требуется механизм активации голосового режима по ключевому слову (wake word). В литературе и практике встречаются два основных класса решений:

- **Локальное keyword spotting (KWS):** ключевое слово детектируется на клиенте по аудиопризнакам (например, MFCC) с использованием DTW-сопоставления или небольшой нейросети.
- **Серверная проверка ключевого слова через ASR:** на клиенте выделяются короткие сегменты речи (VAD), после чего на сервер отправляется короткий фрагмент для распознавания, и активация выполняется по текстовой транскрипции.

Плюсом локального KWS является минимальная задержка и отсутствие сетевых обращений в режиме ожидания. Минусы проявляются в прикладном сценарии стоматологии: существенно меняются микрофоны/акустика, вариативны голоса, а обучение на каждого врача (персонализация) ухудшает удобство внедрения. Среди open-source решений KWS можно отметить проект *openWakeWord* [?], ориентированный на CPU-исполнение, однако перенос KWS непосредственно в браузер требует стабильной цепочки извлечения признаков и калибровки.

Серверная проверка ключевого слова через ASR проще интегрируется в существующий серверный контур распознавания речи. Кроме того, этот подход позволяет обойти необходимость обучения на каждо-

го врача, однако он увеличивает число обращений к ASR по сравнению с чисто локальным KWS.

В дипломной работе исследовались оба варианта (см. раздел ??): клиентский вариант был реализован как прототип, но не прошёл испытания из-за высокой доли пропусков (false negatives) и из-за необходимости обучения под конкретного врача; в итоговой версии используется серверная проверка ключевого слова на основе ASR.

## **2.3. Распознавание речи (ASR) и выбор сервиса**

После выделения аудиосегмента система должна получить текстовую транскрипцию команды. В практических системах используются как локальные движки распознавания (например, Vosk), так и облачные сервисы. В данной работе выбран сервис Yandex SpeechKit [?], поскольку он предоставляет готовый API для русского языка.

## **2.4. Интерпретация команд и ограничение допустимых действий**

В рамках данной работы для интерпретации команд рассматриваются облачные LLM-модели; в прототипе используется qwen3-235b-a22b-fp8, также исследуется альтернатива gpt-oss-20b.

Полученный текст команды необходимо преобразовать в действие внутри МИС. Классические подходы включают грамматические правила и intent-классификацию (intent + slots) для фиксированного набора команд. Однако естественная речь врача вариативна, поэтому жёсткие шаблоны быстро усложняются и требуют постоянной поддержки.

Перспективный подход — использование языковой модели (LLM) для семантической интерпретации команды. При этом критически важно ограничить допустимый вывод модели, чтобы исключить неконтролируемые действия. В прикладных системах это достигается комбинацией:

- строгого формата ответа (например, JSON) и серверной валидации;
- ограниченного справочника действий (список допустимых интенгов) и параметров (слоты, значения из справочников);
- повторного запроса к модели при нарушении формата или при низкой уверенности.

## 2.5. Обзор используемых технологий

**Захват аудио в браузере.** Для доступа к микрофону применяется API `MediaDevices.getUserMedia` [?], а для потоковой обработки и/или записи — Web Audio API [?] и MediaRecorder API [?]. Указанные технологии доступны во всех современных браузерах и не требуют установки дополнительного ПО на рабочее место врача.

**Браузерный VAD.** В качестве VAD-ядра выбран `@ricky0123/vad-web` [?]. Данный вариант обеспечивает баланс между простотой интеграции (JS, события `onSpeechStart/onSpeechEnd`) и ресурсной эффективностью (WebAssembly + оптимизированные вычисления ONNX Runtime Web [?]). В работе используется завершение сегмента после паузы фиксированной длительности (например, 2 секунды), что соответствует требованиям к выделению команд.

**Серверная обработка и интеграция.** Серверный контур реализован на Python (Flask) и принимает аудио сегменты как файл `multipart/form-data`. При необходимости входной формат приводится к Ogg Opus с помощью `ffmpeg` [?], после чего аудио подаётся в модуль распознавания речи (Yandex SpeechKit [?]). Далее результат передаётся в модуль интерпретации команд (правила/LLM), который возвращает структурированное действие для заполнения полей карты пациента.

## 2.6. Выводы

По результатам обзора для прототипа выбрана архитектура *VAD в браузере + ASR и интерпретация на сервере*. Она позволяет:

- снизить стоимость и сетевую нагрузку за счёт отправки на сервер только речевых сегментов, релевантных командам;
- сохранить качество распознавания русской речи благодаря использованию готового ASR-сервиса;
- минимизировать требования к рабочему месту врача, ограничившись браузером и микрофоном.

### **Браузер (рабочее место врача)**

(1) getUserMedia → (2) VAD → (3) сегменты речи  
(4) детектор активации (wake word) → (5) сегмент команды

### **Сервер (Flask, в контуре клиники)**

(6) приём файла `file` → (7) транскодирование (при необходимости)  
(8) ASR (Yandex SpeechKit) → (9) интерпретация команды (rules/LLM)  
(10) запись результата в STOMMIS / возврат ответа клиенту

Рис. 1: Укрупнённая схема обработки аудиопотока и команд.

**Клиент (браузер):** Web Audio API → VAD → (активация) → сегмент команды → HTTP multipart/form-data (`file`)

**Сервер (Python):** приём `file` → (FFmpeg) → ASR (Yandex SpeechKit) → интерпретация (правила/LLM) → действие в МИС

Рис. 2: Конвейер обработки голосовой команды

## **3. Описание решения**

В данном разделе описывается реализованный прототип подсистемы голосового управления для МИС «СТОММИС». Решение построено как конвейер: на стороне клиента в браузере выделяются речевые сегменты и, при активации голосового режима, формируется аудиофрагмент команды; на сервере выполняются транскодирование (при необходимости), распознавание речи и интерпретация команды в контролируемое действие внутри МИС.

### **3.1. Архитектура и поток данных**

Поток обработки состоит из клиентской и серверной частей (рис. 2). Клиент отвечает за захват аудиопотока и предварительную обработку с минимальной вычислительной нагрузкой. Сервер отвечает за «дорогие» операции (ASR и семантическая интерпретация), а также за интеграцию результата с МИС «СТОММИС».

Ключевой принцип архитектуры — минимизировать «дорогие» опе-

рации распознавания речи, выполняя на клиенте лёгкую предварительную фильтрацию (VAD и логика активации). Это снижает стоимость и сетевую нагрузку, а также уменьшает объём данных, передаваемых за пределы браузера.

## 3.2. Клиентский модуль: захват аудио и VAD

Клиентская часть реализована как веб-страница (HTML + JavaScript + jQuery), которая получает доступ к микрофону и запускает VAD-движок. Доступ к микрофону осуществляется через `getUserMedia` [?]; обработка аудио выполняется в браузере с использованием Web Audio API [?].

Для выделения речи используется библиотека `@ricky0123/vad-web` [?], предоставляющая событийную модель:

- `onSpeechStart` — фиксируется начало речевого сегмента;
- `onSpeechEnd` — сегмент считается завершённым после паузы (периода «тишины») заданной длительности;
- `onUpdate` — обновление вероятности речи (для визуализации и отладки).

Пороговые параметры VAD настраиваются экспериментально под условия кабинета: `positiveSpeechThreshold` (чувствительность старта), `negativeSpeechThreshold` (чувствительность окончания), а также `redemptionFrames`, задающий длительность «тишины» до завершения сегмента. В прототипе используется правило «конец команды — 2 секунды тишины», реализуемое через `redemptionFrames`.

## 3.3. Клиентский модуль: активация и выделение команды

Основная проблема непрерывного распознавания речи состоит в том, что на сервер уходит *весь* аудиопоток (включая фоновые разговоры

и паузы), что приводит к росту стоимости и нагрузки. Для решения используется двухэтапная логика:

1. **Режим мониторинга:** VAD работает постоянно, но речевые сегменты не отправляются на сервер.
2. **Режим команды:** после обнаружения ключевого слова (например, «старт») ближайший речевой сегмент считается командой и отправляется на сервер.

В прототипе предусмотрены два варианта активации:

- **Ручная активация** (кнопка / горячая клавиша) — используется как контрольный сценарий и позволяет сравнить качество выделения сегмента команды;
- **Голосовая активация** (wake word) — ключевое слово определяется по коротким речевым сегментам, полученным от VAD. Практический вариант для прототипа состоит в проверке ключевого слова по текстовой транскрипции короткого сегмента на сервере (ASR), что позволяет избежать сложной KWS-модели на клиенте.

После активации команда считается завершённой, когда VAD фиксирует паузу заданной длительности. Получившийся сегмент команды кодируется в аудиофайл и отправляется на сервер как `multipart/form-data` с полем `file` (см. раздел 3.4).

### 3.4. Серверный модуль: приём аудио, ASR и интерпретация

Серверная часть реализована на Python (Flask) и предоставляет HTTP-endpoint для приёма командных аудиосегментов. Клиент отправляет запрос вида:

- метод `POST`;
- тело `FormData` с полем `file`;

- дополнительные поля метаданных (идентификатор врача, пациента, режим команд), если требуется контекст.

На сервере выполняются шаги:

1. проверка наличия поля `file` и чтение байтов аудио в память;
2. приведение формата аудио к поддерживаемому входу ASR: если MIME-тип отличается от ожидаемого, выполняется транскодирование в Ogg Opus через `ffmpeg` по пайпам (без записи на диск) [?];
3. распознавание речи посредством Yandex SpeechKit [?];
4. интерпретация результата: преобразование текста в структурированную команду (например, `intent` + набор параметров), пригодную для заполнения полей карты пациента.

Интерпретация реализуется гибридно: для короткого списка команд задаётся допустимый набор интенгов, а для сопоставления свободных формулировок используется LLM-модуль. Результат возвращается клиенту в JSON и/или применяется на стороне STOMMIS.

### 3.5. Интеграция со STOMMIS и особенности предметной области

В отличие от решений, ориентированных на «протоколирование» консультации, в рассматриваемом сценарии цель — заполнение структурированной карты. Это требует:

- приведения распознанного текста к заранее определённым категориям (диагноз, манипуляция, зуб, рекомендация);
- сопоставления со справочниками (например, список процедур и заболеваний), используемыми в STOMMIS;
- минимизации количества действий врача в интерфейсе (особенно при работе в перчатках).

Командный режим позволяет врачу произносить короткие инструкции, которые конвертируются в конкретные изменения в карте. Например, команда «старт, кариес на шестом верхнем справа» может быть интерпретирована как выбор диагноза «кариес» и установка соответствующего зуба в формуле.

### 3.6. Нефункциональные требования: ресурсы и приватность

Решение спроектировано с учётом ограничений клиники:

- **Ресурсоэффективность:** постоянная работа VAD в браузере должна быть существенно дешевле постоянного ASR;
- **Минимизация трафика:** на сервер отправляются только короткие сегменты команд;
- **Приватность:** исключается постоянная передача полного аудио приёма на сервер; объём передаваемых данных ограничивается командными сегментами.

Перечисленные свойства проверяются экспериментально в разделе 6.

## 4. Алгоритмы обработки речевого сигнала

Обработка речевого сигнала в разрабатываемой системе осуществляется в несколько этапов, каждый из которых направлен на снижение вычислительной нагрузки, уменьшение количества ошибок распознавания и повышение удобства использования системы в реальных условиях медицинского приёма.

Основная идея алгоритмического подхода заключается в том, что непрерывный звуковой поток не передаётся целиком на сервер и не подвергается постоянному распознаванию. Вместо этого используется последовательная фильтрация аудиоданных, позволяющая выделять только информативные участки речи и игнорировать фоновые шумы и нерелевантные звуки.

### 4.1. Захват и предварительная обработка аудиосигнала

Захват аудиосигнала осуществляется на стороне клиента с использованием стандартных средств Web Audio API. Входной сигнал представляет собой непрерывный поток аудиоданных, поступающих с микрофона пользователя.

На этапе предварительной обработки к аудиосигналу применяются базовые методы улучшения качества записи, включая подавление шума, автоматическую регулировку усиления и компенсацию эха. Данные методы позволяют стабилизировать амплитуду сигнала и повысить устойчивость последующих этапов обработки к изменениям громкости речи и внешним акустическим условиям.

### 4.2. Детекция речевой активности

Для разделения непрерывного аудиопотока на участки речи и условной тишины в системе используется алгоритм детекции речевой активности (Voice Activity Detection, VAD). Задачей данного алгоритма является определение моментов начала и окончания речевых сегментов в

звуковом сигнале.

Использование VAD позволяет отказаться от постоянной записи и передачи аудиоданных на сервер, что существенно снижает объём обрабатываемой информации и нагрузку на вычислительные ресурсы. Алгоритм VAD работает в режиме реального времени и принимает решение о наличии речи на основе анализа кратковременных характеристик сигнала.

### **4.3. Выделение участков тишины**

Завершение речевой команды определяется на основе анализа продолжительности интервалов тишины между фрагментами речи. После обнаружения речевого сегмента система продолжает наблюдение за входным сигналом и фиксирует момент завершения команды при достижении заданной длительности условной тишины.

Использование интервалов тишины в качестве критерия завершения команды позволяет отказаться от явных маркеров окончания речи и делает взаимодействие с системой более естественным для пользователя. Такой подход особенно удобен в медицинской практике, где речь врача может сопровождаться паузами, не являющимися признаком завершения команды.

### **4.4. Активация системы по ключевому слову**

Для предотвращения случайной активации системы и снижения количества ложных срабатываний используется механизм активации по ключевому слову. В неактивном состоянии система выполняет только детекцию речевой активности и анализ коротких речевых фрагментов.

При обнаружении короткого речевого сегмента он передаётся на сервер для проверки на соответствие ключевой фразе активации. Только после подтверждения ключевого слова система переходит в активный режим и начинает обработку основной команды врача.

Данный подход позволяет существенно сократить количество обращений к сервису распознавания речи и, как следствие, снизить эксплу-

атационные затраты и задержки обработки.

## **4.5. Оптимизация вычислительных ресурсов**

Одним из ключевых требований к разрабатываемой системе является минимизация вычислительных затрат при сохранении приемлемой точности распознавания. Для достижения данной цели применяется многоуровневая фильтрация аудиосигнала.

На клиентской стороне выполняется первичная фильтрация и сегментация речи с использованием VAD. На сервер передаются только короткие аудиофрагменты, потенциально содержащие команды или ключевые слова. Полноценное распознавание речи и семантическая интерпретация выполняются исключительно для подтверждённых сегментов.

Таким образом, большая часть нерелевантных данных отбрасывается на ранних этапах обработки, что обеспечивает эффективное использование вычислительных ресурсов и повышает масштабируемость системы.

## **4.6. Выводы по главе**

В данной главе был рассмотрен алгоритмический подход к обработке речевого сигнала, основанный на последовательной фильтрации аудиоданных и активации системы по ключевому слову. Предложенные алгоритмы позволяют обеспечить устойчивую работу системы в реальных условиях медицинского приёма при ограниченных вычислительных ресурсах.

Использование детекции речевой активности и анализа интервалов тишины делает взаимодействие с системой естественным и не требует от врача дополнительного обучения или изменения привычного рабочего процесса.

## **5. Алгоритмы обработки команд и автоматического заполнения медицинской карты**

После завершения этапа распознавания речи система получает текстовую транскрипцию голосовой команды врача. Однако прямое использование данного текста для заполнения медицинской документации является неэффективным и может приводить к ошибкам, поскольку речь врача часто содержит уточнения, комментарии и фразы, не предназначенные для прямого сохранения в медицинской карте.

В связи с этим в разрабатываемой системе реализован отдельный этап интерпретации команд, целью которого является преобразование неструктурированной текстовой информации в формализованное представление, пригодное для автоматизированного внесения данных в электронную медицинскую карту пациента.

### **5.1. Разделение речевых фрагментов и управляющих команд**

Речь врача в процессе приёма пациента может содержать как управляющие команды, адресованные системе, так и пояснительные комментарии, не имеющие прямого отношения к заполнению медицинской карты. Для корректной обработки голосового ввода необходимо различать данные типы речевых фрагментов.

В предлагаемом подходе речевая команда рассматривается как управляющее высказывание, содержащее указание на конкретное действие системы, например выбор диагноза, добавление симптома или заполнение определённого поля медицинской карты. Все остальные речевые фрагменты рассматриваются как нерелевантные для автоматической обработки и отбрасываются на данном этапе.

## 5.2. Формирование структурного представления команды

Для обеспечения однозначной интерпретации команд врача используется структурное представление команды, включающее тип действия и набор параметров. Каждая команда преобразуется в формализованную структуру следующего вида:

$$(тип\_команды, объект, параметры)$$

Тип команды определяет общее направление действия, например добавление записи, выбор значения или подтверждение состояния пациента. Объект команды указывает на элемент медицинской карты, к которому относится действие, а параметры содержат дополнительные данные, необходимые для выполнения команды.

## 5.3. Использование языковой модели для интерпретации команд

Для преобразования текстовой транскрипции в структурированное представление используется языковая модель, способная выполнять семантический анализ естественного языка. Языковая модель получает на вход текст команды и контекст текущего состояния медицинской карты пациента.

На основе анализа входных данных модель определяет наиболее вероятный тип команды и соответствующие параметры. Использование языковой модели позволяет учитывать вариативность формулировок команд, характерную для естественной речи, и снижает зависимость системы от строго фиксированного набора шаблонов.

## 5.4. Ограничение пространства решений и фильтрация команд

Для повышения надёжности работы системы и предотвращения некорректных действий языковая модель не имеет доступа к произ-

вольному пространству решений. Вместо этого интерпретация команд выполняется в рамках заранее определённого множества допустимых действий и объектов.

В частности, выбор диагнозов, симптомов и процедур осуществляется исключительно из предопределённых медицинских справочников и классификаторов. Такой подход позволяет снизить вероятность логических ошибок и обеспечивает соответствие результатов обработки требованиям медицинских информационных систем.

## **5.5. Особенности заполнения стоматологической карты**

Стоматологическая медицинская карта обладает высокой степенью структурированности и включает множество взаимосвязанных параметров, таких как состояние отдельных зубов, наличие патологий и выполненные процедуры. Ручной выбор соответствующих значений из иерархических списков является трудоёмкой задачей и отвлекает врача от лечебного процесса.

В разрабатываемой системе голосовые команды используются для навигации по структуре стоматологической карты и выбора необходимых элементов без прямого взаимодействия с пользовательским интерфейсом. Это позволяет врачу сосредоточиться на пациенте и существенно ускоряет процесс заполнения документации.

## **5.6. Интеграция с медицинской информационной системой**

После формирования структурированного представления команды выполняется её интеграция с медицинской информационной системой. На данном этапе проверяется корректность параметров команды и соответствие текущему состоянию медицинской карты пациента.

При успешной проверке система автоматически вносит изменения в соответствующие поля медицинской карты. В случае возникновения

неоднозначностей или ошибок команда отклоняется, а врач получает уведомление о необходимости уточнения запроса.

## **5.7. Сравнение с традиционным подходом**

В традиционных системах голосового ввода распознанная речь сохраняется в медицинской карте в виде текстового комментария без дальнейшей обработки. Такой подход не позволяет автоматизировать работу с данными и требует последующего ручного анализа.

В отличие от этого, предлагаемый в работе метод ориентирован на интерпретацию речи как источника управляющих команд. Результатом обработки является не текст, а структурированные изменения в медицинской карте, что обеспечивает более высокий уровень автоматизации и снижает нагрузку на медицинский персонал.

## **5.8. Выводы по главе**

В данной главе был рассмотрен алгоритмический подход к интерпретации голосовых команд врача и автоматизированному заполнению медицинской карты пациента. Использование языковой модели в сочетании с ограничением пространства допустимых решений позволяет обеспечить надёжную и контролируемую обработку команд в медицинской информационной системе.

## 6. Эксперимент

Цель эксперимента — проверить, что предложенная архитектура (VAD в браузере + отправка только командных сегментов) действительно уменьшает ресурсоёмкость и стоимость обработки по сравнению с наивным подходом непрерывного распознавания речи, сохраняя приемлемое качество выделения и распознавания команд.

### 6.1. Условия эксперимента

Эксперимент проводился в условиях, приближенных к рабочему месту врача:

- **Клиент:** современный браузер (Chrome/Chromium), запуск веб-страницы с VAD; доступ к микрофону через `getUserMedia`.
- **Сервер:** Python 3.x, Flask, модуль транскодирования через `ffmpeg` [?], модуль распознавания речи Yandex SpeechKit [?].
- **Сеть:** локальная сеть клиники / тестовая сеть; фиксируются объём отправленных данных и задержки.

**Данные.** Для оценки использовались записи (или воспроизведение через микрофон) типовых сценариев приёма: команды врача, обычная речь (объяснения пациенту), паузы, а также фоновые шумы кабинета. Разметка включает:

- временные границы команд (начало/конец);
- текстовую расшифровку команды;
- ожидаемое структурированное действие в системе (intent + параметры).

При сборе данных предполагается соблюдение требований приватности: данные деперсонифицированы, не содержат ФИО и иной чувствительной информации.

## 6.2. Исследовательские вопросы

**RQ1:** Насколько уменьшается объём аудио, отправляемого на сервер, и связанная с этим стоимость распознавания по сравнению с непрерывным ASR?

**RQ2:** Каково качество выделения и распознавания команд (precision/recall) при использовании VAD + активации по ключевому слову?

**RQ3:** Как влияет настройка параметров VAD (пороги, длительность «тишины» до завершения) на ложные срабатывания и задержку?

## 6.3. Метрики

Для ответа на исследовательские вопросы используются следующие метрики:

- **Доля переданного аудио**  $S = \frac{T_{\text{sent}}}{T_{\text{total}}}$ , где  $T_{\text{sent}}$  — суммарная длительность сегментов, отправленных на сервер,  $T_{\text{total}}$  — длительность исходного аудиопотока (RQ1).
- **Сетевой трафик** (байты/МБ), отправленный клиентом на сервер (RQ1).
- **Задержка**  $L$  — время от окончания произнесения команды до получения серверного ответа (медиана и 95-й перцентиль) (RQ2, RQ3).
- **Качество выделения команд:** precision, recall и  $F_1$  по факту обнаружения команды и её корректной интерпретации (RQ2).
- **Ресурсоёмкость клиента:** средняя загрузка CPU и потребление памяти браузером в режиме мониторинга и в режиме команды (RQ1).

Таблица 1: Сравнение базового подхода и предложенного решения.

Метрика	Baseline	Proposed
Доля переданного аудио $S$	—	—
Трафик, МБ	—	—
Задержка $L$ , мс (медиана)	—	—
Precision команд	—	—
Recall команд	—	—
CPU в мониторинге, %	—	—

## 6.4. Результаты

Результаты эксперимента представляются в виде таблиц и графиков, сопоставляющих базовый подход и предложенное решение.

### Сравниваемые варианты.

1. **Baseline:** непрерывная запись и отправка аудио на сервер с последующим ASR.
2. **Proposed:** VAD в браузере + активация + отправка только сегментов команд.

**Шаблон таблицы результатов.** В таблице 1 приводится рекомендуемый формат представления ключевых метрик (значения подставляются по результатам замеров).

### 6.4.1. RQ1: ресурсоёмкость и стоимость

Ожидается, что предложенное решение значительно уменьшит  $T_{\text{sent}}$  и сетевой трафик за счёт исключения пауз и нерелевантной речи. Дополнительно фиксируется загрузка CPU в браузере: в режиме мониторинга должна наблюдаться существенно меньшая нагрузка, чем при постоянной записи/кодировании и передаче аудио.

### 6.4.2. RQ2: качество команд

Для качества выделения команд важно учитывать два типа ошибок: ложные срабатывания (команда выделена, но её не было) и пропуски

(команда произнесена, но не выделена/не распознана). Оценка проводится на размеченных сценариях и отражается через  $\text{precision}/\text{recall}/F_1$ .

#### 6.4.3. RQ3: влияние параметров

Проводится серия запусков при различных значениях порогов VAD и длительности паузы завершения. Анализируется компромисс между задержкой (слишком длинная пауза увеличивает  $L$ ) и устойчивостью (слишком короткая пауза может преждевременно обрывать команду).

### 6.5. Обсуждение результатов

При интерпретации результатов важно учитывать особенности стоматологического кабинета: наличие фоновой речи, инструмента и оборудования, а также различия в манере диктовки у разных врачей. Планируется отдельно анализировать типовые причины ошибок (например, смешение речи врача и пациента, тихая речь, высокая шумовая нагрузка) и влияние этих факторов на качество системы.

### 6.6. Угрозы нарушения корректности

Основные угрозы валидности эксперимента:

- **Разметка:** неточность временных границ команд и субъективность в определении «команды».
- **Сеть и инфраструктура:** задержки и потери в сети могут влиять на измеряемую латентность.

### 6.7. Воспроизводимость

Для воспроизводимости эксперимента в репозитории проекта фиксируются:

- версии зависимостей клиентского и серверного модулей;
- параметры VAD, использованные в замерах;

- скрипты для запуска и сбора метрик (тайминг, объём отправленных данных);
- (при возможности) набор обезличенных тестовых аудиофайлов или инструкции по его получению.

## 7. Заключение

В работе разработан и реализован прототип голосового управления для стоматологической ИС, ориентированный на минимизацию ресурсоёмкости и стоимости распознавания за счёт предварительной фильтрации аудиопотока на клиенте.

Получены следующие результаты:

- Реализован клиентский модуль захвата аудио в браузере и определения участков с речью/без речи в реальном времени на основе браузерного VAD (@ricky0123/vad-web) (результат к задаче №1, раздел 3.2).
- Разработана логика активации и выделения команд (wake word + завершение по паузе), позволяющая отправлять на сервер только командные сегменты, тем самым снижая сетевую нагрузку и стоимость обработки (результат к задаче №2, раздел 3.3).
- Выполнена интеграция с серверным контуром на Flask: приём сегмента как поля `file`, приведение формата аудио через `ffmpeg`, распознавание речи Yandex SpeechKit и преобразование текста команды в структурированное действие для заполнения карты пациента в STOMMIS (результат к задаче №3, раздел 3.4).
- Предложен дизайн эксперимента и набор метрик для оценки ресурсоёмкости, задержки и качества выделения команд, позволяющий валидировать работоспособность решения в условиях клиники (раздел 6).

Дальнейшее развитие работы включает: расширение словаря команд и справочников предметной области, улучшение механизма ключевого слова (в том числе за счёт специализированного KWS-модуля), а также проведение полномасштабной апробации на данных нескольких врачей с анализом устойчивости к шумам и индивидуальным особенностям речи.