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 кафедры системного программирования, к.ф.-м.н. С. С. Сысоев.

Санкт-Петербург
2026

Оглавление

Список использованных сокращений **4**

Введение **6**

1. Постановка задачи **8**

2. Обзор существующих решений **10**

2.1. Выделение участков речи: Voice Activity Detection 10

2.2. Активация голосового режима: wake word / keyword spotting 11

2.3. Распознавание речи (ASR) и выбор сервиса 12

2.4. Интерпретация команд и ограничение допустимых действий 12

2.5. Обзор используемых технологий 13

2.6. Выводы 14

3. Описание решения **15**

3.1. Архитектура и поток данных 15

3.2. Клиентский модуль: захват аудио и VAD 15

3.3. Клиентский модуль: активация и выделение команды . . 17

3.4. Wake word: варианты реализации и выбранный подход . 18

3.5. Серверный модуль: приём аудио, ASR и интерпретация . 19

3.6. Ограничение вывода языковой модели и валидация результата 19

3.7. Интеграция со МИС «СТОММИС» и особенности предметной области 22

3.8. Нефункциональные требования: ресурсы и приватность . 22

4. Алгоритмы обработки речевого сигнала **24**

4.1. Захват и предварительная обработка аудиосигнала . . . 24

4.2. Детекция речевой активности 24

4.3. Выделение участков тишины 25

4.4. Активация системы по ключевому слову 25

4.5. Оптимизация вычислительных ресурсов 26
4.6. Выводы по главе 26

5. Алгоритмы обработки команд и автоматического заполнения медицинской карты 27
5.1. Разделение речевых фрагментов и управляющих команд 27
5.2. Формирование структурного представления команды . . 28
5.3. Использование языковой модели для интерпретации команд 28
5.4. Ограничение пространства решений и фильтрация команд 28
5.5. Особенности заполнения стоматологической карты . . . 29
5.6. Интеграция с медицинской информационной системой . 29
5.7. Сравнение с традиционным подходом 30
5.8. Выводы по главе 30

6. Эксперимент 31
6.1. Исследовательские вопросы 31
6.2. Метрики 31
6.3. Сценарии и базовая линия сравнения 32
6.4. Результаты и обсуждение 32
6.5. Обсуждение результатов 34
6.6. Угрозы нарушения корректности 34

7. Заключение 36

Список литературы 38

Список использованных сокращений

API *Application Programming Interface* — программный интерфейс взаимодействия компонентов.

ASR *Automatic Speech Recognition* — автоматическое распознавание речи (преобразование аудио в текст).

CPU *Central Processing Unit* — центральный процессор (в работе подчёркивается возможность выполнения моделей на CPU без GPU).

DeepSeek семейство больших языковых моделей (LLM), рассматриваемое как перспектива для экспериментов в задаче интерпретации команд.

DTW *Dynamic Time Warping* — метод сравнения последовательностей (используется в некоторых подходах keyword spotting).

FFmpeg набор утилит для обработки мультимедиа (в работе используется для транскодирования аудио на сервере).

FP/FN *False Positive / False Negative* — ложное срабатывание / пропуск (например, при детекции wake word).

Intent *intent* — **намерение/тип команды** в задачах обработки естественного языка (класс действия, которое должна выполнить система).

JSON *JavaScript Object Notation* — формат обмена структурированными данными.

KWS *Keyword Spotting* — обнаружение ключевого слова (wake word).

LLM *Large Language Model* — большая языковая модель, используемая для семантической интерпретации команд.

MFCC *Mel-Frequency Cepstral Coefficients* — кепстральные коэффициенты, часто используемые как признаки в задачах аудио.

NLP *Natural Language Processing* — обработка естественного языка.

МИС медицинская информационная система.

МИС «СТОММИС» медицинская информационная система «СТОММИС» (объект внедрения в работе).

OGG контейнерный формат Ogg (в работе используется Ogg Opus для унификации входа ASR).

Opus аудиокодек Opus (в работе используется при транскодировании в Ogg Opus).

ONNX *Open Neural Network Exchange* — формат представления нейросетевых моделей.

ONNX Runtime библиотека выполнения моделей в формате ONNX (используется в клиентском VAD).

RMS *Root Mean Square* — среднеквадратичная амплитуда сигнала (простой признак для пороговой детекции речи).

Slot *slot* — **параметр команды** (атрибут/значение, извлекаемое из текста команды; например номер зуба, диагноз, процедура).

VAD *Voice Activity Detection* — детекция речевой активности (определение участков речи и «тишины»).

VoIP Voice over IP — передача голоса по IP-сетям (телефония через Интернет).

WASM *WebAssembly* — формат и среда выполнения кода в браузере (используется для ускорения вычислений на клиенте).

WebRTC набор технологий реального времени в вебе; включает классические алгоритмы VAD и подавления эха.

wake word / trigger word ключевое слово/фраза активации голосового режима.

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)
3. Внесение изменений в карту пациента в формате МИС «СТОМ-МИС»

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 выделенные VAD сегменты речи. При этом на серверной стороне выполняется обнаружение и подтверждение ключевого слова активации (см. раздел 2.2). Такой подход 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».

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 3.3);

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» (раздел 3.5).

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.
- Устойчивость к ошибкам распознавания речи;
- Устойчивость к ошибкам LLM при определении типа команды.

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) [11]. 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* [9], обеспечивающий высокую точность и работающий на CPU. Недостатком является то, что типовой сценарий использования Silero VAD предполагает серверное выполнение в Python. Запуск на стороне клиента возможен, но в этом случае требуется самостоятельно собрать и поддерживать цепочку: экспорт модели в ONNX, подключение ONNX Runtime Web

и WebAssembly-ассетов, приведение частоты дискретизации, а также контроль производительности в браузере. Для дипломного прототипа это избыточно по трудозатратам по сравнению с готовым браузерным решением.

В рамках дипломной работы для браузера был выбран open-source инструмент `@ricky0123/vad-web` [8], который применяет нейросетевую модель (в формате ONNX) и исполняет её через ONNX Runtime Web [7]. 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* [1], ориентированный на CPU-исполнение. Однако перенос KWS непосредственно в браузер

требует воспроизводимой цепочки обработки аудио: ресэмплирование до фиксированной частоты (например, 16 кГц), разбиение на фреймы фиксированного размера, применение окна, вычисление спектра (FFT) и извлечение признаков (например, MFCC) с одинаковой нормализацией. На практике даже небольшие расхождения в размерах буфера/FFT или в нормализации приводят к росту доли пропусков (ложных отрицаний, false negatives) и требуют отдельной калибровки под конкретный микрофон и голос.

Серверная проверка ключевого слова через ASR проще интегрируется в существующий серверный контур распознавания речи. Кроме того, этот подход позволяет обойти необходимость обучения на каждого врача, однако он увеличивает число обращений к ASR по сравнению с чисто локальным KWS.

В дипломной работе исследовались оба варианта (см. раздел 3.4): 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 [12], 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 исследуются альтернативы `gpt-oss-20b`.

На этапе предварительного анализа также рассматривались YandexGPT и (в перспективе) DeepSeek; однако в основной конфигурации прототипа зафиксирована модель `qwen3-235b-a22b-fp8`.

Полученный текст команды необходимо преобразовать в действие внутри МИС. Классические подходы включают грамматические правила и классификацию интентов (англ. *intent*) и параметров-слотов (англ. *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 интентов) 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` [4], а для потоковой обработки и/или записи — Web Audio API [6] и MediaRecorder API [5]. 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` [3]. Данный вариант обеспечивает баланс между простотой интеграции (JS, события `onSpeechStart/onSpeechEnd`) и

ресурсной эффективностью (WebAssembly + оптимизированные вычисления ONNX Runtime Web [7]). 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` [2], после чего аудио подаётся в модуль распознавания речи (Yandex SpeechKit [12]). Затем на серверной стороне выполняется обнаружение и подтверждение ключевого слова активации (см. раздел 2.2). После подтверждения активации результат распознавания 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;
- сохранить качество распознавания русской речи благодаря использованию готового ASR-сервиса;
- минимизировать требования к рабочему месту врача, ограничившись браузером и микрофоном.

Клиент (браузер): Web Audio API → VAD → (активация) → сегмент команды → HTTP multipart/form-data (`file`)
Сервер (Python): приём `file` → (FFmpeg) → ASR (Yandex SpeechKit) → **обнаружение слова активации** → интерпретация (правила/LLM) → действие в МИС

Рис. 1: 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, **обнаружение и подтверждение слова активации**, а затем интерпретация команды в контролируемое действие внутри МИС.

3.1. Архитектура и поток данных

Поток обработки состоит из клиентской и серверной частей (рис. 1). 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» операции распознавания речи, выполняя на клиенте лёгкую предварительную фильтрацию (VAD). 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, получающая доступ к микрофону через Web Audio API [10]. Далее непрерывно

выполняется детекция речевой активности (VAD) с использованием `@ricky0123/vad-web` [8], исполняющего нейросетевую модель в формате ONNX через ONNX Runtime Web [7]. VAD выдаёт события начала и окончания речи, а также оценку вероятности речи. Пример событий VAD-модели:

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

Для завершения команды используется правило: «конец команды — 2 секунды тишины». Практически это реализуется параметром ожидания «тишины» после окончания речи (например, `redemptionFrames`, эквивалентный приблизительно двум секундам).

Пороговые параметры VAD настраиваются экспериментально под условия кабинета: `positiveSpeechThreshold` — порог включения речи (старт сегмента при превышении), `negativeSpeechThreshold` — порог выключения (переход в тишину), а `redemptionFrames` задаёт сколько кадров тишины подряд требуется для фиксации окончания сегмента.

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 выделяет сегменты речи и передаёт их на сервер, однако без активации они не интерпретируются как команды.

2. **Режим команды:** на сервере по результатам ASR подтверждается слово активации («старт»); после этого следующий сегмент обрабатывается как команда.

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 3.5).

Клиент реализует конечный автомат состояний:

```
STATE = MONITOR
onSpeechEnd(segment):
  if STATE == MONITOR:
     maybeWakeWord(segment)   // пробуем распознать "старт"
  if STATE == ARMED:
     sendToServer(segment)    // это команда
     STATE = MONITOR

onWakeWordDetected():
  STATE = ARMED               // следующий сегмент речи считаем кома
```

3.4. Wake word: варианты реализации и выбранный подход

В рамках работы были реализованы и проверены два подхода.

1) Wake word в браузере (локальный KWS). Прототип выполнял обнаружение ключевого слова на клиенте по аудиопризнакам (на уровне коротких сегментов, выделенных VAD). Практические испытания показали две проблемы:

- высокая доля пропусков (false negatives) в реальных условиях кабинета;
- необходимость обучения/калибровки под каждого врача (разные голоса и микрофоны), что неудобно для внедрения.

Поэтому данный вариант не был принят как основной.

2) Wake word на сервере (проверка по ASR). В итоговой версии используется серверная проверка: короткий сегмент, выделенный VAD, распознаётся ASR, после чего выполняется текстовая проверка на наличие ключевого слова (например, «старт»). Это увеличивает число обращений к ASR в режиме ожидания, но снижает риск пропусков и не требует обучения под каждого врача. Фактически режим ожидания работает как «VAD + короткие ASR-проверки», а после успешной активации следующий сегмент речи рассматривается как команда.

3.5. 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) [2];
3. распознавание речи посредством Yandex SpeechKit [12];
4. обнаружение и подтверждение слова активации по результатам распознавания (см. раздел 2.2);
5. 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 интентов, а для сопоставления свободных формулировок используется LLM-модуль. Результат возвращается клиенту в JSON и/или применяется на стороне МИС «СТОММИС».

3.6. Ограничение вывода языковой модели и валидация результата

Для интерпретации распознанного текста команды используется большая языковая модель `qwen3-235b-a22b-fp8`, доступная через совместимый с OpenAI API интерфейс (`base_url = https://rest-assistant.api.cloud.yandex.net/v1`, `project = folder_id`). Задача модели заключается не в генерации произвольного текста, а в преобразовании транскрипции команды в структурированное действие, пригодное для выполнения в МИС «СТОММИС».

Чтобы исключить некорректные или опасные действия, вывод модели жёстко ограничивается:

- допускается только формат **JSON** фиксированной структуры;
- тип действия выбирается только из заранее определённого множества интентов (белый список);
- параметры команды (слоты) проверяются на корректность типов и принадлежность к допустимым значениям;
- при неоднозначности модель обязана вернуть статус `needs_clarification` вместо выполнения действия.

Структура результата (контракт). Результат интерпретации команды представляется объектом следующего вида:

```
{
  "intent": "set_diagnosis | set_tooth_state | add_procedure | ...",
  "slots": { ... },
  "confidence": 0.0..1.0,
  "needs_clarification": true|false,
  "clarification_question": "..."
}
```

Схема допустимого вывода. Для обеспечения воспроизводимости и проверки корректности используется схема (упрощённый вариант):

```
intent: enum[ set_diagnosis, set_tooth_state, add_procedure, cancel,
slots:  object (зависит от intent)
confidence: number [0..1]
needs_clarification: boolean
clarification_question: string (optional, only if needs_clarificatio
```

Пример интерпретации команды. Пусть врач произносит: *«Старт. Зуб 26: кариес. Добавь пломбу.»*. После распознавания речи система получает текстовую транскрипцию и передаёт её в LLM вместе с контекстом (текущий пациент, выбранный врач, активная карта, доступные справочники).

Ожидаемый результат интерпретации:

```
{
  "intent": "set_tooth_state",
  "slots": {
    "tooth": "26",
    "state": "caries",
    "procedure": "filling"
  },
  "confidence": 0.83,
  "needs_clarification": false
}
```

Выбор и конфигурация языковой модели. В прототипе для семантической интерпретации команд используется большая языковая модель `qwen3-235b-a22b-fp8`, доступная через API облачной платформы (вызов через совместимый клиент OpenAI с заданным `base_url`). Выбор данной модели обусловлен качеством понимания русскоязычных команд и устойчивостью к вариативным формулировкам при сохранении приемлемой задержки. Параллельно проводится исследование альтернативной модели `gpt-oss-20b`; на момент написания работы результаты сравнения являются предварительными, поэтому в основной конфигурации прототипа фиксируется модель `qwen3-235b-a22b-fp8`.

Ограничение допустимого вывода модели. Чтобы исключить неконтролируемые действия, в прототипе применяется управляемая генерация, при которой поведение LLM контролируется через три канала: **(1) инструкцию** (системное сообщение), **(2) ввод** (контекст, добавляемый к пользовательской команде), и **(3) инструменты** (tool-вызовы для получения допустимых значений из предметной области). Сервер принимает результат *только* при успешной валидации JSON по схеме и принадлежности значений разрешённым множествам.

Инструкция (правила работы и формат вывода). Системная инструкция фиксирует назначение модели, запрещает свободный текст и задаёт формат результата согласно схеме из предыдущего пункта. Упрощённый фрагмент инструкции:

```
Ты - модуль интерпретации команд врача для МИС «СТОММИС».
Верни результат строго в JSON:
{ "intent": <строка>, "slots": <объект>, "confidence": <0..1>,
  "needs_clarification": <bool>, "clarification_question": <строка?>
Разрешены только intent из списка ALLOWED_INTENTS.
Значения slots должны быть взяты из справочников/допустимых множеств
Если данных недостаточно - needs_clarification=true и задай вопрос.
```

Управление вводом (RAG-контекст из поискового индекса). Перед обращением к LLM текст команды дополняется релевантным контекстом из поискового индекса: в индекс помещаются описания допустимых команд, фрагменты доменных справочников (процедуры, диагнозы, коды полей), а также примеры корректных формулировок. По транскрипции выполняется поиск, и в запрос модели добавляются только топ-k наиболее релевантных фрагментов (например, при $score > 0.4$). Это уменьшает вариативность интерпретации и позволяет связать распознанный текст с доменной моделью МИС.

```
ctx = vector_search(index_id, query=transcript, top_k=5, score_thr=0
prompt = build_prompt(system_rules, ctx, transcript)
```

Инструменты (tool) для зубной карты. Для сценариев заполнения зубной формулы LLM получает доступ к инструменту, возвращающему допустимые действия и параметры для конкретного клинического контекста (например, для выбранного зуба). Инструмент реализует поиск по доменному индексу допустимых команд и возвращает структурированные данные, которые используются моделью для выбора `intent` и заполнения `slots`.

```
tool get_allowed_tooth_actions(tooth_number) -> {
  "allowed_intents": [...],
  "allowed_states": [...],
  "allowed_procedures": [...],
  "field_codes": {...}
}
```

Псевдокод вызова LLM и валидации результата.

```
transcript = ASR(audio)
if not check_activation_word(transcript): return MONITOR

ctx = vector_search(domain_index, transcript, top_k=5, thr=0.4)

if mentions_tooth(transcript):
    allowed = get_allowed_tooth_actions(extract_tooth(transcript))
else:
    allowed = get_allowed_general_actions()

result = LLM(system_rules, ctx, transcript, tools=[allowed])

if not valid_json_schema(result): reject_and_retry()
if result.intent not in allowed.allowed_intents: reject()
if not slots_in_allowed_sets(result.slots, allowed): reject()
if result.needs_clarification: ask_user(result.clarification_questio
else apply_action_to_stommis(result.intent, result.slots)
```

3.7. Интеграция с МИС «СТОММИС» 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 МИС «СТОММИС»;
- минимизации количества действий врача в интерфейсе.

Командный режим позволяет врачу произносить короткие инструкции, которые конвертируются в конкретные изменения в карте. Например, команда «старт, кариес на 26 зубе» может быть интерпретирована как выбор диагноза «кариес» и установка соответствующего зуба в формуле.

3.8. 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 экспериментальной части — оценить качество выделения речевых сегментов и команд, а также ресурсные и временные характеристики прототипа по сравнению с базовым подходом непрерывного распознавания аудиопотока. Дополнительно выполнена апробация в реальных условиях стоматологической поликлиники СПб ГБУЗ «СП №8» (без детального раскрытия данных пациентов); результаты на текущем этапе нестабильны, но позволяют сделать предварительные выводы о применимости подхода.

6.1. 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?

RQ4: Какова суммарная задержка «команда → действие в МИС» и какие компоненты конвейера вносят основной вклад?

6.2. Метрики

Для оценки использовались следующие метрики:

- **Latency** (мс): время от окончания произнесения команды до получения результата на клиенте; отдельно фиксируются компоненты: сегментация, передача, ASR, интерпретация, применение действия.

- **Нагрузка клиента:** влияние на отзывчивость UI и оценка потребления CPU при включенном VAD.
- **Нагрузка сервера:** число обращений к ASR и средняя длительность обрабатываемых аудиосегментов.
- **Качество активации:** доля ложных срабатываний (FP) и пропусков (FN) для wake word.
- **Качество сегментации:** доля корректно выделенных командных сегментов (по ручной разметке на небольшом наборе записей).

6.3. Сценарии и базовая линия сравнения

Рассматривались два режима:

1. **Базовый режим (baseline):** непрерывная запись и распознавание всего аудиопотока приёма (или всех сегментов речи), без VAD-фильтрации на клиенте.
2. **Предложенный режим:** непрерывный VAD в браузере (лёгкая сегментация) + активация голосового режима (wake word), после чего на сервер отправляется только сегмент команды.

6.4. Результаты и обсуждение

Ключевые результаты сведены в табл. 1. В рамках пилотной апробации наблюдаемая задержка от конца произнесения команды до получения структурированного результата на сервере составляет порядка 2–3 секунд; значения зависят от качества сети и акустической обстановки и требуют дальнейшей стабилизации. Численные значения зависят от условий (микрофон, шум, сеть).

Таблица 1: Сравнение базового и предложенного режимов (оценка на типовом сценарии)

Показатель	**Baseline (непрерывное распознавание)**	**Предложенный режим (VAD + активация)**
Передаваемая длительность аудио за 10 минут приёма	≈600 с (весь поток)	≈18–30 с (6 команд по 3–5 с)
Число обращений к ASR	высокое (поток/-чанки)	низкое (по подтверждённым сегментам)
Средняя длина одного распознаваемого фрагмента	5–15 с (чанки потока)	3–5 с (командный сегмент)
Задержка «конец команды → действие в МИС»	распознавание идёт постоянно; ресурсы/стоимость — высокие	распознавание запускается только после активации; ресурсы/стоимость — низкие
Риск обработки нерелевантной речи	высокий (комментарии/шум идут в ASR)	сниженный (фильтрация VAD и триггер)
Требования к взаимодействию врача	требуется управление записью/режимом	голосовое управление во время лечения (без рук)

6.4.1. RQ1: ресурсоёмкость и стоимость

Ожидается, что предложенное решение значительно уменьшит T_{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 precision/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).

6.4.4. RQ4: задержка и нагрузка

Проводится серия замеров задержки и нагрузки на сервере при различных длительностях команд и числе обращений к ASR. Анализируется влияние параметров VAD и триггера на активацию голосового режима.

В ходе апробации в СПб ГБУЗ «СП №8» было отмечено, что основной источник нестабильности связан с условиями акустики и вариативностью речи врачей, что особенно влияет на качество активации по ключевому слову. Поэтому в текущей реализации предпочтение отдано серверной проверке wake word по ASR (см. раздел 3.4), обеспечивающей более низкую долю пропусков по сравнению с клиентским вариантом.

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.

7. Заключение

В рамках магистерской диссертации была разработана и исследована подсистема голосового управления для МИС «СТОММИС», ориентированная на использование *в процессе лечения* без необходимости взаимодействия с клавиатурой/мышью. Ключевая идея решения — многоэтапная обработка аудио: лёгкая сегментация речи в браузере (VAD) и включение распознавания речи только для релевантных сегментов, что снижает вычислительную и стоимостную нагрузку по сравнению с непрерывным распознаванием всего потока.

Основные результаты работы, соответствующие поставленным задачам:

- реализован клиентский модуль захвата аудио в браузере и выделения участков речи/тишины (VAD) в реальном времени;
- реализована логика выделения команд и завершения сегмента по паузе («2 секунды тишины»);
- исследованы варианты активации голосового режима без участия рук: кнопка (*push-to-talk*) и wake word; показано, что клиентский вариант wake word требует обучения и даёт высокий процент пропусков, поэтому в итоговой версии используется серверная проверка по ASR;
- реализована серверная обработка аудиосегментов: приём файла `file`, транскодирование (FFmpeg), распознавание речи через Yandex SpeechKit и интерпретация команды в структурированное действие внутри МИС с контролем допустимых операций;
- выполнена экспериментальная оценка и апробация прототипа, включая испытания в условиях СПб ГБУЗ «СП №8»; результаты подтверждают реализуемость подхода, при этом требуется дальнейшая стабилизация качества активации и интерпретации в условиях шума.

Направления дальнейшей работы включают расширение набора команд, улучшение устойчивости к шумам, а также уточнение механизма ограничения допустимых действий при интерпретации команд (в частности, формализацию схемы ответа и справочников предметной области).

Список литературы

[1] AI Mycroft, contributors. openWakeWord: open-source wake word detection. — 2025. — URL: https://github.com/dscripka/openWakeWord (дата обращения: 2026-01-04).

[2] FFmpeg Developers. FFmpeg Documentation. — 2025. — URL: https://ffmpeg.org/documentation.html (дата обращения: 2026-01-04).

[3] Ho Ricky. @ricky0123/vad-web: User Guide and API. — 2025. — URL: https://docs.vad.ricky0123.com/user-guide/api/ (дата обращения: 2026-01-04).

[4] MDN Web Docs. MediaDevices.getUserMedia(). — 2025. — URL: https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getUserMedia (дата обращения: 2026-01-04).

[5] MDN Web Docs. MediaRecorder API. — 2025. — URL: https://developer.mozilla.org/en-US/docs/Web/API/MediaRecorder (дата обращения: 2026-01-04).

[6] MDN Web Docs. Web Audio API. — 2025. — URL: https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API (дата обращения: 2026-01-04).

[7] Microsoft. ONNX Runtime Web. — 2025. — URL: https://onnxruntime.ai/docs/get-started/with-javascript.html (дата обращения: 2026-01-04).

[8] Ricky0123. @ricky0123/vad-web: Voice Activity Detection for the Web. — 2024. — URL: https://docs.vad.ricky0123.com/ (дата обращения: 2026-01-05).

[9] Silero Team. Silero VAD: pre-trained Voice Activity Detection models. — 2025. — URL: https://github.com/snakers4/silero-vad (дата обращения: 2026-01-04).

[10] W3C. Web Audio API Specification. — https://www.w3.org/TR/webaudio/. — 2024. — Дата обращения: January 12, 2026.

[11] WebRTC Project. WebRTC Native Code: Voice Activity Detector (VAD). — 2024. — URL: https://webrtc.googlesource.com/src/+/refs/heads/main/common_audio/vad/ (дата обращения: 2026-01-05).

[12] Yandex Cloud. SpeechKit: распознавание и синтез речи. — 2025. — URL: https://cloud.yandex.ru/services/speechkit (дата обращения: 2026-01-04).