

**Общество с ограниченной ответственностью
«РЖД – Цифровые пассажирские решения»**

*ОГРН 5167746236906 ИНН 9717045555
107078, г. Москва, ул. Новорязанская, д. 8А, стр. 2*

**ПРОГРАММА ДЛЯ ЭВМ
«АВТОПРОТОКОЛ»**

**Описание технической архитектуры программного
обеспечения**

На 12 листах

Москва, 2026

АННОТАЦИЯ

Настоящий документ содержит описание технической архитектуры программы для ЭВМ «Автопротокол» (далее – ПО) – совокупности важнейших решений об организации программной системы, а именно: выбор структурных элементов и их интерфейсов, с помощью которых составлена система; соединение выбранных элементов структуры; архитектурный стиль, направляющий организацию системы, ее элементы, интерфейсы и их взаимодействие.

Документ подготовлен в составе комплекта документов к заявлению о включении сведений о ПО в единый реестр российских программ для электронных вычислительных машин и баз данных и следует рассматривать совместно с документами «Описание функциональных характеристик программного обеспечения» (функциональный состав) и «Руководство по установке и эксплуатации программного обеспечения» (порядок развертывания).

СОДЕРЖАНИЕ

1. Общие положения	4
1.1. Термины и сокращения	4
2. Архитектурный стиль	5
3. Структурные элементы системы	7
4. Интерфейсы структурных элементов.....	8
4.1. Внешние интерфейсы	8
4.2. Внутренние интерфейсы	9
5. Соединение структурных элементов	9
5.1. Сетевая организация и изоляция	10
6. Жизненный цикл задачи обработки	10
7. Схемы развертывания	11
8. Заключение	12

1. Общие положения

ПО представляет собой серверную программную систему автоматической транскрибации аудио- и видеозаписей устных коммуникаций и интеллектуальной саммаризации (формирования итоговых документов) с веб-интерфейсом пользователя и административной панелью. Архитектура системы спроектирована исходя из следующих определяющих требований: обработка ресурсоемких вычислений искусственного интеллекта на графических ускорителях; эксплуатация в закрытом контуре организации без обращений к внешним сервисам; горизонтальная масштабируемость; отказоустойчивость длительных фоновых задач; развертывание как на одном сервере, так и распределенно.

1.1. Термины и сокращения

Термин / сокращение	Определение
ПО	Совокупность программ для ЭВМ, включая исходный текст, объектный код и сопроводительную документацию
Реестр	Единый реестр российских программ для электронных вычислительных машин и баз данных, ведение которого осуществляется в соответствии с постановлением Правительства Российской Федерации от 16.11.2015 № 1236
Правила	Правила формирования и ведения единого реестра российских программ для электронных вычислительных машин и баз данных, утвержденные постановлением Правительства Российской Федерации от 16.11.2015 № 1236
Правообладатель	ООО «РЖД – Цифровые пассажирские решения» – обладатель исключительного права на ПО
Транскрибация	Автоматическое преобразование речи, содержащейся в аудио- или видеозаписи, в текст (расшифровка записи)
Саммаризация	Автоматическое формирование краткого структурированного изложения (протокола, выжимки, перечня поручений) на основе полного текста расшифровки
Диаризация	Автоматическое разделение речевого потока по говорящим (спикерам) с привязкой реплик к конкретному говорящему
ASR	Automatic Speech Recognition – автоматическое распознавание речи
LLM	Large Language Model – большая языковая модель, применяемая для саммаризации
ИИ	Искусственный интеллект

API	Application Programming Interface – программный интерфейс взаимодействия приложений
REST	Representational State Transfer – архитектурный стиль построения программных интерфейсов поверх протокола HTTP
WebSocket	Протокол двунаправленного обмена сообщениями поверх TCP-соединения, применяемый для доставки статусов обработки в реальном времени
GPU	Graphics Processing Unit – графический процессор (ускоритель вычислений), используемый для инференса моделей ИИ
СУБД	Система управления базами данных
S3	Протокол объектного хранилища данных (Simple Storage Service), реализуемый в составе ПО хранилищем MinIO или иным совместимым хранилищем
Брокер сообщений	Программное средство асинхронного обмена сообщениями между компонентами (поддерживаются RabbitMQ и Apache Kafka)
Контейнер	Изолированная среда исполнения компонента ПО, формируемая средствами контейнеризации Docker
Закрытый контур	Режим эксплуатации, при котором обработка данных выполняется исключительно на вычислительных мощностях эксплуатирующей организации без обращения к внешним публичным сервисам
Микросервисная архитектура	Архитектурный стиль, при котором система составляется из небольших автономных сервисов, взаимодействующих по сети через формализованные интерфейсы
Событийно-ориентированное взаимодействие	Асинхронный обмен сообщениями (событиями, задачами) между компонентами через брокер сообщений
Воркер	Фоновый вычислительный компонент, потребляющий задачи из очереди и выполняющий их обработку
DLQ	Dead Letter Queue – очередь недоставленных (повторно ошибочных) сообщений
Инференс	Исполнение обученной модели машинного обучения
Compose-профиль	Именованный набор компонентов в декларативной конфигурации Docker Compose, определяющий схему развертывания

2. Архитектурный стиль

Организация Программного обеспечения направляется сочетанием следующих архитектурных стилей и принципов:

– **микросервисная архитектура** – система составлена из автономных сервисов, каждый из которых решает одну функциональную задачу (прием запросов, транскрибация, саммаризация, отправка почты, статистика),

поставляется отдельным контейнером Docker и может обновляться и масштабироваться независимо;

– **событийно-ориентированное (асинхронное) взаимодействие** – сервисы обмениваются задачами и событиями через брокер сообщений (durable-очереди с персистентными сообщениями); длительные вычисления не блокируют пользовательские запросы, а сбой отдельного компонента не приводит к потере задач;

– **разделение управляющего и вычислительного контуров** – легкие управляющие сервисы (шлюз, оркестратор, почта, статистика) отделены от ресурсоемких вычислительных компонентов (GPU-воркер транскрибации, сервер инференса языковой модели), что позволяет масштабировать вычисления по числу графических ускорителей и серверов;

– **хранение состояния вне сервисов (stateless-сервисы)** – медиафайлы, статусы и результаты размещаются в объектном S3-хранилище, метаданные и настройки – в реляционной СУБД; сами сервисы не хранят состояние и допускают перезапуск и горизонтальное масштабирование без потери данных;

– **контейнеризация и декларативное развертывание** – все компоненты поставляются образами Docker и управляются конфигурацией Docker Compose с профилями, определяющими схему развертывания (одноузловая, распределенная);

– **закрытый контур** – все вычисления, включая инференс моделей искусственного интеллекта, выполняются на оборудовании эксплуатирующей организации; обращения к внешним публичным сервисам конструктивно исключены;

– **отказоустойчивость** – автоматический повтор задач при первичной ошибке, изоляция «ядовитых» сообщений в очереди недоставленных (DLQ), сторожевые таймеры, автоматический перезапуск зависших компонентов, проверки живости (healthchecks);

– **наблюдаемость** – сквозные события статистики, телеметрия воркеров, централизованное журналирование, мониторинг состояния в административной панели.

3. Структурные элементы системы

Состав структурных элементов (компонентов) Программного обеспечения и их назначение приведены в таблице.

Компонент	Назначение	Использование GPU
frontend	Веб-интерфейс пользователя и административная панель; веб-сервер Nginx: терминирование TLS, маршрутизация запросов, проксирование API и потоковых ответов языковой модели	нет
api-gateway	Точка входа REST/WebSocket: прием и валидация файлов, ограничение частоты запросов, ведение статусов задач, административный API, генерация документов DOCX	нет
Воркер транскрибации (GPU-воркер)	Конвейер интеллектуальной обработки записи: нормализация громкости, нейросетевое шумоподавление, диаризация, распознавание речи, пословное выравнивание, согласование говорящих	да
orchestrator	Координатор саммаризации: пакетный отбор задач, взаимодействие с языковой моделью, каскад отказоустойчивых стратегий (map-reduce, повтор без reasoning, усечение), запись результатов	опосредованно
vllm	Сервер инференса больших языковых моделей с OpenAI-совместимым программным интерфейсом (модель настраивается эксплуатирующей организацией)	да
email-sender	Формирование и отправка электронных писем с результатами (SMTP, STARTTLS), генерация вложений Protocol.docx и Transcription.html	нет
stats-service	Прием событий статистики и телеметрии, запись в СУБД, обслуживание срока хранения метрик	нет
Брокер сообщений	RabbitMQ (в составе поставки) либо Apache Kafka – асинхронный обмен задачами и событиями между компонентами	нет

MinIO (S3)	Объектное хранилище медиафайлов, статусов, результатов, шаблонов и снапшота настроек; допустимо любое S3-совместимое хранилище	нет
PostgreSQL	Реляционная СУБД: справочники типов документов, runtime-настройки, статистика, журнал аудита, телеметрия	нет

Инфраструктурные элементы (брокер, объектное хранилище, СУБД, сервер инференса) входят в состав поставки в виде контейнеров и отдельной установки не требуют; при наличии у эксплуатирующей организации собственных экземпляров соответствующих средств допускается подключение к ним.

4. Интерфейсы структурных элементов

4.1. Внешние интерфейсы

Интерфейс	Протокол / стандарт	Назначение
Веб-интерфейс пользователя и административная панель	HTTP/HTTPS (TLS)	Работа пользователей и администраторов через веб-браузер; специального клиентского ПО не требуется
Открытый REST API	HTTP/HTTPS, JSON	Постановка задач обработки (POST /api/transcribe), получение статусов и результатов (GET /api/status/{id}, GET /api/result/{id}, DOCX), справочники; условные запросы (ETag/304)
Канал статусов реального времени	WebSocket	Доставка стадий обработки и позиции в очереди; восстановление сессии в течение 60 минут
Административный API	HTTP/HTTPS, Basic Auth	Управление типами документов, шаблонами, runtime-настройками, просмотр статистики и телеметрии
Интерфейс языковой модели	OpenAI-совместимый API (HTTP, SSE)	Доступ доверенных внутренних клиентов организации к локальной языковой модели через проксирование веб-сервером
Почтовый выход	SMTP (STARTTLS)	Отправка результатов на адреса получателей через SMTP-сервер организации

4.2. Внутренние интерфейсы

Интерфейс	Протокол / стандарт	Назначение
Очереди задач и событий	AMQP 0-9-1 (RabbitMQ) / Apache Kafka	Задачи транскрибации, саммаризации, отправки почты; события статистики и телеметрии; контракты сообщений формализованы и переносимы между брокерами
Объектное хранилище	S3 API (HTTP)	Чтение и запись медиафайлов, статусов, результатов, шаблонов DOCX, снапшота настроек
Реляционная СУБД	SQL (PostgreSQL, asyncpg)	Справочники, настройки, статистика, аудит, телеметрия
Санитарная перекодировка медиа	ffmpeg (внутренний вызов)	Приведение входных файлов к унифицированному WAV с отбрасыванием контейнеров и метаданных

5. Соединение структурных элементов

Схема соединения элементов (направление инициирования взаимодействия, протокол, характер обмена) приведена в таблице.

Источник	Приемник	Протокол	Характер
Браузер пользователя	frontend (Nginx)	HTTPS, WebSocket	синхронный
frontend	api-gateway	HTTP (проксирование)	синхронный
frontend	vllm	HTTP, SSE (проксирование /v1)	синхронный, потоковый
api-gateway	Объектное хранилище (S3)	S3 API	синхронный
api-gateway	Брокер сообщений	AMQP / Kafka (публикация задач)	асинхронный
api-gateway	PostgreSQL	SQL	синхронный
Воркер транскрибации	Брокер сообщений	AMQP / Kafka (потребление, публикация)	асинхронный
Воркер транскрибации	Объектное хранилище (S3)	S3 API (чтение аудио, запись результата)	синхронный
orchestrator	Брокер сообщений	AMQP / Kafka (потребление, публикация)	асинхронный

orchestrator	vllm	OpenAI-совместимый API (HTTP)	синхронный, пакетный
orchestrator	Объектное хранилище (S3)	S3 API	синхронный
email-sender	Брокер сообщений	AMQP / Kafka (потребление)	асинхронный
email-sender	SMTP-сервер организации	SMTP (STARTTLS)	синхронный
stats-service	Брокер сообщений	AMQP / Kafka (потребление событий)	асинхронный
stats-service	PostgreSQL	SQL	синхронный

Все связи между компонентами замыкаются во внутренних сетях развертывания; единственные внешние соединения – входящие подключения браузеров и интеграционных клиентов к веб-серверу и исходящие подключения к SMTP-серверу организации (отключаемы).

5.1. Сетевая организация и изоляция

Компоненты разделены по трем внутренним сетям Docker: frontend (веб-сервер и шлюз), backend (шлюз, воркеры, брокер, хранилище, СУБД) и llm (оркестратор и сервер инференса). Внешний доступ предоставляется только для портов веб-сервера; административная панель защищена аутентификацией; при распределенном развертывании межсерверное взаимодействие ограничивается внутренними портами брокера и хранилища.

6. Жизненный цикл задачи обработки

Сквозной сценарий обработки записи через соединенные элементы системы:

- прием файла шлюзом (валидация формата, размера, согласий; ограничение частоты запросов);
- санитарная перекодировка в унифицированный WAV и размещение аудио и статуса в объектном хранилище;
- публикация задачи транскрибации в брокер; отображение позиции в очереди;

- многоступенчатая обработка GPU-воркером (нормализация → шумоподавление → диаризация → распознавание → выравнивание → согласование говорящих) с записью промежуточных статусов;
- публикация задачи саммаризации; пакетный отбор задач оркестратором и обращение к языковой модели с каскадом отказоустойчивых стратегий;
- запись итогового результата в хранилище; публикация задачи отправки почты;
- формирование и отправка письма с вложениями; доставка статуса «готово» в браузер по WebSocket;
- регистрация событий статистики на каждой стадии; автоматическое удаление исходных файлов и результатов по истечении срока хранения.

7. Схемы развертывания

Декларативная конфигурация с профилями обеспечивает следующие схемы развертывания без изменения исходного текста ПО:

- **одноузловая** – все компоненты, включая GPU-воркер и сервер инференса, на одном сервере с графическим ускорителем;
- **распределенная** – управляющее ядро (веб-сервер, шлюз, брокер, хранилище, СУБД, почта, статистика) на одном сервере; вычислительные компоненты (GPU-воркеры, сервер инференса) – на выделенных GPU-серверах, подключаемых по внутренним портам брокера и хранилища;
- **многоускорительная** – на сервере с несколькими графическими ускорителями языковая модель распределяется средствами тензорного параллелизма, воркеры транскрибации запускаются по одному на ускоритель.

Порядок развертывания, требования к среде функционирования и процедуры обновления приведены в документе «Руководство по установке и эксплуатации программного обеспечения».

8. Заключение

Техническая архитектура ПО – микросервисная, событийно-ориентированная, контейнеризованная система с разделением управляющего и вычислительного контуров, хранением состояния во внешних хранилищах и формализованными интерфейсами (REST, WebSocket, S3, SQL, AMQP/Kafka, OpenAI-совместимый интерфейс инференса, SMTP). Принятые архитектурные решения обеспечивают отказоустойчивость обработки, горизонтальную масштабируемость вычислений искусственного интеллекта, эксплуатацию в закрытом контуре организации и гибкость схем развертывания – от одного сервера до распределенной конфигурации с несколькими GPU-серверами.