УТВЕРЖДАЮ

Руководитель проекта

_____ Е.Ю.Тихомирова

МП

«31» июля 2022 г.

СОГЛАСОВАНО

Директор

ООО «Войс Коммьюникэйшн»

_____ С.В.Полковникова

«31» июля 2022 г.

Система обеспечения безопасности в помещениях на базе речевых технологий и искусственных нейронных сетей

Архитектура системы

2022

Содержание

Перечень терминов и сокращений

Термин/Сокращение	Определение
API (Application Programming Interface)	Описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой
FIFO (First In, First Out)	Очередь FIFO представляет собой циклический буфер, где будут храниться помещаемые в очередь данные. Есть два указателя: указатель на «голову» очереди (head) и указатель на «хвост» очереди (tail). Новый записываемый элемент помещается в ячейку, на которую указывает head, затем этот указатель перемещается на следующую ячейку памяти буфера. Выбираются элементы из очереди по указателю tail, после того, как элемент выбран из очереди, указатель tail так же передвигается вперед к следующей ячейке
gRPC (Remote Procedure Calls)	Система удалённого вызова процедур с открытым исходным кодом, которая предоставляет такие функции как аутентификация, двунаправленная потоковая передача и управление потоком, блокирующие или неблокирующие привязки, а также отмена и тайм-ауты
HTTP (HyperText Transfer Protocol)	Протокол передачи гипертекста – протокол прикладного уровня передачи данных
ID	Уникальный признак объекта, позволяющий отличать его от других объектов
IP-микрофон	Самостоятельное сетевое интеллектуальное устройство, имеющее свой IP адрес, встроенный веб-интерфейс и питание от локальной сети по PoE, предназначенное для преобразования акустического сигнала в цифровой аудио поток, для последующей его передачи по каналам Ethernet и Internet с использованием различных протоколов на аудио сервер или систему видеонаблюдения для записи, обработки и хранения
JSON (JavaScript Object Notation)	Текстовый формат обмена данными, основанный на JavaScript.
JWT-токен	Открытый стандарт (RFC 7519) для создания токенов доступа, основанный на формате JSON. Используется для передачи данных для аутентификации в клиент-серверных приложениях. Токены создаются сервером, подписываются секретным ключом и передаются клиенту, который в дальнейшем использует данный токен для подтверждения своей личности
MQ (Messages Queye)	Архитектура и ПО промежуточного уровня, которое занимается сбором, хранение и маршрутизацией (распределением) сообщений между компонентами
Power On Self Test	Процедура самопроверки при включении
REST (Representational State Transfer)	Архитектурный стиль взаимодействия компонентов распределённого приложения в сети.
RTSP (Real Time Streaming Protocol)	Протокол прикладного уровня для настройки и управления доставкой данных со свойствами в реальном времени
URL (Uniform Resource Locator)	Система унифицированных адресов электронных ресурсов, или единообразный определитель местонахождения ресурса (файла)

Термин/Сокращение	Определение
БД	База данных
Бэкенд Системы	Программно-аппаратная часть Системы
Инстанс	Запущенный на компьютере экземпляр приложения, сервиса
Нейронные сети	Одно из направлений искусственного интеллекта, цель которого смоделировать аналитические механизмы, осуществляемые человеческим мозгом
Протокол AWS S3 (Amazon Simple Storage Service)	Онлайн веб-служба, предлагаемая Amazon Web Services, предоставляет возможность хранения и получения любого объёма данных в любое время из любой точки сети (файловый хостинг)
Система	SecureTrack.AI. Разрабатываемая информационная система обеспечения безопасности в помещениях на базе речевых технологий и искусственных нейронных сетей
Фронтенд Системы	Клиентская сторона пользовательского интерфейса к программно-аппаратной части Системы
Чанк	Блок аудиоданных определенного формата
Эндпоинт	Конечное устройство

1 Общие сведения

1.1 Наименование системы

Полное наименование системы: Система обеспечения безопасности в помещениях на базе речевых технологий и искусственных нейронных сетей.

Краткое наименование системы: SecureTrack.AI, Система.

2 Архитектура системы

Архитектура Системы состоит из следующих составляющих (Рисунок 1):

- графический интерфейс, предоставляет рабочий кабинет для администратора и оператора Системы;
- подсистема записи, осуществляет взаимодействие с микрофонным оборудованием и дальнейшую передачу звукового трафика внутри Системы;
- сервисы, осуществляют детектирование событий, архивирование аудиозаписей, оповещение, логирование, а также ряд вспомогательных функций;
- бэкенд системы, предоставляет REST API для графического интерфейса и других компонентов Системы, в том числе, сервисов; реализует большую часть бизнес-логики Системы.

Рисунок 1 – Общая архитектура Системы

Система также включает в себя базу данных, файловое хранилище и брокер сообщений, которые представляют собой продукты с открытым исходным кодом и распространяющиеся под свободной лицензией:

- база данных Postgresql13;

- файловое хранилище S3;
- брокер сообщений RabbitMQ.

Все подсистемы работают с одной и той же БД, с одними и теми же таблицами БД. Такой подход позволяет уменьшить объём взаимодействия между подсистемами по REST протоколу и упрощает механизмы обеспечения консистентности данных.

2.1 Сервис BackEnd

Бэкенд Системы взаимодействует с несколькими другими элементами Системы (Рисунок 2).

Бэкенд имеет только REST API интерфейс, который делится на два блока:

- блок работы с приложением графического интерфейса. Эндпоинты данного раздела позволяют графическому интерфейсу взаимодействовать с данными в Системе, получать и обновлять данные и настройки.
- блок для коллбэков, приходящих от детектора событий.

Для простоты на схеме не указана подсистема авторизации, которая описана в отдельном разделе.

Если Система должна послать сообщение во внешнюю систему, бэкенд отправляет соответствующий запрос на REST API сервиса оповещений.

Если необходимо дополнительное управление инстансами подсистемы записи, бэкенд может отправить соответствующие команды через MQ-брокер сообщений.

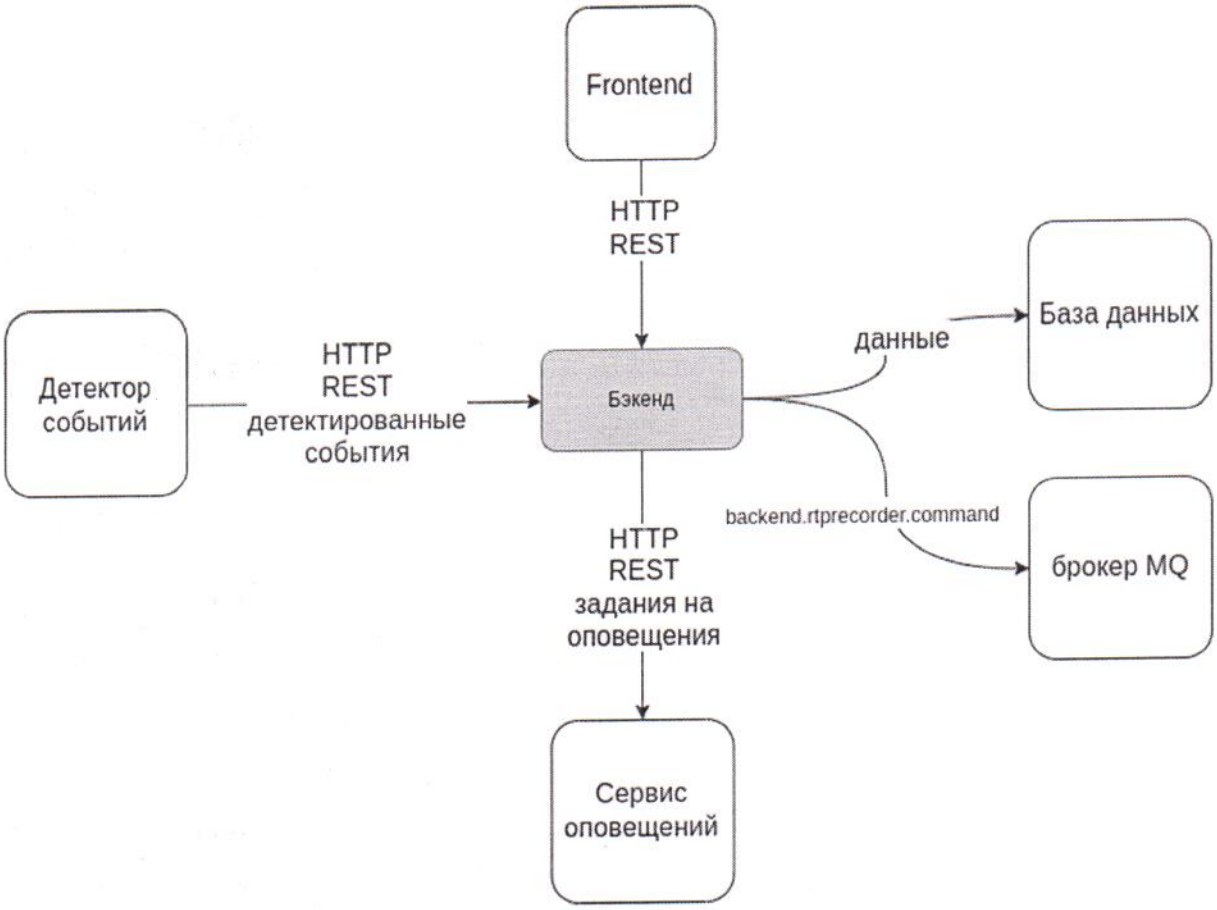

Рисунок 2 – Взаимодействие BackEnd с другими компонентами

Сервис бэкенда поддерживает горизонтальное масштабирование путём запуска дополнительных инстансов сервиса. Для этого системный администратор должен предусмотреть механизмы балансировки HTTP запросов от смежных подсистем ко всем инстансам бэкенда.

2.2 Детектор событий

Схема работы детектора событий представлена на рисунке ниже (Рисунок 3).

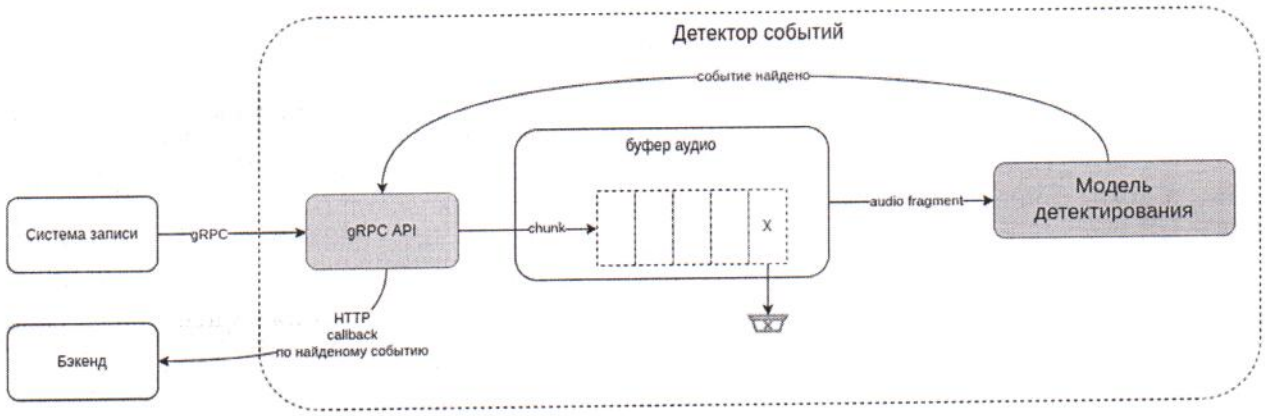

Рисунок 3 – Схема работы детектора событий

Модуль gRPC API предоставляет gRPC-интерфейс для системы записи. Как только с помощью микрофона начинают прослушивание, система записи инициализирует соединение с детектором событий и начинает отправлять в него чанки аудиоданных. Размер чанка настраивается, но оптимальным значением можно считать чанк длительностью 250 мс. На каждый микрофон инициализируется по одному gRPC-соединению. Таким образом, необходимо предусмотреть, чтобы детектор событий успевал обрабатывать то же количество соединений, что и количество активных микрофонов, зарегистрированных в Системе.

В момент старта компонента детектор инициализирует модель детектирования, после чего он становится готов к обработке аудиопотоков.

После того, как система записи инициализировала gRPC-соединение, детектор инициализирует буфер аудио по типу FIFO (по одному буферу на соединение), т.е. после того как в буфер был отправлен очередной чанк, самый старый чанк удаляется из него. Буфер всегда содержит последнюю секунду аудиопотока, реализуя функцию «скользящего окна». После удаления устаревшего чанка содержимое всего буфера анализируется моделью детектирования.

Если модель детектировала какой-то значимое событие, она сообщает об этом главному процессу декодера (на схеме обозначен как gRPC API). Главный процесс производит постобработку этих сигналов, стараясь исключить множественные срабатывания на одном и том же событии. Если главный процесс уверен в том, что было детектировано новое событие, он отправляет отклик на URL бэкенда. Бэкенд принимает данное сообщение и обрабатывает его.

Детектор событий может масштабироваться путём запуска нескольких инстансов параллельно. В этом случае инженеру Системы необходимо предусмотреть механизмы балансировки gRPC-соединений.

2.3 Подсистема записи

Система записи поддерживает работу с IP-микрофонами по протоколу RTSP. Данный протокол оптимально подходит для использования в системах, работающих с мультимедийными данными, поскольку позволяет удалённо управлять потоком данных с медиасервера через отправку команд на него. Простота протокола обеспечила его поддержку в большинстве IP-микрофонов, а значит, подсистема записи потенциально сможет работать с подавляющим большинством IP-микрофонов, существующих на рынке на текущий день.

Задачи подсистемы:

- собирать медиатрафик с IP-микрофонов и транслировать его в подсистему детектирования событий;

– буферизировать принимаемый медиапоток, агрегировать его в отрезки указанной длины и отправлять их через MQ-шину с целью их последующего архивирования;
– оповещать Систему о возникающих проблемах с микрофонами или связью с подсистемой детектирования.

Основные компоненты подсистемы записи представлены на рисунке ниже (Рисунок 4).

Рисунок 4 – Основные компоненты подсистемы записи

Модуль DB reader осуществляет всё взаимодействие с базой данных:

– проверяет базу данных на наличие свободных микрофонов;
– меняет статусы микрофонов в базе данных;
– обновляет в базе данных таймеры, показывающие активность инстанса подсистемы записи и активность занятых микрофонов.

Модуль Master (центральный элемент подсистемы) осуществляет следующие операции:

– взаимодействует с IP-микрофонами;
– поднимает RTSP-сессию;
– инициализирует воркеры gRPC и буферы медиатрафика queue.

Модуль MQ publisher взаимодействует с брокером MQ:

– один раз в N минут агрегирует отрезок аудио из буфера queue и отправляет его в очередь MQ с целью его дальнейшего архивирования;
– принимает команды на сброс буфера, освобождение или отключение микрофонов, отключение инстанса;
– отправляет в специальные очереди логи и оповещения.

Модуль gRPC осуществляет взаимодействие с сервисом детектирования событий:

– принимает чанки из очереди queue;

- при необходимости перекодирует чанки в формат, подходящий сервису детектирования;
- инициализирует сессию с сервисом детектирования по протоколу gRPC;
- отправляет чанки в требуемом формате на сервис детектирования в течение всего периода прослушивания IP-микрофона.

Таблицы базы данных, с которыми работает подсистема записи, представлены на рисунке ниже (Рисунок 5).

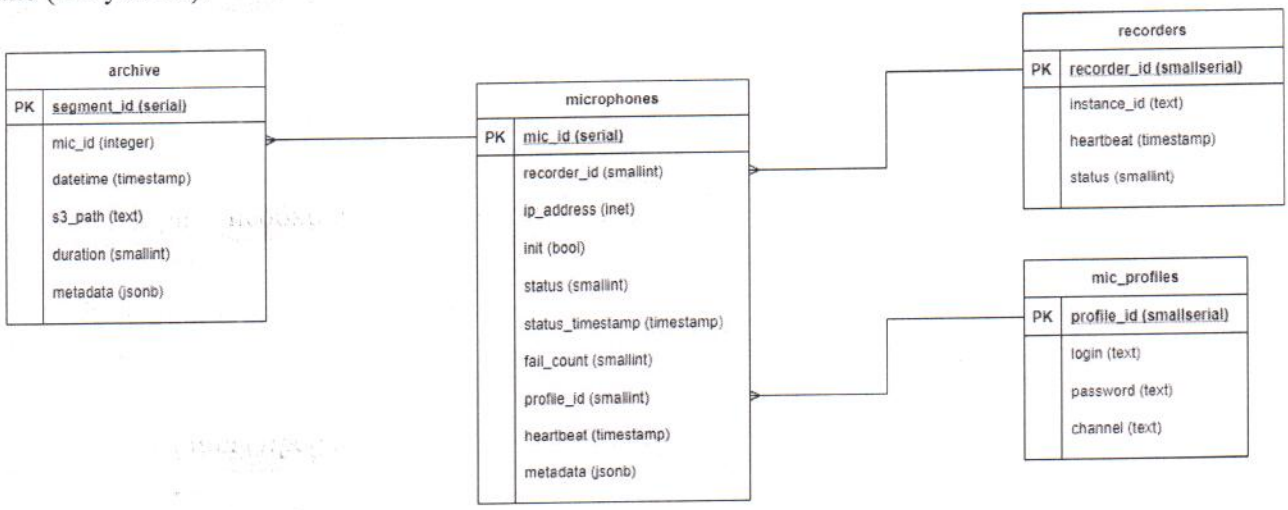

Рисунок 5 – Таблицы БД для взаимодействия с подсистемой записи

Выделение микрофонных профилей в таблицу mic_profiles позволяет описать общие для всех микрофонов данные и уменьшить трудоёмкость этапа регистрирования микрофонов в Системе. Кроме того, потенциально это позволяет скрыть секретные данные о логинах и паролях микрофонов от широкого круга пользователей, разрешив доступ к данным таблицы только определённым ролям пользователей в Системе.

Таблица recorders содержит перечень инстансов с указанием их ID и статуса. При старте инстанс проверяет, что его ID есть в БД, иначе останавливает запуск. В течение работы инстанс обновляет поле recorders.heartbeat, что позволяет отслеживать его работоспособность. Поле recorders.status позволяет включать или выключать инстансы, что бывает полезно при их обслуживании. Корректное выключение инстанса позволяет аккуратно освободить микрофоны и перевести их на другие инстансы подсистемы.

Таблица microphones содержит зарегистрированные в системе микрофоны, их статусы и привязку к инстансам (если они в данный момент прослушиваются ими).

Таблица archive содержит информацию об архивированных отрезках аудио. Эти отрезки можно будет прослушать в интерфейсе пользователя, в случае, если оператору это будет необходимо.

Алгоритм работы подсистемы позволяет запускать несколько инстансов подсистемы записи одновременно. В этом случае свободные микрофоны распределяются равномерно по всем работающим инстансам. Если один из инстансов планово или аварийно завершил работу и высвободил микрофоны, то другие инстансы подсистемы записи забирают эти микрофоны в работу. Таким образом, реализован принцип горизонтальной масштабируемости и отказоустойчивости.

2.4 Сервис Watcher

Сервис Watcher предназначен для отслеживания зависших процессов в Системе. В основном это касается подсистемы записи. Она спроектирована таким образом, чтобы иметь возможность работы для одновременно сразу нескольких инстансов с целью обеспечения отказоустойчивости и распределения нагрузки.

На рисунке (Рисунок 4) представлены таблицы, с которыми работает подсистема записи. С этими же таблицами работает и сервис Watcher.

Поля microphones.heartbeat и recorders.heartbeat постоянно обновляются в процессе активности системы записи. Первое поле относится к микрофону и показывает, что запись с микрофона активна. Второе поле относится к самой подсистеме записи.

Прекращение обновления данных полей одновременно со статусом ACTIVE является признаком ошибки в Системе. Watcher отслеживает таймауты обновления данных полей и создаёт оповещение для администратора Системы, если обнаруживает приостановку активности.

Таким образом, администратор Системы обратит внимание на проблему и сможет отработать её.

2.5 Сервис архивирования аудиофрагментов

Схема работы сервиса архивирования аудиофрагментов представлена на рисунке ниже (Рисунок 6).

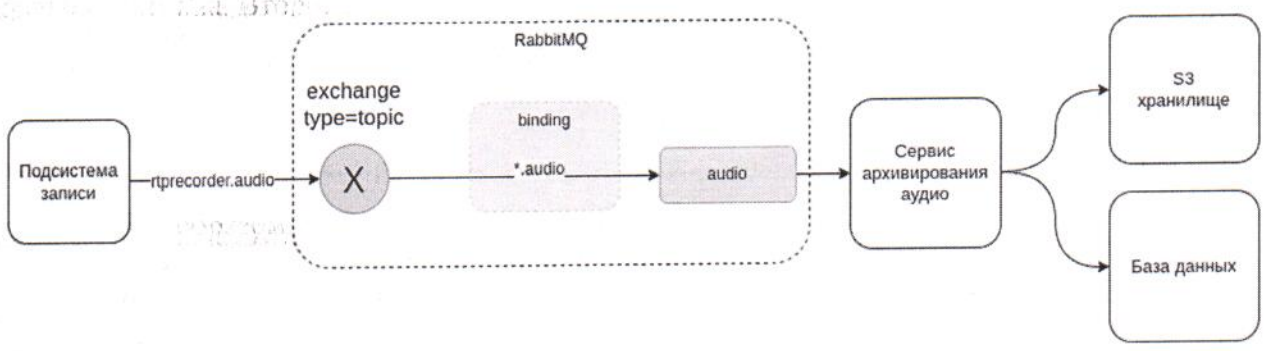

Рисунок 6 – Сервис архивирования аудиофрагментов

Сервис принимает сообщения, содержащие аудиофрагменты с метаданными по MQ-шине, шифрует аудиофрагмент, передает его в объектное хранилище, совместимое с протоколом AWS S3, а затем передает данные по этому аудиофрагменту в БД Системы.

Шифрование аудиофайлов ключом, сформированным на основе алгоритма AES-256, позволяет добиться большей безопасности, надёжно защищая данные, даже если злоумышленники получат доступ к хранилищу. Алгоритм AES-265 на текущий момент считается одним из наиболее криптостойких симметричных алгоритмов шифрования. Надёжный алгоритм шифрования позволяет хранить архив даже в облачных системах хранения.

2.6 Аудит действий пользователей

Поскольку Система является многопользовательской, важно отслеживать любые изменения в настройках, производимых пользователями в Системе.

Каждое действие пользователя, приводящее к изменению настроек, сохраняется в журнале аудита.

Поскольку все изменения производятся через бэкенд Системы, то сохранение логов аудита – это его задача.

2.7 Сервис оповещения

Подсистема оповещения является интеграционным звеном между системой детектирования событий и внешними системами. Подсистема оповещения является буфером сообщений и старается доставить все сообщения нужным адресатам. В алгоритм работы подсистемы входит обработка ошибок отправки, тайм-аутов соединений и хранение данных по каждому сообщению и адресату.

Более подробно архитектура данной подсистемы будет прорабатываться на следующем этапе работ.

2.8 Графический интерфейс

Графический интерфейс Системы включает в себя следующие страницы:

- страница входа в Систему;
- главная страница;
- карты;
- события;

– отчёты;
– настройки.

Ниже представлены макеты интерфейса (Рисунок 7 – Рисунок 9). Детально интерфейс будет дорабатываться на следующих этапах.

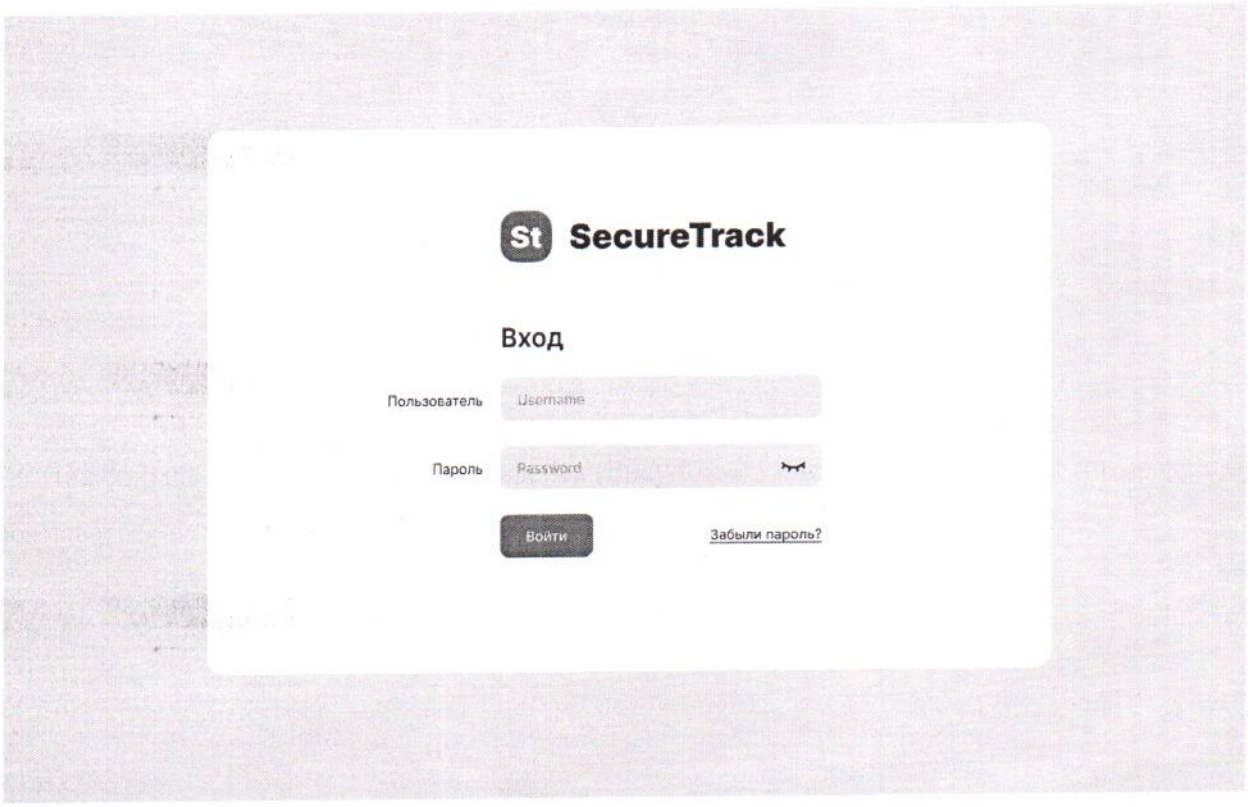

Рисунок 7 – Страница входа

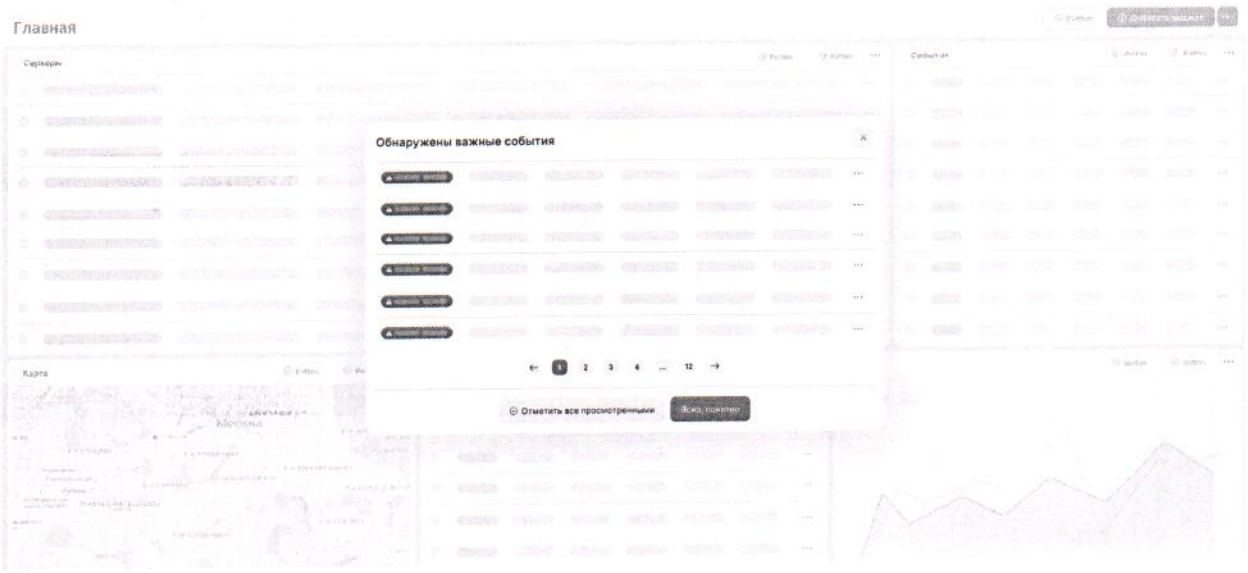

Рисунок 8 – Главная страница

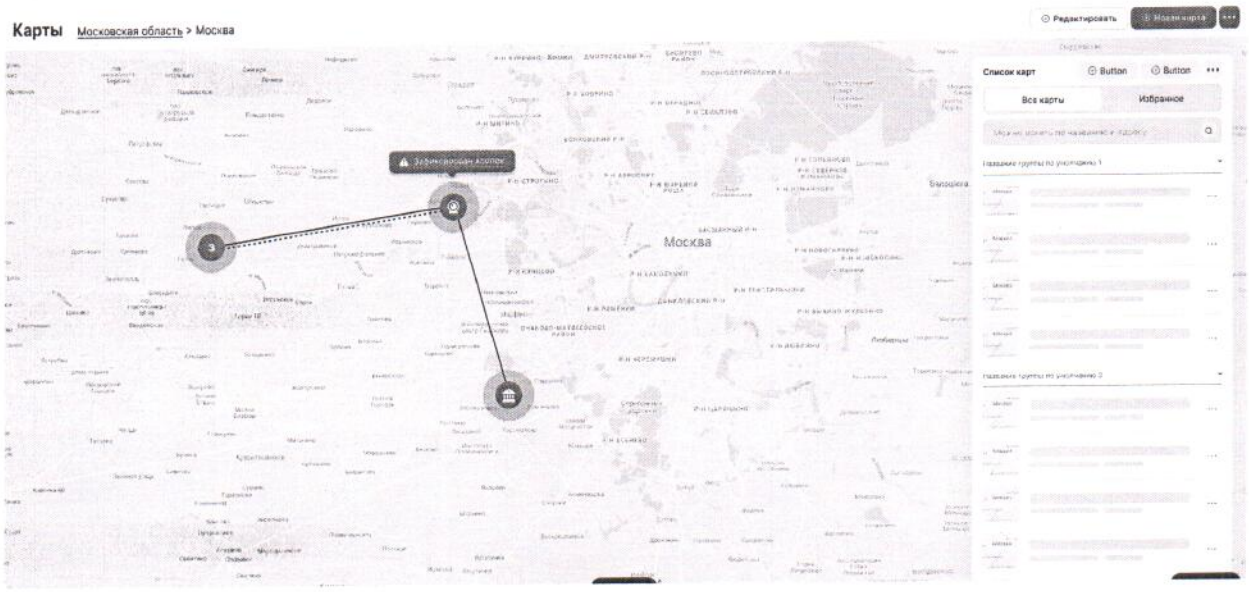

Рисунок 9 – Карты

2.9 Сервис авторизации и аутентификации

Данный сервис предназначен для аутентификации пользователей и авторизации их действий в Системе. Любое действие пользователя в графическом интерфейсе Системы порождает запрос от приложения графического интерфейса к REST API бэкенда Системы. Данные запросы и авторизуются с помощью указанной подсистемы.

Схема взаимодействия фронтенд-, бэкенд-приложений и подсистемы авторизации и аутентификации действий пользователя представлена на рисунке ниже (Рисунок 10).

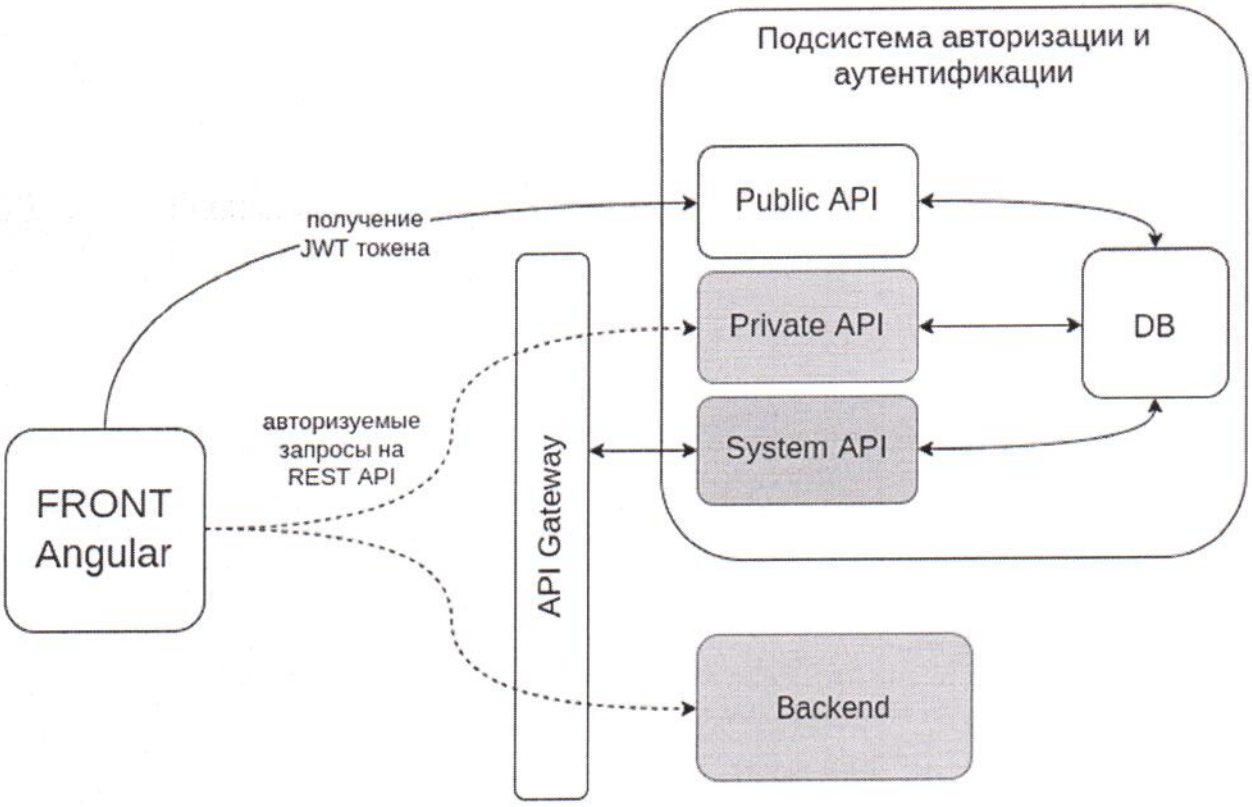

Рисунок 10 – Авторизация и аутентификация

Подсистема имеет три группы эндпоинтов REST API:

- Public API, необходим для неаутентифицированных пользователей. Он позволяет получить JWT-токен по связке логин-пароль пользователя;
- Private API, доступен только авторизованным пользователям. Обычно это пользователи с ролью администратора Системы. Данная группа эндпоинтов позволяет управлять пользователями Системы;
- System API, необходим для взаимодействия с API Gateway. Пользователи никак не взаимодействуют с System API.

Порядок аутентификации пользователя:

- клиентское приложение отправляет запрос на какой-либо эндпоинт бэкенда Системы;
- если JWT-токен клиента невалиден или его не существует, то на запрос возвращается код ошибки HTTP 401. При получении хотя бы одного такого ответа, фронтенд должен сбросить сессию и отобразить пользователю страницу авторизации;
- пользователь вводит логин и пароль, которые отправляются на Public-эндпоинт /api/auth/login. Если связка логин-пароль верная, то клиенту выдаются два токена – JWT и Refresh;

- после успешной авторизации фронтенд должен запросить имя роли пользователя и полный список разрешений для его роли через эндпоинт /auth/permissions. Перечень разрешений позволяет Front-приложению определить, какие элементы интерфейса отображать, а на какие установить права «только для чтения»;
- если JWT-токен невалиден, но имеется выданный ранее Refresh-токен, то фронтенд должен попытаться получить новый JWT токен без необходимости нового ввода логина-пароля.

Порядок авторизации действий:

- все обращения к эндпоинтам бэкенда Системы идут через API Gateway;
- API Gateway отправляет авторизационный запрос на сервис авторизации, передавая в нём запрашиваемый URL, тип действия (GET, POST и т.д.), и JWT-токен клиента;
- сервис авторизации проверяет валидность JWT-токена. Если он невалиден, то возвращается код ошибки 401;
- если JWT-токен валиден, сервис авторизации проверяет, доступно ли для пользователя с его ролью запрашиваемое действие. Если нет, то возвращается код ошибки 403;
- если действие для пользователя разрешено, то сервис авторизации возвращает код 200. Кроме того, он может вернуть ряд дополнительных параметров для пользователя (например, уникальный идентификатор пользователя user_id, который впоследствии используется бэкендом). Данные параметры берутся из таблицы users БД и добавляются к запросу силами API Gateway. Если они уже присутствовали в запросе, то должны быть перезаписаны. Перечень дополнительных параметров будет задаваться в настройках подсистемы авторизации. Фактически это перечень столбцов из таблицы users, которые нужно вернуть в ответе;
- авторизованные запросы передаются на бэкенд;
- бэкенд не занимается авторизацией. Всё что пришло на него по умолчанию разрешено.

Таким образом, подсистема авторизации и аутентификации действий пользователя осуществляет надёжную и эффективную защиту Системы, позволяя не внедрять данные механизмы непосредственно в бэкенд Системы.

2.10 Брокер RabbitMQ

Система предусматривает активное взаимодействие ряда подсистем через брокер сообщений. Данный подход является альтернативой взаимодействию по REST-протоколу, при этом обладая рядом преимуществ.

Общий подход к взаимодействию подсистем через брокер сообщений изображён на рисунке ниже (Рисунок 11).

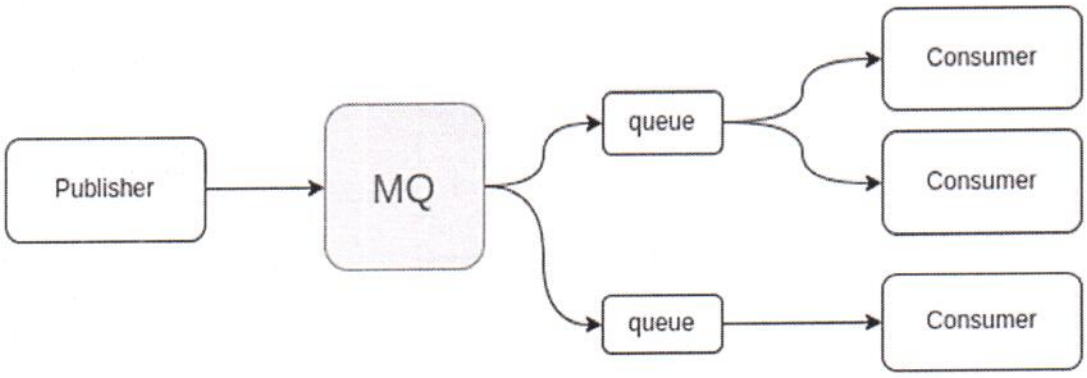

Рисунок 11 – Работа RabbitMQ

Каждая из подсистем может выступать в роли отправителя сообщений (publisher) либо в роли приёмщика сообщений (consumer). Принимать сообщения из очереди queue можно по различным стратегиям:

- несколько инстансов принимают сообщения по очереди, распределяя таким образом нагрузку между собой. Такой подход позволяет обеспечивать горизонтальное масштабирование Системы;
- все инстансы принимают все копии сообщений. Это позволяет обрабатывать одни и те же сообщения всеми заинтересованными участниками.

В разрабатываемой Системе применяются оба этих подхода в зависимости от условий.

Точка обмена сообщениями (exchange) с типом topic представлена на рисунке ниже (Рисунок 12). Consumers могут получать не все сообщения, а только те, маска topic которых совпадает с требуемой.

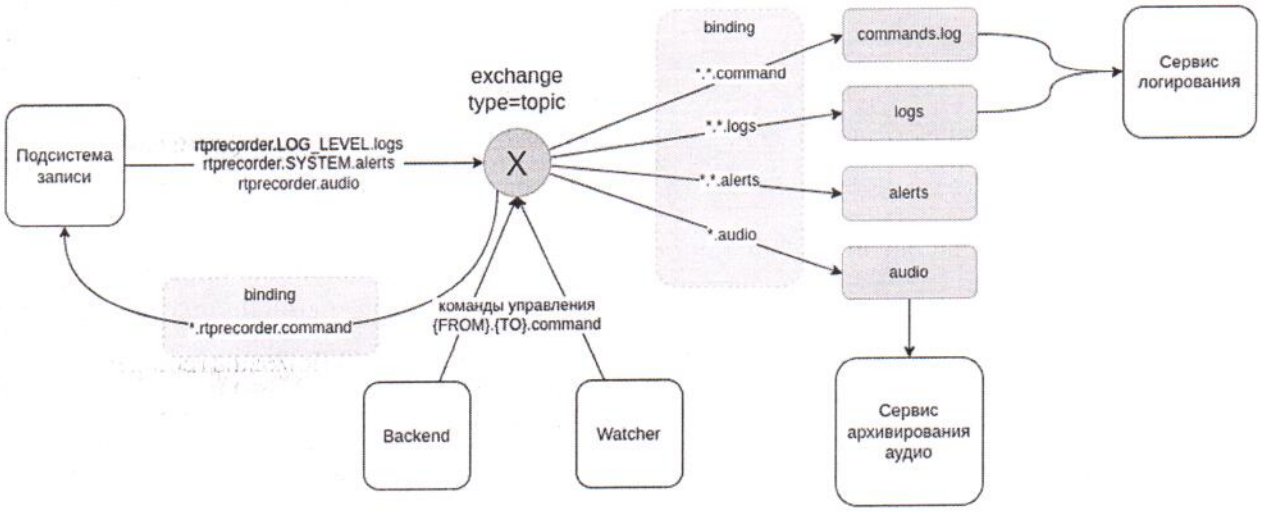

Рисунок 12 – Обмен данными

Для примера возьмём сервис архивирования аудио. Он подписывается на все сообщения, у которых topic = *.audio. Такие сообщения рассылает подсистема записи, они содержат фрагменты аудиопотоков длительностью N минут, предназначенные для архивирования.

Другой пример — подсистема записи. Она подписывается на сообщения с topic = *.rtprecorder.command. Такие сообщения рассылают любые сервисы, которые хотят управлять инстансами подсистемы записи. Одновременно сервис логирования подписывается на сообщения с topic = *.*.command. В этом случае команда, отправленная через exchange, будет получена и подсистемой записи и сервисом логирования. Это очень удобная возможность, так она позволяет прозрачно логировать все команды, проходящие через Систему, не вмешиваясь в её работу. В ряде случаев это может помочь системному администратору в поддержке Системы. Сервис логирования может в любой момент подключаться либо отключаться, а также менять маску подписки, никак при этом не влияя на работу Системы.