**«Санкт-Петербургский государственный электротехнический университет**

**«ЛЭТИ» им. В.И.Ульянова (Ленина)»**

**(СПбГЭТУ «ЛЭТИ»)**

|  |  |  |
| --- | --- | --- |
| **Направление** | 09.04.04 – Программная инженерия | |
| **Программа** | Математическое и программное обеспечение систем искусственного интеллекта | |
| **Факультет** | КТИ | |
| **Кафедра** | МО ЭВМ | |
| *К защите допустить* |  | |
| Зав. кафедрой |  | А.А. Лисс |

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

магистра

Тема: Разработка программного комплекса записи экрана и аудио устройств рабочего места оператора call-центра

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент |  |  |  | Э.Э. Абибулаев |
|  |  | *подпись* |  |  |
| Руководитель | к.т.н., доцент |  |  | С.В. Родионов |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |
| Консультанты | к.т.н., доцент |  |  | М.М. Заславский |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |
|  |  |  |  | М.С. Шахметов |
|  |  | *подпись* |  |  |
|  | к.т.н., доцент |  |  | А.Н. Иванов |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |

Санкт-Петербург

2024

**ЗАДАНИЕ**

**на выпускную квалификационную работу**

|  |  |
| --- | --- |
|  | Утверждаю |
|  | Зав. кафедрой МО ЭВМ |
|  | \_\_\_\_\_\_\_\_\_\_\_\_ А.А. Лисс |
|  | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ г. |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Студент | Абибулаев Э.Э. | | | |  | Группа | 8304 | |
| Тема работы: разработка программного комплекса записи экрана и аудио устройств рабочего места оператора call-центра | | | | | | | | |
| Место выполнения ВКР: ООО «ПРОТЕЙ-ЛАБ» | | | | | | | | |
| Исходные данные (технические требования):  Поставлена задача разработки программного комплекса для записи экрана и аудио устройств рабочего места оператора call-центра | | | | | | | | |
| Содержание ВКР:  «Введение», «Обзор аналогов», «Выбор технологий для решения», «Разработка программного обеспечения», «Оценка быстродействия», «Заключение». | | | | | | | | |
| Перечень отчетных материалов: пояснительная записка, иллюстративный материал. | | | | | | | | |
| Дополнительные разделы: Специальные вопросы обеспечения безопасности. | | | | | | | | |
|  | | | | | | | | |
| Дата выдачи задания | | | Дата представления ВКР к защите | | | | | |
| «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ г. | | | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ г. | | | | | |
|  | | |  | | | | | |
| Студент | |  | | Э.Э. Абибулаев | | | |  |
| Руководитель к.т.н., доцент | |  | | С.В. Родионов | | | |  |
| *(Уч. степень, уч. звание)* | |  | |  | | | |  |
| Консультант к.т.н., доцент | |  | | М.М. Заславский | | | |  |
| *(Уч. степень, уч. звание)* | |  | |  | | | |  |
| Консультант | |  | | М.С. Шахметов | | | |  |

**календарный план выполнения**

**выпускной квалификационной работы**

|  |  |
| --- | --- |
|  | Утверждаю |
|  | Зав. кафедрой МО ЭВМ |
|  | \_\_\_\_\_\_\_\_\_\_\_\_ А.А. Лисс |
|  | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ г. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | Абибулаев Э.Э. |  | Группа | 8304 |
| Тема работы: разработка программного комплекса записи экра-на и аудио устройств рабочего места оператора call-центра | | | | |

|  |  |  |
| --- | --- | --- |
| № п/п | Наименование работ | Срок выполнения |
| 1 | Обзор литературы по теме работы | 05.02 – 19.02 |
| 2 | Обзор предметной области | 20.02 – 20.03 |
| 3 | Формулировка требований к решению | 21.03 – 23.03 |
| 4 | Разработка ПО | 24.03 – 30.04 |
| 5 | Оформление пояснительной записки | 31.04 – 16.05 |
| 6 | Предварительная защита | 17.05 – 20.05 |
| 7 | Оформление иллюстративного материала | 20.05 – 23.05 |

|  |  |  |
| --- | --- | --- |
| Студент |  | Э.Э. Абибулаев |
| Руководитель к.т.н., доцент |  | С.В. Родионов |
| *(Уч. степень, уч. звание)* |  |  |
| Консультант к.т.н., доцент |  | М.М. Заславский |
| *(Уч. степень, уч. звание)* |  |  |
| Консультант |  | М.С. Шахметов |

**РЕФЕРАТ**

Пояснительная записка 81 стр., 5 рис., 9 табл., 47 ист.

CALL-ЦЕНТР, ЗАПИСЬ ЭКРАНА, ЗАПИСЬ АУДИО, КОДИРОВАНИЕ ВИДЕО, КОДИРОВАНИЕ АУДИО, HTTP/2

**Объектом исследования** является процесс работы сотрудников Call-центра.

**Предметом исследования** являются программные средства и библиотеки, позволяющие реализовать комплекс для записи работы оператора Call-центра.

**Цель работы:** разработать программный комплекс для записи экрана и аудио устройств рабочего места оператора call-центра с возможностью передачи записей на сервера для хранения.

В рамках данной работы был осуществлён обзор аналогов разрабатываемого программного комплекса, обзор литературы и сравнение данных аналогов. Был приведён, обзор технологий, использованных при разработке программного комплекса, описана мотивация использования данных технологий. Описаны технические ограничения, появившиеся при выполнении данной работы и методы их преодоления. Приведены результаты нагрузочного тестирования программного комплекса.

**ABSTRACT**

This work includes a review of analogues of the software package under development, a literature review, and comparison of the data of these analogues. We provide an overview of the technologies used in the development of the software package and describe the motivation for using these technologies. We also describe the technical limitations encountered during this work and the methods of overcoming them. The results of load testing of the software package are presented.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 10](#_Toc166725017)

[1 Обзор предметной области 12](#_Toc166725018)

[1.1 Язык программирования С++ 12](#_Toc166725019)

[1.2 Transmission Control Protocol 13](#_Toc166725020)

[1.3 HTTP/2 16](#_Toc166725021)

[1.4 Кодирование видео 21](#_Toc166725022)

[2 Анализ требований и разработка схемы проекта 30](#_Toc166725023)

[2.1 Общие функциональные требования 30](#_Toc166725024)

[2.2. Аппаратные требования 31](#_Toc166725025)

[2.3 Схема системы записи 32](#_Toc166725026)

[2.4 API используемый системой записи 35](#_Toc166725027)

[3 Разработка проекта 43](#_Toc166725028)

[3.1 Получение изображений экрана 43](#_Toc166725029)

[3.2 Получение аудио 46](#_Toc166725030)

[3.3 Кодирование 49](#_Toc166725031)

[3.4 Сетевое взаимодействие 50](#_Toc166725032)

[3.5 Обеспечение возможности восстановления работы сервера 51](#_Toc166725033)

[3.6 Обеспечение возможности восстановления работы клиента 52](#_Toc166725034)

[3.7 Отправка готовых записей на сервер «Mediator» 52](#_Toc166725035)

[3.8 Развёртывание проекта 53](#_Toc166725036)

[4 Нагрузочное тестирование 57](#_Toc166725037)

[4.1 Цели нагрузочного тестирования 57](#_Toc166725038)

[4.2 Результаты нагрузочного тестирования 57](#_Toc166725039)

[5 Специальные вопросы обеспечения безопасности 60](#_Toc166725040)

[5.1 Общая характеристика алгоритма оценки кардио-васкулярной реактивности при психофизиологической нагрузке 60](#_Toc166725041)

[5.2 Классификация разрабатываемого программного обеспечения 61](#_Toc166725042)

[5.3 Анализ риска потенциальных опасностей 64](#_Toc166725043)

[5.4 Качество программного обеспечения 65](#_Toc166725044)

[5.5 Безопасность ПО и его оценка 71](#_Toc166725045)

[5.6 Требования к физическому и графическому интерфейсу 72](#_Toc166725046)

[5.7 Требования к охране труда за ПК 73](#_Toc166725047)

[5.8 Выводы к 4 главе 76](#_Toc166725048)

[ЗАКЛЮЧЕНИЕ 77](#_Toc166725049)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 79](#_Toc166725050)

**ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ**

Call-центр ­– специализированная организация или выделенное подразделение в организации, занимающиеся обработкой обращений и информированием по голосовым каналам связи в интересах организации-заказчика или головной организации [1].

ЯП – язык программирования.

C++ – ЯП «С++».

Live-stream – потоковое вещание текущей записи в режиме реального времени [2].

«ВЕБ АРМ» – название программного проекта реализующего рабочее место оператора call-центра.

HTTP – (англ. HyperText Transfer Protocol) протокол передачи гипертекста [3].

HTTP2 – вторая версия протокола HTTP [4].

HTTP2 Streams – нововведение в протокол HTTP из второй версии, позволяющее создавать параллельные запросы без создания дополнительных соединений, как это было в протоколах HTTP предыдущих версий [4].

Google Chrome – браузер, разрабатываемый компанией Google на основе свободного браузера Chromium и движка Blink [5].

Mozilla Firefox – свободный браузер на движке Quantum, разработкой и распространением которого занимается Mozilla Corporation [6].

Yandex browser – браузер, созданный компанией «Яндекс» на основе движка Blink, используемого в открытом браузере Chromium [7].

X86 – архитектура процессора и одноимённый набор команд, впервые реализованные в процессорах компании Intel [8].

Windows 10 – операционная система для персональных компьютеров и рабочих станций, разработанная корпорацией Microsoft в рамках семейства Windows NT [9].

Astra Linux – Cертифицированная ОС со встроенными средствами защиты информации (СЗИ) для стабильной и безопасной работы ИТ-инфраструктур любого масштаба и обработки информации различной степени конфиденциальности [10].

## ВВЕДЕНИЕ

Call-центры в настоящее время являются неотъемлемой частью нашей повседневной жизни. Они представляют собой службы поддержки для различных предприятий, а также играют важную роль в обеспечении безопасности населения, примером чему служат службы 112 и 122.

Эффективность работы call-центров во многом определяется профессионализмом и навыками операторов. В процессе работы могут возникать спорные ситуации, связанные с нештатными случаями, которые требуют тщательного анализа и проверки действий оператора.

**Актуальность** данной работы объясняется тем, что для предотвращения подобных ситуаций и обеспечения высокого качества обслуживания клиентов, важно иметь записи работы операторов. Это позволяет анализировать их действия, выявлять ошибки и улучшать качество обслуживания.

**Целью работы** является разработка специализированного программного комплекса для записи и анализа действий операторов call-центра. Данный комплекс позволит осуществлять мониторинг работы операторов, выявлять проблемные моменты и принимать меры по их устранению.

**Задачи работы**, которые необходимо было решить для достижения поставленной цели:

* Провести общую оценку технических характеристик персональных компьютеров операторов call-центра, чтобы определить объём ресурсов, выделяемых для функционирования программного обеспечения.
* Разработать программное обеспечение для ПК оператора call-центра согласно полученным требованиям и с учётом ограничений, определяемых техническими характеристиками ПК.
* Разработать программное обеспечение для серверной части системы.

**Объектом исследования** является процесс работы сотрудников на посту оператора в call-центре.

**Предметом исследования** являются программные средства и библиотеки, позволяющие реализовать комплекс для записи работы оператора Call-центра.

**Практическая ценность** данной работы заключается в создании программного обеспечения, которое позволяет записывать изображение с экранов и звук с аудиоустройств персональных компьютеров операторов call-центра, а также передавать полученные данные на удалённые серверы для хранения. Это обеспечивает возможность мониторинга и контроля работы операторов, что способствует повышению качества обслуживания клиентов и эффективности деятельности call-центра в целом.

В рамках данной работы было проведено исследование существующих решений и технологий, которые могут быть использованы для разработки программного обеспечения. Были определены требования к программному обеспечению, а также ограничения, связанные с техническими характеристиками персональных компьютеров операторов. На основе полученных данных было разработано программное обеспечение, которое соответствует всем требованиям и ограничениям.

Программное обеспечение, разработанное в рамках данной работы, позволяет записывать изображение с экрана и звук с аудиоустройств персонального компьютера оператора call-центра. Полученные данные передаются на удалённый сервер для хранения. Программное обеспечение также обеспечивает возможность просмотра записанных данных и анализа работы операторов.

Разработанное программное обеспечение может быть использовано для контроля работы операторов call-центра и повышения качества обслуживания клиентов. Оно также может быть адаптировано для использования в других сферах, где требуется контроль работы персонала и мониторинг качества обслуживания клиентов.

## 1 Обзор предметной области

### 1.1 Язык программирования С++

C++ — это статически типизированный, компилируемый высокоуровневый язык программирования общего назначения [11].

C++ был разработан Бьёрном Страуструпом в 1979 году в компании Bell Labs как расширение языка C. Целью разработки было создание языка, который бы объединял эффективность и гибкость языка C с объектно-ориентированными возможностями [11].

C++ широко используется для разработки программного обеспечения, особенно системного программирования, прикладных программ, драйверов устройств, игр и многого другого. Он также используется в академических исследованиях и обучении.

Одной из ключевых особенностей C++, унаследованных от C, является его низкоуровневая природа, которая позволяет разработчикам иметь прямой доступ к аппаратному обеспечению и памяти. Это делает C++ подходящим для разработки операционных систем, драйверов и других приложений, требующих тесного взаимодействия с аппаратным обеспечением.

C++ является одним из самых популярных языков программирования. Он широко используется в различных областях, включая разработку игр, операционных систем и высокопроизводительных приложений. C++ также является основой для многих других языков программирования, таких как C# и Java.

Основные особенности C++:

* Компилируемый язык: исходный код преобразуется в машинный код перед выполнением.
* Объектно-ориентированный: поддерживает инкапсуляцию, наследование и полиморфизм.
* Низкоуровневый доступ: позволяет разработчикам взаимодействовать с аппаратным обеспечением и памятью.
* Высокая производительность: обеспечивает эффективное выполнение кода.
* Кроссплатформенность: программы, написанные на C++, могут быть скомпилированы и выполнены на различных платформах.

В целом, C++ является мощным и гибким языком программирования, который предоставляет разработчикам широкий спектр возможностей для создания сложных и высокопроизводительных приложений.

### 1.2 Transmission Control Protocol

Transmission Control Protocol (TCP) — это один из основных протоколов транспортного уровня модели OSI, который обеспечивает надёжный, упорядоченный и контролируемый поток данных между хостами в сети. [12]

TCP был разработан Винтом Серфом и Бобом Каном в 1974 году в рамках проекта ARPANET. Целью разработки было создание протокола, который бы обеспечивал надёжную передачу данных в условиях ненадёжных сетей.

Основные особенности TCP:

* **Надёжность**: TCP обеспечивает надёжную передачу данных, гарантируя, что данные будут доставлены. Это достигается за счёт использования механизма подтверждения получения данных и повторной передачи данных в случае потери или искажения данных.
* **Упорядоченность**: TCP гарантирует, что данные будут доставлены в правильном порядке. Это важно для приложений, которые чувствительны к порядку данных, таких как аудио- и видеопотоки.
* **Контроль перегрузки**: TCP контролирует скорость передачи данных, чтобы предотвратить перегрузку сети. Это достигается за счёт использования алгоритма управления перегрузкой, который регулирует скорость передачи данных в зависимости от состояния сети. Управление количеством передаваемых данных происходит при помощи изменения величины «окна» пакетов.
* **Управление соединением**: TCP является протоколом, который позволяет устанавливать и разрывать соединения. В протоколе есть механизмы позволяющие узлам соединения понимать, в каком состоянии находится соединение и возможно ли дальнейшее его использование.
* **Мультиплексирование**: TCP позволяет нескольким приложениям использовать одно и то же IP соединение. Это упрощает разработку приложений, которые используют TCP, так как не требуется создание механизма позволяющего определить к какому приложению относятся данные.

TCP используется для передачи данных в большинстве современных сетей, включая Интернет. Он является основой для многих приложений, таких как веб-браузеры, электронная почта, обмен файлами и многое другое.

В целом, TCP является важным и широко используемым протоколом, который обеспечивает надёжную и эффективную передачу данных в современных сетях.

#### 1.2.1 Применение TCP

TCP используется в различных приложениях, где требуется надёжная и упорядоченная передача данных. Некоторыми примерами данных приложений являются:

* Веб-браузеры: TCP используется для передачи данных между веб-браузером и веб-сервером.
* Электронная почта: TCP используется для передачи сообщений электронной почты между почтовыми серверами.
* Обмен файлами: TCP используется для передачи файлов между компьютерами.
* Потоковое мультимедиа: TCP используется для передачи аудио- и видеопотоков между устройствами.

TCP является важным компонентом современных сетей и приложений. Он обеспечивает надёжную и эффективную передачу данных, что делает его незаменимым протоколом.

#### 1.2.2 Libevent2

HTTP/2 функционирует на прикладном уровне сетевой модели OSI. Для его работы требуется наличие протокола транспортного уровня. В случае с HTTP/2 таким протоколом выступает TCP.

Ядро операционной системы отвечает за обработку TCP и протоколов, расположенных ниже в модели OSI.

Данные, передаваемые по TCP, получаются с использованием библиотеки libevent2.

Libevent2 — это кроссплатформенная библиотека, предназначенная для разработки событийно-ориентированных приложений. Она предоставляет разработчикам набор функций и методов для обработки событий, таких как сетевые соединения, таймеры, файлы и другие.

Libevent2 является открытой библиотекой с открытым исходным кодом, написанной на языке C. Она позволяет разработчикам создавать приложения, которые могут эффективно обрабатывать события и обеспечивать высокую производительность.

Основные функции и методы, предоставляемые Libevent2, включают:

* Обработка событий: Libevent2 предоставляет функции для обработки различных типов событий, таких как сетевые соединения, таймеры, файлы и другие. Это позволяет разработчикам создавать приложения, которые могут реагировать на события в реальном времени.
* Поддержка различных платформ: Libevent2 поддерживает различные платформы, такие как Windows, Linux, macOS и другие. Это делает её универсальной библиотекой, которая может быть использована для разработки приложений на разных платформах.
* Гибкость и расширяемость: Libevent2 является гибкой и расширяемой библиотекой. Разработчики могут легко добавлять новые функции и методы в библиотеку, чтобы создавать приложения, соответствующие их потребностям.

Использование Libevent2 позволяет разработчикам создавать высокопроизводительные приложения, которые могут обрабатывать события. Это делает Libevent2 важным инструментом для разработчиков, создающих приложения, работающие с сетевыми соединениями, таймерами, файлами и другими событиями.

### 1.3 HTTP/2

HTTP/2 — это вторая версия протокола передачи данных HTTP, который является основным протоколом для обмена информацией в сети Интернет.

HTTP/2 был разработан в 2012–2015 годах группой инженеров и специалистов по информационным технологиям с целью улучшения производительности и эффективности передачи данных по сравнению с предыдущей версией протокола HTTP/1.1.

Основные особенности HTTP/2:

* Мультиплексирование: позволяет отправлять и получать несколько запросов и ответов одновременно по одному соединению.
* Сжатие заголовков: уменьшает размер передаваемых заголовков запросов и ответов, что ускоряет передачу данных.
* Приоритезация запросов: позволяет серверу указывать приоритет запросов, что позволяет более эффективно использовать ресурсы сервера.
* Серверные push-уведомления: позволяют серверу отправлять данные клиенту без запроса, что может ускорить загрузку страниц.

HTTP/2 является более эффективным и производительным протоколом по сравнению с HTTP/1.1, что делает его предпочтительным выбором для современных веб-приложений и сервисов.

HTTP/2 используется в различных веб-приложениях и сервисах, где требуется высокая производительность и эффективность передачи данных [4]. Вот некоторые примеры:

* Веб-браузеры: HTTP/2 используется для загрузки веб-страниц и ресурсов, таких как изображения, стили, скрипты и т. д.
* Мобильные приложения: HTTP/2 используется для обмена данными между мобильными приложениями и серверами.
* Облачные сервисы: HTTP/2 используется для доступа к облачным сервисам, таким как хранилища данных, базы данных, вычислительные ресурсы и т. д.

В целом, HTTP/2 является важным шагом в развитии протокола HTTP и обеспечивает более высокую производительность и эффективность передачи данных в современных веб-приложениях и сервисах.

#### 1.3.1 Потоки и мультиплексирование соединения

Поток представляет собой независимую двунаправленную последовательность фреймов, которыми обмениваются клиент и сервер в рамках соединения HTTP/2. Потокам присущи следующие характеристики:

* В одном соединении HTTP/2 может быть создано несколько одновременно открытых потоков, при этом любой из участников соединения может чередовать фреймы из разных потоков.
* Потоки могут быть созданы и использованы в одностороннем порядке или совместно любым из участников соединения.
* Любой из участников соединения может закрыть поток.

Порядок отправки фреймов имеет важное значение. Получатель обрабатывает фреймы в том порядке, в котором они были получены. В частности, порядок фреймов HEADERS и DATA семантически значим.

Потоки идентифицируются с помощью целого числа. Идентификаторы потоков присваиваются потокам тем участником соединения, который инициирует поток.

#### 1.3.2 Сжатие заголовков

Сжатие заголовков — это механизм, используемый в протоколе HTTP/2 для уменьшения объёма передаваемой информации и повышения эффективности передачи данных. Этот механизм позволяет уменьшить размер заголовков HTTP-сообщений, что приводит к уменьшению общего объёма передаваемых данных и, как следствие, к более быстрой передаче информации.

В протоколе HTTP/2 сжатие заголовков реализуется с помощью алгоритма HPACK (HTTP/2 Header Compression). Этот алгоритм использует механизм кодирования и декодирования заголовков, что позволяет уменьшить их размер и тем самым повысить эффективность передачи данных.

Алгоритм HPACK работает следующим образом:

* При первом обмене данными между клиентом и сервером происходит обмен всеми заголовками HTTP-сообщений.
* После этого клиент и сервер создают таблицу соответствия между заголовками и их значениями.
* При последующих обменах данными клиент и сервер используют эту таблицу для кодирования и декодирования заголовков.
* Если заголовок уже был отправлен ранее, то клиент и сервер могут использовать его номер в таблице вместо отправки полного заголовка.

Таким образом, сжатие заголовков позволяет уменьшить объём передаваемой информации, что приводит к более быстрой передаче данных и повышению эффективности работы протокола HTTP/2.

#### 1.3.3 Приоритезация запросов

Приоритезация запросов в протоколе HTTP/2 — это механизм, который позволяет определить порядок обработки запросов сервером. Механизм приоритезации запросов основан на концепции потоков (streams) и позволяет серверу отдавать предпочтение одним запросам перед другими.

В протоколе HTTP/2 каждый запрос представляет собой поток данных, который имеет свой уникальный идентификатор. Сервер может устанавливать приоритеты для потоков, чтобы обеспечить более быструю обработку важных запросов.

Механизм приоритезации запросов реализуется с помощью следующих методов:

* Приоритезация запросов в протоколе HTTP/2 — это механизм, который позволяет определить порядок обработки запросов сервером. Механизм приоритезации запросов основан на концепции потоков (streams) и позволяет серверу отдавать предпочтение одним запросам перед другими.
* В протоколе HTTP/2 каждый запрос представляет собой поток данных, который имеет свой уникальный идентификатор. Сервер может устанавливать приоритеты для потоков, чтобы обеспечить более быструю обработку важных запросов.

Механизм приоритезации запросов позволяет серверу оптимизировать обработку запросов и обеспечить более быстрое обслуживание важных запросов. Это особенно важно для приложений, которые требуют быстрой реакции сервера, таких как онлайн-игры, видеоконференции и другие приложения реального времени.

#### 1.3.4 Серверные push-уведомления

Серверные push-уведомления в протоколе HTTP/2 представляют собой механизм, который позволяет серверу инициировать передачу данных клиенту без прямого запроса от последнего. Этот подход направлен на улучшение производительности веб-приложений, так как позволяет заранее отправлять ресурсы, которые, как ожидается, будут затребованы клиентом в ближайшем будущем.

В традиционном HTTP/1.1, клиент должен явно запрашивать каждый ресурс, что может приводить к задержкам, особенно когда требуется множество файлов (например, стили, скрипты, изображения) для отображения страницы. HTTP/2 вводит концепцию server push, которая позволяет серверу предугадать и отправить ресурсы до того, как клиент сделает соответствующий запрос.

Механизм server push работает следующим образом:

* Клиент отправляет запрос на получение веб-страницы (например, HTML).
* Сервер, анализируя запрос, определяет, какие дополнительные ресурсы (например, CSS, JavaScript, изображения) будут необходимы для полного отображения страницы.
* Сервер инициирует push-уведомления, отправляя эти ресурсы клиенту без ожидания запросов на их получение.
* Клиент принимает эти ресурсы и сохраняет их в кэше для последующего использования.

Важно отметить, что server push не является универсальным решением для всех сценариев и может не приносить ощутимого улучшения производительности, особенно если ресурсы уже находятся в кэше клиента или если клиент работает в ограниченных сетевых условиях.

Для эффективного использования server push необходимо тщательное планирование и анализ, чтобы гарантировать, что отправляемые ресурсы действительно необходимы и не будут дублировать уже имеющиеся данные в кэше клиента.

Серверные push-уведомления в HTTP/2 являются важным инструментом для оптимизации веб-приложений, но их использование должно быть основано на тщательном анализе и тестировании, чтобы обеспечить максимальную эффективность и производительность.

#### 1.3.5 libcno

Cno — это библиотека, предназначенная для работы с протоколом HTTP/2. Она предоставляет набор функций и методов для взаимодействия с серверами, поддерживающими этот протокол.

Cno является открытой библиотекой с открытым исходным кодом, напи-санной на языке C. Она позволяет создавать приложения, способные эффектив-но использовать преимущества HTTP/2, такие как мультиплексирование, сжатие заголовков и приоритезация запросов.

Основные функции и методы, предоставляемые Cno, включают:

* Создание и управление соединениями: Cno позволяет разработчикам создавать соединения с серверами, поддерживающими HTTP/2, и управлять ими.
* Отправка запросов и получение ответов: Cno предоставляет функции для отправки запросов и получения ответов от серверов, поддержива-ющих HTTP/2.
* Обработка заголовков: Cno позволяет разработчикам обрабатывать за-головки запросов и ответов, отправленных по протоколу HTTP/2.

Использование Cno позволяет создавать приложения, которые могут эф-фективно использовать преимущества протокола HTTP/2 для улучшения произ-водительности и эффективности работы. Это делает Cno важным инструментом для разработчиков, создающих приложения, работающие с данными, получен-ными через интернет.

Однако стоит отметить, что использование Cno требует определённых знаний и навыков в области работы с протоколами передачи данных. Разработ-чики должны понимать основы работы с HTTP/2 и уметь применять их для со-здания своих приложений.

### 1.4 Кодирование видео

Для кодирования видео используются специальные видеокодеки. Видеокодек – это алгоритм или набор алгоритмов, используемых для сжатия и распаковки цифрового видеосигнала с целью уменьшения его размера без значительной потери качества.

Кодеки используются в различных областях, таких как потоковое мультимедиа, видеоконференции, видео по запросу и других, где требуется передача и хранение видеоданных.

Существует множество различных видеокодеков, каждый из которых имеет свои особенности и характеристики. Некоторые из наиболее распространённых кодеков включают H.264, H.265, VP9, AV1 и другие.

Основные характеристики видеокодеков:

* **Степень сжатия**: определяет, насколько сильно видеоданные могут быть сжаты без потери качества.
* **Скорость передачи данных**: определяет, сколько данных необходимо передать для воспроизведения видео с определённым качеством.
* **Качество изображения**: определяет, насколько близко восстановленное видео соответствует оригинальному.
* **Поддержка различных разрешений и форматов**: определяет, с какими форматами и разрешениями видеоданных может работать кодек.

Выбор видеокодека зависит от конкретных требований к качеству, скорости передачи данных и совместимости с различными устройствами и платформами.

В целом, видеокодеки являются важными инструментами для эффективного хранения, передачи и воспроизведения видеоданных в различных приложениях и устройствах.

#### 1.4.1 Advanced Video Coding (AVC)

Advanced Video Coding (AVC) — это стандарт видеосжатия, разработанный группой экспертов по видеокодированию (VCEG) и группой экспертов по движущимся изображениям (MPEG). Стандарт был опубликован в 2003 году и является основой для форматов сжатия видео H.264 и MPEG-4 AVC [13].

AVC был разработан с целью улучшения качества видеосжатия по сравнению с предыдущими стандартами, такими как MPEG-2 и H.263. Стандарт использует более сложные алгоритмы для сжатия видеоданных, что позволяет достичь более высокого качества изображения при меньшем размере файла.

Основные особенности AVC:

* **Улучшенное качество изображения**: AVC использует более сложные алгоритмы для сжатия видеоданных, что позволяет достичь более высокого качества изображения при меньшем размере файла.
* **Поддержка различных разрешений и форматов**: AVC поддерживает широкий спектр разрешений и форматов видеоданных.
* **Высокая степень сжатия**: AVC обеспечивает высокую степень сжатия видеоданных без значительной потери качества.
* **Поддержка многопоточности**: AVC поддерживает многопоточность, что позволяет ускорить процесс сжатия и распаковки видеоданных.

AVC является одним из наиболее распространённых стандартов видеосжатия и используется в различных приложениях, таких как потоковое мультимедиа, видеоконференции, видео по запросу и других.

В целом, AVC является важным стандартом видеосжатия, который обеспечивает высокое качество изображения, высокую степень сжатия и поддержку различных разрешений и форматов.

#### 1.4.2 High Efficiency Video Coding (HEVC)

High Efficiency Video Coding (HEVC) — это стандарт видеосжатия, разработанный группой экспертов по видеокодированию (VCEG) и группой экспертов по движущимся изображениям (MPEG). Стандарт был опубликован в 2013 году и является развитием стандарта AVC (H.264/MPEG-4 AVC) [14].

HEVC был разработан с целью дальнейшего улучшения качества видеосжатия и снижения требований к пропускной способности по сравнению с AVC. Стандарт использует более сложные алгоритмы для сжатия видеоданных, что позволяет достичь более высокого качества изображения при меньшем размере файла.

Основные особенности HEVC:

* **Улучшенное качество изображения**: HEVC использует более сложные алгоритмы для сжатия видеоданных, что позволяет достичь более высокого качества изображения при меньшем размере файла по сравнению с AVC.
* **Высокая степень сжатия**: HEVC обеспечивает высокую степень сжатия видеоданных без значительной потери качества.
* **Поддержка многопоточности**: HEVC поддерживает многопоточность, что позволяет ускорить процесс сжатия и распаковки видеоданных.
* **Снижение требований к пропускной способности**: HEVC позволяет снизить требования к пропускной способности по сравнению с AVC, что делает его более подходящим для передачи видео по сетям с ограниченной пропускной способностью.

HEVC является одним из наиболее перспективных стандартов видеосжатия и используется в различных приложениях, таких как потоковое мультимедиа, видеоконференции, видео по запросу и других.

В целом, HEVC является важным стандартом видеосжатия, который обеспечивает высокое качество изображения, высокую степень сжатия и снижение требований к пропускной способности.

#### 1.4.3 VP9

VP9 — это видеокодек, разработанный компанией Google. Он был представлен в 2010 году как открытый стандарт для сжатия видео с высокой степенью сжатия и качеством изображения [15].

VP9 является одним из наиболее перспективных видеокодеков и используется в различных приложениях, таких как потоковое мультимедиа, видеоконференции, видео по запросу и других.

Основные характеристики VP9:

* Высокая степень сжатия. VP9 обеспечивает высокую степень сжатия видеоданных без значительной потери качества.
* Поддержка многопоточности. VP9 поддерживает многопоточность, что позволяет ускорить процесс сжатия и распаковки видеоданных.
* Высокое качество изображения. VP9 использует сложные алгоритмы для сжатия видеоданных, что позволяет достичь высокого качества изображения при меньшем размере файла.
* Открытый стандарт. VP9 является открытым стандартом, что делает его доступным для использования в различных приложениях и устройствах.

Кодек VP9 используется в различных приложениях и устройствах, где требуется эффективное сжатие видеоданных. Например, его применяют для:

* стриминговых сервисов, таких как YouTube и Vimeo;
* видеоконференций, например, Zoom и Microsoft Teams;
* онлайн-кинотеатров, таких как Кинопоиск и Иви;
* облачных хранилищ, например, Google Диск и Яндекс.Диск.

Благодаря высокой степени сжатия и качеству изображения, кодек VP9 становится всё более популярным выбором среди разработчиков и пользователей.

#### 1.4.4 AV1

AV1 (AOMedia Video 1) — это видеокодек, разработанный Alliance for Open Media (AOMedia) с целью обеспечения высокого качества видеосжатия при сохранении открытого исходного кода.

AV1 был представлен в 2018 году как открытый стандарт для сжатия видео с высокой степенью сжатия и качеством изображения. Он основан на алгоритмах кодирования видео, разработанных в рамках проекта Daala, который был запущен в 2012 году.

Основные особенности AV1:

* **Высокая степень сжатия**. AV1 обеспечивает высокую степень сжатия видеоданных без значительной потери качества.
* **Поддержка многопоточности**. AV1 поддерживает многопоточность, что позволяет ускорить процесс сжатия и распаковки видеоданных.
* **Высокое качество изображения**. AV1 использует сложные алгоритмы для сжатия видеоданных, что позволяет достичь высокого качества изображения при меньшем размере файла.
* **Открытый исходный код**. AV1 является открытым стандартом, что делает его доступным для использования в различных приложениях и устройствах.

В целом, AV1 является важным видеокодеком, который обеспечивает высокое качество изображения, высокую степень сжатия и поддержку многопоточности.

AV1 используется в различных приложениях, где требуется эффективное сжатие видеоданных. Вот некоторые примеры:

* стриминговые сервисы, такие как YouTube и Vimeo;
* видеоконференции, например, Zoom и Microsoft Teams;
* онлайн-кинотеатры, такие как Кинопоиск и Иви;
* облачные хранилища, например, Google Диск и Яндекс.Диск.,

#### 1.4.5 Intel Quick Sync Video

Intel Quick Sync Video — это технология, разработанная компанией Intel для ускорения кодирования и декодирования видеоданных. Она интегрирована в некоторые процессоры Intel и позволяет выполнять эти операции с высокой производительностью.

Технология Intel Quick Sync Video использует специализированные аппаратные блоки в процессоре для выполнения операций кодирования и декодирования видео. Это позволяет ускорить процесс обработки видеоданных и снизить нагрузку на центральный процессор.

Основные особенности Intel Quick Sync Video:

* **Высокая производительность**: технология обеспечивает высокую производительность при кодировании и декодировании видеоданных.
* **Поддержка различных форматов**: технология поддерживает широкий спектр форматов видеоданных, включая H.264, H.265 и другие.
* **Низкое энергопотребление**: технология оптимизирована для работы с низким энергопотреблением, что делает её подходящей для мобильных устройств.

Intel Quick Sync Video используется в различных приложениях, таких как видеоконференции, потоковое мультимедиа, видео по запросу и других, где требуется обработка видеоданных. Технология позволяет ускорить процесс кодирования и декодирования видео, что повышает производительность и эффективность работы приложений. Поддержка аппаратного кодирования на процессорах intel для приведённых выше форматов описана далее (таблица 1).

Таблица 1 ­­– поддержка аппаратного кодирования видео на процессорах intel

|  |  |  |
| --- | --- | --- |
| Кодек | Кодирование | Декодирование |
| H.264(AVC) | Начиная с Ivy Bridge (2012) | Начиная с Cantiga (2008) |
| H.265(HEVC) | Начиная со Skylake (2015) | Начиная с Braswell/Cherry Trail (2013) |
| VP9 | Начиная c Ice Lake (2019) | Кроссплатформенное, начиная с Kaby Lake  (2016) |
| AV1 | Посредством iGPU, начиная c Meteor Lake (2023) | Начиная с Tiger Lake (2020) |

#### 1.4.6 MP4

MP4 (Moving Picture Experts Group, версия 4) — это формат мультимедийных файлов, который используется для хранения, передачи и воспроизведения аудио- и видеоданных. MP4 является одним из самых популярных форматов для обмена мультимедийным контентом в интернете.

MP4 был разработан как преемник формата MPEG-1. Он поддерживает более широкий спектр возможностей, таких как поддержка высокого разрешения видео, поддержка различных аудиоформатов и возможность встраивания субтитров.

Формат MP4 основан на технологии MPEG-4, которая представляет собой набор стандартов для кодирования, сжатия и воспроизведения аудио- и видеоданных. MPEG-4 включает в себя множество различных спецификаций, таких как MPEG-4 Part 2 (видеокодек), MPEG-4 Part 3 (аудиокодек) и MPEG-4 Part 10 (профиль кодека для мобильных устройств).

Файлы MP4 обычно имеют расширение .mp4 или .m4p. Они могут содержать различные типы аудио- и видеоданных, такие как аудиопотоки, видеопотоки, субтитры и метаданные.

Файлы MP4 широко используются в различных приложениях и устройствах, таких как проигрыватели мультимедиа, смартфоны, планшеты и веб-браузеры. Они обеспечивают высокое качество воспроизведения аудио- и видеоконтента и поддерживают широкий спектр возможностей для персонализации воспроизведения.

Формат MP4 является важным инструментом для обмена мультимедийным контентом и обеспечивает высокое качество воспроизведения аудио- и видеоданных в различных устройствах и приложениях.

#### 1.4.7 LibAV

LibAV — это библиотека с открытым исходным кодом, предназначенная для обработки аудио- и видеофайлов. LibAV предоставляет набор функций для кодирования, декодирования, преобразования и воспроизведения мультимедийного контента.

Библиотека LibAV разработана для обеспечения высокой производительности и эффективности при работе с мультимедийными данными. Она поддерживает широкий спектр форматов аудио и видео, включая MPEG-1, MPEG-2, MPEG-4, H.264, H.265, MP3, AAC и другие.

Основные функции LibAV включают в себя:

* Кодирование и декодирование аудио- и видеоданных.
* Трансформацию мультимедийных файлов, например изменение разрешения, частоты кадров, битрейта и других параметров.
* Воспроизведение аудио- и видеоконтента.
* Доступ к метаданным файлов, таким как информация о кодеке, разрешении, частоте кадров и других характеристиках.

LibAV широко используется в различных областях, включая разработку мультимедийных приложений, создание видеоигр, научные исследования и другие области, где требуется обработка аудио- и видеоданных. Библиотека доступна для различных платформ, включая Windows, Linux, macOS и другие операционные системы.

## 2 Анализ требований и разработка схемы проекта

### 2.1 Общие функциональные требования

1. Система записи данных должна быть интегрирована в проект «ВЕБ АРМ» и состоять из двух частей: клиентской и серверной.

2. Клиентская часть устанавливается на персональный компьютер оператора и предоставляет возможность управления началом и остановкой записи.

3. Для проверки статуса записи и её параметров со стороны «ВЕБ АРМ» необходимо реализовать ответ на запросы по пути «/ping».

4. Серверная часть собирает данные в медиаконтейнер, который предоставляет возможность просмотра записей с возможностью перемотки позднее.

5. Необходимо предусмотреть возможность конфигурации параметров записи:

* битрейт записи видео;
* частота кадров видео;
* битрейт записи аудио;
* интервал, в течение которого ожидается следующий «ping».
* URI «сервера»;
* URI хранилища, куда готовая запись должна быть отправлена.

6. Также с каждой записью необходимо передавать метаданные, которые должны быть прозрачно переданы в хранилище.

7. Важным требованием является простота использования со стороны оператора. То есть, оператор не должен производить дополнительных действий для работы системы записи (не давать разрешений, запускать самостоятельно и т. п.).

8. Должна быть предоставлена возможность обновления комплекса системы записи на ПК оператора.

9. Серверная часть должна поставляться и работать внутри docker-контейнера.

10. Необходимо разработать механизмы отказоустойчивости системы при недоступности некоторых серверов или при аварийной остановке работы.

11. Клиентская часть должна позволять записывать звуковые сигналы со всех аудиоустройств и изображение со всех мониторов ПК оператора call-центра.

12. Записи, производимые программным комплексом, должны укладываться в норматив – 150 ТБ хранилища в год при параметрах:

* FPS: 2
* На видеозаписи должны быть различимы символы на экране ПК оператора.
* Аудиозапись должна быть битрейтом не ниже 20 кбит

Дополнительно, представлен ожидаемый план нагрузки системы описан ниже (таблица 2):

Таблица 2 ­­– ожидаемый план нагрузки на систему записи

|  |  |
| --- | --- |
| Средняя длительность вызова ФЛ, минуты | 5 |
| Средняя длительность вызова ЮЛ, минуты | 6 |
| Средняя длительность вызова по общим вопросам, минуты | 4,5 |
| Количество вызовов ФЛ | 14000 |
| Количество вызовов ЮЛ | 4400 |
| Количество вызовов по общим вопросам | 881 |

### 2.2. Аппаратные требования

#### 2.2.1 Аппаратные требования для клиентской части

Предполагается, что на компьютере оператора будут установлены:

* 2-ядерный процессор на архитектуре X86;
* 4 ГБ оперативной памяти;
* операционная система Windows 10 или Astra Linux.

Учитывая, что параллельно с предполагаемым приложением будет использоваться браузер (Google Chrome, Mozilla Firefox, Yandex Browser и т. д.), а также ресурсы потребляет операционная система Windows 10, необходимо разработать приложение с приоритетом на максимально эффективное использование ресурсов на машине оператора.

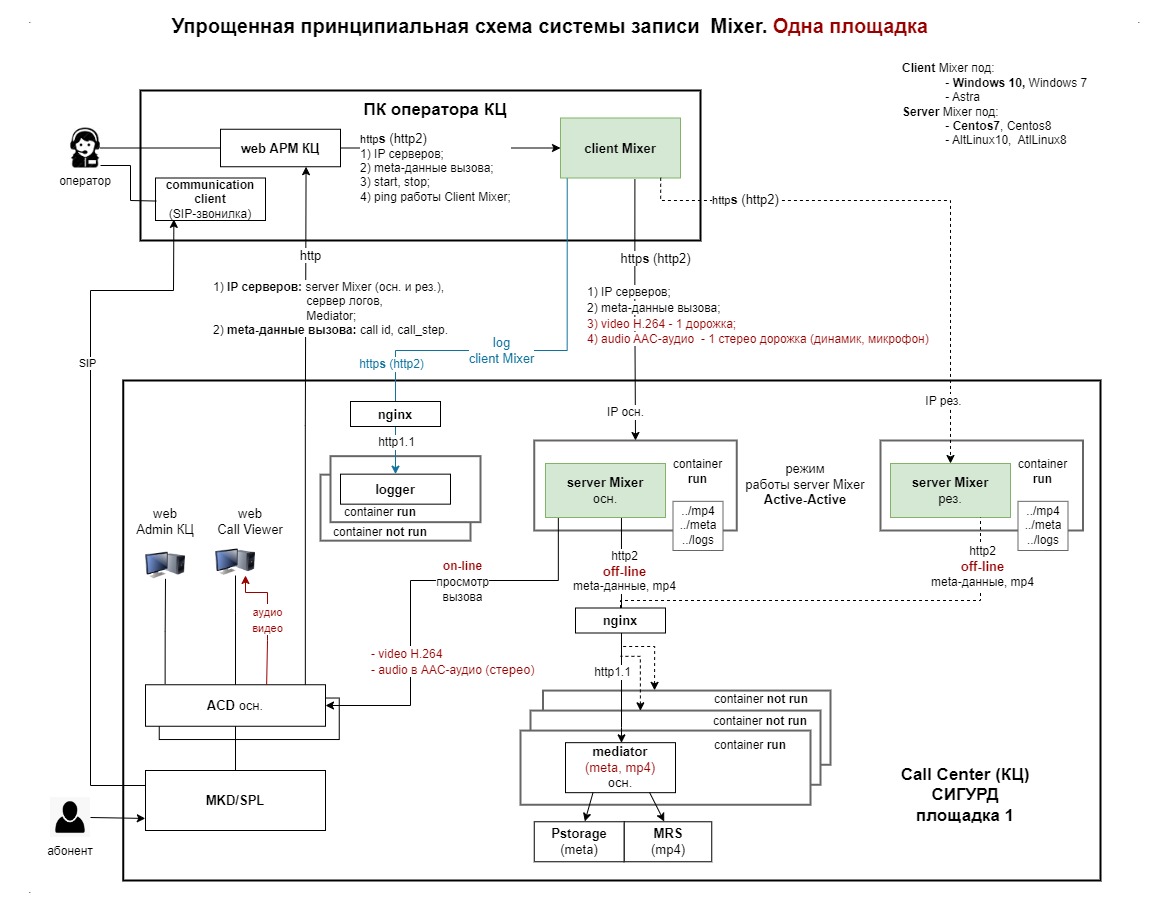
Это означает, что клиентская часть приложения должна потреблять возможный минимум аппаратных характеристик.

#### 2.2.2 Аппаратные требования для серверной части

Аппаратные требования к серверу не предъявляются и ожидаются от команды разработки.

### 2.3 Схема системы записи

Для реализации проекта была разработана схема системы записи, приведённая ниже (Рис. 1).

Рисунок 1 – общая схема системы записи

Согласно данной схеме, приложение клиента ожидает запросы по протоколу HTTP/2 поверх протокола TLS по localhost. Использование TLS обосновано тем, что большинству браузеров запрещено использовать незащищённое соединение при посылке запросов не в направлении сервера данного приложения.

Сервер также ожидает от клиента запросы по протоколу HTTP/2 поверх протокола TLS для защиты информации, передаваемой по сети. Также, сервер соединяется с серверами для хранения данных называемыми «Mediator» по протоколу HTTP/2 и передаёт данные в формате «multipart/form-data» [16].

#### 2.3.1 Запрос начала записи

Клиент принимает запрос на начало записи по пути «/start». При получении запроса на начало записи клиент ожидает следующую информацию:

* Набор IP адресов и TCP портов серверов системы записи;
* Параметры записи, которые требовались в функциональных требованиях (пункт 5);
* Meta-данные вызова, которые необходимо передать в неизменном виде.

При получении запроса на получение записи клиент пытается установить соединение с одним из предоставленных в наборе серверов для передачи видео, а также с сервером для передачи информационных сообщений о работе программы. После осуществления соединения, начинается запись изображения с экранов и звуков с аудиоустройств.

Сервер начинает принимать пакеты закодированного аудио и видео от клиента.

#### 2.3.2 Запрос «/ping»

Во время записи работы оператора КЦ клиент ожидает запросы /ping от веб-приложения «АРМ КЦ» в течении определённого времени. Внутри запроса веб-приложение присылает параметры записи, которые клиент сверяет с параметрами текущей записи. В случае, если параметры не сходятся, клиент посылает код ошибки и прекращает запись.

При истечении времени ожидания запроса, клиент прекращает запись и оповещает об этом сервер. Такой сценарий выполняется с целью экономии пространства в хранилищах и вычислительной мощности сервера.

#### 2.3.3 Запрос на прекращение записи

Запрос на прекращение записи поступает на путь «/stop». При получении запроса на прекращение записи, по аналогии с запросом «/ping» клиент сверяет параметры записи и код возврата зависит от совпадения параметров текущей записи с запрашиваемыми. Помимо этого, клиент останавливает запись и оповещает об этом сервер. После этого, клиент отключается от сервера передачи информационных сообщений.

Сервер финализирует запись и сохраняет её для последующей передачи в хранилище.

Согласно информации, описанной выше, создана схема, представляющая цикл одной записи (Рис. 2):

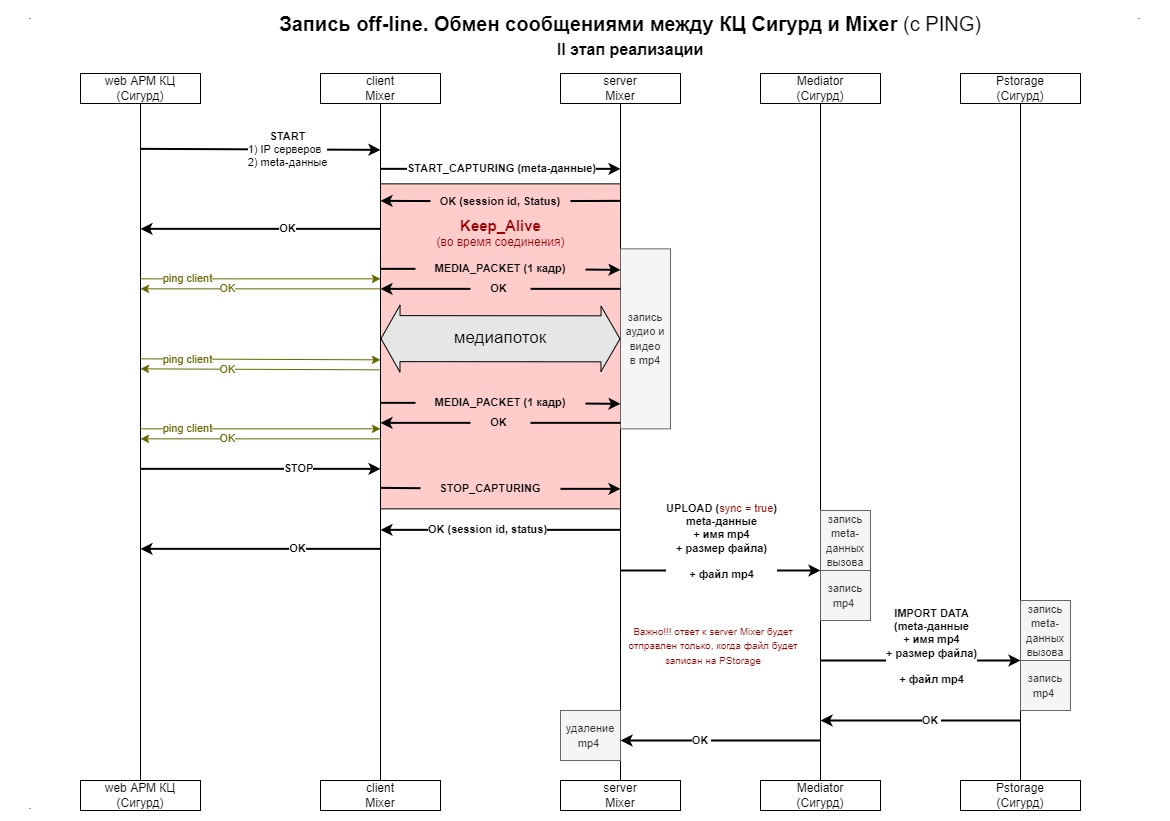


Рисунок 2 – схема процесса одной записи

Система записи, описываемая в данной работе, состоит из client Mixer и server Mixer.

### 2.4 API используемый системой записи

Перед тем, как описать API взаимодействия элементов системы записи, стоит определить, в каком виде передаётся метаинформация о записи от web АРМ КЦ до client Mixer.

Главным элементом идентификации записи в системе является структура «CallInfo» (Рис. 3):

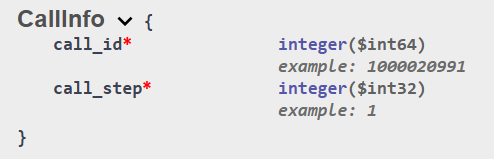


Рисунок 3 – структура CallInfo

Поля данной структуры описывают идентификатор записываемого звонка (call\_id) и порядковый номер записи звонка (call\_step). Порядковый номер записи звонка необходим для будущей обработки видео-отрывков одной записи при разных сценариях.

#### 2.4.1 API взаимодействия client Mixer и WEB АРМ КЦ

Запрос на начало записи имеет следующую структуру:

* Тип: POST
* Путь: «/start»
* Заголовки: отсутствуют
* Тип данных: application/json (таблица 3)
* Ожидаемые коды возврата:
  + 200 – OK
  + 400 – Некорректно сформировано тело запроса
  + 500 – Внутренняя ошибка

Таблица 3 ­­– поля объекта JSON из запроса «/start»

|  |  |  |
| --- | --- | --- |
| Наименование | Тип | Назначение |
| server\_mixer | List | основной и резервный адреса формате ip:port |
| logger\_url | String | URL сервера для отправки информационных сообщений о работе клиента |
| mediator\_url | String | URL сервера для последующей загрузки записи в хранилище |
| collection\_name | String | Наименование коллекции |
| params | Object | Объект, прозрачно передаваемый на сервер хранилища |
| call\_info | CallInfo | Объект для индентификации записи |
| bitrate\_video | Number(int32) | Ограничение битрейта видеодорожки. Задаётся в кбит/с |
| bitrate\_audio | Number(int32) | Ограничение битрейта аудиодорожки. Задаётся в кбит/с |
| fps | Number(int32) | Требуемое количество кадров в секунду от видеодорожки |
| log\_level | String | Уровень информационных сообщений, отправляемых на сервер во время записи |
| audio | Boolean | Требуется ли запись аудиодорожки |
| video | Boolean | Требуется ли запись видеодорожки |
| ping\_timeout | Number(uint64) | Время ожидания следующего запроса «/ping» в миллисекундах |

Запрос проверки записи:

* Тип: POST
* Путь: «/ping»
* Заголовки: отсутствуют
* Тип данных: application/json (рис. 3)
* Ожидаемые коды возврата:
  + 200 – OK
  + 400 – Некорректно сформировано тело запроса
  + 404 – Не найден call\_id/call\_step
  + 500 – Внутренняя ошибка

Запрос на остановку записи:

* Тип: POST
* Путь: «/stop»
* Заголовки: отсутствуют
* Тип данных: application/json (рис. 3)
* Ожидаемые коды возврата:
  + 200 – OK
  + 400 – Некорректно сформировано тело запроса
  + 404 – Не найден call\_id/call\_step
  + 500 – Внутренняя ошибка

#### 2.4.2 API взаимодействия client Mixer и Logger

Взаимодействие с сервером логов ведётся согласно API JSON Log Message Format [17]:

* Тип: POST
* Путь: logger\_url/path (из запроса «/start»)
* Заголовки: отсутствуют
* Тип данных: application/json (рис. 4)
* Ожидаемые коды возврата:
  + 200 – OK
  + 400 – Некорректно сформировано тело запроса
  + 500 – Внутренняя ошибка

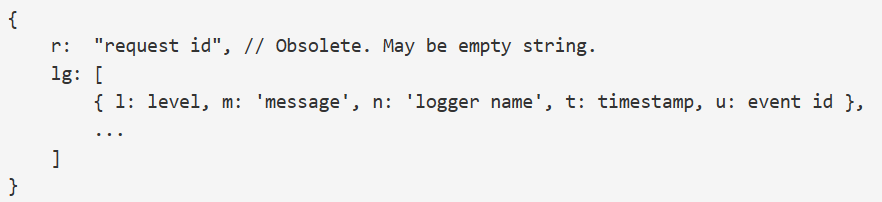


Рисунок 4 – пример запроса на сохранение

информационного сообщения на сервере.

#### 2.4.3 API взаимодействия client Mixer и server Mixer

Для создания контекста записи сервер Mixer ожидает следующую команду:

* Тип: POST
* Путь: /strt
* Заголовки: присутствуют (таблица 4)
* Тип данных: application/json
* Ожидаемые коды возврата:
  + 200 – OK
  + 400 – Некорректно сформировано тело запроса
  + 500 – Внутренняя ошибка

Таблица 4 ­­– заголовки запроса «/strt» на server Mixer

|  |  |  |
| --- | --- | --- |
| Наименование | Тип преобразования | Назначение |
| mediator-url | Строка (без преобразования) | URL сервера для отправки завершённой записи в хранилище |
| file\_name | Строка (без преобразования) | Имя файла |
| video\_width | Целое | Длина кадра видео в пикселях |
| video\_height | Целое | Высота кадра видео в пикселях |
| sample\_rate | Целое | Частота дискретизации аудиозаписи |
| video | Целое | Если == 1, то видеодорожка присутствует |
| audio | Целое | Если == 1, то аудиодорожка присутствует |

В сегменте данных запроса отправляется JSON объект с единственным объектом «Params» из запроса начала записи клиенту от Web АРМ КЦ, который система должна передать в хранилище.

При успешном создании контекста записи, помимо кода возврата, клиенту высылается заголовок с его уникальным идентификатором записи «uid», который необходимо использовать в последующих запросах в рамках данной записи.

Для посылки очередного пакета закодированного видео (h264) или аудио (aac) используется следующий запрос:

* Тип: POST
* Путь: /mpkt
* Заголовки: присутствуют (таблица 5)
* Тип данных: application/octet-stream
* Ожидаемые коды возврата:
  + 200 – OK
  + 400 – Некорректно сформировано тело запроса
  + 500 – Внутренняя ошибка

Таблица 5 ­­– заголовки запроса «/mpkt» на server Mixer

|  |  |  |
| --- | --- | --- |
| Наименование | Тип преобразования | Назначение |
| uid | Строка (без преобразования) | Уникальный идентификатор сессии записи |
| stream-id | Целое | Идентификатор дорожки внутри контейнера MP4 |
| pts | Целое | Presentation Timestamp – параметр в пакете, отвечающий за время начала его воспроизведения |
| dts | Целое | Decode Timestamp – параметр в пакете, отвечающий за время начала его декодирования |
| flags | Целое | Флаги отправляемого пакета |

В сегменте данных данного запроса передаётся поток байт из которых состоит очередной закодированный пакет медиа.

Для посылки запроса на окончание записи используется следующий запрос:

* Тип: POST
* Путь: /mpkt
* Заголовки: присутствуют
* Тип данных: отсутствует
* Ожидаемые коды возврата:
  + 200 – OK
  + 400 – Некорректно сформировано тело запроса
  + 500 – Внутренняя ошибка

Единственный заголовок, ожидаемый server Mixer при получении запроса на окончание записи – это «uid».

#### 2.4.4 API взаимодействия server Mixer и Mediator

В отличии от описанных ранее протоколов взаимодействия узлов системы, отличительной особенностью API Mediator является использование заголовков в «url-encoded» формате, а передача данных в формате «multipart/form-data».

Запрос загрузки файла на Mediator выглядит следующим образом:

* Тип: POST
* Путь: /upload
* Заголовки: присутствуют (таблица 6)
* Тип данных: multipart/form-data
* Ожидаемые коды возврата:
  + 200 – OK
  + 400 – Некорректно сформировано тело запроса
  + 500 – Внутренняя ошибка

Таблица 6 ­­– заголовки запроса «/upload» на server Mediator

|  |  |  |
| --- | --- | --- |
| Наименование | Тип преобразования | Назначение |
| sync | логическое | Отвечает за тип операции загрузки. По требованиям компании заказчика должно быть всегда true. В таком случае запрос закончится только когда данные будут сохранены в хранилище |

|  |  |  |
| --- | --- | --- |
| collection | Строка (без преобразования) | Параметр «collection\_name» из запроса «/start»  (Таблица 3) |
| params | Строка (без преобразования) | JSON объект «params» из запроса «/start»  (Таблица 3) |

## 3 Разработка проекта

### 3.1 Получение изображений экрана

Создание программного комплекса было начато с реализации функционала позволяющего получить изображения со всех рабочих столов ПК оператора КЦ.

Первоочередной задачей было решить проблему, связанную с тем, что мониторы могут различаться по своим характеристикам: иметь разное разрешение и располагаться в разных местах.

Для решения этой задачи было принято решение располагать изображения с мониторов слева направо в порядке их обнаружения, независимо от их программного (а в некоторых случаях и реального) расположения. При этом длина кадра подбирается как сумма длин всех изображений, а высота — как максимальная из высот всех изображений.

У такого решения есть недостаток: возможное непонимание реального расположения мониторов и, как следствие, трудности с определением положения указателя мыши при переходе с одного экрана на другой. Однако этот недостаток не является критичным для заказчика, поскольку основная цель записи экрана — возможность проверки добросовестности исполнения оператором своих обязанностей.

#### 3.1.1 Windows 10

Для получения снимков экрана программным способом в Windows 10 был использован Windows API. Windows API предоставляет разработчикам набор функций и структур для взаимодействия с операционной системой Windows [18]. Одной из таких функций является функция для получения изображений экрана.

Для получения изображений экрана с нескольких мониторов в Windows можно использовать функцию GetDC(). Эта функция возвращает дескриптор контекста устройства (DC), который представляет собой область памяти, содержащую информацию об устройстве вывода (например, мониторе).

Для каждого монитора можно получить свой дескриптор DC, а затем использовать его для получения изображения экрана. Для этого можно использовать функцию BitBlt() или GetDIBits(). Эти функции позволяют скопировать изображение из DC в буфер обмена или в память приложения.

Функция EnumDisplayMonitors является частью Windows API и используется для перечисления всех доступных мониторов в системе [19]. Эта функция принимает три параметра:

* LPRECT \*lpRect — указатель на структуру RECT, которая будет заполнена информацией о границах каждого монитора;
* LPDISPLAY\_DEVICEEX \*lpdd — указатель на структуру DISPLAY\_DEVICEEX, которая содержит информацию о мониторе;
* LPENUM\_PROC lpfnEnum — указатель на функцию обратного вызова, которая будет вызываться для каждого монитора.

Функция EnumDisplayMonitors перебирает все мониторы в системе и вызывает функцию обратного вызова для каждого из них. Функция обратного вызова должна заполнить структуру DISPLAY\_DEVICEEX информацией о мониторе.

После того как функция EnumDisplayMonitors завершает перебор всех мониторов, она возвращает значение TRUE, если были найдены мониторы, или FALSE, если мониторы не были найдены.

После того как изображения всех мониторов будут получены, их можно сохранить в файл или обработать каким-либо образом.

#### 3.1.2 Astra Linux

Astra Linux использует оконную систему X Window System, т.к. ОС основана на дистрибутиве Debian [10]. X Window System — это оконная система, которая широко используется в Unix-подобных операционных системах. Она позволяет приложениям создавать окна и отображать графику на экране.

Для получения изображения экрана в X Window System можно использовать функцию XGetImage(). Эта функция принимает следующие параметры:

* Display \*dpy — дескриптор соединения с сервером X;
* Window \*win — идентификатор окна, изображение которого нужно получить;
* XImage \*image — структура, в которую будет записано полученное изображение;
* int x, int y, unsigned int width, unsigned int height — координаты и размеры области экрана, изображение которой нужно получить.

Для получения дескриптора соединения с сервером X в X Window System необходимо использовать функцию XOpenDisplay(). Эта функция принимает один параметр — имя дисплея в формате host:display\_number, где host — имя хоста, а display\_number — номер дисплея.

Для получения координат области экрана в X Window System можно использовать функцию XGetGeometry. Эта функция принимает следующие параметры:

* Display \*dpy — дескриптор соединения с сервером X;
* Window \*win — идентификатор окна, координаты которого нужно получить;
* int \*x, int \*y — указатели на переменные, в которые будут записаны координаты верхнего левого угла окна;
* unsigned int \*width, unsigned int \*height — указатели на переменные, в которые будут записаны ширина и высота окна.

После того как координаты области экрана будут получены, их можно использовать для определения области, изображение которой нужно получить с помощью функции XGetImage.

Все вышеописанные функции данного подраздела возвращают значение 0 в случае успешного выполнения или -1 в случае ошибки.

После того как изображение экрана будет получено, его можно сохранить в файл или обработать каким-либо образом.

### 3.2 Получение аудио

Для кодирования кодеком aac необходимо количество сэмплов аудио (audio sample) равное 512, 1024 ил 2048 в формате плавающей точки, одинарной точности (IEEE 754). В проекте был настроен режим, требующий 512 сэмплов.

Вне зависимости от операционной системы был получен список всех аудиоустройств и разделён на устройства записи и воспроизведения. Далее, считанные сэмплы аудио объединяются в одну дорожку посредством суммирования и передаются для кодирования кодеком aac.

#### 3.2.1 Windows 10

Для получения списка аудиоустройств был использован объект WinAPI IMMDeviceEnumerator [19].

Объект IMMDeviceEnumerator представляет собой интерфейс, который позволяет получить доступ к списку доступных аудиоустройств в системе. Этот объект является частью Windows API (Winapi) и используется для перечисления всех доступных аудиоустройств, таких как звуковые карты, микрофоны и динамики.

Объект IMMDeviceEnumerator предоставляет следующие методы:

* EnumAudioEndpoints — перечисляет все доступные аудиоустройства в системе.
* GetDefaultAudioEndpoint — получает информацию о текущем аудиоустройстве по умолчанию.
* Reset — сбрасывает состояние объекта IMMDeviceEnumerator.

Для использования объекта IMMDeviceEnumerator был создан его экземпляр и вызван метод EnumAudioEndpoints. Этот метод принимает два параметра:

* DATAFORMAT — формат аудиоданных, который будет использоваться для перечисления аудиоустройств.
* DWORD — флаг, который определяет, какие аудиоустройства будут перечислены.

Метод EnumAudioEndpoints возвращает структуру IMMDeviceCollection, которая содержит список доступных аудиоустройств. Эта структура предоставляет следующие методы для работы с аудиоустройствами:

* Item — даёт доступ к определённому аудиоустройству в списке.
* Count — возвращает количество аудиоустройств в списке.

После того как объект IMMDeviceEnumerator был использован, его необходимо уничтожить с помощью метода Release.

Также, в проекте используется объект наследующий интерфейс IMMNotificationClient. IMMNotificationClient — это интерфейс в библиотеке Windows API (WinAPI), который используется для получения уведомлений о событиях, связанных с мультимедиа-устройствами.

Основные функции интерфейса IMMNotificationClient:

* Регистрация для получения уведомлений от системы.
* Обработка уведомлений о событиях мультимедиа-устройств, таких как подключение, отключение, изменение состояния и т. д.
* Предоставление информации о мультимедиа-устройствах, подключенных к системе.

Интерфейс IMMNotificationClient используется в приложениях, которые должны получать уведомления о событиях мультимедиа-устройств. Это могут быть приложения для управления мультимедиа-устройствами, приложения для мониторинга состояния устройств и т. д.

Для использования интерфейса IMMNotificationClient в приложении выполнены следующие шаги:

* Получен указатель на интерфейс IMMNotificationClient с помощью функции CoCreateInstance.
* Объект зарегистрирован для получения уведомлений с помощью функции IMMNotificationClient::Register.
* Уведомления обрабатываются с помощью функции IMMNotificationClient::OnNotification.

Данный интерфейс используется для регистрации случаев изменения списка аудиоустройств. В таких случаях приложение клиента перезагружает стек аудио.

#### 3.2.2 Astra Linux

Astra Linux использует Pulse Audio сервер, т.к. он основан на дистрибутиве Debian. Pulse Audio — это звуковая система, которая широко используется в Linux и других Unix-подобных операционных системах. Она позволяет приложениям получать доступ к аудиоустройствам и воспроизводить звук.

Для получения списка всех устройств в Pulse Audio API можно использовать функцию pa\_context\_get\_devices(). Эта функция принимает один параметр — указатель на контекст Pulse Audio, который должен быть предварительно создан с помощью функции pa\_context\_new().

Функция pa\_context\_get\_devices() возвращает список всех устройств, подключённых к контексту Pulse Audio. Этот список представлен в виде структуры PaSystemDeviceInfo, которая содержит следующую информацию о каждом устройстве:

* name — имя устройства;
* description — описание устройства;
* driver — драйвер устройства;
* device\_id — идентификатор устройства;
* default\_sample\_spec — спецификация сэмпла по умолчанию для устройства.

Для чтения аудиоданных из Pulse Audio используется функция pa\_simple\_read. Эта функция принимает следующие параметры:

* handle — дескриптор аудиопотока;
* buffer — указатель на буфер, в который будут записаны аудиоданные;
* frames — количество аудиосэмплов, которые будут прочитаны;
* err — переменная, в которую будет записана ошибка, если она произошла.

### 3.3 Кодирование

Для кодирования данных была использована библиотека libAV. Ключевыми факторами, способствовавшими использованию, данной библиотекой являются:

* **Широкая распространённость.** По данной причине в открытых источниках информации гораздо легче найти решение некоторых проблем, возникающих при разработке программного комплекса.
* **Кроссплатформенность.** Данный фактор позволил написать код не требующий существенных изменений при портировании приложения под разные операционные системы.
* **Унифицированный интерфейс.** Библиотека предоставляет унифицированный интерфейс для настройки и использования кодеков и медиа-контейнеров. Этот фактор крайне удобен тем, что вероятность допущения ошибок в коде уменьшается, т.к. большинство схожих участков кода объединяются в функции, проверить корректность работы которых гораздо проще.

#### 3.3.1 Кодирование видео

Для данного проекта было принято решение использовать кодек H264. Этот выбор был обусловлен следующими причинами:

* H264 является одним из наиболее распространённых кодеков в мире, что делает его идеальным выбором для обеспечения широкой совместимости.
* Большинство современных видеоадаптеров поддерживают аппаратное кодирование видео с использованием кодека H264, что обеспечивает высокую эффективность и производительность при обработке видео. Это означает, что видео, закодированное с помощью H264, может быть воспроизведено без значительной нагрузки на процессор, что особенно важно для устройств с ограниченной вы-числительной мощностью.

Таким образом, выбор H264 для этого проекта был обоснован его широкой распространённостью и поддержкой аппаратного кодирования.

#### 3.3.2 Кодирование аудио

Для кодирования аудио был выбран кодек AAC. Использование кодека AAC в контейнере MP4 для записи работы оператора call-центра было обосновано следующими причинами:

* Эффективность сжатия: кодек, который обеспечивает более эффективное сжатие аудио по сравнению с MP3 при сохранении или даже улучшении качества звука. Это означает, что файлы будут занимать меньше места на диске без потери качества звука, что является важным фактором при хранении большого количества записей звонков.
* Качество звука: AAC обеспечивает высокое качество звука, даже при низких битрейтах. Это особенно важно для call-центров, где четкость и ясность звука могут существенно влиять на качество обслуживания клиентов.

### 3.4 Сетевое взаимодействие

#### 3.4.3 TCP

Для облегчения сетевого взаимодействия была выбрана библиотека libevent2. Данный выбор был сделан, принимая во внимание следующие преимущества использования библиотеки:

* Поддержка различных платформ: Libevent2 поддерживает различные платформы, такие как Windows, Linux, macOS и другие. Это позволило писать гораздо меньше кода для использования сетевого стека TCP на разных операционных системах.
* Гибкость и расширяемость: из-за возможности дополнения функционала библиотеки посредством добавления своего исполняемого кода, библиотеку крайне удобно интегрировать в проект.

#### 3.4.4 HTTP2

Основными кандидатами на интеграцию в проект в качестве библиотеки для работы с HTTP/2 были libnghttp2 и libcno. При разработке проекта для взаимодействия по протоколу HTTP/2 была выбрана библиотека libcno. Не смотря на то, что CNO реализует менее богатый функционал для работы по протоколу HTTP/2 (например, отсутствие поддержки приоритезации потоков) по сравнению с nghttp2 и гораздо меньшей распространённости, libcno существенно выигрывает в скорости обработки запросов.

Тестирование скорости библиотек производилось при помощи нагрузочного приложения h2load. Параметры загрузки были следующие:

* Количество запросов: 10^7
* Количество параллельных запросов внутри одного соединения: 100

Запросы были максимально простыми: минимум заголовков и отсутствие сегмента данных.

Характеристики ПК, на котором производилось сравнение:

* ЦП: Intel Core i5 10400
* RAM: 16 Gb (свободно 80%)

Полученный результат:

* libnghttp2: 505 \* 10^3 запросов/сек.
* libcno: 1203 \* 10^3 запросов/сек.

### 3.5 Обеспечение возможности восстановления работы сервера

Для обеспечения возможности восстановить работу сервера при аварийных или штатных завершениях работы была разработана следующая структура каталогов в папке с сервером:

* audio – текущие аудиозаписи в формате mp4
* video – текущие видеозаписи в формате mp4
* meta – метаданные текущих записей
* done – метаданные для завершённых записей

Пока сервер принимает фрагменты записи от приложения клиента, он записывает фрагменты в соответствующие папки (audio или video). При этом, в папку meta сохраняется файл JSON с полем «params», которое необходимо будет передать в хранилище.

При окончании записи, создаётся файл JSON в папке Done, который содержит следующую информацию:

* Имя файла, к которому относится данный JSON
* URL адрес сервера Mediator, на который необходимо отправить файл.
* Поле «params»

Предполагается, что данные папки будут существовать вне docker-контейнера и привязываться к директориям внутри него. При этом, возможен повторный запуск приложения поверх существующих папок. Если работа сервера была завершена аварийно, то будут потеряны только нефинализированные записи. Сервер при подобном запуске продолжит приём новых запросов на запись и отправку готовых записей.

### 3.6 Обеспечение возможности восстановления работы клиента

Для перезапуска приложения клиента была разработана небольшая утилита «stream app monitor». Данная утилита следит за списком запущенных процессов и, в случае если не находит там client Mixer, перезапускает его. Утилита находится в одной папке с приложением клиента.

### 3.7 Отправка готовых записей на сервер «Mediator»

Согласно функциональным требованиям, готовые записи должны быть отправлены на Mediator в формате Multipart/form-data. Однако согласно спецификации формата, при отправке данных с его использованием, необходимо найти уникальную строку-разделитель. Данная строка представляет собой последовательность символов длиною от 1 до 70 символов, которая является уникальной и не встречается ни в одном из объектов данных. Прямое выполнение данного требования – поиск уникальной подстроки символов внутри mp4 файла, может значительно замедлить работу сервера.

Решением данной задачи стала передача данных, закодированных в формате base64. В данном формате ограниченное количество символов, а следственно растёт объём передаваемых данных: на каждые 3 байта данных – 4 байта переданной информации. Однако, ограниченный набор символов делает тривиальной задачу поиска уникальной последовательности символов для разделителя. В проекте разделителем является символ «\_» (нижняя черта).

Согласно договорённости с разработчиками Mediator, принимающая сторона декодирует данные согласно проставленным заголовкам «mime-type» и «content-encoding», и сохраняет полученные файлы в хранилище с расширением mp4.

### 3.8 Развёртывание проекта

#### 3.8.1 client Mixer

Для распространения приложения клиента была создана программа установки при помощи комплекса утилит NSIS (Null Software Installer). Для установки клиента требуются права администратора, для добавления приложения в исключения брандмауэра windows, добавления SSL сертификатов приложения в список доверенных и установки автозапуска приложения и утилиты для перезапуска при старте операционной системы.

Конфигурация client Mixer происходит при помощи файла «conf.json» в папке config. Текущие поля для конфигурации клиента:

* domain [строка] – IP-адрес или домен, на который ВЕБ АРМ должен посылать запросы.
* port [строка] – TCP порт, на который ВЕБ АРМ должен посылать запросы
* out\_video\_height [целое] – высота в пикселях кадров исходящего видеопотока
* out\_video\_width [целое] – длина в пикселях кадров исходящего видеопотока
* log\_ssl [логическое] – параметр контролирует создание и дополнение файлов ssl\_log\_server(client). В них записываются ключи для TLS соединений с ВЕБ АРМ и Server MIXER соответственно. Данный параметр необходим для отладки

#### 3.8.2 server Mixer

Сервер MIXER представляет из себя приложение со статическими библиотеками и распространяется как исполняемый файл внутри docker-контейнера.

Конфигурация server Mixer происходит при помощи файла «conf.json» в папке config. Текущие поля для конфигурации сервера:

* domain [строка] – IP-адрес или домен, который будет прослушивать сервер.
* port [строка] – TCP порт, который будет прослушивать сервер.
* log\_level [целое] – число обозначающее уровень максимальный уровень логов, для вывода:0 - SUPER\_PRIORITY - выводятся только самые критичные логи (например, обработки конфига)10 - NOTICE - выводятся все логи уровня VERBOSE, кроме дампа заголовков сообщений15 - VERBOSE - выводятся все логи уровня DEBUG, кроме дампа заголовков http пакетов20 - DEBUG - выводятся абсолютно все логи.
* video\_output\_dir [строка] – путь к директории для временного хранения записанных файлов .mp4 с присутствующей видео-дорожкой
* audio\_output\_dir [строка] – путь к директории для временного хранения записанных файлов .mp4 с отсутствующей видео-дорожкой
* meta\_output\_dir [строка] – путь к директории для временного хранения метаданных вызова (json файла с объектом "params") переданных ВЕБ АРМ КЦ
* done\_output\_dir [строка] – путь к директории для временного хранения метаданных файлов записей, которые используются для их посылки на MEDIATOR
* mediator\_send\_query\_size [целое] – количество готовых записей mp4 одновременно посылаемых на MEDIATOR
* tcp\_client\_reconnect\_timeout [целое] – время в секундах, которое должно пройти прежде чем произойдёт попытка переподключения к MEDIATOR
* record\_session\_lifetime [целое] – время в секундах через которое будет удалён контекст записи. При этом, записываемый файл .mp4 финализируется
* file\_output\_buffer\_size [целое] – размер буфера в байтах для буферизованной записи файлов на диск
* file\_send\_buffer\_size [целое] – размер буфера в байтах для загрузки файла перед отправкой. То есть, из файла будет считываться указанное количество байт и пока они не отправятся, данные из файла считываться не будут
* key\_file [строка] – путь к файлу ключа формата .key
* cert\_file [строка] – путь к файлу сертификата формата .pem
* log\_on [логическое] – если установлено начение true, то сервер пишет логи в файлы и на консоль
* log\_ssl [логическое] – если установлено начение true, то сервер пишет лог ssl ключей для декодирования пакетов. Также, при включённом состоянии выводится предупреждение при запуске сервера

Для работы сервера необходимы:

* файл с описанием конфигурации работы под названием conf.json в директории config, в папке сервера;
* Сертификат формата .pem и ключом формата .key, на которые должен указывать conf.json.

## 4 Нагрузочное тестирование

### 4.1 Цели нагрузочного тестирования

Главной целью нагрузочного тестирования стал анализ поведения server Mixer, при параллельной записи аудио- и видеозаписей mp4. Для создания нагрузки было написано программное обеспечение на языке Python с использованием библиотек pynghttp2 и asyncio.

### 4.2 Результаты нагрузочного тестирования

Конфигурационный файл сервера описан следующим образом:

{

"domain" : "0.0.0.0",

"port" : "65500",

"log\_level" : 1000,

"video\_output\_dir": "video",

"audio\_output\_dir": "audio",

"meta\_output\_dir": "meta",

"done\_output\_dir": "done",

"mediator\_send\_query\_size": 10,

"record\_session\_lifetime" : 30,

"file\_output\_buffer\_size": 524288,

"file\_send\_buffer\_size": 65535,

"key\_file": "certs/server.key",

"cert\_file": "certs/server.pem",

"log\_on" : false,

"log\_ssl" : true

}

Характеристики сервера:

* ЦП: Core i9 12900k
* RAM: 125 GiB (Free 34 GiB)

Параметры записей, которые посылались на server Mixer:

* Аудио. Битрейт - 19 кбит/c;
* Видео. Битрейт - 60 кбит/c, 2 fps

Результаты нагрузочного тестирования представлены ниже (таблица 7)

Таблица 7 – Результаты нагрузочного тестирования

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Кол-во клиентов | Параметры теста | CPU, %, пиковое значение | Память, MB | Диск, MB/s, Средняя скорость записи на диск | Сеть, MB/s, средняя скорость приёма |
| 0 | Без логов | 0.2 | 8.695 | 0.176 | 0.008 |
|  | С логами | 0.1 | 8.691 | 27.5 | 1.189 |
| 1 | По сети - без логов | 1.2 | 11.824 | 15.5 | 2.4 |
|  | По сети - с логами | 1.6 | 11.766 | 10.9 | 0.772 |
| 10 | По сети - без логов | 14 | 38.5 | 10.4 | 1.130 |
|  | По сети - с логами | 27 | 38.71 | 11.9 | 1.067 |
| 50 | По сети - без логов | 15 | 34 | 157.1 | 10.7 |
|  | По сети - с логами | 25 | 39 | 156.9 | 9.8 |
| 100 | По сети - без логов | 56,00 | 306,00 | 5.286 | 3.033 |
|  | По сети - с логами | 62 | 305 | 13.1 | 3.448 |
| 300 | По сети - без логов | 86,00 | 900 | 3.478 | 7.594 |
|  | По сети - с логами | 96 | 900 | 13.4 | 8.863 |
| 330 | По сети - без логов | 100,00 | 992 | 3.868 | 8.736 |
|  | По сети - с логами | 100,00 | 993 | 14.7 | 9.245 |

Проанализировав результаты, был сделан вывод, что процесс логирования может значительно замедлять работу сервера из-за создания большой нагрузки на подсистему хранения данных (например, если это жёсткий диск). Также, был исследован максимум количества клиентов, обрабатываемых одним сервером в данной конфигурации.

## 5 Специальные вопросы обеспечения безопасности

### 5.1 Общая характеристика алгоритма оценки кардио-васкулярной реактивности при психофизиологической нагрузке

В данной работе представлено программное обеспечение для записи изображения с экранов и звуков с аудиоустройств ПК оператора call-центра во время работы, который разработан на языке программирования С++ с использованием соответствующих библиотек, а также с набором необходимых функций в среде программирования CLion2022.2.2.

Программный комплекс, о котором идет речь в данном контексте, представляет собой набор программ, который может функционировать как автономное приложение или быть интегрированным в состав других программных комплексов. Основная цель его использования — запись изображений с экранов и аудио с аудиоустройств и сохранение их в предоставленном хранилище в формате MP4.

Данный программный комплекс состоит из двух частей. Клиентская часть устанавливается на персональный компьютер оператор call-центра, а серверная часть устанавливается на ЭВМ, соответствующую аппаратным требованиям, установленным для серверной части комплекса. Это позволяет управляющему персоналу call-центра просматривать видеозаписи с работой оператора непосредственно со своего рабочего места, что значительно повышает эффективность их работы в области физиологии и психофизиологии.

Программный модуль был разработан с учетом потребностей заказчиков (управляющего персонала call-центров) и направлен на улучшение качества их работы. Он не требует дополнительного обслуживания после первоначальной установки и настройки, а также проведения инструктажа, так как является частью более крупного программного комплекса WEB Автоматизированного Рабочего Места.

Однако стоит отметить, что в процессе использования результатов работы программы оператор в течение рабочего дня взаимодействует с ПК. В связи с этим, несоблюдение требований безопасности может привести к возникновению ряда неприятных симптомов, таких как головные боли, резь в глазах, усталость, раздражительность, нарушение сна и аппетита, ухудшение зрения, боли в руках, шее, пояснице и т.д. При ненормированной работе возможно нервное истощение. Поэтому крайне важно соблюдать нормы и рекомендации по безопасности при работе за ПК [20]. Ниже будет рассмотрен более подробный анализ риска потенциальных опасностей, связанных с использованием разрабатываемого ПО и конечным пользователем – врачом.

### 5.2 Классификация разрабатываемого программного обеспечения

При оценке возможного вреда или опасности, которые может представлять ПО для человека, технической системы или окружающей среды, важно провести классификацию данного ПО или набора программных средств. Один из методов такой классификации представлен в ГОСТ Р ИСО/МЭК ТО 12182–2002 «Информационная технология (ИТ). Классификация программных средств»[21-22].

В рамках этого стандарта вводится концепция схемы классификации (categorization schema). Схема классификации представляет собой упорядоченную комбинацию различных видов и классов, которые связаны с программным обеспечением [23-26].

Это означает, что ПО классифицируется в соответствии с определенной схемой, которая включает в себя различные категории и подкатегории. Эти категории могут быть основаны на различных аспектах ПО, таких как его функциональность, сложность, потенциальные риски и так далее.

Такая классификация позволяет более точно оценить потенциальные риски и опасности, которые может представлять программное обеспечение, и принять соответствующие меры для их минимизации. Это особенно важно в контексте здравоохранения, где безопасность пациентов является приоритетом. Классификация также помогает в определении требований к безопасности и надежности программного обеспечения, а также в выборе подходящих методов тестирования и верификации [26].

Проведем оценку разрабатываемого ПО (таблица 8), а также ПО, которое использовалось при разработке (таблица 9), согласно ГОСТ Р ИСО/МЭК ТО 12182–2002, чтобы создать алгоритм оценки кардио-васкулярной реактивности при психофизиологической нагрузке.

Таблица 8 – Оценка разрабатываемого алгоритма

|  |  |
| --- | --- |
| Программное обеспечение | программное обеспечение для записи изображения с экранов и звуков с аудиоустройств ПК оператора call-центра |
| Функция ПО | Программный комплекс для записи видео |
| Режим  эксплуатации | Пакетная обработка данных (ПС получает команды, которые обрабатываются ПС) |
| Прикладная область информационной системы | Прикладная система. Предназначена для использования в сфере записи экранов и аудио |
| Масштаб ПО | Средний (система может расширяться за счет добавления нового функционала) |
| Представление данных | Объектное (формата JSON), форматированный файл |
| Критичность ПО | Секретность (записи для внутреннего использования) |
| Класс  пользователя | Специалист (необходимо прохождение обучения  у производителя ПС) |
| Требуемые рабочие характеристики | Емкость – средняя и высокая (размер записи ограничен только доступностью вычислительных ресурсов);  время отклика – быстрое, порядка 5 миллисекунд; производительность – высокая (одновременно несколько записей) |
| Стабильность ПО | Дискретное внесение изменений, не вызывающее системные сбои |
| Требование защиты | Защита от несанкционированного доступа: средняя;  Контрольный след: сильная (при несовпадении контрольной суммы или постороннем вмешательстве выполнение программы прекращается);  Защита программ и данных: сильная (при постороннем вмешательстве выполнение программы прекращается) |
| Требование надежности | Завершенность: высокая;  отказоустойчивость: слабая; при возникновении системного сбоя  восстанавливаемость ­­— быстрая |
| Вычислительная система и среда | Управляемый ПК (операционная система MS Windows, Astra Linux) |
| Требования к вычислительным ресурсам | Оперативная память – память с произвольной выборкой (согласно рекомендуемым требованиям операционной системы).  Внешняя память – массовое запоминающее устройство (согласно рекомендуемым требованиям операционной системы) |
| Готовность программного продукта | Готовый коммерческий;  запатентованный (лицензионный) |
| Использование программных данных | Для множества пользователей |
| Исходный язык | Компилируемый C++ |

Таблица 9 – Оценка используемого ПО для разрабатываемого алгоритма

|  |  |
| --- | --- |
| Программное обеспечение | Программа, написанная на языке C++ |
| Вид | Класс |
| Функция ПС | Разработка программ |
| Режим  эксплуатации | Отображение биомедицинских сигналов (пользователь ПС вводит команды, которые обрабатываются ПС) |
| Прикладная область информационной системы | Наука. Предназначена для использования в сфере здравоохранения |

### 5.3 Анализ риска потенциальных опасностей

Безопасность жизнедеятельности человека, в частности медицинского персонала, который взаимодействует с разрабатываемым алгоритмом оценки кардио-васкулярной реактивности при психофизиологических нагрузках, во многом определяется правильной оценкой потенциально вредных и опасных производственных факторов. Это важно, поскольку взаимодействие с ПО становится все более распространенным в нашей повседневной жизни, и мы должны быть осведомлены о возможных рисках.

Согласно ГОСТ 12.0.003-2015 сформированы основные типы этих факторов, оказывающих опасное воздействие:

1. Физические факторы: это могут быть различные виды излучения; блескость экрана дисплея; опасность поражения электрическим током; шум, который превышает допустимую норму; недостаточная освещенность рабочего места, температура, влажность воздуха и другие физические условия, которые могут негативно влиять на здоровье человека при взаимодействии с ПО. Например, длительное взаимодействие с компьютером может привести к проблемам со зрением или мускулоскелетным расстройствам (артрит, сколиоз и другие деформации позвоночника) [23-24].
2. Химические факторы: хотя это может быть менее применимо к ПО, однако, можно отметить активные частицы, которые образуются в процессе ионизации воздуха во время работы за ПК и постоянно воздействуют на пользователя (врача-оператора) [23-24].
3. Биологические факторы: это могут быть бактерии, вирусы или другие микроорганизмы, которые могут быть переданы через оборудование или программное обеспечение. Например, если оборудование не очищается должным образом, оно может стать источником инфекции [23-24].
4. Психофизиологические факторы: возможна психофизиологическая нагрузка (стресс, усталость, умственно напряжение и другие психологические или физиологические состояния, которые могут возникнуть при взаимодействии с ПО). Например, перегрузка информацией может привести к ухудшению внимания и концентрации, что в свою очередь может привести к ошибкам [23-24].

Важно помнить, что эти факторы могут влиять не только на здоровье, но и на общую производительность и благополучие человека. Поэтому важно принимать меры для минимизации этих рисков, которые будут рассмотрены в следующем подразделе.

### 5.4 Качество программного обеспечения

На сегодняшний день мы сталкиваемся с активным развитием информационных технологий в области здравоохранения, однако это сопровождается рисками случайных и неблагоприятных событий, связанных с применением медицинского ПО. Эти риски становятся все более заметными. С каждым днем создаются все более сложные медицинские программы, которые помогают врачам принимать решения на самом высоком уровне. Важно помнить не только о безопасности пациента, но и о враче, работающем с любой медицинской информационной системой (МИС), поскольку существует высокая вероятность возникновения рисков неблагоприятных событий при работе с МИС.

Любое разрабатываемое ПО, которое является безопасным как для взаимодействия с пациентом, так и для непосредственного пользователя — врача, в соответствии с целью и желаемым конечным результатом, никогда не может обеспечить 100 % гарантии из-за невозможности устранения всех потенциальных рисков. Поэтому основное внимание следует уделить минимизации рисков настолько, насколько это возможно, но не обязательно их полному устранению, на протяжении всего жизненного цикла медицинского ПО.

Согласно ГОСТ Р ИСО/МЭК 12207-99 «Информационная технология. Процессы жизненного цикла программных средств», процесс разработки и внедрения программного обеспечения включает следующие ключевые этапы:

1. Определение модели жизненного цикла. На этом этапе принимается решение о выборе наиболее подходящей модели жизненного цикла для выбранного проекта.
2. Идентификация и анализ требований. Этот этап включает в себя определение и анализ требований к ПО, где определяются потребности пользователей и происходит преобразование этих потребностей в формализованные требования.
3. Дизайн и проектирование ПО. На этом этапе создается детальная структура ПО, включая архитектуру системы, выбор алгоритмов и структур данных.
4. Программирование ПО. Этот этап включает непосредственное написание кода на основе спроектированной структуры.
5. Тестирование ПО. На этом этапе проводится проверка ПО на наличие ошибок и соответствие требованиям.
6. Реализация (внедрение) ПО. Здесь программное обеспечение устанавливается и настраивается в рабочей среде.
7. Приемка ПО. На этом этапе заказчик проверяет ПО на соответствие его требованиям и принимает решение о его приемке [27].

На рис. 5 схематично приведена каскадная модель жизненного цикла разрабатываемого алгоритма оценки кардиоваскулярной реактивности при психофизиологических нагрузках. Данная модель имеет свои преимущества и недостатки. Которые будут рассмотрены ниже.

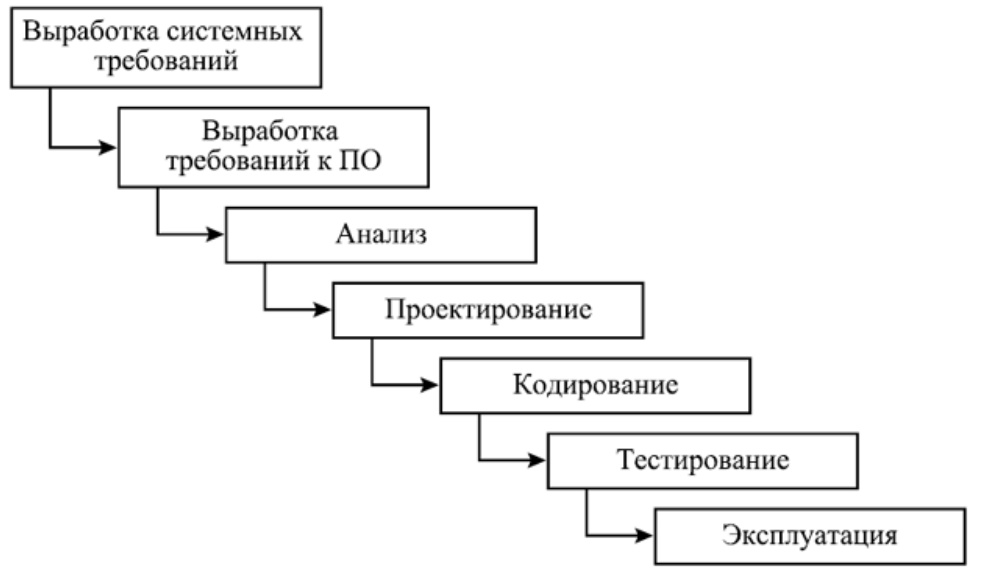


Рисунок 5 – Каскадная модель жизненного цикла

К преимуществам можно отнести:

Выполнение этапов проекта в строго установленной последовательности, что позволяет оценивать качество продукта на каждом этапе.

Однако, существуют и недостатки:

1. Отсутствует обратная связь между этапами, что может привести к проблемам в процессе разработки.
2. Такой подход не всегда соответствует реальным условиям разработки программного продукта, где этапы могут перекрываться или повторяться [27-28].
3. Этот подход относится к первой группе моделей, что означает, что он, может быть, не так эффективен для сложных или динамически изменяющихся систем.

Важно отметить, что большинство ошибок возникает на ранних стадиях разработки, поэтому при выборе модели жизненного цикла программного обеспечения особое внимание уделяется вопросам последовательности выполнения и взаимосвязи включения важных видов работ. Кроме того, очень важно провести тщательный анализ требований для разработки ПО.

Возможны ситуации, когда требования могут быть двусмысленными. В одном случае, двусмысленность может заключаться в том, что пользователь, а именно заказчик ПО, может интерпретировать одно и то же требование по-разному. В другом случае, у разных читателей может возникнуть разное понимание того, что означает требование. Это подчеркивает важность ясности и однозначности при формулировании требований [29].

Класс безопасности медицинского ПО регулируется следующим ГОСТом Р МЭК 62304-2013 «Национальный стандарт Российской Федерации. Изделия медицинские. Программное обеспечение. Процессы жизненного цикла», согласно которому разработанный программный модуль можно отнести к классу В, который означает возможность незначительных травм [30].

#### 5.4.1 Представление пользователя и разработчика

Разработка пользовательского интерфейса представляет собой неотъемлемую часть любого проекта, связанного с созданием ПО. Интерфейс пользователя является ключевым моментом взаимодействия человека и программы, которая часто обладает сложной функциональностью. От качества интерфейса зависит общее восприятие программы пользователем, и именно на основе этого впечатления пользователь принимает решение о дальнейшем использовании ПО. Следовательно, успех разработки продукта непосредственно зависит от того, насколько удобным будет его пользовательский интерфейс [31].

Наличие нечитаемого кода в программе влечет за собой риск трудностей в изменении и поддержке программного обеспечения для представления разработчика.

Также для пользователя важным вопросом становится проблема безопасности данных и системы, а также возможность безопасного взаимодействия с внешней информацией и обеспечения стабильной работы компьютера. В условиях активного развития Интернета, посредством которого распространяются вирусы и вредоносные программы, такие как троянские программы и вирусы, внешние угрозы становятся все более актуальными и серьезными [31-32].

#### 5.4.2 Методы обнаружения нежелательного ПО

При проверке программных средств на наличие вирусов существуют две основные группы методов обнаружения и защиты от них: программные и аппаратно-программные. Простейший метод обнаружения вирусов — это сканирование, когда антивирусная программа последовательно проверяет файлы на наличие сигнатур известных вирусов. Сигнатура представляет собой уникальную последовательность байтов, свойственную конкретному вирусу, и не встречающуюся в других программах [33].

Эвристический анализ в соответствии с ГОСТ Р 51188-98 «Испытания программных средств на наличие компьютерных вирусов» представляет собой новый метод обнаружения вирусов, позволяющий находить ранее неизвестные угрозы. Он сканирует другие программы и загрузочные секторы дисков без необходимости предварительного сбора данных о файловой системе, что отличает его от метода обнаружения изменений [33].

Вакцинирование является способом защиты конкретной программы от вирусов путем подключения к ней специального модуля контроля, следящего за целостностью файла. Метод использует контрольную сумму программы или другие характеристики, и, если файл заразился вирусом, модуль контроля обнаружит изменение и сообщит об этом.

Кроме программных методов, существуют аппаратно-программные методы защиты от вирусов, которые основаны на использовании специализированного устройства — контроллера, подключаемого к ПК через разъем расширения. Этот контроллер управляется специальным ПО и реализует один из вышеописанных методов защиты [34].

#### 5.4.3 Основные положения эргономики

Одним из ключевых требований в процессе разработки интерфейса между человеком и системой являются нормативные указания, представленные в ГОСТ Р ИСО 9241-100:2010. Согласно этому стандарту, эргономика определяется как научное направление, изучающее вопросы взаимодействия человека с другими элементами системы. При проектировании программного модуля необходимо учитывать рекомендации эргономики при формировании пользовательского интерфейса, включая использование технологии отображения, в частности дисплеев [35].

Оценка качества изображения на дисплее основана на нескольких критериях, которые регламентируются эргономическими стандартами. Во-первых, размер видимого изображения на экране должен соответствовать определенным параметрам: по ГОСТу предпочтительно, чтобы диагональ экрана составляла не менее 14 дюймов [35-36].

Во-вторых, частота обновления изображения имеет значительное значение для комфортного восприятия пользователем. Согласно современным стандартам, рекомендуется, чтобы частота обновления была не менее 70 Гц, чтобы предотвратить мерцание изображения, что может вызывать раздражение и утомление глаз [35].

Кроме того, символы на экране должны быть четко различимы и легко узнаваемы пользователем, что обеспечивает комфортное взаимодействие с программным интерфейсом.

Наконец, регулировка яркости и контрастности между символами и фоном также имеет важное значение для обеспечения оптимального уровня комфорта пользователей. Эти критерии являются основополагающими для создания пользовательского интерфейса, который соответствует требованиям эргономики и обеспечивает комфортное использование программного обеспечения [35-36].

### 5.5 Безопасность ПО и его оценка

Безопасность программного обеспечения играет ключевую роль в предотвращении психофизиологически вредных факторов, связанных с его использованием. Качество ПО, оцененное в соответствии с ГОСТ ИСО/МЭК 9126-2001, обеспечивает ряд положительных характеристик, способствующих снижению риска и сохранению здоровья [37]:

Устойчивость: ПО с высоким уровнем устойчивости снижает вероятность возникновения сбоев и ошибок, что помогает избежать стресса и психологического дискомфорта у пользователей (в нашем случае у врача‑оператора).

Удобство использования: ПО с интуитивно понятным интерфейсом и легким доступом к функциям уменьшает нагрузку на психические ресурсы пользователей, делая процесс работы с ПО более комфортным и эффективным.

Производительность: эффективное ПО позволяет пользователям выполнять задачи быстро и без лишних усилий, что снижает время, проведенное перед экраном, и сокращает психофизиологическую усталость.

Безопасность: ПО с высоким уровнем защиты данных и надежной системой безопасности минимизирует риск нарушений конфиденциальности и несанкционированного доступа, обеспечивая психологическое спокойствие пользователей.

Доступность: ПО, которое легко доступно и понятно для пользователей с разным уровнем навыков, снижает адаптационные трудности и улучшает удовлетворенность пользователей от использования программы.

Для снижения риска негативного воздействия на психофизиологическое состояние пользователей и обеспечения сохранения здоровья, рекомендуется выбирать качественное программное обеспечение, соответствующее требованиям ГОСТ ИСО/МЭК 9126-2001, соблюдать меры информационной безопасности и регулярно обновлять ПО для минимизации уязвимостей. Кроме того, обучение пользователей по безопасному и эффективному использованию программы, а также предоставление возможности настройки интерфейса в соответствии с индивидуальными предпочтениями, также способствует уменьшению риска и повышению общего уровня удовлетворенности пользователей [37].

### 5.6 Требования к физическому и графическому интерфейсу

Требования к физическому и графическому интерфейсу ПО играют ключевую роль в процессе разработки. Они определяют способ взаимодействия пользователя с системой, влияя на удобство использования и эффективность работы.

Физический интерфейс ПО охватывает все аспекты взаимодействия пользователя с системой, которые имеют физическое измерение. Это может включать такие элементы, как размер экрана, разрешение, частота обновления изображения и другие параметры дисплея. Важно, чтобы физический интерфейс был удобен для пользователя и обеспечивал четкое и ясное отображение информации [38].

Научно-исследовательский анализ принципов разработки ПО выдвигает ряд требований к физическому и графическому интерфейсам. В соответствии с работами Марина и Холланда (2017) и Смита (2019), физический интерфейс должен обладать:

Простотой использования, т. е. интерфейс должен быть интуитивно понятным для пользователя (врача-оператора), даже без предварительного обучения.

Эргономичностью, следовательно, интерфейс должен быть адаптирован к особенностям человеческого восприятия и движений с целью уменьшения усталости и повышения производительности пользователя.

Консистентностью, т. е. элементы интерфейса должны быть последовательными и предсказуемыми. Это позволяет пользователям легко ориентироваться и уменьшает вероятность ошибок [39, 40].

Графический интерфейс пользователя (GUI) представляет собой метод взаимодействия пользователя с компьютером с использованием графических элементов, таких как окна, кнопки и меню. Он должен быть интуитивно понятным и легким в использовании. Ключевыми требованиями к GUI являются четкость символов, яркость и контраст между символами и фоном, а также возможность корректировки этих параметров пользователем [41]. Разрабатываемый алгоритм оценки кардио-васкулярной реактивности при психофизиологической нагрузке соответствует вышеуказанным требованиям.

В свою очередь, графический интерфейс, согласно исследованиям Джонсона (2018) и Ли (2020), должен характеризоваться привлекательным дизайном, ясной структурой, индивидуальностью, интерактивностью и игровым подходом. Эти характеристики направлены на повышение удовлетворенности и привлекательности пользовательского опыта [42, 43].

Одним из основных требований к хорошему графическому интерфейсу ПО является концепция «делай то, что я имею в виду» или DWIM (Do What I Mean). Согласно этой концепции, система должна быть способна интерпретировать действия пользователя так, как он их предполагал, даже если они не были выполнены точно [44-45].

Эти аспекты обусловлены стремлением к максимальной эффективности взаимодействия пользователя с программным продуктом.

### 5.7 Требования к охране труда за ПК

Основным руководящим документом в области охраны труда при работе с ПК «Инструкция по охране труда при работе на персональном компьютере» ТОИ Р-45-084-, в котором отмечены следующие моменты [46]:

1. Рекомендуется ограничить время работы за компьютером шестью часами в смену.
2. Перерывы в работе за ПК рекомендуется делать каждые 50 мин продолжительностью 10 мин.
3. Непрерывная работа за компьютером без перерыва не должна превышать двух часов.
4. Во время перерывов целесообразно выполнять комплексы упражнений.

В настоящее время «Инструкция по охране труда при работе на ЭВМ» должна быть разработана на каждом предприятии или организации с учетом последних законодательных и нормативных документов, согласована с профсоюзной организацией и утверждена руководителем.

При интенсивной работе за ПК могут возникнуть головные боли, нервные срывы и прочие неприятные ощущения.

Оценивая условия труда по методике Р 2.2.2006-05, данная деятельность могла быть отнесена к категории 3.1 — вредной и напряженной. Это обусловлено следующими факторами:

1. Интеллектуальная нагрузка включает решение простых задач по известным алгоритмам, оценку изображений и их сопоставление, а также работу с дефицитом времени.
2. Сенсорная нагрузка связана с продолжительным сосредоточенным наблюдением (51-57 %) и работой с медицинскими приборами на протяжении большей части рабочего времени.
3. Эмоциональная нагрузка определяется степенью ответственности за результаты работы, при этом риск для собственной жизни исключен.
4. Режим работы характеризуется фактической продолжительностью рабочего дня в 10-12 час при трехсменной смене, включая работу ночью.

По новой методике приказа Минтруда и социальной защиты № 33н условия труда оцениваются классом 2 (допустимые). Тем не менее даже эти условия труда требуют особого внимания и управления риском для обеспечения здоровья и безопасности медицинских работников.

#### 5.7.1 Рабочее место специалиста

В связи с тем, что врач проводит большую часть рабочего времени в сидячем положении, не менее важной является конфигурация его рабочего места. Оптимальная высота стола составляет в среднем 65-75 см. По данным ГОСТ Р ИСО 9241-5–2009. «Эргономические требования к проведению офисных работ с использованием видеодисплейных терминалов (VDT). Часть 5. Требования к расположению рабочей станции и осанке оператора» и СанПиН 2.4.3648-20 «Гигиенические требования к организации труда операторов персональных электронно-вычислительных машин», при работе с органами управления, такими как мышь и клавиатура, кисти рук должны находиться на одном уровне и не должны быть перекошены вверх, вниз или в стороны. Рекомендуется размещать клавиатуру на поверхности стола на расстоянии от 100 до 300 см от его края [47].

Здесь важно учесть, что эргономически правильное размещение оборудования способствует уменьшению риска развития множества профессиональных заболеваний, таких как синдром карпального канала (атрофия мышц руки) и болезни опорно-двигательного аппарата. Врачи, работающие на ПК, должны быть осведомлены об этих рекомендациях и стремиться к созданию наиболее комфортных условий труда для себя [47].

Также работа с ПК сопровождается такими недостатками как появление прямой блескости, ухудшение контрастности между изображением и фоном, отражение экрана [47].

Наряду с этим, освещение помещения оказывает не маловажное влияние на самочувствие, психофизиологическое состояние и здоровье работника.

Учитывая ограниченность естественного освещения, необходимо обеспечить дополнительное искусственное освещение на рабочем месте. В соответствии с санитарными нормами (СанПиН 1.2.3685-21), уровень искусственного освещения в зоне рабочего места должен составлять 400 лк при общем освещении, а освещенность экрана монитора не должна превышать 200 лк. Яркость светящихся поверхностей в поле зрения пользователя ПК не должна превышать 500 кд/м2 [47].

### 5.8 Выводы к 5 главе

В ходе выполнения данной главы дана общая программного обеспечения для записи работы оператора call-центра и были получены следующие результаты:

1. Классификация разрабатываемого программного обеспечения.
2. Рассмотрены процессы жизненного цикла разработанного ПО.
3. Приведены методы обнаружения нежелательного ПО.
4. Изучены и проанализированы основные положения эргономики.
5. Оценена безопасность ПО.
6. Представлены требования к физическому и графическому интерфейсу.
7. Представлены общие рекомендации при работе с ПК.
8. Рассмотрены требования к рабочему месту оператора.
9. Выявлен класс условий труда.
10. Определен класс безопасности ПО.

## ЗАКЛЮЧЕНИЕ

В результате выполнения работы был разработан инструмент для записи изображений с мониторов и аудио с источников записи и воспроизведения ПК оператора КЦ и передачи их на сервера хранения.

На основе анализа доступных программных средств были выбраны наиболее подходящие по характеристикам для данного проекта и интегрированы в решение. Критерии выбора каждого средства были описаны в данной работе.

Разработанный программный комплекс соответствует всем представленным функциональным и аппаратным требованиям. Созданы docker-контейнеры для развёртывания серверной части и программа установки для клиентской части.

Были произведены анализы производительности серверной части и сделаны выводы, позволяющие продолжить дальнейшую разработку и улучшение программного комплекса в следующих версиях.

Ниже приведены основные направления и идеи для улучшения программного комплекса:

* **Дальнейшая оптимизация сервера.** На текущий момент сервер является однопоточным приложением, значительную часть процессорного времени которого отнимают системные вызовы для работы с сетью, файлами и написания информационных сообщений. По имеющимся данным, системные вызовы занимают около 65% процессорного времени при нагрузочном тестировании. Предлагается создать параллельный поток, который возьмёт на себя системные вызовы.
* **Возможность трансляции записи.** Предлагается добавить возможность просмотра трансляции выбранной ведущейся записи. Данный функционал позволит помочь оператору КЦ в режиме реального времени или предотвратить нежелательную ситуацию, не дожидаясь окончания записи.
* **Возможность «парковать» записи.** В некоторых сценариях сотрудник КЦ может работать поочерёдно одновременно с несколькими вызовами. В данном случае было бы удобно записывать фрагменты каждого отдельного разговора, объединять их и отправлять на сервер уже целую запись

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. "Contact Center vs Communication Center vs Call Center". EWA Bespoke Communications. 2010-03-26.
2. Kramer, Bryan. "How Live-Streaming is Going to Crush it in 2016". SocialMediaToday. Archived from the original on October 2, 2016. Retrieved September 30, 2016.
3. HTTP/1.1, RFC 9112, Abstract, ссылка: https://datatracker.ietf.org/doc/html/rfc9112
4. M. Thomson, Ed. (June 2022) Mozilla, C. Benfield, Ed. Apple Inc. Hypertext Transfer Protocol/2 (HTTP/2). RFC 9113
5. Google Chrome, Веб-ресурс, ссылка: https://www.google.com/chrome/
6. Mozilla Firefox, Веб-ресурс, ссылка: https://www.mozilla.org/en-US/about/
7. Яндекс Браузер, Веб-ресурс, ссылка: https://yandex.ru/company/technologies/browser
8. x86, Веб-ресурс, ссылка: https://www.xda-developers.com/x86/
9. Windows 10, Веб-ресурс, ссылка: https://www.lenovo.com/us/en/glossary/what-is-windows-10/
10. Astra Linux, Веб-ресурс, ссылка: https://wiki.astralinux.ru/
11. Stroustrup, Bjarne (1997). "1". The C++ Programming Language (Third ed.). Addison-Wesley. ISBN 0-201-88954-4. OCLC 59193992.
12. Eddy, Wesley M., ed. (August 2022). Transmission Control Protocol (TCP). doi:10.17487/RFC9293. RFC 9293.
13. MPEG-4, Advanced Video Coding (Part 10) (H.264) (Full draft). Sustainability of Digital Formats. Washington, D.C.: Library of Congress. December 5, 2011. Retrieved December 1, 2021.
14. High Efficiency Video Coding (HEVC) Family, H.265, MPEG-H Part 2 (Preliminary draft). Sustainability of Digital Formats. Washington, D.C.: Library of Congress. November 19, 2020. Retrieved December 1, 2021.
15. VP9 Bitstream & Decoding Process Specification, Version 0.7, Adrian Grange – Google, Peter de Rivaz – Argon Design, Jonathan Hunt – Argon Design, Веб-ресурс, ссылка: https://storage.googleapis.com/downloads.webmproject.org/docs/vp9/vp9-bitstream-specification-v0.7-20170222-draft.pdf
16. Returning Values from Forms: multipart/form-data, L. Masinter – Adobe, (July 2015), RFC 7587, Веб-ресурс, ссылка: https://datatracker.ietf.org/doc/html/rfc7578
17. JSON Log Message Format, Веб-ресурс, ссылка: https://jsnlog.com/Documentation/HowTo/LogMessageFormat
18. Capturing an Image, Веб-ресурс, ссылка: https://learn.microsoft.com/en-us/windows/win32/gdi/capturing-an-image
19. EnumDisplayMonitors function, Веб-ресурс, ссылка: <https://learn.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-enumdisplaymonitors>
20. Требования охраны труда при работе на ПК // prepress-book.narod.ru. URL: http://www.prepress-book.narod.ru/Trebovanija-okhrany-truda/Trebovanija-okhrany-truda-pri-rabote-na-PK.htm (дата обращения: 21.05.2022).
21. Техника безопасности при работе с персональным компьютером // oxrana-bez.ru. URL: https://oxrana-bez.ru/stati/tehnika-bezopasnosti-pri-rabote-s-personalnym-kompyuterom/ (дата обращения: 21.05.2022).
22. ГОСТ 12.0.003-2015 "Опасные и вредные производственные факторы"
23. В. А. Буканин, А. Н. Иванов, В. Н. Павлов, А. О. Трусов. Анализ, оценка и управление процессом снижения технического риска опасности: учеб. пособие. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2023. 112 c
24. ГОСТ Р ИСО/МЭК ТО 12182–2002 «Информационная технология (ИТ). Классификация программных средств».
25. Вопросы безопасности в жизненном цикле технологий программных средств, устройств и систем [Текст] : учеб. пособие / [В. А. Буканин [и др.] ; Санкт-Петербургский государственный электротехнический университет им. В.И. Ульянова (Ленина) "ЛЭТИ" . - СПб. : Изд-во СПбГЭТУ "ЛЭТИ", 2017. - 95, [1] с.]
26. «ГОСТ Р ИСО/МЭК 12207-99 Информационная технология. Процессы жизненного цикла программных средств»
27. Модели жизненного цикла программного обеспечения https://habr.com/ru/articles/111674/
28. 38. Петрова Е. С. Информационная безопасность: особенности современных вирусных угроз [Электронный ресурс]. URL: http://sisupr.mrsu.ru/2012-1/PDF/14\_inf/Sidorova\_Petrova.pdf
29. ГОСТ Р МЭК 62304-2013. Национальный стандарт российской федерации изделия медицинские. М.: Управление качеством медицинских изделий, 2015
30. Ю. А. Романова. Основные риски в процессах разработки программного обеспечения. Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова, 243-246 с.
31. ГОСТ Р 51275-99. Защита информации. Объект информатизации. Факторы, воздействующие на информацию. – М.: Изд-во стандартов, 2003. – 12 с.
32. ГОСТ Р 51188-98 "Испытания программных средств на наличие компьютерных вирусов"
33. Стратегии управления рисками, связанными с вредоносными программами [Электронный ресурс]. URL: http://www.oszone.net/4752
34. ГОСТ Р ИСО 9241-100:2010
35. Smith, J. (2021). Principles of Ergonomics in Software Development. Journal of Human-Computer Interaction, 15(3), 78-91.
36. ГОСТ ИСО/МЭК 9126-2001
37. ребования к графическим интерфейсам: одна памятка ответит на все ваши вопросы / Хабр
38. Марин, А., & Холланд, Б. (2017). Принципы дизайна пользовательского интерфейса. Издательство Харвардского университета.
39. Смит, Д. (2019). Исследование гибкости физического интерфейса в программном обеспечении. Журнал Человек и Компьютер, 10(2), 45-60.
40. Графический интерфейс пользователя — Википедия
41. Джонсон, Э. (2018). Эстетика и гармония в графическом дизайне интерфейса. Журнал Визуальные Коммуникации, 15(3), 112-125.
42. Ли, Г. (2020). Эмоциональная связь и интерактивность в графическом интерфейсе программного обеспечения. Исследовательские заметки, 25(4), 78-91.
43. Документирование требований — IT1410: Разработка требований к программному обеспечению — Бизнес-информатика
44. Техническое условие организации труда на персональных электронно-вычислительных машинах (ТОИ Р-45-084-01). Министерство труда и социальной защиты Российской Федерации, 2018.
45. СанПиН 2.4.3648-20 "Гигиенические требования к организации труда операторов персональных электронно-вычислительных машин" (утвержден Роспотребнадзором 12.07.2020).
46. ГОСТ 12.2.032-78. Рабочее место при выполнении работ сидя. М.: Государственного комитета стандартов Совета Министров СССР, 1978
47. ГОСТ Р ИСО 9241-100. Эргономика взаимодействия человек-система. М.: Приказом Федерального агентства по техническому регулированию и метрологии, 2014