ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии

СОГЛАСОВАНО	*УТВЕРЖДАЮ*
Старший преподаватель департамента программной инженерии факультета компьютерных наук	Академический руководитель образовательной программы «Программная инженерия» профессор департамента программной инженерии, канд. техн. наук
_____ А. В. Поповкин	_____ В. В. Шилов
«_____» _____ 2021 г.	«_____» _____ 2021 г.

ПРИЛОЖЕНИЕ ДЛЯ СОВМЕСТНОГО ПРОСМОТРА ФИЛЬМОВ

Техническое задание

Лист Утверждения

RU.17701729.02.06-01 ТЗ 01-1-ЛУ

Исполнитель: Студент группы БПИ194
_____ В. А. Анненков
«_____» _____ 2021 г.

Москва 2021

Утверждён
RU.17701729.02.06-01 ТЗ 01-1-ЛУ

ПРИЛОЖЕНИЕ ДЛЯ СОВМЕСТНОГО ПРОСМОТРА ФИЛЬМОВ

Техническое задание

RU.17701729.02.06-01 ТЗ 01-1

Листов 21

Инв. № подл	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

Москва 2021

Содержание

АННОТАЦИЯ

Техническое задание – это основной документ, оговаривающий набор требований и порядок создания программного продукта, в соответствии с которым производится разработка программы, её тестирование и приёмка.

Настоящее Техническое задание на разработку серверной части «Приложения для совместного просмотра фильмов» содержит следующие разделы: «Введение», «Основания для разработки», «Назначение разработки», «Требования к программе», «Требования к программным документам», «Технико-экономические показатели», «Стадии и этапы разработки», «Порядок контроля и приёмки» и приложения.

В разделе «Введение» указано наименование и краткая характеристика области применения программы.

В разделе «Основания для разработки» указан документ, на основании которого ведётся разработка и наименование темы разработки.

В разделе «Назначение разработки» указано функциональное и эксплуатационное назначение программного продукта.

Раздел «Требования к программе» содержит основные требования к функциональным характеристикам, к надёжности, к условиям эксплуатации, к составу и параметрам технических средств, к информационной и программной совместимости, к маркировке и упаковке, к транспортировке и хранению, а также специальные требования.

Раздел «Требования к программным документам» содержит предварительный состав программной документации и специальные требования к ней.

Раздел «Технико-экономические показатели» содержит ориентировочную экономическую эффективность, предполагаемую годовую потребность, экономические преимущества разработки программы.

Раздел «Стадии и этапы разработки» содержит стадии разработки, этапы и содержание работ.

В разделе «Порядок контроля и приёмки» указаны общие требования к приёмке работы.

Настоящий документ разработан в соответствии с требованиями:

1) ГОСТ 19.103-77 Обозначения программ и программных документов;
2) ГОСТ 19.104-78 Основные надписи;
3) ГОСТ 19.105-78 Общие требования к программным документам;
4) ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом;
5) ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению.

Изменения к Техническому заданию оформляются согласно ГОСТ 19.603-78, ГОСТ 19.604-78.

Перед прочтением данного документа рекомендуется ознакомиться с терминологией, приведённой в Приложении 1 настоящего Технического задания.

1. Введение

1.1. Наименование программы

1.1.1. Наименование программы на русском языке

Приложение для совместного просмотра фильмов.

1.1.2. Наименование программы на английском языке

Application for Collective Movie Watching.

1.2. Краткая характеристика области применения

В период пандемии друзья и знакомые стараются меньше контактировать друг с другом в реальной жизни и переносят общение в онлайн. Наиболее актуальной становится сфера онлайн-развлечений: фильмы, игры, социальные сервисы.

Разрабатываемое приложение предназначено для онлайн-просмотра фильмов и других видеофайлов с синхронизацией потока между несколькими пользователями. В приложении ожидается акцент на удобное социальное взаимодействие пользователей во время просмотра фильма.

2. Основания для разработки

Основанием для разработки является учебный план подготовки бакалавров по направлению 09.03.04 “Программная инженерия” и утвержденная академическим руководителем тема курсового проекта.

3. Назначение разработки

3.1. Функциональное назначение

Функциональным назначением «приложения для совместного просмотра фильмов» является предоставление пользователям возможности смотреть фильмы на расстоянии. Пользователи могут загружать видео на сервер, воспроизводить видео, общаться с другими пользователями посредством сообщений и «реакций».

Сама программа реализует серверную часть сервиса для совместного просмотра фильмов. Основная задача программы – предоставить приложениям-клиентам всю необходимую информацию для реализации логики совместного просмотра фильмов.

Отдельный клиент может загрузить видео на сервер, после чего любой из посторонних клиентов сможет скачать это видео. Для синхронизации воспроизведения видео клиенты могут использовать WebSocket интерфейс сервера.

Каждое загруженное на сервер видео обрабатывается и разделяется на небольшие фрагменты по 10 секунд. Такой подход позволяет клиентам быстрее скачивать видео небольшими порциями, а также мгновенно переключаться между различными разрешениями в процессе просмотра.

Видео преобразовывается в формат HLS (HTTP Live Streaming). Особенность этого формата в том, что он поддерживается любыми операционными системами: Windows, macOS, Android, iOS и т. д. Несмотря на то, что обработка видео в формат HLS требует длительного времени, с текущим форматом это не является проблемой. Видео, которые преобразованы в HLS лишь частично, могут быть воспроизведены клиентами до момента последнего обработанного фрагмента. Таким образом клиенты могут воспроизводить фильмы мгновенно после загрузки на сервер, не дожидаясь полной обработки. Обработка идёт параллельно.

3.2. Эксплуатационное назначение

Сервер может быть использован frontend-клиентами для реализации логики совместного просмотра фильмов.

«Приложение для совместного просмотра фильмов» может быть использовано пользователями для совместного просмотра фильмов или других видео на расстоянии. Для этого пользователю требуется только открыть сайт и загрузить видео. После этого можно сразу же начать просмотр фильма.

В приложении реализован широкий функционал для общения и взаимодействия пользователей друг с другом. Можно писать сообщения или отправлять реакции – всё для того, чтобы добиться максимального погружения в процесс совместного просмотра фильма.

Данное приложение становится особенно актуальным, когда требуется что-то посмотреть наиболее простым способом. Для этого в приложении реализован минималистичный интерфейс, возможность быстро загрузить собственный файл с фильмом и сразу же начать его просмотр. Приложение не ограничивает пользователя собственной базой фильмов, предоставляя пользователю возможность самостоятельно загружать файлы с фильмами.

4. Требования к программе

4.1. Требования к функциональным характеристикам

Программа состоит из двух основных компонент: клиентской и серверной частей, между которыми должно быть налажено взаимодействие. В данном документе описаны требования к серверной части.

4.1.1. Требования к серверной части

На серверной части должен быть реализован алгоритм по преобразованию единого видеоролика в набор видеороликов меньшей длительности (сегментов). Также требуется добавить конвертацию видеоролика в видеоролики с меньшим качеством (Рис. 1).

Алгоритм конвертации видеоролика следующий:

1) Клиент загружает видеофайл на сервер.

2) Сервер конвертирует видеофайл в видеофайлы меньших разрешений.

$$video1080 \Rightarrow videos = \{video1080, video720, video480, video360, video240, video144\}$$

Данное действие требуется сделать для обеспечения возможности менять качество воспроизводимых роликов в плеере клиента, а также для сжатия самих файлов с целью ускорения кэширования видеороликов клиентом.

3) Сервер разделяет каждый видеоролик из множества видеороликов *videos* на небольшие фрагменты (1 сек.).

$$video = part1 + part2 + \cdots, \quad video \in videos$$

Данное действие требуется сделать для обеспечения возможности измененять качество и настройки видео в процессе его воспроизведения (без повторного кэширования и приостановки воспроизведения).

Требуется реализовать API для обеспечения взаимодействия сервера с клиентами. Реализованные методы API и протоколы взаимодействия должны быть задокументированы.

Должно быть реализовано взаимодействие с базой данных для хранения данных о комнатах.

4.1.2. Требования к взаимодействию клиентской и серверной частей

Взаимодействие между клиентом и сервером должно осуществляться посредством HTTP-запросов и WebSocket-подключений.

При получении HTTP-запроса (GET, POST, UPDATE, DELETE и т.д.) от клиента, сервер должен ответить сообщением в формате JSON, содержащим необходимую информацию для работы клиента.

Для синхронизации видеопотока между разными клиентами используется протокол WebSocket. В целях обеспечения наименьшей рассинхронизации видеопотока между клиентами, требуется хранить переменные $diff\_sc$ и $diff\_cs$ для каждого клиента. Данные переменные будут содержать в себе информацию о количестве затрачиваемого времени при передаче данных от сервера к клиенту или от клиента к серверу.

Формула для определения переменных $diff\_sc$ и $diff\_cs$:

$$diff\_sc = time\_sc_2 - time\_sc_1$$
$$diff\_cs = time\_cs_2 - time\_cs_1$$

, где:

- $time\_sc_1$ — время отправки сообщения сервером;
- $time\_sc_2$ — время получения сообщения клиентом;
- $diff\_sc$ — разница между временем отправки и временем получения при передаче сообщения от сервера к клиенту;
- $time\_cs_1$ — время отправки сообщения клиентом;
- $time\_cs_2$ — время получения сообщения сервером;
- $diff\_cs$ — разница между временем отправки и временем получения при передаче сообщения от клиента к серверу.

Так как возможно подключение нескольких клиентов, требуется хранить содержимое значений $diff\_sc_i$ и $diff\_cs_i$ для всех n клиентов.

Алгоритм (Рис. 2) синхронизации видео единый:

1) Клиент k отправляет серверу запрос на действие d (перемотку/приостановку/возобновление) видео.

2) Сервер отправляет всем клиентам сообщение с текущим серверным временем. Клиенты принимают значение и считают разницу $diff\_sc_i$ с учётом своего времени. Затем клиенты отправляют посчитанную разницу времени в миллисекундах обратно серверу, а также отправляют текущее клиентское время. С учётом полученной информации сервер считает разницу $diff\_cs_i$.

3) Сервер отправляет всем клиентам команду выполнить действие d и передаёт каждому клиенту задержку $delay_i$, которая считается по формуле:

$$delay_i = \max(diff\_sc_1, \ldots, diff\_sc_n) - diff\_sc_i + diff\_cs_k, \quad i \in [1, \ldots, n]$$

Значение $delay_i$ требуется по-разному использовать в различных ситуациях. При организации совместного просмотра фильма возможны следующие сценарии:

- **Перемотка** видео на позицию t мс одним из клиентов.

  При получении такой команды все клиенты (кроме клиента-инициатора) выполняют перемотку видео на позицию $(t + delay_i)$ мс.

– **Приостановка** видео одним из клиентов.

  При получении такой команды все клиенты (кроме клиента-инициатора) приостанавливают воспроизведение и выполняют перемотку видео на $delay_i$ мс назад.

– **Возобновление** видео одним из клиентов.

  При получении такой команды все клиенты (кроме клиента-инициатора) возобновляют воспроизведение и выполняют перемотку видео на $delay_i$ мс вперёд.

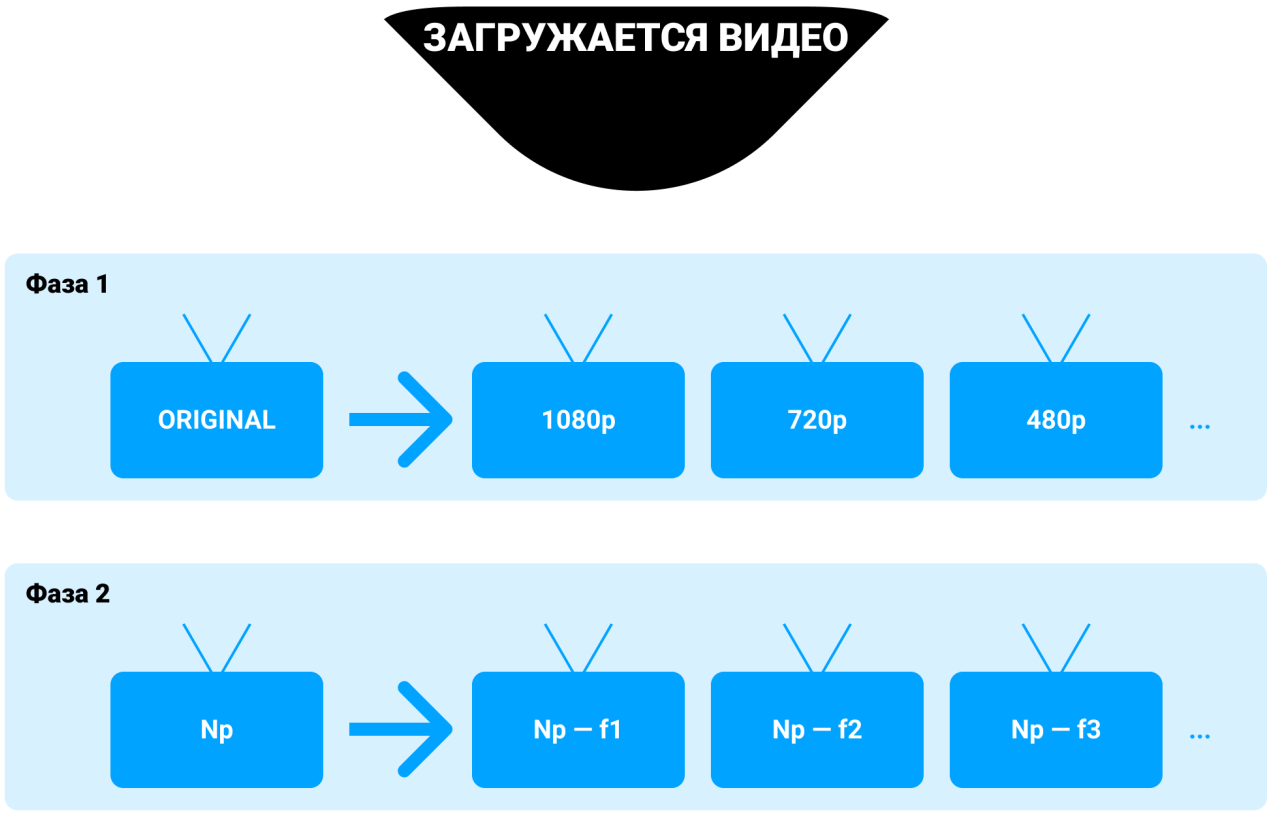

Рис. 1 — Алгоритм конвертации видео

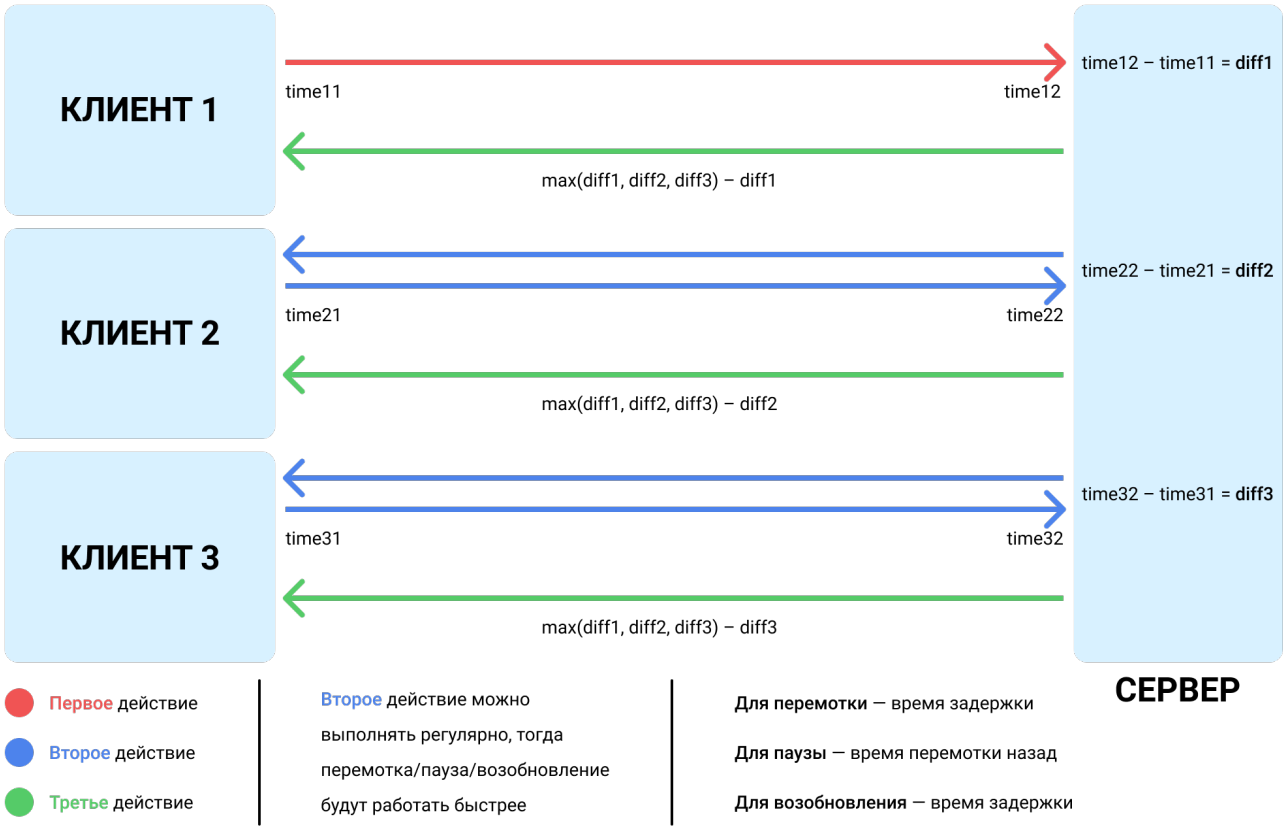

Рис. 2 — Алгоритм синхронизации видеопотока

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.06-01 ТЗ 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

4.1.3. Требования к организации входных данных

Входными данными для серверной части приложения будут являться:

1) HTTP-запросы от клиентской части приложения, которые могут содержать в себе тело в формате JSON;
2) запросы от клиентов с помощью протоколов на базе UDP.

4.1.4. Требования к организации выходных данных

Выходными данными для приложения будут являться:

1) HTTP-ответы на запросы клиентской части, которые будут содержать следующую информацию:
   - код состояния,
   - тело ответа в формате JSON,
   - различные заголовки;
2) ответы для клиентов с помощью протоколов на базе UDP.

4.2. Требования к надёжности

4.2.1. Требования к обеспечению надёжного (устойчивого) функционирования программы

Разрабатываемый сервер должен:

1) не завершаться аварийно при возникающих ошибках;
2) быть устойчивым к атакам следующего типа:
   - Cross-Site Scripting (XSS),
   - SQL Injection,
   - Local File Inclusion (LFI),
   - Distributed Denial of Service (DDoS).

4.2.2. Время восстановления после отказа

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, не должно превышать времени, необходимого на перезагрузку операционной системы и запуск программы, при условии соблюдения условий эксплуатации технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на переустановку программных средств.

4.2.3. Отказы из-за некорректных действий оператора

Во избежание возникновения отказов программы по причине некорректных действий оператора следует обеспечить работу конечного пользователя без предоставления ему административных привилегий.

4.3. Условия эксплуатации

Для корректной эксплуатации системы необходим системный администратор, знакомый с языком программирования Java, а также СУБД PostgreSQL.

Прочие требования эксплуатации приложения определяются требованиями к условиям эксплуатации технических средств.

4.4. Требования к составу и параметрам технических средств

В состав технических средств сервера должен входить компьютер или система компьютеров. Допускается использование облачных сервисов.

Рекомендуется следующие характеристики:

1) Объём оперативной памяти 8 Гб;
2) Объём физической памяти не менее 50 Гб.

4.5. Требования к информационной и программной совместимости

4.5.1. Требования к исходным кодам и языкам программирования

Исходные коды для сервера должны быть реализованы с использованием следующих языков и технологий:

1) Java;
2) Spring Boot — для реализации базовой архитектуры сервера;
3) Spring Web — для реализации REST API методов и панели администрирования;
4) Spring Thymeleaf — для миграций и версионирования базы данных;
5) Hibernate — для работы с базой данных, для связки таблиц базы данных с классами Java;
6) ffmpeg — для обработки видеофайлов: нарезка и изменение качества;
7) PostgreSQL — использовать в качестве СУБД;
8) Docker, docker-compose — для обеспечения переносимости сервера.

4.5.2. Требования к программным средствам, используемым программой

Требования к информационным и программным характеристикам сервера:

1) Наличие доступа по SSH;
2) Открытые порты: 22, 40, 80, 443, 3000, 5432, 5454, 8080-8100;
3) Установленные утилиты: nano, apt-get;
4) Установленные: docker, docker-compose, docker-machine;
5) Установленный с настройками по-умолчаию: Nginx, пользователь с доступом по SSH должен иметь права на редактирование конфигурации;
6) Операционная система Ubuntu 16.04.

4.6. Требования к маркировке и упаковке

Не предъявляются.

5. Требования к программной документации

5.1. Состав программной документации

В состав программной документации должны входить следующие компоненты:

1) Техническое задание (ГОСТ 19.201-78)
2) Программа и методика испытаний (ГОСТ 19.301-78)
3) Пояснительная записка (ГОСТ 19.404-79)
4) Руководство оператора (ГОСТ 19.505-79)
5) Текст программы (ГОСТ 19.401-78)

5.2. Специальные требования к программной документации

Документы к программе должны быть выполнены в соответствии с ГОСТ 19.106-78 и ГОСТами к каждому виду документа (см. п. 5.1.);

Пояснительная записка должна быть загружена в систему Антиплагиат через LMS «НИУ ВШЭ».

Документация и программа сдаются в электронном виде в формате .pdf или .docx. в архиве формата .zip или .rar;

За три дня до защиты комиссии все материалы курсового проекта:

– техническая документация,

– программный проект,

– исполняемый файл,

– отзыв руководителя,

– лист Антиплагиата

должны быть загружены одним или несколькими архивами в проект дисциплины «Курсовой проект, 2 курс ПИ» в личном кабинете в информационной образовательной среде LMS (Learning Management System) НИУ ВШЭ.

6. Технико-экономические показатели

6.1. Предполагаемая потребность

Приложение могут использовать пользователи для совместного просмотра фильмов на расстоянии. Для этого пользователю требуется только открыть сайт и загрузить видео. После этого можно сразу же начать просмотр фильма.

В приложении реализован широкий функционал для общения и взаимодействия пользователей друг с другом. Можно писать сообщения или отправлять реакции – всё для того, чтобы добиться максимального погружения в процесс совместного просмотра фильма.

6.2. Экономические преимущества разработки по сравнению с отечественными и зарубежными аналогами

Большинство существующих сервисов поддерживают воспроизведение видео из существующей базы, не предоставляя пользователям возможность смотреть любые желаемые видео. Кроме того, сервисы, которые всё-таки имеют возможность загрузки видео, не позволяют просматривать видео в процессе его обработки – для просмотра требуется дождаться полной обработки, а этот процесс может быть длительным.

Ещё одно преимущество перед аналогами – наличие «реакций», которые реализованы в виде смайликов. С их помощью можно выразить какую-то эмоцию, не потратив время на набор текстового сообщения.

7. Стадии и этапы разработки

7.1. Стадии и этапы разработки

1) техническое задание:
   - этапы разработки:
     а) обоснование необходимости разработки программы;
     б) постановка задачи;
     в) сбор исходных материалов;
     г) выбор и обоснование критериев эффективности и качества разрабатываемой программы;
     д) обоснование необходимости проведения научно-исследовательских работ;
   - разработка и утверждение технического задания:
     а) определение требований к программе;
     б) определение стадий, этапов и сроков разработки программы и документации на неё;
     в) согласование и утверждение технического задания;
2) технический проект:
   - разработка технического проекта:
     а) уточнение структуры входных и выходных данных;
     б) разработка алгоритма решения задачи;
     в) определение формы представления входных и выходных данных;
     г) разработка структуры программы;
     д) окончательное определение конфигурации технических средств.
   - утверждение технического проекта:
     а) разработка пояснительной записки;
     б) согласование и утверждение технического проекта.
3) рабочий проект:
   - разработка программы:
     а) программирование и отладка программы.
   - разработка программной документации:
     а) разработка программных документов в соответствии с требованиями гост 19.101-77.
   - испытания программы:
     а) разработка, согласование и утверждение порядка и методики испытаний;
     б) корректировка программы и программной документации по результатам испытаний.
4) подготовка и передача программы:

– утверждение даты защиты программного продукта;
– подготовка программы и программной документации для презентации и защиты;
– представление разработанного программного продукта руководителю и получение отзыва;
– загрузка Пояснительной записки в систему Антиплагиат через ЛМС НИУ ВШЭ;
– загрузка материалов курсового проекта (курсовой работы) в ЛМС, проект дисциплины «Курсовая работа 2019» (п. 5.2);
– защита программного продукта (курсового проекта) комиссии.

7.2. Сроки разработки и исполнители

Разработка должна закончиться в мае 2021 года.

Исполнители:

– Панфилов Егор Павлович, студент группы БПИ194 факультета компьютерных наук НИУ ВШЭ.

  Разработчик WEB-клиента.

– Анненков Владислав Алексеевич, студент группы БПИ194 факультета компьютерных наук НИУ ВШЭ.

  Разработчик сервера.

8. Порядок контроля и приёмки

Проверка программного продукта, в том числе и на соответствие техническому заданию, осуществляется исполнителем вместе с заказчиком согласно «Программе и методике испытаний», а также пункту 5.2.

Защита выполненного проекта осуществляется комиссии, состоящей из преподавателей департамента программной инженерии, в утверждённые приказом декана ФКН сроки.

9. Источники, использованные при разработке

1) ГОСТ 19.101-77 Виды программ и программных документов. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

2) ГОСТ 19.102-77 Стадии разработки. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

3) ГОСТ 19.103-77 Обозначения программ и программных документов. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001

4) ГОСТ 19.104-78 Основные надписи. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

5) ГОСТ 19.105-78 Общие требования к программным документам. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

6) ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001

7) ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

8) Загрузка файлов [Электронный ресурс] / Spring-Projects. Режим доступа: https://spring-projects.ru/guides/uploading-files/, свободный. (Дата обращения: 15.05.2020)

9) Как установить и настроить PostgreSQL в MacOS [Электронный ресурс] / 900913. Режим доступа: `https://900913.ru/note/b/postgresql-macos-9da176/`, свободный. (Дата обращения: 15.05.2020)

10) Курс по Spring / ВТБ // [Электронный ресурс]: Google Drive. Режим доступа: `https://drive.google.com/drive/folders/1PQspMs1gm8aIFNua9l-wDsjgPxidqfhC`, свободный. (Дата обращения: 15.05.2021)

11) Тянем ролик с Youtube и раздаем по WebRTC в реалтайме [Электронный ресурс] / Flashphoner. Режим доступа: https://flashphoner.com/tyanem-rolik-s-youtube-i-razdaem-po-webrtc-v-realtaj me/?lang=ru, свободный. (Дата обращения: 15.05.2020)

12) Adaptive HTTP Streaming Technologies: HLS vs. DASH / Tech Blog // [Электронный ресурс]: StriveCast. Режим доступа: `https://strivecast.com/hls-vs-mpeg-dash/`, свободный. (Дата обращения: 15.05.2021)

13) Create Multiple tables [Электронный ресурс] / LaunchSchool. Режим доступа: `https://launchschool.com/books/sql_first_edition/read/multi_tables`, свободный. (Дата обращения: 15.05.2020)

14) Docker документация [Электронный ресурс] / Docker. Режим доступа: `https://docs.docker.com/`, свободный. (Дата обращения: 15.05.2020)

15) ffmpeg документация [Электронный ресурс] / ffmpeg. Режим доступа: `https://ffmpeg.org/ffmpeg.html`, свободный. (Дата обращения: 15.05.2020)

16) GitHub [Электронный ресурс]. Режим доступа: https://github.com/, свободный. (Дата обращения: 15.05.2020)

17) How do I use ffmpeg to get the video resolution? / Vladimir Stazhilov // [Электронный ресурс]: SuperUser. Режим доступа: https://superuser.com/questions/841235/how-do-i-use-ffmpeg-to-get-the-video-resolution, свободный. (Дата обращения: 15.05.2021)

18) How to create .mpd or .m3u8 video file on the server using FFMPEG for Adaptive Streaming / Mayur Solanki // [Электронный ресурс]: Medium. Режим доступа: https://mayur-solanki.medium.com/how-to-create-mpd-or-m3u8-video-file-from-server-using-ffmpeg-97e9e1fbf6a3, свободный. (Дата обращения: 15.05.2021)

19) How to read ffmpeg response from java and use it to create a progress bar? / shalki // [Электронный ресурс]: StackOverflow. Режим доступа: https://stackoverflow.com/questions/10927718/how-to-read-ffmpeg-response-from-java-and-use-it-to-create-a-progress-bar, свободный. (Дата обращения: 15.05.2021)

20) How video streaming works on the web: An introduction / Paul Berberian // [Электронный ресурс]: Medium. Режим доступа: https://medium.com/canal-tech/how-video-streaming-works-on-the-web-an-introduction-7919739f7e1, свободный. (Дата обращения: 15.05.2021)

21) Kotlin документация [Электронный ресурс] / JetBrains. Режим доступа: https://kotlinlang.org/docs/, свободный. (Дата обращения: 15.05.2020)

22) PostgreSQL документация [Электронный ресурс] / PostgreSQL. Режим доступа: https://www.postgresql.org/docs/, свободный. (Дата обращения: 15.05.2020)

23) Spring документация [Электронный ресурс] / Spring. Режим доступа: https://spring.io/, свободный. (Дата обращения: 15.05.2020)

24) Spring Cloud Demo Project / gonwan // [Электронный ресурс]: GitHub. Режим доступа: https://github.com/gonwan/spring-cloud-demo, свободный. (Дата обращения: 15.05.2021)

25) Spring Cloud Netflix Microservices – start project (серия статей) – часть 4 / Kirill Sereda // [Электронный ресурс]: Medium. Режим доступа: https://medium.com/@kirill.sereda/spring-cloud-netflix-microservices-start-project-%D1%81%D0%B5%D1%80%D0%B8%D1%8F-%D1%81%D1%82%D0%B0%D1%82%D0%B5%D0%B9-%D1%87%D0%B0%D1%81%D1%82%D1%8C-4-d2137d19d783, свободный. (Дата обращения: 15.05.2021)

26) StackOverflow [Электронный ресурс]. Режим доступа: https://stackoverflow.com/, свободный. (Дата обращения: 15.05.2020)

27) streaming an mkv file while processing with ffmpeg / Phani Rithvij // [Электронный ресурс]: StackOverflow. Режим доступа: https://stackoverflow.com/questions/55460359/streaming-an-mkv-file-while-processing-with-ffmpeg, свободный. (Дата обращения: 15.05.2021)

ПРИЛОЖЕНИЕ 1

Терминология

HLS (HTTP Live Streaming) – протокол для потоковой передачи медиа, разработанный компанией Apple.

HTTP – протокол передачи данных, работающий в режиме «запрос-ответ».

IoC (Inversion of Control) – архитектурный принцип объектно-ориентированного программирования, используемый для уменьшения зацепления (связанности) в компьютерных программах.

WebSocket – протокол поверх TCP-соединения, позволяющий взаимодействовать клиенту и серверу в режиме реального времени.

База данных – совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.

Клиент – приложение, которое подключается к серверу.

Комната – виртуальное пространство, в котором воспроизводится видео и к которому могут подключаться пользователи.

Реакция – всплывающий смайлик для быстрой передачи эмоций во время просмотра.

Терминал – разновидность текстового интерфейса между человеком и компьютером, в котором инструкции компьютеру даются в основном путём ввода с клавиатуры текстовых строк (команд).

Сегмент – небольшой отрывок из исходного видео.

Сервер – выделенный или специализированный компьютер для выполнения сервисного программного обеспечения.

Серверное приложение – программа, запущенная на сервере.

СУБД – система управления базами данных.

Фреймворк – компонент к программе, который включает в себя дополнительный реализованный функционал, упрощающий разработку.

Лист регистрации изменений

Изм.	Номера листов				Всего листов в документе	№ документа	Входящий № сопровод. докум. и дата	Подпись	Дата
	измененных	замененных	новых	аннулированных					