

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
**ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ  
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)**

Кафедра компьютерных систем в управлении и проектировании (КСУП)

**СОЗДАНИЕ ВЕБ-ПРИЛОЖЕНИЯ ВИДЕОХОСТИНГА**

Курсовая работа по дисциплине «Разработка приложений»

Студент гр. 571-2

\_\_\_\_\_ А.Я. Гринев

«\_\_» \_\_\_\_ 2024 г.

Старший преподаватель

КСУП

\_\_\_\_\_ Е.С. Мурзин

«\_\_» \_\_\_\_ 2024 г.

Оценка

Томск 2024

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ  
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании (КСУП)  
УТВЕРЖДАЮ

Заведующий кафедрой КСУП, д-р  
техн. наук, профессор

\_\_\_\_\_ Ю.А. Шурыгин

«\_\_\_» \_\_\_\_\_ 20\_\_ г.

**ЗАДАНИЕ**

на курсовую работу по дисциплине «Разработка приложений» студенту  
Гриневу Алексею Ярославовичу группы 571-2 факультета вычислительных  
систем.

1 Тема работы: Создание веб-приложения видеоХостинга с  
учебно-исследовательской базой данных.

(утверждена приказом по вузу от \_\_\_\_\_ № \_\_\_\_\_ )

2 Срок сдачи студентом законченной работы: до «\_\_\_» \_\_\_\_\_ 20\_\_ г.

3 Назначение и область применения:

Разрабатываемая система предназначена для предоставления  
пользователям возможности находить и просматривать видеоконтент.

4 Требование к работе:

4.1 Проектирование предметной области

4.1.1 Описание предметной области

4.1.2 Обзор аналогов

4.2 Проектирование системы

4.2.1 Проектирование базы данных

4.2.2 Проектирование структуры программы

**4.3 Составление руководства пользователя**

**4.3.1 Указание функционала**

**4.3.2 Составление руководства по использованию**

**4.4 Проведение тестирования и демонстрации работы**

**4.4.1 Составление плана тестирования**

**4.4.2 Проведение тестирования**

**4.4.3 Показ демонстрации работы**

**5 Содержание пояснительной записки:**

**5.1 Титульный лист**

**5.2 реферат на русском языке;**

**5.3 задание;**

**5.4 содержание;**

**5.5 введение;**

**5.6 Описание предметной области;**

**5.7 обоснование выбора программных средств;**

**5.8 описание прикладной программы;**

**5.9 руководство пользователю;**

**5.10 заключение;**

**5.11 список использованных источников;**

**5.12 приложения (листинг программы).**

**6 Дата выдачи задания:**

**«\_\_» \_\_\_\_\_ 20\_\_ г.**

**Задание согласовано:**

**Руководитель работы**

**Е.С. Мурзин**

**«\_\_» \_\_\_\_\_ 20\_\_ г.**

\_\_\_\_\_

**Задание принято к исполнению**

**А.Я. Гринев**

**«\_\_» \_\_\_\_\_ 20\_\_ г.**

\_\_\_\_\_

## **РЕФЕРАТ**

Курсовая работа, 30 страниц, 13 рисунков, 3 источника, 9 приложений.

**ВЕБ-ПРИЛОЖЕНИЕ, РЕЛЯЦИОННЫЕ МОДЕЛИ, БАЗА ДАННЫХ,  
ВИДЕОХОСТИНГ, ВИДЕО, ОНЛАЙН, ВЕБ.**

Объектом исследования является веб-приложение и база данных в области видеохостинга.

Цель работы: создать веб-приложение видеохостинг.

В процессе работы была проанализирована предметная область «видеохостинга», определены требования к базе данных и программному комплексу. Помимо этого было разработано веб-приложение для удобной и эффективного чтения данных с базы данных и просмотра видео материалов.

Для разработки программного комплекса были использованы инструменты среды разработки Visual Studio Code и pgAdmin 4.

В итоге были созданы база данных и веб-приложение, предназначенный просмотра видео материалов.

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ	8
1 КОНЦЕПТУАЛЬНОЕ (ИНФОЛОГИЧЕСКОЕ) ПРОЕКТИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ (ПО)	9
1.1 Неформальное описание ПО с использованием естественного языка	9
1.2 Описание бизнес-процессов ПО в методологии функционального моделирования IDEF0	9
1.3 Цели автоматизации ПО	11
1.3.1. Бизнес-процессы, подлежащие автоматизации.	12
1.3.2. Описание бизнес-процессов ПО в IDEF0 после внедрения автоматизированной информационной системы.	12
1.4 Концептуальная информационная модель данных для ПО.	13
1.4.1. Основные объекты ПО, информация о которых будет накапливаться в БД. Их характеристики и свойства (атрибуты).	13
1.4.2. Определение связей между объектами ПО.	14
1.4.3. Графическое представление концептуальной информационной модели данных.	14
2 ПРОЕКТИРОВАНИЕ РЕЛЯЦИОННОЙ МОДЕЛИ БАЗЫ ДАННЫХ	15
2.1 Логическое (даталогическое) проектирование модели данных	15
2.1.1. Определение отношений и связей между отношениями на основе концептуальной информационной модели. Первичные и внешние ключи.	15
2.1.2. Нормализация логической модели данных.	15
2.1.3. Графическое представление логической модели данных в методологии IDEF1x.	16
2.2 Физическое проектирование с учетом выбранной СУБД	17
2.2.1. Определение типов данных атрибутов отношений.	17

2.2.2. Графическое представление физической модели данных в методологии IDEF1x.	18
<b>3 ЗАЩИТА ИНФОРМАЦИИ В БАЗЕ ДАННЫХ</b>	<b>19</b>
3.1 Определение уязвимостей	19
3.2 Создание ролей и пользователей	20
3.3 Разграничение прав доступа	20
3.4 Применение других средств для устранения уязвимостей	22
<b>4 ПРОГРАММНЫЙ КОМПЛЕКС ДЛЯ РАБОТЫ С СУБД</b>	<b>27</b>
<b>ЗАКЛЮЧЕНИЕ</b>	<b>38</b>
<b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ</b>	<b>39</b>
<b>ПРИЛОЖЕНИЕ А (обязательное) Программный код формы с главной страницей программы</b>	<b>40</b>
<b>ПРИЛОЖЕНИЕ Б (обязательное) Программный код формы для редактирования данных о проектах</b>	<b>42</b>
<b>ПРИЛОЖЕНИЕ В (обязательное) Программный код формы для редактирования данных программистов</b>	<b>45</b>
<b>ПРИЛОЖЕНИЕ Г (обязательное) Программный код формы для редактирования данных о заказчиках</b>	<b>47</b>
<b>ПРИЛОЖЕНИЕ Д (обязательное) Программный код формы для редактирования данных о заказчиках</b>	<b>50</b>
<b>ПРИЛОЖЕНИЕ Е (обязательное) Программный код формы для редактирования данных о командах проектов</b>	<b>52</b>

## **ВВЕДЕНИЕ**

Системы видеохостинга представляют собой программное обеспечение, позволяющее пользователям просматривать видеоконтент и получать информацию о нём. Такие платформы являются важной частью цифровой экосистемы, предоставляя удобный доступ к разнообразным материалам для обучения, развлечения и информирования.

Целью данной курсовой работы является разработка базовой платформы видеохостинга, которая обеспечивает пользователям возможность выбора и просмотра видеоматериалов из общего списка с отображением соответствующих данных о каждом видео.

В рамках работы будут реализованы ключевые функции системы, включая отображение общего списка видеоматериалов, предоставление информации о каждом видео и создание удобного интерфейса для просмотра контента.

Результатом работы станет прототип видеохостинга, демонстрирующий основные возможности системы для просмотра видеоматериалов, который может служить основой для последующего расширения функционала.

# **1 ПРОЕКТИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ**

## **1.1 Описание предметной области и постановка задачи**

Предметная область систем видеохостинга охватывает широкий спектр действий, связанных с созданием, распространением и потреблением видеоконтента. Платформы, подобные YouTube, стали не только средством развлечения, но и мощным инструментом для образовательных, маркетинговых и социальных проектов. Основные требования к системам видеохостинга включают высокую доступность, масштабируемость и возможность поддерживать большое количество видеоматериалов.

Задачи, которые ставятся перед разработкой данной системы, включают разработку функционала для просмотра видеоматериалов и легкой навигации для поиска видео.

## **1.2 Обзор аналогов**

В качестве основных аналогов системы можно рассмотреть существующие популярные платформы видеохостинга, такие как YouTube, Вк Видео, Rutube Vimeo, Dailymotion и TikTok. Каждый из этих сервисов имеет свои особенности:

YouTube – крупнейший в мире видеохостинг, предоставляющая пользователям возможность загружать, просматривать и комментировать видео. Отличается широкой аудиторией, продвинутыми алгоритмами рекомендаций и поддержкой видео высокого качества (до 8К).

**Схожесть:** Платформа является прямым вдохновением, предлагая функции загрузки, просмотра, комментирования и взаимодействия с видео.

**Отличие:** Более гибкая политика модерации и новые алгоритмы рекомендаций, которые ориентированы на индивидуальные интересы пользователей, а не только на тренды и популярный контент.

Vimeo – платформа, ориентированная на профессионалов, позволяющая пользователям загружать и транслировать видео высокого качества, с акцентом на бизнес-решения и видеомаркетинг.

**Схожесть:** Поддержка качественного видеоконтента.

**Отличие:** Проект не фокусируется на бизнес-решениях и профессиональном видео маркетинге, а ориентирован на массовую аудиторию и развлекательный контент.

Newgrounds – медиаплатформа платформа ориентированная на артистов, позволяющая пользователям свободно самовыражаться через анимированный или нарисованный контент, с малыми ограничениями на производимый контент.

**Похожесть:** Возможность творческого самовыражения.

Отличие: Проект направлен на видеоформат, а не на контент в любом формате.

Dailymotion – аналогичная YouTube платформа, с акцентом на предоставление видеоконтента в области новостей и развлекательных шоу.

Схожесть: Общие функции видеохостинга, включая просмотр и размещение контента.

Отличие: Проект имеет более широкую целевую аудиторию и не ограничивается видео новостей и шоу.

TikTok – платформа для краткосрочных видеороликов, ориентированная на мобильные устройства, с возможностью редактирования и обмена видео.

Схожесть: Возможность индивидуализированных рекомендаций.

Отличие: Проект не фокусируется на коротких видеороликах и мобильной аудитории, предлагая универсальную платформу для контента разной длительности.

Coub – платформа для краткосрочных зацикленный видеороликов с дополнительным звуковым не зацикленным звуковым сопровождением.

Схожесть: Поддержка пользовательского творчества.

Отличие: Проект не ограничивается зацикленными роликами, предоставляя более широкий формат контента.

RuTube – Отечественный аналог YouTube, с акцентом на предоставление видеоконтента в области телевизионных передач, но не ограниченный только данной областью.

Схожесть: Отечественная альтернатива YouTube с возможностью размещения пользовательских видео.

Отличие: Проект ориентирован на международную аудиторию и алгоритмы рекомендаций.

Вк Видео – Платформа на базе социальной сети Вконтакте, позволяющая пользователям загрузки контента любого характера, его организации и распространения.

Схожесть: Возможность загружать и распространять видео.

Отличие: Проект является отдельной платформой, не интегрированной с социальной сетью.

Dzen – это рекомендательная медиаплатформа, где пользователи создают и потребляют разнообразный контент, включая статьи, видео и блоги, чаще всего развлекательного, образовательного или лайфстайл-характера, с подборкой под персональные интересы аудитории.

Схожесть: Персонализированные рекомендации.

Отличие: Проект полностью сосредоточен на видеоконтенте, исключая статьи и блоги.

Видео Mail.ru – это российская платформа для хранения, просмотра и обмена видео, предлагающая разнообразный контент, включая фильмы, сериалы, шоу и пользовательские ролики, с акцентом на удобство интеграции с другими сервисами экосистемы ВКонтакте.

Схожесть: Поддержка разнообразного видеоконтента.

Отличие: Проект не зависит от экосистемы сторонних сервисов.

Sibnet Видео – это видео хостинговая платформа, популярная в Сибири и России, предоставляющая возможность загружать, хранить и просматривать видео различных жанров, включая фильмы, сериалы и пользовательский контент, с акцентом на свободный доступ и удобный интерфейс.

Схожесть: Поддержка пользовательских роликов и свободный доступ.

Отличие: Проект ориентирован на глобальную аудиторию, а не только на региональный рынок.

Разработка новой системы предполагает создание новой уникальной среды, предоставляющей гибкость настроек модерации, универсальный формат видеоконтента и персонализированные алгоритмы рекомендаций, которые обеспечат лучший пользовательский опыт.

### **1.3 Выбор средств реализации проекта**

Для реализации веб-сайта видеохостинга доступного из веб-браузеров таких как Chromium, Firefox, Safari. Необходимо выбрать подходящие инструменты, обеспечивающие высокую производительность, масштабируемость и безопасность системы. В качестве базовой инфраструктуры были рассмотрены следующие решения:

Frontend:

HTML5, CSS3, JavaScript – для создания динамического и удобного пользовательского интерфейса с поддержкой всех современных браузеров и мобильных устройств.

Backend:

Node.js – для реализации серверной логики.

База данных:

PostgreSQL – реляционная СУБД для хранения данных пользователей, информации о видео и статистики.

Данный набор технологий позволит создать надежное, масштабируемое и удобное веб-приложение для видеохостинга, удовлетворяющее потребности конечных пользователей и авторов контента.

## **1.4 Проект системы**

В данном разделе представлена модель системы «ВидеоХостинга», в которую входит:

- структура программы;
- функциональные диаграммы системы.

Модель проекта была разработана на основе методологий IDEF и UML, с целью наглядного представления функциональных и структурных аспектов приложения.

### **Логическая модель данных**

Логическая модель данных представлена диаграммой классов, включающей ключевые сущности:

Пользователь: основные атрибуты - идентификатор, имя, email, пароль, аватар, подписки, описание, дата создания, статус, псевдоним, рейтинг.

Комментарий: основные атрибуты - идентификатор, автор, текст, дата создания, положительные и негативные оценки, приложение.

Видео: основные атрибуты - идентификатор, название, автор, описание, просмотры, описание, дата загрузки, время года, ссылка на проигрыватель, ссылка на страницу. А так же содержит: директора, положительные и негативные оценки.

Тег: основные атрибуты - идентификатор, имя, тип.

Видео\_тег: основные атрибуты — идентификатор видео, идентификатор тег.

Видео\_комментарий: основные атрибуты - идентификатор видео, идентификатор комментарий.

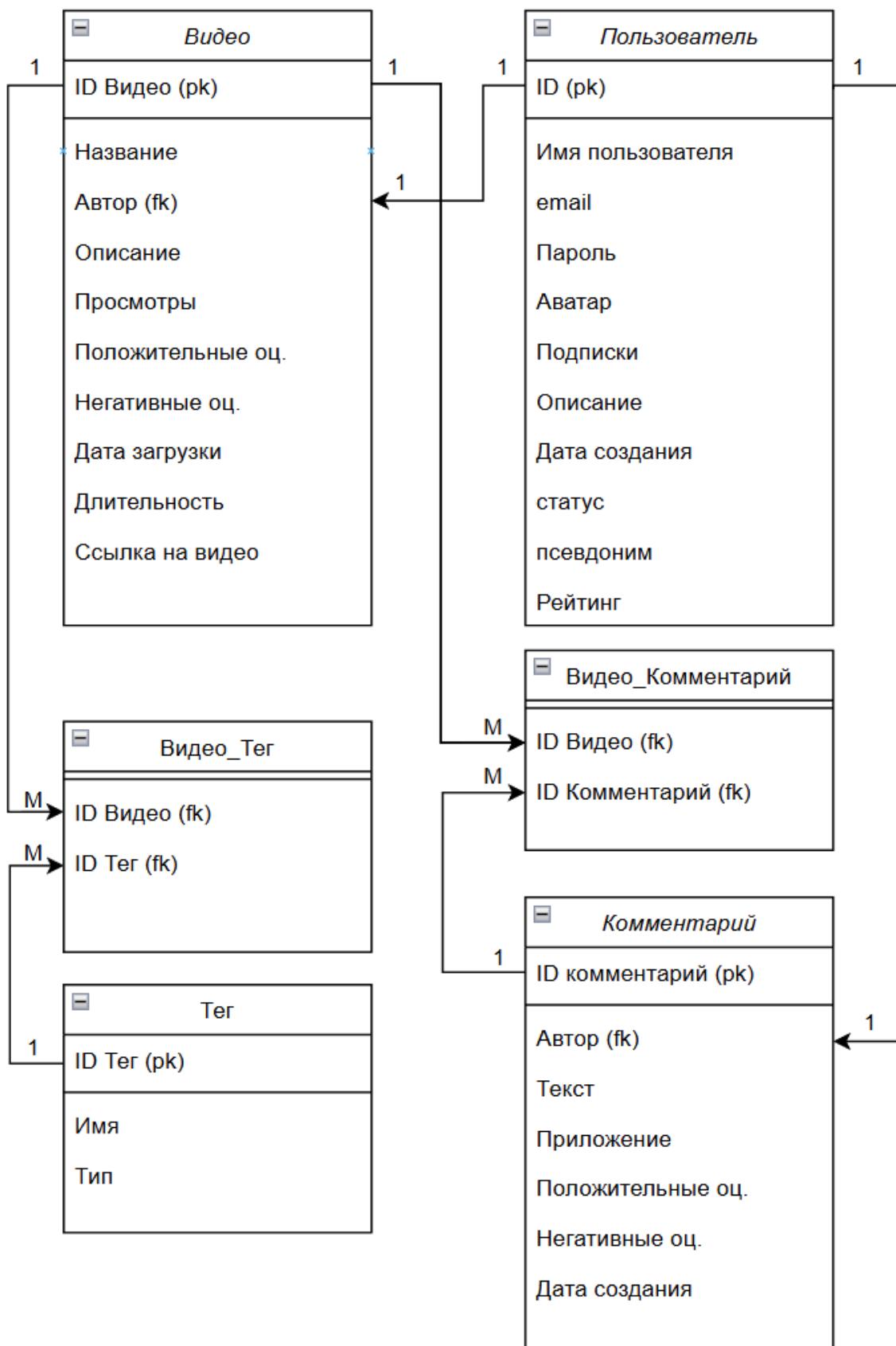


Рисунок 1.1 - Графическое представление логической модели данных.

## Структура программы

Структура программы включает следующие модули:

Модуль App - основной модуль отвечающий за работу сервера.

Модуль Watch - отвечает за страницу с выбранным видео.

Модуль Index - отвечает за главную страницу сайта и выдачу списка доступных видео

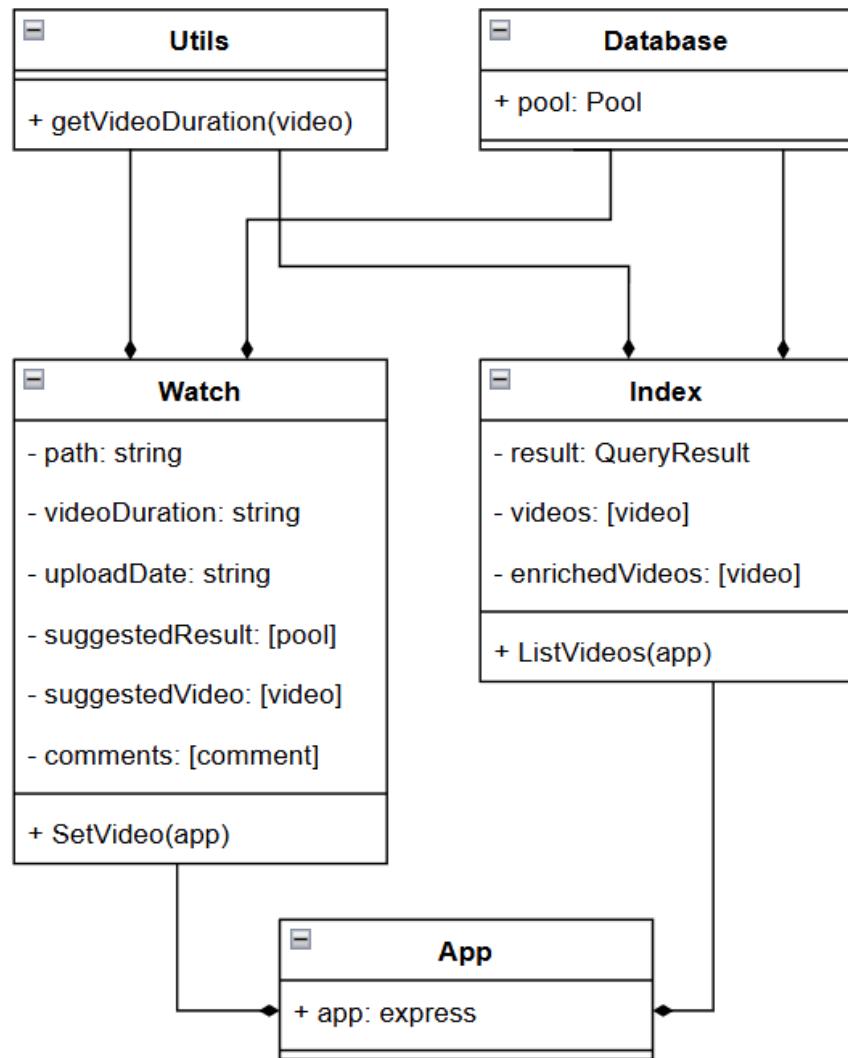


Рисунок 1.2 - Диаграмма классов

## Функциональные диаграммы

Для визуализации работы системы были сделаны следующие функциональные диаграммы:

IDEF0-диаграмма - описывает основную функциональность системы.

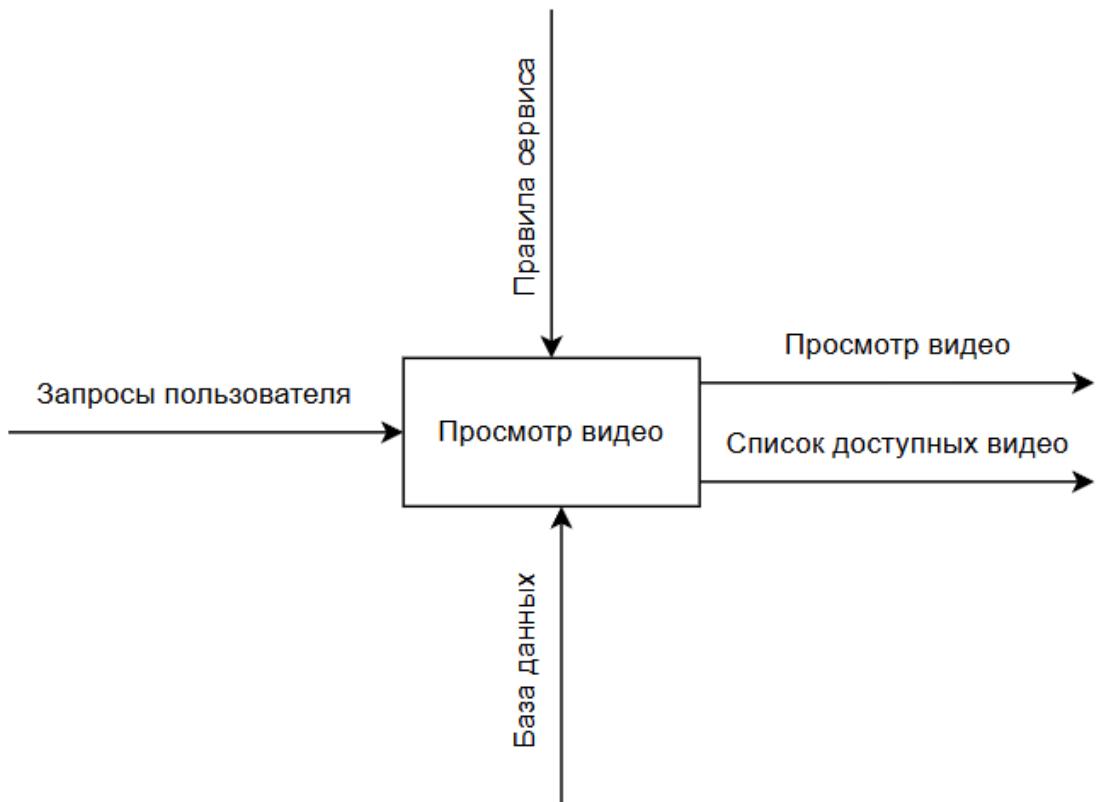


Рисунок 1.3 - IDEF0-диаграмма системы

Диаграмма последовательности - описывает процесс взаимодействия компонентов системы при авторизации и показывает взаимодействие разных компонентов системы между собой.

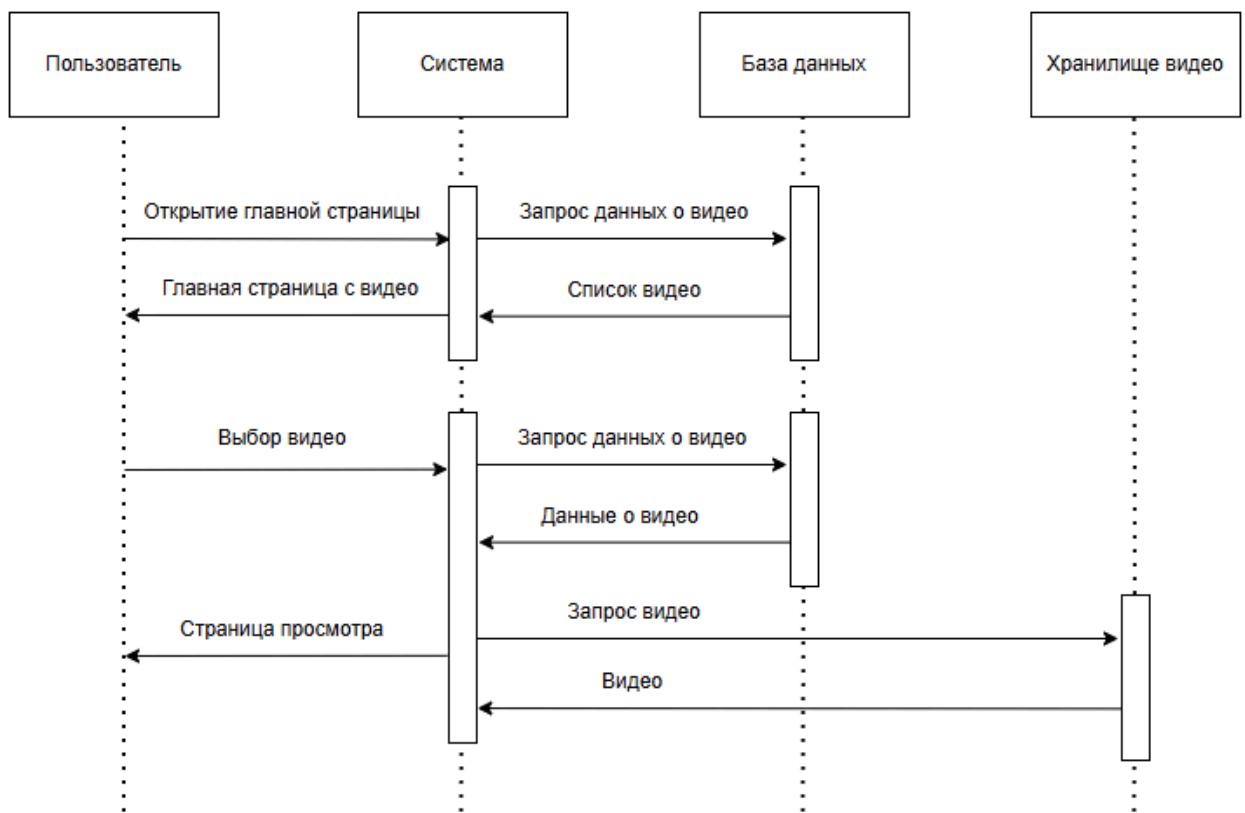


Рисунок 1.4 – Диаграмма последовательности

## **2 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

### **2.1 Общая информация**

Данный проект — это веб-приложение для размещения и просмотра видео.

Пользователи могут:

- Просматривать видео.
- Просматривать список.
- Просматривать статистику видео.

### **2.2 Основные функции**

Главная страница:

Содержит список доступных видео, представленных в виде блоков (миниатюра, название, автор, количество просмотров, длительность).

При клике на блок видео открывается страница его просмотра.

Страница видео:

Включает:

1. Видео плеер для воспроизведения.
2. Информацию о видео (название, количество просмотров, описание, дата загрузки).
3. Колонку с имеющимися видео.

### **2.3 Руководство по использованию**

Запуск приложения:

1. Убедитесь, что установлен Node.js, PostgreSQL 17.
2. Внутри папки проекта, запустите сервер используя команду: node app.js

3. Перейдите в браузере по адресу <http://localhost:3000>.

Просмотр видео:

1. На главной странице выберите интересующее видео.
2. Просмотрите видео и по желанию ознакомьтесь с описанием.

Просмотр других видео:

Выберите видео в колонке "Предлагаемые" справа от страницы просмотра или нажмите на ссылку "Главная" и осуществите действия указанное в "Просмотр видео" пункт 2.

### **3 ТЕСТИРОВАНИЕ И ДЕМОНСТРАЦИЯ РАБОТЫ**

План тестирования:

Функциональное:

- Загрузка главной страницы  
Ожидаемый результат: Список видео отображается корректно
- Клик на видео на главной странице  
Ожидаемый результат: Открывается страница видео с корректным контентом
- Увеличение просмотров при открытии страницы видео  
Ожидаемый результат: Число просмотров увеличивается в базе данных и в интерфейсе.

Интерфейса:

- Проверка масштабируемости главной страницы  
Ожидаемый результат: Контент корректно отображается на разных разрешениях.
- Проверка дизайна страницы видео  
Ожидаемый результат: Элементы отображаются по сетке, нет смещений.
- Проверка отображения миниатюр видео  
Ожидаемый результат: Отображается корректная миниатюра или заглушка.

Далее представлены результаты ручного тестирования.

### **3.1 Проверка интерфейса главной страницы.**

На рисунке 3.1.1 изображено отображение верстки при горизонтальном расположении и разрешении 2560 на 1440 точки, отображение корректное.

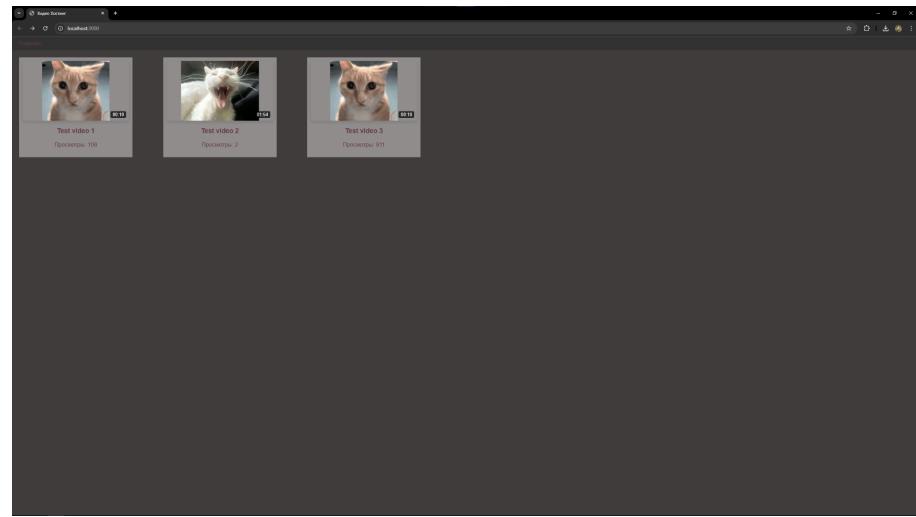


Рисунок 3.1.1 - Главная страница размер 1.

На рисунке 3.1.2 изображено отображение верстки при вертикальном расположении окна в режиме смартфона, отображение корректное.

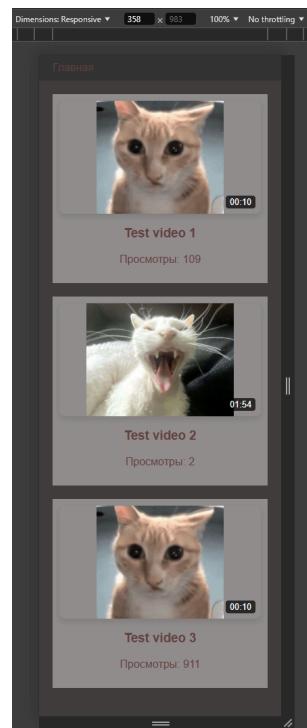


Рисунок 3.1.2 - Главная страница размер 2.

На рисунке 3.1.3 изображено отображение верстки при нестандартном размере окна, отображение корректное.

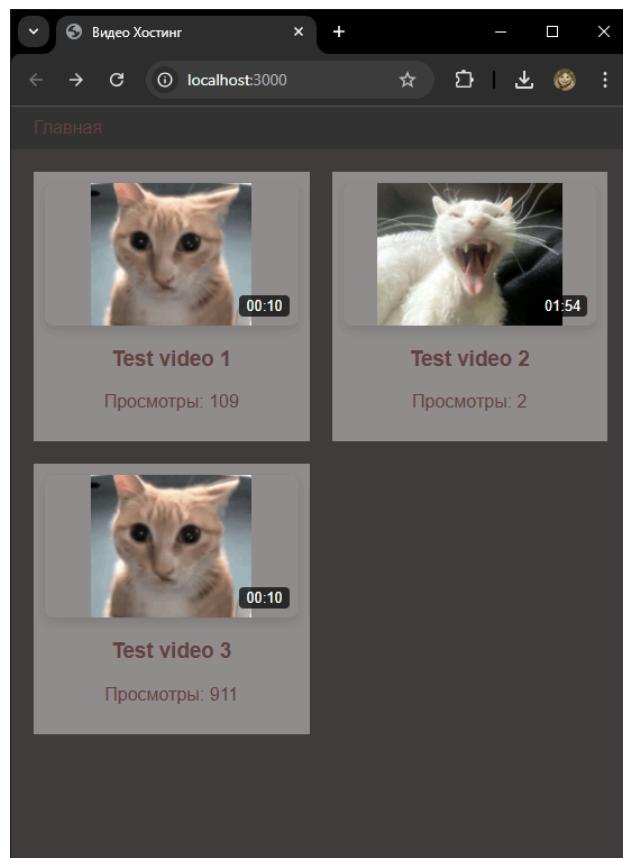


Рисунок 3.1.3 - Главная страница размер 3

В ряде представленных так же было проведена проверка отображения иконок видео и их замена при отсутствии. Для тестового видео 1 и 3 файлы иконок были повреждены и они были автоматически заменены иконкой плейсхолдером. Для тестового видео 2 файл иконки был считан корректно.

Также отображение названий, иконок, длительности видео и количества просмотров на видео отображаются корректно и соответствует своим данным в базе данных.

### 3.2.1 Проверка интерфейса страницы просмотра видео.

На рисунке 3.2.1 изображено отображение верстки при горизонтальном расположении и разрешении 2560 на 1440 точки, отображение корректное.

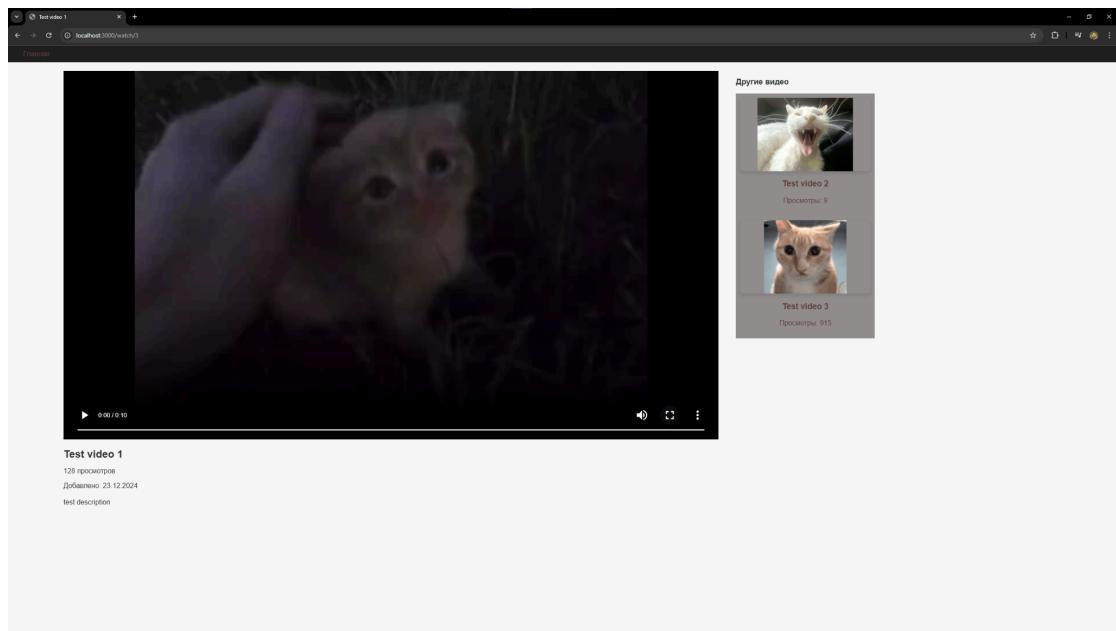


Рисунок 3.2.1 - Страница просмотра размер 1.

На рисунке 3.2.2 изображено отображение верстки при вертикальном расположении окна в режиме смартфона, отображение корректное.

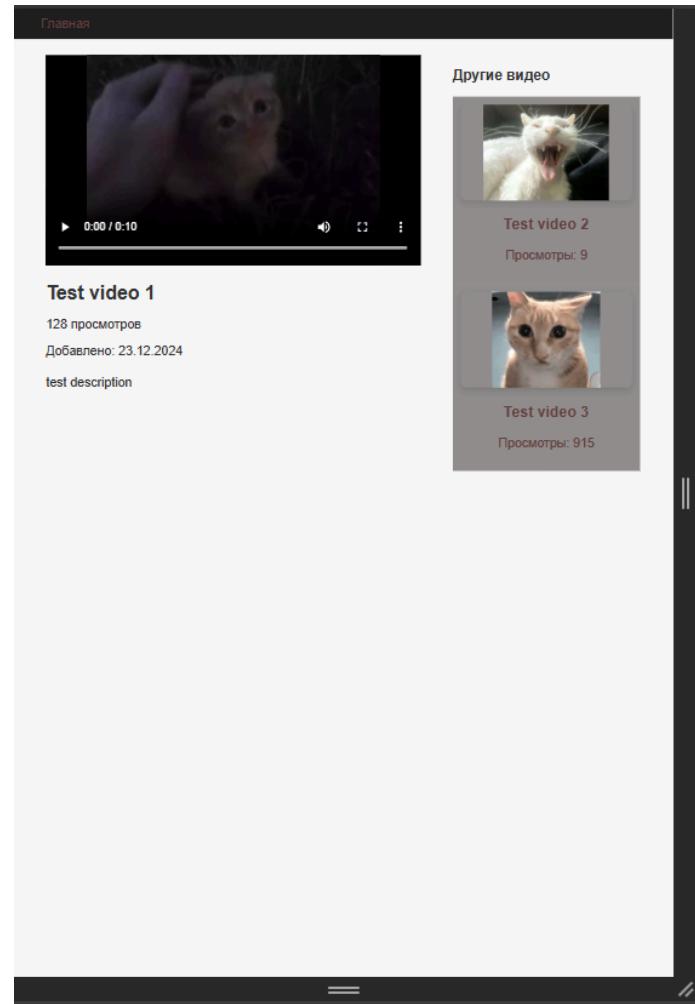


Рисунок 3.2.2 - Страница просмотра размер 2.

На рисунке 3.2.3 изображено отображение верстки при нестандартном размере окна, отображение корректное.

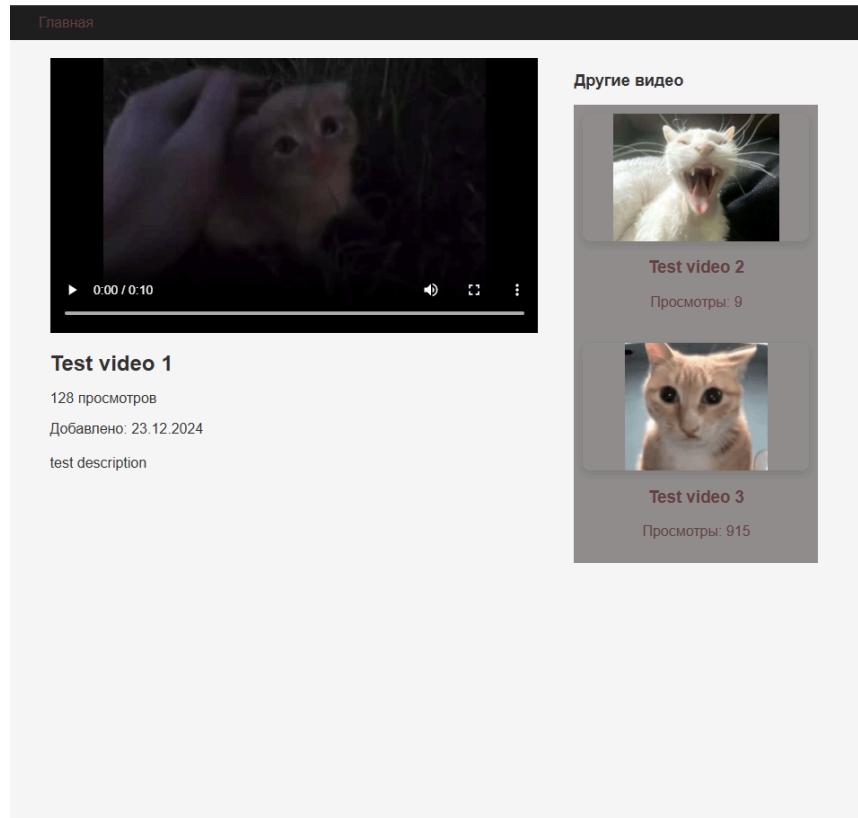


Рисунок 3.2.3 - Страница просмотра размер 3

Отображение названий, иконок, количества просмотров, описание и дата загрузки на видео отображаются корректно и соответствует своим данным в базе данных. Верстка ведет как и задумано без артефактов независимо от формата окна.

Далее на рисунках с 3.2.4 по 3.2.6 будут представлены результаты выбора видео со страницы просмотра.

Переход будет осуществлен на видео 2, после на видео 3 и в конце на видео 1.

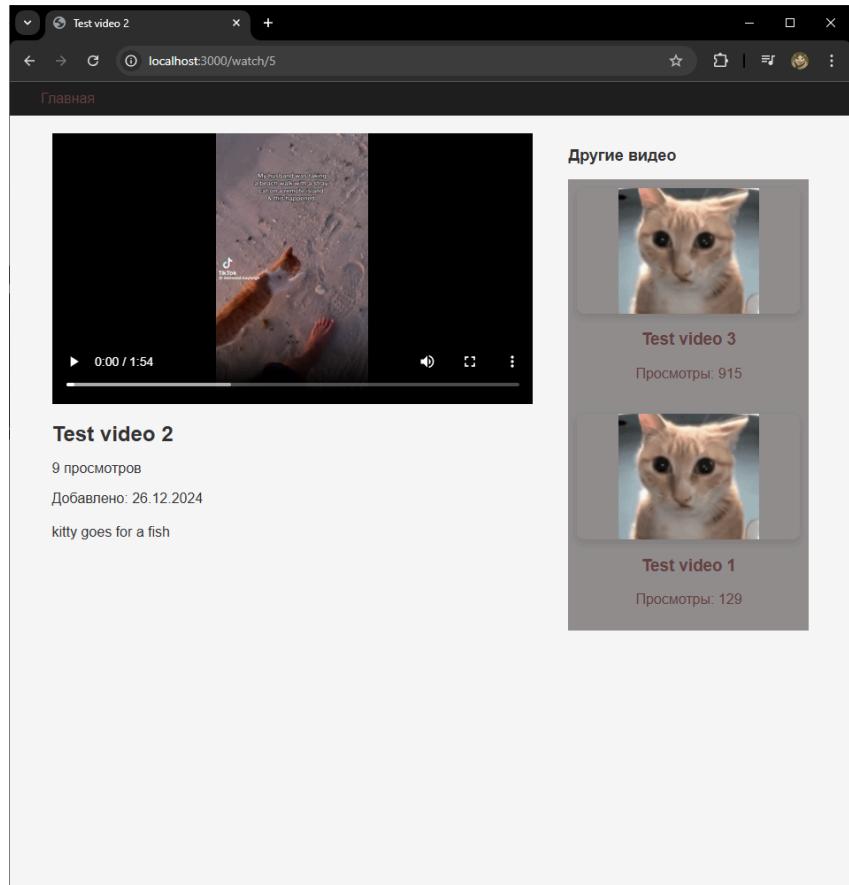


Рисунок 3.2.4 - Страница просмотраа видео 2

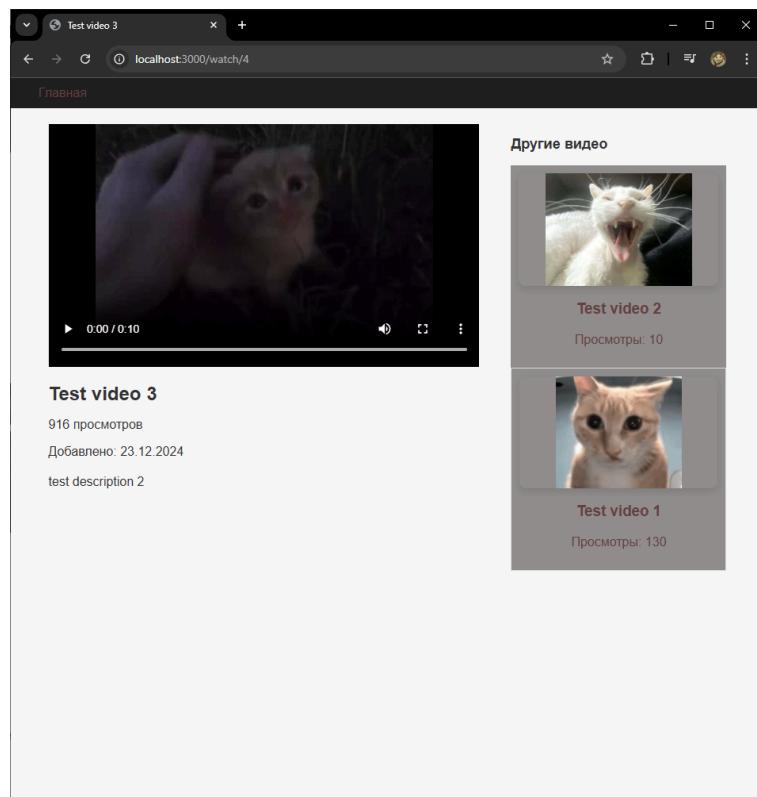


Рисунок 3.2.5 - Страница просмотраа видео 3

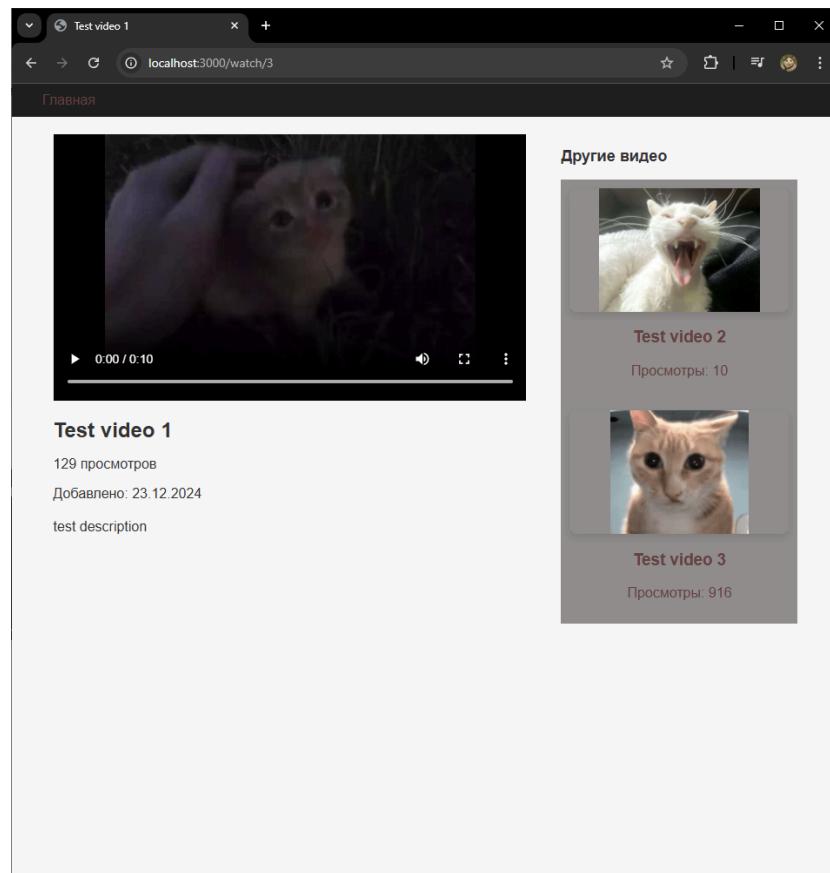


Рисунок 3.2.6 - Страница просмотра видео 1

Данный тест был пройден успешно выбранные видео не появляются в списке видео и видео соответствуют своему названию и приписанным данным.

Общий результат тестирования положительный ошибок не было выявлено, весь предполагаемый функционал работает корректно.

## **ЗАКЛЮЧЕНИЕ**

В процессе выполнения курсовой работы по разработке веб-приложения видеохостинга были применены навыки проектирования баз данных. Такими навыками являются разработка реляционной модели базы данных. Также были изучены и применены навыки написания веб-приложений.

Программный комплекс был реализован на языках программирования js, html, css с использованием IDE Visual Studio Code. Для работы с базой данных в качестве СУБД были выбраны PostgreSQL и среда pgAdmin 4. Веб-приложение может быть улучшено путем внедрения новых функций, таких как возможность создания аккаунтов, загрузки видеоматериалов, система тегов, использование алгоритмов рекомендаций для индивидуального подбора видеоматериалов.

Таким образом, цель курсовой работы – создание прототипа базового веб-приложения видеохостинга – была успешно достигнута.

При выполнении курсовой работы были приобретены такие компетенции, как инициализация инсталлирование программного и аппаратного обеспечения для информационных и автоматизированных систем, а также освоение методики использования программных средств для решения практических задач.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ОС ТУСУР 01-2021. Работы студенческие по направлениям подготовки и специальностям технического профиля. Общие требования и правила оформления. Томск: Томский гос. ун-т систем упр. и радиоэлектроники, 2021. 52 с.
2. The Node.js Documentation. Официальная документация Node.js. — [Электронный ресурс]. — URL: <https://nodejs.org/en/docs/> (дата обращения: 25.12.2024).
3. Mozilla Developer Network (MDN). JavaScript: современный учебник по JavaScript. — [Электронный ресурс]. — URL: <https://developer.mozilla.org/ru/docs/Web/JavaScript> (дата обращения: 13.12.2024).
4. PostgreSQL Documentation. Руководство по PostgreSQL (официальная документация). — [Электронный ресурс]. — URL: <https://www.postgresql.org/docs/> (дата обращения: 29.12.2024).
5. Simpson, K. You Don't Know JS (серия книг). — [Электронный ресурс]. — URL: <https://github.com/getify/You-Dont-Know-JS> (дата обращения: 29.12.2024).
6. Learn JavaScript. Современный учебник JavaScript. — [Электронный ресурс]. — URL: <https://learn.javascript.ru> (дата обращения: 28.12.2024).

# ПРИЛОЖЕНИЕ А

## (обязательное)

### Программный код скрипта инициализации

```
// Подтягиваем зависимости
const express = require('express');
const path = require('path');
const app = express();
const listVideos = require('./models/Index');
const SetVideo = require('./models/watch')

// порт на котором будет хостится сайт
const PORT = 3000;

// Устанавливаем папки для видео и изображений
const videosPath = path.join(__dirname, './Debug/Videos');
const imagesPath = path.join(__dirname, './Debug/Images');

// Устанавливаем статические папки
app.use('/videos', express.static(videosPath));
app.use('/Debug/Images', express.static(imagesPath));
app.use(express.static('public'));

// Настройка шаблонизатора
app.set('view engine', 'ejs');
app.set('views', './public/views');

// инициализация для списка видео на главной странице
listVideos(app);

// инициализация видео
SetVideo(app);

// Запуск сервера
app.listen(PORT, () => {
    console.log(`Сервер запущен на http://localhost:${PORT}`);
});
```

# ПРИЛОЖЕНИЕ Б

## (обязательное)

### Программный код модуля страницы просмотра

```
const pool = require('./database'); // Подключаем базу данных

// выделяем поля
let commentsResult = null;
let videoResult = null;
let video = null;
let videoId = null;
let comments = null;
let suggestedResult = null;
let suggestedVideos = null;
let uploadDate = null;
let path = null;

function SetVideo(app){
    app.get('/watch/:id', async (req, res) => {
        videoId = req.params.id;

        try {
            // Получение данных о выбранном видео
            videoResult = await pool.query('SELECT * FROM videos WHERE id = $1', [videoId]);
            video = videoResult.rows[0];

            if (!video) {
                return res.status(404).send('Видео не найдено');
            }

            // Получение списка рекомендуемых видео
            suggestedResult = await pool.query('SELECT * FROM videos WHERE id != $1 LIMIT 20', [videoId]);
            suggestedVideos = suggestedResult.rows;

            // Получение даты выхода
            uploadDate = new Date(video.uploaded_at).toLocaleDateString('ru-RU');

            // указываем путь на видеофайл
            path = video.video_file

            // Увеличиваем число просмотров
            await pool.query('UPDATE videos SET views = views + 1 WHERE id = $1', [videoId]);

            res.render('watch', { video, suggestedVideos, uploadDate, path });
        } catch (err) {
            console.error('Ошибка загрузки видео:', err);
            res.status(500).send('Ошибка загрузки видео');
        }
    });
}

module.exports = SetVideo;
```

**ПРИЛОЖЕНИЕ В**  
**(обязательное)**  
**Программный код модуля главной страницы**

```
const pool = require('./database'); // Подключаем базу данных
const getVideoDuration = require('./utils');

let result;
let videos;
let enrichedVideos;

function listVideos(app) {
    app.get('/', async (req, res) => {
        try {
            // Запрос для получения списка видео
            result = await pool.query('SELECT * FROM videos');
            videos = result.rows;

            // Обогащение каждого видео его длительностью
            enrichedVideos = videos.map(video => ({
                ...video,
                duration: getVideoDuration(video)
            }));
        }

        res.render('index', { videos: enrichedVideos });
    } catch (err) {
        console.error('Ошибка обработки видео:', err);
        res.status(500).send('Ошибка сервера');
    }
};

module.exports = listVideos;
```

## ПРИЛОЖЕНИЕ Г

### (обязательное)

### Программный код модуля utils.js

```
// парсит интервалы postgres в переменные времени
let parseInterval = (interval) => {
    const hours = interval.hours || 0; // Часы, если есть
    const minutes = interval.minutes || 0; // Минуты, если есть
    const seconds = interval.seconds || 0; // Секунды

    return { hours, minutes, seconds };
};

// форматирует из времени в читаемый формат времение MM:SS или HH:MM:SS
let intervalToTime = (interval) => {
    const { hours, minutes, seconds } = interval;

    // Если меньше часа, возвращаем MM:SS
    if (hours === 0) {
        return `${String(minutes).padStart(2, '0')}:${String(seconds).padStart(2, '0')}`;
    }

    // Если больше часа, возвращаем HH:MM:SS
    return `${String(hours).padStart(2, '0')}:${String(minutes).padStart(2, '0')}:${String(seconds).padStart(2, '0')}`;
};

/// Форматирование длительности видео в читаемый формат
function getVideoDuration(video){
    return intervalToTime(parseInterval(video.duration))
}

module.exports = getVideoDuration;
```

**ПРИЛОЖЕНИЕ Д**  
**(обязательное)**  
**Программный код модуля database.js**

```
const { Pool } = require('pg');

// Настройки подключения к базе данных
const pool = new Pool({
    user: 'VideoHosting',      // Имя пользователя PostgreSQL
    host: 'localhost',         // Хост базы данных
    database: 'video_hosting', // Имя базы данных
    password: "password",     // Пароль пользователя
    port: 5432,                // Порт подключения
});

// Проверка подключения
pool.connect((err, client, release) => {
    if (err) {
        console.error('Ошибка подключения к базе данных:', err.stack);
    } else {
        console.log('Успешное подключение к базе данных');
        release();
    }
});

module.exports = pool;
```

**ПРИЛОЖЕНИЕ Е**  
**(обязательное)**

**Программный код листинга Главной страницы**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Видеохостинг</title>
    <link rel="stylesheet" href="/css/style.css">
</head>
<body>
    <header>
        <nav>
            <a href="/">Главная</a>
            <a href="/profile">Профиль</a>
        </nav>
    </header>
    <main>
        <div class="video-grid">
            <% videos.forEach(video => { %>
                <div class="video-item">
                    <a href="/watch/<%= video.id %>">
                        <div class="thumbnail">
                            " onerror="this.onerror=null; this.src='./Resorces/brainless confusion.jpg';">
                            <span class="duration"><%= video.duration %></span>
                        </div>
                        <h3><%= video.title %></h3>
                        <p>Автор: <%= video.author %></p>
                        <p>Просмотры: <%= video.views %></p>
                    </a>
                </div>
            <% }); %>
        </div>
    </main>
</body>
</html>
```

# ПРИЛОЖЕНИЕ Ё

## (обязательное)

### Программный код листинга Главной страницы

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="/css/style.css">
    <link rel="stylesheet" href="/css/watch_style.css">
    <title><%= video.title %></title>
</head>
<body>
    <header>
        <nav>
            <a href="/">Главная</a>
        </nav>
    </header>

    <div class="container">
        <!-- Main Content -->
        <div class="main-content">
            <div class="video-player">
                <video src="<%= path %>" controls></video>
            </div>
            <div class="video-info">
                <h1><%= video.title %></h1>
                <p><%= video.views %> просмотров</p>
                <p>Добавлено: <%= uploadDate %></p>
                <div class="description">
                    <p id="description-text"><%= video.description %></p>
                </div>
            </div>
        </div>
        <!-- Suggested Videos -->
        <div class="suggested-videos">
            <h2>Другие видео</h2>
            <ul>
                <% suggestedVideos.forEach(video => { %>
                    <div class="video-item">
                        <a href="/watch/<%= video.id %>">
                            <div class="thumbnail">
                                " onerror="this.onerror=null; this.src='/Resorces/brainless confusion.jpg';">
                            </div>
                            <h3><%= video.title %></h3>
                            <!-- <p>Автор: <%= video.author %></p> -->
                            <p>Просмотры: <%= video.views %></p>
                        </a>
                    </div>
                <% }); %>
            </ul>
        </div>
    </div>
</body>
</html>
```

**ПРИЛОЖЕНИЕ Ж**  
**(обязательное)**  
**Программный код стиля Главной страницы**

```
body {  
    font-family: Arial, sans-serif;  
    margin: 0;  
    padding: 0;  
    background: #413c3c;  
}  
  
header {  
    background: #333;  
    color: #fff;  
    padding: 10px 20px;  
}  
  
header nav a {  
    color: #fff;  
    text-decoration: none;  
    margin-right: 20px;  
}  
  
.video-grid {  
    display: grid;  
    grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));  
    max-width: 1200px;  
    gap: 20px;  
    padding: 20px;  
}  
  
.video-item {  
    text-align: center;  
    background: #918d8d;  
    padding: 10px;  
    /* border: 1px solid #ddd; */  
    max-width: 300px;  
}  
  
.thumbnail {  
    position: relative;  
    width: 100%;  
    padding-top: 56.25%;  
    overflow: hidden;  
    border-radius: 8px;  
    max-width: 300px;  
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);  
}  
  
.thumbnail .background {  
    position: absolute;  
    top: 0;  
    left: 0;  
    width: 100%;  
    height: 100%;  
    background-size: cover;  
    background-position: center;  
    filter: blur(10px);
```

```

        transform: scale(1.1);
        z-index: 1;
    }

.thumbnail img {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    object-fit: contain;
    z-index: 2;
}

.thumbnail .duration {
    position: absolute;
    bottom: 8px;
    right: 8px;
    background: rgba(0, 0, 0, 0.7);
    color: white;
    font-size: 0.8rem;
    padding: 2px 6px;
    border-radius: 4px;
    z-index: 3;
}

.duration {
    position: absolute;
    bottom: 5px;
    right: 5px;
    background: rgba(0, 0, 0, 0.7);
    color: #fff;
    padding: 2px 5px;
    font-size: 12px;
}

@media (min-width: 768px) {
    .thumbnail {
        transition: transform 0.22s ease;
    }
    .thumbnail:hover {
        transform: scale(1.15);
    }
    .thumbnail:hover {
        max-width: 320px;
    }
}

.video-item:hover {
    border: 1px solid #ddd;
}

a {
    color: rgb(255, 255, 255);
    text-decoration: none;
}
a:hover {
    color: rgba(255, 0, 0, 0.616);
}

a:visited {
    color: rgba(255, 190, 190, 0.945);
}

```

```
a:active {  
    color: rgb(209, 62, 99);  
}
```

## **ПРИЛОЖЕНИЕ 3**

(обязательное)

### **Программный код стиля Страницы просмотра**

```
body {  
    font-family: Arial, sans-serif;  
    margin: 0;  
    padding: 0;  
    background: #413c3c;  
    color: #333;  
}
```

```
header {  
    background-color: #222;  
    color: #fff;  
    padding: 10px 20px;  
    display: flex;  
    justify-content: space-between;  
    align-items: center;  
}
```

```
header nav a {  
    color: #fff;  
    text-decoration: none;  
    margin-left: 15px;  
}
```

```
header nav a:hover {  
    text-decoration: underline;  
}
```

```
.container {  
    display: flex;  
    justify-content: space-between;  
    margin: 20px 5%;  
}
```

```
.main-content {  
    flex: 2;  
    margin-right: 20px;  
}
```

```
.video-player {  
    position: relative;  
    width: 100%;  
    padding-top: 56.25%;  
    background-color: #000;  
    overflow: hidden;  
}
```

```
.video-player video {  
    position: absolute;  
    top: 0;  
    left: 0;  
    width: 100%;  
    height: 100%;  
    object-fit: contain;
```

```
}

.video-info {
    margin-top: 20px;
    color: #bbb;
}

.video-info h1 {
    font-size: 24px;
    font-weight: bold;
}

.buttons {
    margin-top: 10px;
    display: flex;
    align-items: center;
    gap: 10px;
}

.buttons button {
    background: #ddd;
    border: none;
    padding: 5px 10px;
    border-radius: 5px;
    cursor: pointer;
}

.buttons button:hover {
    background: #bbb;
}

.channel-info {
    margin-top: 20px;
    display: flex;
    align-items: center;
    gap: 10px;
}

.channel-info img {
    width: 50px;
    height: 50px;
    border-radius: 50%;
}

.description {
    margin-top: 20px;
}

.description button {
    display: block;
    margin: 10px 0;
    background: #ddd;
    border: none;
    padding: 5px 10px;
    border-radius: 5px;
    cursor: pointer;
}

.comments-section {
    margin-top: 20px;
}
```

```
.comments-section textarea {  
    width: 100%;  
    resize: none;  
    height: 60px;  
}  
  
.comments-section button {  
    display: none;  
    margin-top: 5px;  
    background: #ddd;  
    border: none;  
    padding: 5px 10px;  
    border-radius: 5px;  
    cursor: pointer;  
}  
  
.comments-section textarea:focus + button {  
    display: inline-block;  
}  
  
.suggested-videos {  
    flex: 1;  
    margin-left: 20px;  
    color:#bbb;  
}  
  
.suggested-videos h2 {  
    font-size: 18px;  
    margin-bottom: 10px;  
}  
  
.suggested-videos ul {  
    list-style: none;  
    padding: 0;  
}  
  
.suggested-videos li {  
    margin-bottom: 10px;  
}
```