Оглавление

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *ДП 00.00.ПЗ* | | | | | |
|  |  |  |  |  |
|  |  | Ф.И.О | Подпись | Дата |
| Разраб | | Окулич Д.Ю. |  |  | *Оглавление* | Лит. | | | Лист | Листов |
| Пров. | | Белодед Н.И. |  |  |  | У |  | 1 | 3 |
|  | |  |  |  | БГТУ 1-40 01 01, 2025 | | | | |
| Н. контр. | | Белодед Н.И. |  |  |
| Утв. | | Смелов В.В. |  |  |

[Реферат 7](#_Toc199962355)

[Abstract 8](#_Toc199962356)

[Введение 9](#_Toc199962357)

[1 Постановка задачи и аналитический обзор аналогичных решений 10](#_Toc199962358)

[1.1 Обзор аналогичных решений 10](#_Toc199962359)

[1.1.1 Веб-приложение «YouTube» 10](#_Toc199962360)

[1.1.2 Веб-приложение «RUTUBE» 11](#_Toc199962361)

[1.1.3 Веб-приложение «VK Video» 12](#_Toc199962362)

[1.2 Постановка задачи 13](#_Toc199962363)

[1.3 Выводы по разделу 13](#_Toc199962364)

[2 Проектирование веб-приложения 14](#_Toc199962365)

[2.1 Функциональность веб-приложения 14](#_Toc199962366)

[2.2 Логическая схема базы данных 16](#_Toc199962367)

[2.3 Архитектура веб-приложения 17](#_Toc199962368)

[2.4 Блок-схема алгоритма подключения к совместному просмотру 17](#_Toc199962369)

[2.5 Выводы по разделу 18](#_Toc199962370)

[3 Разработка веб-приложения 19](#_Toc199962371)

[3.1 Разработка серверной части веб-приложения 19](#_Toc199962372)

[3.1.1 Используемые библиотеки 19](#_Toc199962373)

[3.1.2 Структура проекта серверной части 19](#_Toc199962374)

[3.1.3 Разработка уровня доступа к данным 20](#_Toc199962375)

[3.1.4 Разработка уровня бизнес-логики 21](#_Toc199962376)

[3.1.5 Разработка уровня представления 22](#_Toc199962377)

[3.1.6 Настройка безопасности 23](#_Toc199962378)

[3.1.7 Используемые паттерны 24](#_Toc199962379)

[3.2 Клиентская часть 24](#_Toc199962380)

[3.2.1 Используемые библиотеки 24](#_Toc199962381)

[3.2.2 Структура проекта клиентской части 25](#_Toc199962382)

[3.2.3 Конфигурация проекта 25](#_Toc199962383)

[3.3 Реализация функций 25](#_Toc199962384)

[3.3.1 Регистрация 25](#_Toc199962385)

[3.3.2 Авторизация 26](#_Toc199962386)

[3.3.3 Просмотр публичных видео 27](#_Toc199962387)

[3.3.4 Просмотр комментариев под видео 28](#_Toc199962388)

[3.3.5 Поиск публичных видео 28](#_Toc199962389)

[3.3.6 Поиск по названию 29](#_Toc199962390)

[3.3.7 Сортировка видео 29](#_Toc199962391)

[3.3.8 Поиск по тегам 29](#_Toc199962392)

[3.3.9 Удаление канала 30](#_Toc199962393)

[3.3.10 Редактирование канала 30](#_Toc199962394)

[3.3.11 Создание канала 31](#_Toc199962395)

[3.3.12 Загрузка видео 31](#_Toc199962396)

[3.3.13 Редактирование данных видео 33](#_Toc199962397)

[3.3.14 Удаление видео 34](#_Toc199962398)

[3.3.15 Удаление комментариев под своими видео 35](#_Toc199962399)

[3.3.16 Оставление жалобы под видео 36](#_Toc199962400)

[3.3.17 Оставление комментариев под видео 37](#_Toc199962401)

[3.3.18 Удаление своих комментариев 38](#_Toc199962402)

[3.3.19 Редактирование своих комментариев 38](#_Toc199962403)

[3.3.20 Совместный просмотр 39](#_Toc199962404)

[3.3.21 Создание плейлистов 40](#_Toc199962405)

[3.3.22 Удаление видео из плейлиста 41](#_Toc199962406)

[3.3.23 Добавление видео в плейлиста 41](#_Toc199962407)

[3.3.24 Воспроизведение плейлиста 43](#_Toc199962408)

[3.3.25 Удаление плейлиста 44](#_Toc199962409)

[3.3.26 Удаление любых комментариев администратором 45](#_Toc199962410)

[3.3.27 Скрытия видео администратором 45](#_Toc199962411)

[3.3.28 Удаление видео администратором 46](#_Toc199962412)

[3.3.29 Просмотр жалоб пользователей администратором 47](#_Toc199962413)

[3.3.30 Блокировка канала администратором 48](#_Toc199962414)

[3.3.31 Удаление канала администратором 48](#_Toc199962415)

[3.4 Маршруты для контролеров 50](#_Toc199962416)

[3.5 Выводы по разделу 50](#_Toc199962417)

[4 Тестирование веб-приложения 51](#_Toc199962418)

[4.1 Функциональное тестирование 51](#_Toc199962419)

[4.2 Автоматизированное тестирование 53](#_Toc199962420)

[4.3 Нагрузочное тестирования 54](#_Toc199962421)

[4.4 Выводы по разделу 54](#_Toc199962422)

[5 Руководство пользователя 55](#_Toc199962423)

[5.1 Регистрация 55](#_Toc199962424)

[5.2 Авторизация 56](#_Toc199962425)

[5.3 Просмотр публичных видео. 57](#_Toc199962426)

[5.4 Просмотр комментариев под видео 57](#_Toc199962427)

[5.5 Поиск публичных видео 58](#_Toc199962428)

[5.6 Поиск по названию 58](#_Toc199962429)

[5.7 Сортировка видео 58](#_Toc199962430)

[5.8 Поиск по тегам 58](#_Toc199962431)

[5.9 Удаление канала 59](#_Toc199962432)

[5.10 Редактирование канала 59](#_Toc199962433)

[5.11 Создание канала 59](#_Toc199962434)

[5.12 Загрузка видео 60](#_Toc199962435)

[5.13 Редактирование данных видео 61](#_Toc199962436)

[5.14 Удаление видео 62](#_Toc199962437)

[5.15 Удаление комментариев под своими видео 62](#_Toc199962438)

[5.16 Оставление жалобы под видео 62](#_Toc199962439)

[5.17 Оставление комментариев под видео 63](#_Toc199962440)

[5.18 Удаление своих комментариев 63](#_Toc199962441)

[5.19 Редактирование своих комментариев 63](#_Toc199962442)

[5.20 Совместный просмотр 63](#_Toc199962443)

[5.21 Создание плейлистов 64](#_Toc199962444)

[5.22 Удаление видео из плейлиста 65](#_Toc199962445)

[5.23 Добавление видео в плейлиста 65](#_Toc199962446)

[5.24 Воспроизведение плейлиста 66](#_Toc199962447)

[5.25 Удаление плейлиста 66](#_Toc199962448)

[5.26 Удаление любых комментариев администратором 66](#_Toc199962449)

[5.27 Скрытия видео администратором 67](#_Toc199962450)

[5.28 Удаление видео администратором 67](#_Toc199962451)

[5.29 Просмотр жалоб пользователей администратором 67](#_Toc199962452)

[5.30 Блокировка канала администратором 67](#_Toc199962453)

[5.31 Удаление канала администратором 68](#_Toc199962454)

[5.32 Вывод по разделу 68](#_Toc199962455)

[6 Технико-экономическое обоснования проекта 69](#_Toc199962456)

[6.1 Общая характеристика разрабатываемого программного средства 69](#_Toc199962457)

[6.2 Исходные данные для проведения расчетов и маркетинговый анализ 69](#_Toc199962458)

[6.3 Обоснование цены программного средства 70](#_Toc199962459)

[6.3.1 Расчет затрат рабочего времени на разработку программного средства 70](#_Toc199962460)

[6.3.2 Расчет основной заработной платы 71](#_Toc199962461)

[6.3.3 Расчет дополнительной заработной платы 72](#_Toc199962462)

[6.3.4 Расчет отчислений в Фонд социальной защиты населения и по обязательному страхованию 72](#_Toc199962463)

[6.3.5 Расчет суммы прочих прямых затрат 73](#_Toc199962464)

[6.3.6 Расчет суммы накладных расходов 73](#_Toc199962465)

[6.3.7 Сумма расходов на разработку программного средства 73](#_Toc199962466)

[6.3.8 Расходы на сопровождение и адаптацию 74](#_Toc199962467)

[6.3.9 Расчет полной себестоимости 74](#_Toc199962468)

[6.4 Вывод по разделу 74](#_Toc199962469)

[Заключение 76](#_Toc199962470)

[Список использованной литературы 77](#_Toc199962471)

[Диаграмма вариантов использована (ДП 01.00.ГЧ) 79](#_Toc199962472)

[Логическая схема базы данных (ДП 02.00.ГЧ) 80](#_Toc199962473)

[Диаграмма развертывания (ДП 03.00.ГЧ) 81](#_Toc199962474)

[Блок-схема подключения к совместному просмотру (ДП 04.00.ГЧ) 82](#_Toc199962475)

[Диаграмма компонентов (ДП 05.00.ГЧ) 83](#_Toc199962476)

[Скриншот работы приложения (ДП 06.00.ГЧ) 84](#_Toc199962477)

[Приложение А 85](#_Toc199962478)

[Приложение Б 88](#_Toc199962479)

[Приложение В 90](#_Toc199962480)

[Приложение Г 95](#_Toc199962481)

[Приложение Д 98](#_Toc199962482)

Реферат

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *ДП 00.00.ПЗ* | | | | | |
|  |  |  |  |  |
|  |  | Ф.И.О | Подпись | Дата |
| Разраб | | Окулич Д.Ю. |  |  | *Реферат* | Лит. | | | Лист | Листов |
| Пров. | | Белодед Н.И. |  |  |  | У |  | 1 | 1 |
|  | |  |  |  | БГТУ 1-40 01 01, 2025 | | | | |
| Н. контр. | | Белодед Н.И. |  |  |
| Утв. | | Смелов В.В. |  |  |

Пояснительная записка дипломного проекта содержит 99 страниц пояснительной записки, 19 таблиц, 9 формул, 19 иллюстрация, 27 источников литературы, 5 приложений.

DOCKER 4.39.0, ASP .NET CORE 8, C# 12.0, SIGNALR 8.0.7, EF 8.0.1, POSTGRESQL 17, REACT.JS 18.3.1, MOBX 6.13.5

Целью дипломного проекта является разработка видеохостинга который предоставляет платформу для совместного просмотра видео.

Веб-приложение предоставляет пользователю просмотр видео, создание плейлистов, создание каналов для загрузки своего контента, комментирование видео, так же возможность совместного просмотра видео контента с другими пользователями.

Дипломный проект состоит из оглавления, реферата, введения, 6 разделов, заключения, графической части, приложения.

Во введении рассмотрена актуальность проекта, а также поставлены задачи для дипломного проекта.

В первом разделе проведен аналитический обзор литературы и описана цель дипломного проекта, обзор аналогов.

Во втором разделе представлены используемые средства разработки, архитектура приложения и проектирование веб-приложения.

Третий раздел полностью посвящен разработке программной реализации веб-приложения.

Четвертый раздел посвящен тестированию веб-приложения.

В пятом разделе приведено руководство пользователя.

В шестом разделе приводится расчет экономических параметров.

В заключении приведены результаты проделанной работы.

Abstract

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *ДП 00.00.ПЗ* | | | | | |
|  |  |  |  |  |
|  |  | Ф.И.О | Подпись | Дата |
| Разраб | | Окулич Д.Ю. |  |  | *Abstract* | Лит. | | | Лист | Листов |
| Пров. | | Белодед Н.И. |  |  |  | У |  | 1 | 1 |
|  | |  |  |  | БГТУ 1-40 01 01, 2025 | | | | |
| Н. контр. | | Белодед Н.И. |  |  |
| Утв. | | Смелов В.В. |  |  |

The explanatory note of the graduation project contains 99 pages of explanatory notes, 19 tables, 9 formulas, 19 illustrations, 27 sources of literature, 5 appendices.

DOCKER 4.39.0, ASP .NET CORE 8, C# 12.0, SIGNALR 8.0.7, EF 8.0.1, POSTGRESQL 17, REACT.JS 18.3.1, MOBX 6.13.5

The purpose of the graduation project is to develop a video hosting service that provides a platform for video sharing.

The web application provides the user with watching videos, creating playlists, creating channels for uploading their content, commenting on videos, as well as the ability to share video content with other users.

The graduation project consists of a table of contents, an abstract, an introduction, 6 sections, a conclusion, a graphic part, and an appendix.

In the introduction, the relevance of the project is considered, as well as the tasks for the graduation project are set.

The first section provides an analytical review of the literature and describes the purpose of the thesis project, an overview of analogues.

The second section presents the development tools used, the architecture of the application, and the design of the web application.

The third section is devoted to the development of a software implementation of a web application.

The fourth section is devoted to testing the web application.

The fifth section contains the user's guide.

The sixth section provides the calculation of economic parameters.

In conclusion, the results of the work done are presented.

# **Введение**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *ДП 00.00.ПЗ* | | | | | |
|  |  |  |  |  |
|  |  | Ф.И.О | Подпись | Дата |
| Разраб | | Окулич Д.Ю. |  |  | *Введение* | Лит. | | | Лист | Листов |
| Пров. | | Белодед Н.И. |  |  |  | У |  | 1 | 1 |
|  | |  |  |  | БГТУ 1-40 01 01, 2025 | | | | |
| Н. контр. | | Белодед Н.И. |  |  |
| Утв. | | Смелов В.В. |  |  |

В современном мире видео-контент приобретает все большую актуальность, и несмотря на разнообразие платформ для личного просмотра, люди все чаще стремятся к совместному потреблению медиа. Количество пользователей, для которых онлайн-кинотеатры и видеохостинги стали основным способом развлечения и общения, растет с каждым годом.

Целью дипломного проекта является разработка видеохостинга который предоставляет платформу для совместного просмотра видео. Пользователю доступно возможность создавать каналы, обсуждать ролики в комментариях и наслаждаться контентом вместе, даже находясь далеко друг от друга.

Задачи дипломного проекта:

* постановка задачи и аналитический обзор аналогичных решений;
* проектирование веб-приложения: разработка схемы вариантов использования, разработка логической схемы базы данных, проектирование архитектуры веб-приложения;
* разработка веб-приложения;
* тестирование веб-приложения;
* разработка руководства пользователя;
* проведение технико-экономическое обоснование проекта.

Для разработки веб-приложения была выбрана монолитная архитектура. Серверная часть реализована на платформе ASP.NET Core 8.0 [1], клиентская часть – с использованием библиотеки React [2] и MOBX [3]. В качестве системы управления базами данных используется PostgreSQL [4], с подключением через Entity Framework Core. Для передачи данных используется REST API [5]. Для обеспечения совместного просмотра используется SignalR [6]. Приложение развернуто в Docker [7].

Для обеспечения безопасности пользователей используется валидация данных, управление ролями и политики авторизации.

Приложение предназначено для широкого круга пользователей, включая:

* авторов контента, заинтересованных в создании и публикации своих видео;
* обычных зрителей (гостей), которые хотят найти и просмотреть необходимый им видеоконтент;
* администраторов, отвечающих за модерацию и поддержание контента.

1. Постановка задачи и аналитический обзор аналогичных решений

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *ДП 01.00.ПЗ* | | | | | |
|  |  |  |  |  |
|  |  | Ф.И.О | Подпись | Дата |
| Разраб | | Окулич Д.Ю. |  |  | *1 Постановка задачи и аналитический обзор аналогичных решений* | Лит. | | | Лист | Листов |
| Пров. | | Белодед Н.И. |  |  |  | У |  | 1 | 4 |
|  | |  |  |  | БГТУ 1-40 01 01, 2025 | | | | |
| Н. контр. | | Белодед Н.И. |  |  |
| Утв. | | Смелов В.В. |  |  |

* 1. Обзор аналогичных решений
     1. Веб-приложение «YouTube»

YouTube [8] обладает широким спектром функций, включая загрузку и просмотр видео, создание плейлистов, подписку на каналы, комментирование и оценку видео, а также возможность проведения прямых трансляций. Этот видеохостинг также позволяет пользователям создавать и управлять своими каналами, а также монетизировать контент через рекламу и спонсорство.

YouTube имеет простой и удобный интерфейс, который легко использовать. Дизайн этого видеохостинга состоит из основных функций в верхней части экрана и списка рекомендованных видео в центральной части. YouTube также предлагает различные темы, которые пользователи могут выбрать, чтобы изменить внешний вид сайта.

Дизайн YouTube – минималистичный и простой в использовании. Цветовая схема состоит в основном из белого и красного цветов, что создает ощущение легкости и чистоты для пользователя.

Интерфейс приложения представлен на рисунке 1.1.

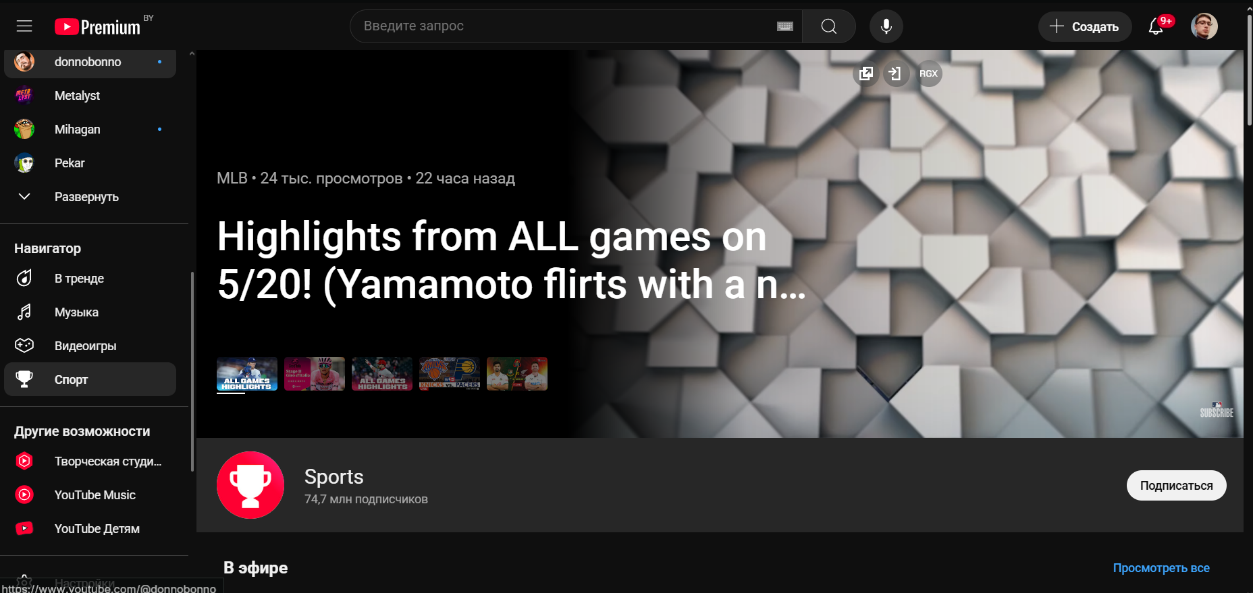


Рисунок 1.1 – Приложение «YouTube»

К преимуществам можно отнести обширную библиотеку контента, развитую инфраструктуру, большое количество авторов, а также удобный интерфейс.

К недостаткам можно отнести большое кол-во рекламного контента, а также закрытость некоторого контента для платного распространения.

* + 1. Веб-приложение «RUTUBE»

RUTUBE [9] предоставляет пользователям широкий спектр возможностей для работы с видео. Платформа поддерживает загрузку и просмотр контента, организацию видео в плейлисты, подписку на интересующие каналы, а также взаимодействие с другими пользователями через комментарии и оценки. Кроме того, RUTUBE предлагает функционал для проведения прямых трансляций, что делает его привлекательным для создателей живого контента.

Одной из ключевых особенностей видеохостинга является возможность создавать и настраивать собственные каналы. Пользователи могут не только публиковать видео, но и управлять их оформлением, а также настраивать параметры отображения контента. Для авторов доступны инструменты монетизации, включая рекламные интеграции и спонсорскую поддержку, что позволяет получать доход от своего творчества.

Интерфейс RUTUBE отличается интуитивной простотой и удобством. Основные элементы управления расположены в верхней части экрана, обеспечивая быстрый доступ к ключевым функциям. В центральной области отображаются рекомендации, подобранные на основе предпочтений пользователя. Также платформа предлагает выбор тем оформления, позволяя адаптировать внешний вид сайта под индивидуальные предпочтения.

Дизайн RUTUBE выполнен в минималистичном стиле, что способствует комфортному взаимодействию с сервисом. Преобладание белого и синего цветов в оформлении создает ощущение легкости и визуальной чистоты, делая платформу приятной для ежедневного использования. Такой подход позволяет пользователям сосредоточиться на контенте, не отвлекаясь на избыточные элементы интерфейса.

Интерфейс приложения представлен на рисунке 1.2.

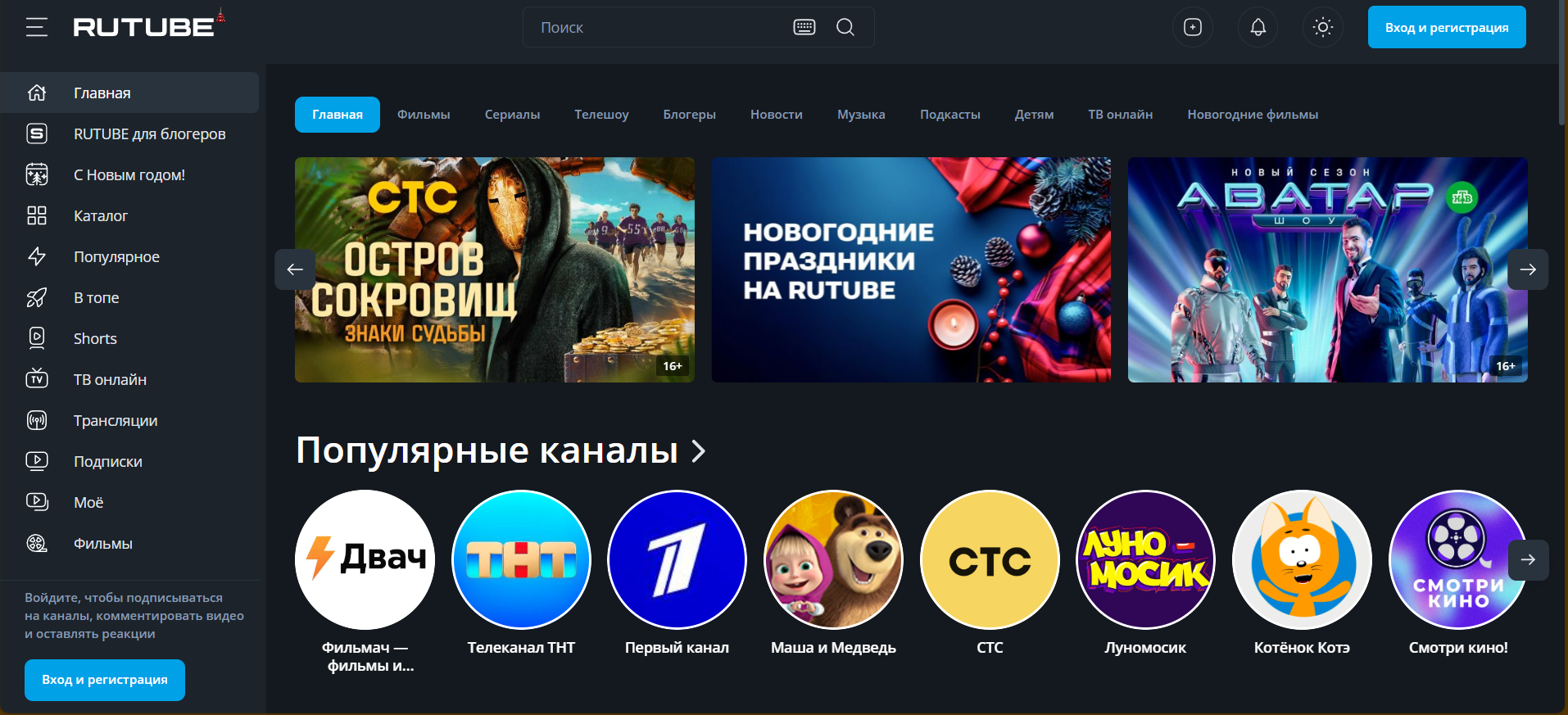


Рисунок 1.2 – Приложение «RUTUBE»

К преимуществам можно отнести обширную библиотеку локального русскоязычного контента, интеграции с кино сервисами, а также удобный интерфейс.

К недостаткам можно отнести большое кол-во рекламного контента, а также слабая инфраструктура.

* + 1. Веб-приложение «VK Video»

VK Video [10] – это мощная платформа для работы с видеоконтентом, предлагающая пользователям разнообразные инструменты для создания, просмотра и взаимодействия с видео. Сервис поддерживает загрузку роликов в различных форматах, а также их удобный просмотр с возможностью регулировки качества. Пользователи могут организовывать видео в тематические плейлисты, подписываться на любимые каналы и выражать свое мнение через комментарии и оценки. Отдельного внимания заслуживает функция прямых трансляций, которая позволяет авторам взаимодействовать с аудиторией в реальном времени.

Одной из ключевых возможностей VK Video является создание и управление персональными каналами. Пользователи могут настраивать оформление, публиковать контент по расписанию и анализировать статистику просмотров. Для авторов предусмотрены инструменты монетизации, включая рекламные интеграции и спонсорскую поддержку, что делает платформу привлекательной для профессиональных создателей контента.

Интерфейс VK Video отличается интуитивной понятностью и продуманной структурой. Основные элементы управления расположены в верхней части экрана, обеспечивая быстрый доступ ко всем важным разделам. Центральная часть страницы отведена под персонализированные рекомендации, которые формируются на основе предпочтений пользователя. Дополнительно платформа предлагает несколько вариантов оформления, позволяя адаптировать дизайн под индивидуальные предпочтения.

Интерфейс приложения представлен на рисунке 1.3.

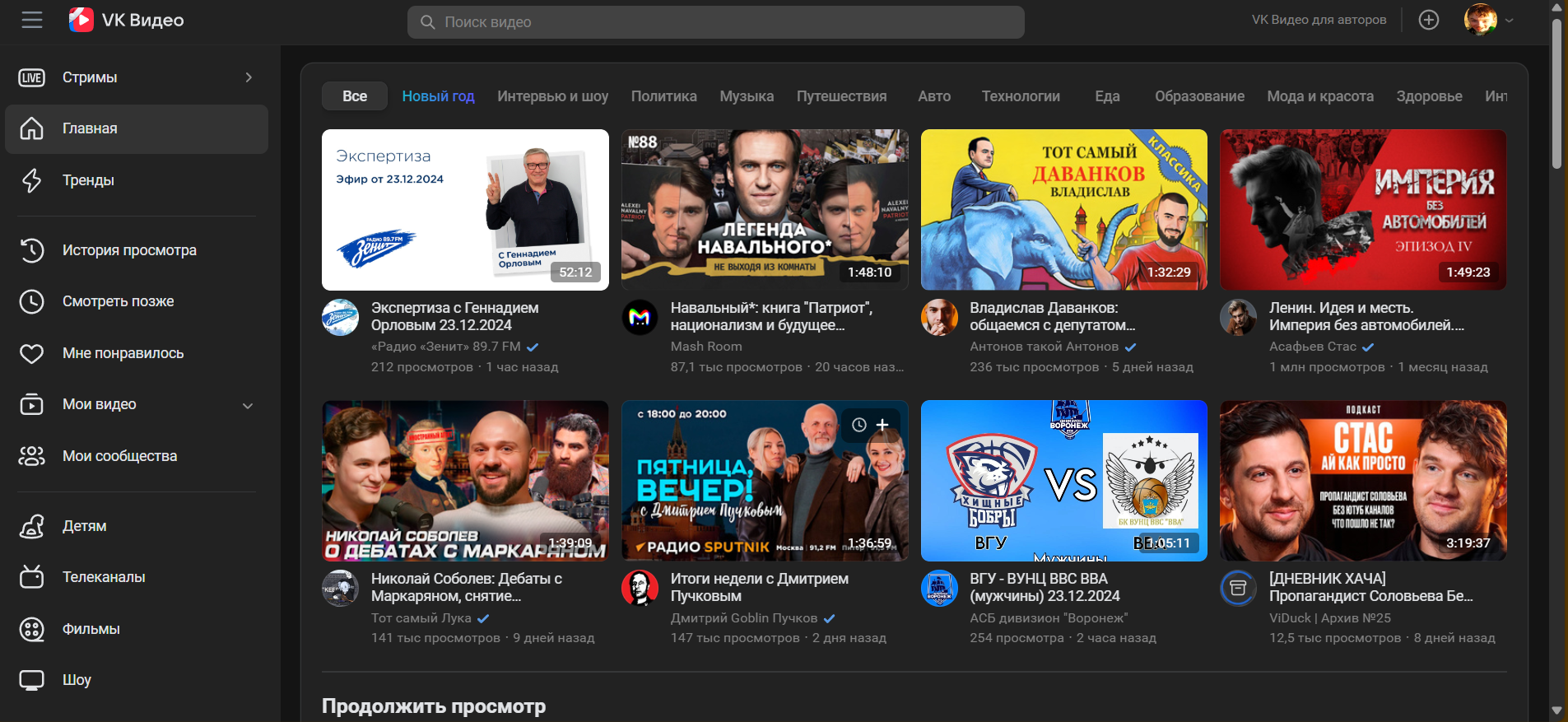


Рисунок 1.3 – Приложение «VK Video»

К преимуществам можно отнести обширную библиотеку локального русскоязычного контента, интеграции с одноименной социальной сетью.

К недостаткам можно отнести большое кол-во рекламного контента, а закрытость платформы что затрудняет взаимодействие зрителя и автора.

* 1. Постановка задачи

Веб-приложение BYTUBE предназначено для просмотра и обмена видео. Оно поддерживает три роли пользователей: Гость, Клиент и Администратор, каждая из которых обладает определенными правами и возможностями.

Все пользователи, независимо от роли, могут сортировать видео, искать их по названию или тегам, просматривать публичный контент и просматривать комментарии под видео.

Гости имеют ограниченный функционал: они могут зарегистрироваться в системе или авторизоваться, чтобы получить доступ к возможностям клиента.

Клиенты обладают широким набором инструментов для взаимодействия с платформой. Они могут создавать и управлять своими каналами, подписываться на других авторов, а также работать с плейлистами создавать их, добавлять или удалять видео, а также воспроизводить сохраненные подборки. Кроме того, клиенты могут загружать собственные видео, редактировать или удалять их, оставлять жалобы на нежелательный контент, а также взаимодействовать с комментариями удалять их под своими видео, оставлять новые или редактировать уже существующие. Одной из интересных возможностей является совместный просмотр видео с другими пользователями.

Администраторы выполняют функции модерации. Они могут скрывать или полностью удалять видео, проверять жалобы от пользователей и удалять комментарии, нарушающие правила платформы.

* 1. Выводы по разделу

В рамках данной главы были поставлены необходимые задачи для реализации веб-приложения. Разобраны три аналогичных решения: «YouTube», «RUTUBE», «VK Video». Из «YouTube» взят основной функционал, из «RUTUBE» взят основной дизайн приложения. Были выявлены преимущества и недостатки аналогичных решений. Было отмечено, что ни один из аналогов не имеет возможности совместного просмотра видео контента, а также что платформы имеют большое количество платного и рекламного контента.

1. Проектирование веб-приложения

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *ДП 02.00.ПЗ* | | | | | |
|  |  |  |  |  |
|  |  | Ф.И.О | Подпись | Дата |
| Разраб | | Окулич Д.Ю. |  |  | *2 Проектирование веб-приложения* | Лит. | | | Лист | Листов |
| Пров. | | Белодед Н.И. |  |  |  | У |  | 1 | 5 |
|  | |  |  |  | БГТУ 1-40 01 01, 2025 | | | | |
| Н. контр. | | Белодед Н.И. |  |  |
| Утв. | | Смелов В.В. |  |  |

* 1. Функциональность веб-приложения

Функциональные возможности веб-приложения представлены в диаграмме вариантов использования, которая приведена в ДП 01.00.ГЧ.

Роли пользователей веб-приложения и их полное описание представлены в таблице 2.1.

Таблица 2.1 – Описание ролей

|  |  |
| --- | --- |
| Роль | Описание |
| Гость | Пользователь, не прошедший регистрацию, с доступом к публичным видео и базовым функциям. |
| Клиент | Зарегистрированный пользователь, имеющий возможность взаимодействовать с контентом. |
| Администратор | Уполномоченный пользователь, отвечающий за модерацию и управление приложением. |

Роли в системе разделены таким образом, чтобы разграничить доступ к функционалу приложения и обеспечить безопасное использование платформы. Каждая роль наделена строго определенным набором возможностей, что упрощает управление и поддержку системы.

Описание функций предоставлена на таблице 2.2.

Таблица 2.2 – описание функций

|  |  |  |  |
| --- | --- | --- | --- |
| № | Функция | Описание | Доступно ролям |
| 1 | 2 | 3 | 4 |
| 1 | Регистрация | Возможность создания нового аккаунта для получения доступа к расширенному функционалу. | Гость |
| 2 | Авторизация | Вход в систему для использования персонализированного функционала. | Гость |
| 3 | Просмотр публичных видео | Возможность смотреть видео, доступные для всех пользователей. | Гость, Клиент, Администратор |
| 4 | Просмотр комментариев под видео | Возможность просматривать комментарии, доступные под видео для всех пользователей. | Гость, Клиент, Администратор |
| 5 | Поиск публичных видео | Возможность поиска видео, доступных для всех пользователей. | Гость, Клиент, Администратор |
| 6 | Поиск по названию | Возможность поиска видео по названию | Гость, Клиент, Администратор |

Продолжение таблицы 2.2

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 7 | Сортировка видео | Возможность сортировать  видео по определенным признакам | Гость, Клиент, Администратор |
| 8 | Поиск по тегам | Возможность поиска видео по тегам | Гость, Клиент, Администратор |
| 9 | Удаление канала | Удаление своего канал пользователем | Клиент, Администратор |
| 10 | Редактирование канала | Редактирование своего канал пользователем | Клиент, Администратор |
| 11 | Создание канала | Создание канал пользователем | Клиент, Администратор |
| 12 | Загрузка видео | Загрузка видео пользователем в рамках канала | Клиент, Администратор |
| 13 | Редактирование данных видео | Редактирование данных видео пользователем в рамках канала | Клиент, Администратор |
| 14 | Удаление видео | Удаление видео пользователем в рамках канала | Клиент, Администратор |
| 15 | Удаление комментариев под своими видео | Удаление любых комментариев автором видео | Клиент, Администратор |
| 16 | Оставление жалобы под видео | Возможность оставить жалобу под видео | Клиент, Администратор |
| 17 | Оставление комментариев под видео | Возможность оставить комментарий под видео | Клиент, Администратор |
| 18 | Удаление своих  комментариев | Возможность удалить свой комментарий | Клиент, Администратор |
| 19 | Редактирование своих комментариев | Возможность редактировать свой комментарий | Клиент, Администратор |
| 20 | Совместный просмотр | Возможность пользователям совместно просматривать видео контент | Клиент, Администратор |
| 21 | Создание плейлистов | Возможность создание плейлиста | Клиент, Администратор |
| 22 | Удаление видео из  плейлиста | Возможность удалять видео из своего плейлиста | Клиент, Администратор |
| 23 | Добавление видео в  плейлиста | Возможность добавлять видео в свой плейлиста | Клиент, Администратор |
| 24 | Воспроизведение  плейлиста | Возможность последовательно воспроизводить видео из плейлиста | Клиент,  Администратор |
| 25 | Удаление плейлиста | Возможность удаления своего плейлиста | Клиент,  Администратор |
| 26 | Удаление любых  комментариев | Возможность удалять любые комментарии | Администратор |
| 27 | Скрытия видео | Возможность скрыть видео администратором | Администратор |
| 28 | Удаление видео | Возможность удаления видео администратором | Администратор |

Продолжение таблицы 2.2

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 29 | Просмотр жалоб  пользователей | Возможность просматривать жалобы пользователей | Администратор |
| 30 | Блокировка канала | Возможность блокировки канала администратором | Администратор |
| 31 | Удаление канала | Возможность удаление канала администратором | Администратор |

Функции приложения разделены между ролями таким образом, чтобы обеспечить удобство использования для каждой категории пользователей. Например, гости могут только просматривать и искать публичные видео, что снижает нагрузку на систему, так как им не нужно хранить данные о своих действиях. Клиенты, в свою очередь, обладают более широким спектром возможностей, включая управление собственным контентом и подписки. Администраторы сосредоточены на обеспечении качества контента и модерации пользовательских жалоб.

Диаграмма вариантов использования позволяет визуализировать все основные функции и роли веб-приложения «BYTUBE». Разделение на роли обеспечивает структурированный доступ к функционалу и улучшает безопасность приложения. Таблицы с описанием ролей и функций дополняют диаграмму, предоставляя детальное представление о возможностях каждой категории пользователей.

* 1. Логическая схема базы данных

Логическая схема базы данных приведена в ДП 02.00.ГЧ. База данных содержит 10 таблиц, таблицы разрабатывались, ориентируясь на реляционную СУБД. Описание таблиц базы данных предоставлено в таблице 2.3.

Таблица 2.3 – Описание таблиц базы данных

|  |  |
| --- | --- |
| Таблица | Описание |
| Users | Хранение информации о пользователях |
| Channels | Хранение информации о каналах |
| Videos | Хранение данных о загруженных видео |
| Comments | Хранение комментариев к видео |
| Playlists | Хранение данных о плейлистах |
| PlaylistItems | Хранение информации о содержимом плейлистов |
| Subscriptions | Хранение информации о подписках пользователей |
| Reports | Хранение жалоб пользователей на видео |
| VideoMarks | Хранение оценок видео |
| VideoViews | Хранение просмотров видео |

Все таблицы разработаны для максимального эффективного хранение данных веб-приложения. Полное описание каждой таблицы предоставлено в приложении А.

* 1. Архитектура веб-приложения

Диаграмма развертывания веб-приложения приведена в ДП 03.00.ГЧ.

Веб-приложение состоит из клиентской и серверной части. Пояснение назначения каждого элемента веб-приложения представлено в таблице 2.4.

Таблица 2.4 – Назначение элементов архитектурной схемы веб-приложения

|  |  |
| --- | --- |
| Элемент | Назначение |
| Client Browser/Edge | Браузер пользователя, через который он взаимодействует с веб-приложением. Обрабатывает и отображает фронтенд, отправляет HTTP-запросы к Backend Server |
| Database Server | Сервер с базой данных. Отвечает за хранение данных |
| Web API Server | Сервер, который обрабатывает запросы от фронтенда, управляет бизнес-логикой приложения, взаимодействует с базой данных |

Описание протоколов, используемых при работе веб-приложений, представлено в таблице 2.5.

Таблица 2.5 – Описание используемых протоколов

|  |  |
| --- | --- |
| Протокол | Назначение |
| HTTPS [11] | Обмен данными между серверной и клиентской части веб-приложения. Обеспечивает безопасную передачу данных путем использования криптографического протокола TLS. |
| TCP [12] | Обмен данными между Database Server и Backend Server. |
| WebSocket [13] | Используется для поточного просмотра видео и работы комнат совместного просмотра. |

Данная архитектура обеспечивает максимально эффективную работу веб-приложения.

* 1. Блок-схема алгоритма подключения к совместному просмотру

Блок-схема подключения к совместному просмотру привалена ДП 04.00.ГЧ.

Данная блок-схема описывает процесс подключения пользователя к комнате для совместного просмотра видео. После создания комнаты к ней подключаются пользователи, каждый из которых проходит проверку на авторизацию. Если пользователь не авторизован, он перенаправляется на главную страницу, в противном случае добавляется в комнату. Далее проверяется, является ли он первым в очереди, если да, ему присваиваются права мастера комнаты. Затем осуществляется проверка, выбрано ли видео для просмотра. Если видео еще не выбрано, остальные участники ожидают действий мастера. После выбора видео происходит синхронизация всех пользователей и завершение процесса.

* 1. Выводы по разделу

Таким образом, в рамках проектирования веб-приложения была разработана диаграмма вариантов использования, благодаря которой был определен необходимый функционал системы, а также установлены основные роли пользователей, включая их права доступа и возможные действия в рамках приложения. На основе выявленных требований была спроектирована архитектура приложения, определена логика взаимодействия между клиентской и серверной частями, выбрана модель размещения компонентов с учетом масштабируемости и отказоустойчивости, а также определены используемые протоколы взаимодействия.

Кроме того, была разработана структура базы данных, включающая основные сущности, такие как пользователи, видео, комментарии, каналы, плейлисты и жалобы. Были определены связи между таблицами, типы данных, индексы и ограничения целостности, что позволило обеспечить консистентность и логическую связанность информации. Также была спроектирована блок-схема алгоритма подключения к функции совместного просмотра видео, описывающая этапы подключения пользователя к комнате в рамках совместного просмотра.

Дополнительно были учтены аспекты безопасности, включая механизмы аутентификации и авторизации с использованием JWT, а также реализация разграничения доступа в зависимости от ролей пользователей. Все это позволило заложить надежную основу для реализации веб-приложения.

Таким образом можно считать, что приложение было спроектировано и готово к реализации.

1. Разработка веб-приложения

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *ДП 03.00.ПЗ* | | | | | |
|  |  |  |  |  |
|  |  | Ф.И.О | Подпись | Дата |
| Разраб | | Окулич Д.Ю. |  |  | *3 Разработка веб-приложения* | Лит. | | | Лист | Листов |
| Пров. | | Белодед Н.И. |  |  |  | У |  | 1 | 31 |
|  | |  |  |  | БГТУ 1-40 01 01, 2025 | | | | |
| Н. контр. | | Белодед Н.И. |  |  |
| Утв. | | Смелов В.В. |  |  |

В этом разделе будет рассмотрен процесс разработки системы, в соответствии с проектными требованиями продукта, описанными в разделе 2 данного дипломного проектирования.

* 1. Разработка серверной части веб-приложения
     1. Используемые библиотеки

Все библиотеки серверной части предоставлены в диаграмме компонентов, которая приведена в ДП 05.00.ГЧ.

Описание библиотек предоставлено в таблице 3.1.

Таблица 3.1 – Серверные зависимости

|  |  |
| --- | --- |
| Библиотека | Описание |
| Microsoft.AspNetCore [14] | Базовые компоненты фреймворка .NET Core. |
| Microsoft.AspNetCore.Session [15] | Библиотека, обеспечивающая работу сессий. |
| Microsoft.AspNetCore.StaticFiles [16] | Библиотека, обеспечивающая работу с файлами и основные веб-функции. |
| Microsoft.AspNetCore.  Authentication.JwtBearer [17] | Библиотека для аутентификации пользователей с использованием JWT. |
| Microsoft.AspNetCore.SignalR.  Common [18] | Поддержка WebSocket-технологии через SignalR для серверной части. |
| Npgsql.EntityFrameworkCore.  PostgreSQL [19] | Библиотека Entity Framework Core. |
| Xabe.FFmpeg [20] | Обертка над FFmpeg для обработки видеофайлов. |
| NAudio [21] | Библиотека для обработки и анализа аудиофайлов. |
| Swashbuckle.AspNetCore.SwaggerGen, SwaggerUI, Annotations [22] | Библиотеки для генерации документации API |
| System.IdentityModel.Tokens.Jwt [23] | Библиотека для работы с JWT-токенами. |

Выбранные библиотеки содержат весь необходимый инструментарий для разработки веб-приложения.

* + 1. Структура проекта серверной части

Для разработки серверной части использовалась монолитная архитектура. сервер состоит из одного приложения, которое отвечает за обработку с всей серверной части.

Для обеспечения упорядоченности серверного кода, было решено отказаться от простейшей монолитной архитектуры в пользу довольно простой N-layer [24] архитектуры, с выделением трех основных слоев: уровень доступа к данным, уровень бизнес-логики, уровень представления.

Описание папок входящих в структуру проекта предоставлено в таблице 3.2.

Таблица 3.2 – Папки входящие в проект

|  |  |
| --- | --- |
| Папка | Описание |
| Assets | Статические данные (изображения и тд.) |
| Controllers | Содержит контроллеры |
| Data | Предназначена для хранения тяжелых данных (изображения, видео) |
| Entity | Содержит реализацию базы данных |
| Exceptions | Содержит исключения |
| Helpers | Содержит бизнес логику, которая не поддерживает в DI [25] |
| Hubs | Содержит хаб для совместного просмотра |
| Middleware | Содержит необходимые middleware |
| Migrations | Содержит автоматически созданные миграции |
| Models | Содержит модели данных для передачи с клиентом |
| Public | Содержит index.html |
| Services | Содержит бизнес логику с поддержкой DI |
| wwwroot | Содержит статические файлы необходимые для работы клиентской части (bundle.js) |

За уровень доступа к данным отвечают папки «Models» и «Entity». За уровень бизнес-логики отвечают папки «Services» и «Helpers». За уровень представления отвечает папка «Controllers».

* + 1. Разработка уровня доступа к данным

На уровне доступа к данным система должна обеспечивать удобный и безопасный способ взаимодействия с данными, гарантируя целостность, производительность и масштабируемость.

Модели данных (или сущности) представляют собой абстракции, которые отражают структуру данных в базе данных. Каждая сущность обычно соответствует таблице в реляционной базе данных, а ее свойства – столбцам этой таблицы. В листинге 3.1 предоставлен пример сущности.

public class Comment {

public Guid Id { get; set; }

public required string Message { get; set; }

public List<Guid> Likes { get; set; } = [];

public required DateTime Created { get; set; }

public User? User { get; set; }

public required Guid VideoId { get; set; }

public Video? Video { get; set; } }

Листинг 3.1 – Сущность Comment

EF автоматически генерирует SQL-запросы на основе моделей данных, обеспечивая их выполнение в базе данных MSSQL. На листинге 3.2 приведен пример получения объектов при помощи репозитория.

Comment comment = await \_commentRepository.GetCommentWithAuthor(guid)

?? throw new ServerException("Комментарий не найден", 404);

Листинг 3.2 – Пример выполнения запроса в базу данных

Паттерн репозитория используется для абстрагирования логики доступа к данным. Он позволяет скрыть детали работы с базой данных от остальной части приложения, предоставляя интерфейс для выполнения операций с сущностями, таких как добавление, удаление, обновление и чтение данных. На листинге 3.3 представлен интерфейс обобщенного репозитория, который определяет ряд методов, позволяющих выполнять различные запросы в базу данных, при их использовании можно абстрагироваться от деталей реализации работы самого фреймворка.

public interface IRepository<T>

{

public T? Get(Guid guid);

public List<T> GetAll();

public void Create(T value);

public void Update(T value);

public void Delete(Guid guid);

}

Листинг 3.3 – Интерфейс IRepository

Таблица с описанием всех сущностей, их типами данных и соответствующими типами данных в SQL представлены в приложении А. В приложении Б – скрипт для создания таблиц.

* + 1. Разработка уровня бизнес-логики

При разработке бизнес-логики для WebAPI на C# одним из ключевых аспектов является правильное использование сервисов и паттернов внедрения зависимостей. Важно грамотно организовать слой бизнес-логики (BL), чтобы он оставался гибким, тестируемым и масштабируемым.

Использование DI позволяет передавать сервисы в контроллеры или другие сервисы, делая их более гибкими и независимыми от конкретной реализации. В C# это достигается через механизм DI, который автоматически управляет созданием объектов и их зависимостями. Контроллеры могут получать сервисы через конструктор, что упрощает тестирование и управление зависимостями в проекте, также при помощи DI в слой бизнес-логики попадают объекты слоя доступа к данным, как например сервис авто удаления комнат, данная логика предоставлена в листинге 3.4.

public WatchTogetherLobbyCleaningService(WatchTogetherLobbyService watchTogetherLobby)

{

\_watchTogetherLobby = watchTogetherLobby;

}

Листинг 3.4 – пример передачи зависимостей через конструктор

Таким образом было организованно использование бизнес-логики.

* + 1. Разработка уровня представления

Разработка уровня представления является важным этапом в создании веб-приложений, так как именно через этот слой взаимодействуют пользователи с системой.

Контроллеры в контексте разработки веб-приложений с использованием ASP.NET Core являются классами, которые обрабатывают HTTP-запросы и отвечают на них соответствующими данными. Эти классы обычно наследуются от базового класса ControllerBase, который предоставляет доступ к набору стандартных методов для обработки запросов. Каждый контроллер связан с определенным маршрутом, который указывает на URL, к которому привязан данный контроллер.

Пример контроллера предоставлен в листинге 3.5.

[Route("api/[controller]")] [ApiController]

public class VideoController : ControllerBase {

private readonly PostgresDbContext \_dbContext;

private readonly LocalDataService \_localData;

private readonly VideoMediaService \_videoMedia;

public VideoController(PostgresDbContext db, LocalDataService localData, VideoMediaService videoMedia){

\_dbContext = db;

\_localData = localData;

\_videoMedia = videoMedia;

}

[HttpGet("view")]

public async Task<IResult> GetViews([FromQuery] Guid id)

{ ... }

}

Листинг 3.5 – Пример контроллера

Контроллеры должны обеспечивать чистоту кода и простоту в тестировании, вся логика обработки данных должна быть вынесена в слой бизнес-логики.

* + 1. Настройка безопасности

Одной из распространенных задач в рамках реализации веб-приложения, в котором поддерживаются пользовательские учетные записи – реализация механизма аутентификации и авторизации, чаще всего используется механизм на основе refresh и access токенов.

Создание токенов обычно осуществляется с использованием библиотеки System.IdentityModel.Tokens.Jwt, которая предоставляет все необходимые инструменты для работы с JSON Web Token (JWT) [26]. Access token представляет собой короткоживущий токен, который используется для предоставления доступа к защищенным ресурсам. Refresh token имеет более длительный срок действия и предназначен для обновления access token, когда срок его действия истек. Это снижает частоту запросов к серверу авторизации и улучшает производительность.

Основными параметрами при создании JWT-токенов являются issuer (издатель токена), audience (аудитория токена), claims (утверждения, содержащие информацию о пользователе), и expiration (время истечения токена). Например, issuer используется для указания сервера, который выдал токен, а audience определяет, для какого ресурса предназначен токен. Claims включают такие данные, как идентификатор пользователя, его роль или другие атрибуты. Expiration задает ограничение по времени, что делает токен недействительным после периода.

Код создание токенов предоставлен в листинге 3.6.

public static string GenerateJwtToken(JwtSettings settings, UserData user) {

var key = new SymmetricSecurityKey(Encoding. .GetBytes(settings.SecretKey));

var claims = new[] {

new Claim(ClaimTypes.UserData, user.Id.ToString()),

new Claim(ClaimTypes.Role, user.Role.ToString()),

new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString()) };

var credentials = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

var tokenDescriptor = new SecurityTokenDescriptor

{

Subject = new ClaimsIdentity(claims),

Expires = DateTime.UtcNow.AddMinutes(settings.ExpiryMinutes),

Issuer = settings.Issuer,

Audience = settings.Audience,

SigningCredentials = credentials

};

var tokenHandler = new JwtSecurityTokenHandler();

var token = tokenHandler.CreateToken(tokenDescriptor);

return tokenHandler.WriteToken(token);

}

Листинг 3.6 – Код создание токенов

Я проверки авторизации пользователей используется атрибут для функций [Authorize] который проверяет пользователя на авторизацию, а также если указать параметр Role можно проверить принадлежность пользователю роли.

Таким образом, настройка безопасности через access и refresh токены позволяет создать эффективную и гибкую систему управления доступом в ASP.NET Core. Это решение обеспечивает высокий уровень защиты, удобство для пользователей и возможность масштабирования приложения.

* + 1. Используемые паттерны

Использование ASP .NET Core в качестве фреймворка для разработки приложений влечет за собой необходимость применения ряда паттернов проектирования:

* dependency Injection (DI) – паттерн, который помогает разделить создание зависимостей и их использование. В ASP.NET Core встроена мощная система DI, которая позволяет инжектировать сервисы, такие как репозитории, логеры или контексты баз данных, прямо в контроллеры и другие компоненты через конструктор. Это способствует лучшей модульности, тестируемости и снижению связности между компонентами системы;
* repository [27] – паттерн для абстракции доступа к данным. Он позволяет скрыть детали работы с хранилищем данных (например, базой данных) и предоставляет простые методы для работы с объектами. В ASP.NET Core репозитории используются для инкапсуляции операций CRUD, что упрощает работу с данными и делает код более читаемым и поддерживаемым. Это позволяет легко заменить или изменить реализацию доступа к данным без затрагивания других компонентов.
  1. Клиентская часть
     1. Используемые библиотеки

Все библиотеки клиентской части предоставлены в диаграмме компонентов, которая приведена в ДП 05.00.ГЧ.

Описание библиотек предоставлено в таблице 3.3.

Таблица 3.3 – Клиентские библиотеки

|  |  |
| --- | --- |
| Библиотеки | Описание |
| react, react-dom, react-router-dom | Базовые библиотеки для построения SPA-интерфейса. |
| mobx, mobx-react-lite | Библиотеки для управления состоянием приложения. |
| @mui/material, @mui/icons-material, @mui/lab | Визуальные компоненты и иконки для создания интерфейса. |
| axios | HTTPS-клиент для общения с сервером |
| Localforage | Библиотека для хранения данных на стороне клиента. |
| match-sorter, sort-by | Библиотеки сортировки и фильтрации. |
| react-player | Компонент для проигрывания мультимедийных файлов. |
| @microsoft/signalr | Библиотека для работы с SignalR. |

Данные библиотеки полностью закрывают необходимости клиентской в API части.

* + 1. Структура проекта клиентской части

Структура проекта организованны таким образом, чтобы все файлы были сгруппированы внутри корневой папки src в директории по их назначению.

В таблице 3.4 приведен список основных директорий проекта и их назначение.

Таблица 3.4 – Директории входящие в проект

|  |  |
| --- | --- |
| Директория | Описание |
| assets | Содержит статические файлы |
| components | Содержит реакт компоненты |
| helpers | Содержит вспомогательные функции |
| hooks | Содержит необходимые хуки |
| stores | Содержит MOBX хранилища |
| styles | Содержит общие стили |
| types | Содержит модели данных |

Данная структура обеспечивает доступ ко всем файлам проекта.

* + 1. Конфигурация проекта

Для настройки и сборки клиентской части веб-приложения был выбран сборщик WebPack, который является одним из самых популярных и мощных сборщиков для JavaScript-приложений. WebPack позволяет настроить процесс сборки таким образом, чтобы эти ресурсы правильно компилировались, объединялись в пакеты и оптимизировались для быстрого загрузки в браузере, а также быстрого настройки прокси, что полезно для отладки.

* 1. Реализация функций
     1. Регистрация

Для регистрации пользователя в рамка контроллера «AuthController» был разработан метод «Register». Для регистрации метод принимает имя пользователя, почту, пароль и файл иконки пользователя. Создается новый пользователь с указанным именем, электронной почтой и захешированным паролем. Пользователь сохраняется в базе данных. Если все прошло хорошо возвращает пользователю код 200 OK.

Реализация регистрации предоставлен в листинге 3.7.

[HttpPost("register")]

public async Task<IResult> Register([FromForm] RegisterModel model)

{

try

{

var usr = await \_db.Users.AddAsync(new()

{

Name = model.UserName,

Email = model.Email,

Password = \_passwordHasher.Hash(model.Password),

Role = Entity.Models.User.RoleType.User,

BirthDay = DateOnly.Parse(model.BirthDay)

});

await \_db.SaveChangesAsync();

await \_localDataManager.SaveUserFiles(usr.Entity.Id, model.ImageFile);

}

catch

{

var errorModel = new ServerErrorModel(400);

errorModel.errors.Add("email", ["Пользователь с такой почтой уже сущетсвует"]);

return Results.Json(errorModel, statusCode: 400);

}

return Results.Ok();

}

Листинг 3.7 – Реализация регистрации

Таким образом была реализована регистрация в веб-приложении.

* + 1. Авторизация

Для авторизации в рамках контроллера «AuthController» был разработан метод «SignIn». Метод обрабатывает POST-запрос по маршруту "signin" и принимает на вход модель «SigninModel», содержащую адрес электронной почты и пароль пользователя.

В начале метода производится поиск пользователя в базе данных по переданному адресу электронной почты. Если пользователь не найден или переданный пароль не совпадает с сохраненным хэшем, генерируется исключение с сообщением о несоответствии пароля.

При успешной проверке пароля для пользователя создаются два JWT-токена: access-token и refresh-token. Access-token добавляется в cookies ответа с предустановленными параметрами хранения, а refresh-token сохраняется как в cookies, так и в поле Token соответствующей записи пользователя в БД.

Далее пользовательская запись обновляется в базе данных, изменения сохраняются, и клиенту возвращается код 200 OK, подтверждающий успешную авторизацию. В случае ошибок, связанных с процессом входа, клиент получает JSON-ответ [28] с описанием проблемы и соответствующим HTTP-кодом.

Листинг реализации метода предоставлен в приложении В.

* + 1. Просмотр публичных видео

Для просмотра публичных видео в рамках контролера «VideoController» было разработано два метода «Get» и «StreamVideo».

Для получения подробной информации о видео в рамках контроллера «VideoController» был реализован метод «Get». Метод обрабатывает GET-запрос и принимает идентификатор видео в формате GUID через параметры запроса.

В начале выполнения метода из контекста HTTP-запроса извлекаются данные авторизации пользователя. Затем осуществляется попытка найти видео по переданному идентификатору в базе данных. Если видео не найдено, генерируется исключение с кодом 404 и соответствующим сообщением. После этого производится выборка канала, к которому принадлежит видео, с предварительной загрузкой связанных с ним подписок.

Если видео имеет статус приватного доступа, дополнительно проверяется, авторизован ли пользователь и совпадает ли его идентификатор с идентификатором владельца канала. При нарушении этих условий генерируется исключение с кодом 403, означающее отсутствие прав доступа к видео. Также метод проверяет, не было ли видео или канал заблокированы. В случае блокировки возвращается исключение с тем же кодом доступа и сообщением о недоступности ресурса.

Если все проверки проходят успешно, извлекаются локальные данные видео и канала, а также подсчитывается количество просмотров видео на основе записей в базе данных. Далее формируется объект модели, содержащий полную информацию о видео, и возвращается клиенту в формате JSON. В случае возникновения логических ошибок, метод возвращает структурированный JSON-ответ с описанием ошибки и соответствующим кодом состояния.

Для потоковой передачи видеофайла в рамках контроллера «VideoController» был реализован метод «StreamVideo». Метод принимает идентификатор видео в формате GUID через маршрут запроса и предназначен для выдачи видеофайла с поддержкой обработки диапазонных HTTP-запросов.

В начале метода из HTTP-контекста извлекаются авторизационные данные пользователя. Затем формируется путь к видеофайлу на локальной файловой системе. Если файл не найден по указанному пути, генерируется исключение с кодом 404 и сообщением о его отсутствии.

Далее из базы данных извлекается объект видео с предварительной загрузкой связанного канала. Проводится проверка доступа: если видео является приватным, доступ к нему разрешается только владельцу канала. Если пользователь не авторизован или его идентификатор не совпадает с идентификатором владельца, выбрасывается исключение с кодом 403. Также проверяется статус видео – в случае, если оно заблокировано, передача файла запрещается с соответствующим кодом отказа в доступе.

Если все проверки пройдены успешно, в заголовки ответа добавляется информация о поддержке диапазонных запросов. Открывается поток к видеофайлу, и определяется его длина. Если клиент указал заголовок Range, метод обрабатывает запрос частичного контента: вычисляется начальная и конечная позиция считывания, поток перемещается на заданную позицию, и клиенту возвращается только указанный диапазон данных. В противном случае клиенту возвращается весь файл. В случае ошибок, связанных с логикой доступа или существованием файла, клиент получает JSON-ответ с кодом и описанием ошибки.

Листинг реализация методов «Get» и «StreamVideo» предоставлен в приложении В.

* + 1. Просмотр комментариев под видео

Для просмотра комментариев под видео в рамках контроллера «CommentController» был реализован метод «GetVideoComments». Метод обрабатывает GET-запрос по маршруту "video" и принимает идентификатор видео в виде строки через параметры запроса.

В начале метода извлекаются авторизационные данные пользователя из текущего HTTP-контекста.

После успешного преобразования идентификатора вызывается репозиторий комментариев, чтобы получить все комментарии, относящиеся к указанному видео. Далее каждый комментарий обрабатывается: извлекаются локальные данные пользователя, оставившего комментарий, а также вычисляются два логических признака. Первый – поставил ли текущий пользователь лайк этому комментарию, второй – является ли пользователь владельцем видео, к которому относится комментарий. Листинг с реализацией метода предоставлен в приложении B.

* + 1. Поиск публичных видео

Для поиска публичных видео в рамках контроллера «VideoController» был разработан метод «Select». Данный метод принимает разные параметры в виде модели «SelectOptions». Описание параметров предоставлено в таблице 3.5.

Таблица 3.4 – Описание параметров модели «SelectOptions»

|  |  |
| --- | --- |
| Параметр | Описание |
| 1 | 2 |
| Ignore | Список видео, которые нужно игнорировать |
| SearchPattern | Паттерн название |
| Take | Длинна одной страницы (нужен для пагинации) |
| Skip | Сколько элементов пропустить |

Продолжение таблицы 3.4

|  |  |
| --- | --- |
| 1 | 2 |
| OnlyUnlimited | Искать только видео без ограничений |
| OnlyAllAges | Искать только видео для всех возвратов |
| Favorite | Поиск по понравившемся видео |
| Subscribes | Поиск видео подписанных каналов |
| AsAdmin | Запрос специальной информации (только для администраторов) |
| OrderBy | Сортировка видео по признаку |

Возвращает массив моделей видео в формате JSON c кодом 200 OK. Реализация метода предоставлена в листинге 3.8.

[HttpGet("select")]

public async Task<IResult> Select([FromQuery] SelectOptions options)

{

var authData = AuthorizeData.FromContext(HttpContext);

var videos = await \_repository.Select(options, authData);

var result = videos.Select(video =>

{ ... });

return Results.Json(result);

}

Листинг 3.8 – реализация метода «Select»

Таким образом разлизаны поиск по названию, по тегам и сортировка видео.

* + 1. Поиск по названию

Реализация поиска по названию аналогичная рассмотренной в главе 3.3.5. Для поиска требуется указать название видео в параметр «SearchPattern». Возвращает массив моделей видео в формате JSON c кодом 200 OK.

* + 1. Сортировка видео

Реализация сортировки видео аналогичная рассмотренной в главе 3.3.5. Для сортировки видео требуется указать тип сортировки видео в параметр «OrderBy». Всего существует несколько видов сортировки: по новым, по старым, по популярности, по количеству жалоб (для администраторов). Возвращает массив моделей видео в формате JSON c кодом 200 OK.

* + 1. Поиск по тегам

Реализация поиска по тегам аналогичная рассмотренной в главе 3.3.5. Для поиска требуется указать теги видео в параметр «SearchPattern». Возвращает массив моделей видео в формате JSON c кодом 200 OK.

* + 1. Удаление канала

Для удаления канала в рамках контроллера «ChannelController» был реализован метод «Delete». Метод доступен только авторизованным пользователям и принимает идентификатор канала в формате GUID через параметры запроса.

В начале метода осуществляется получение авторизационных данных пользователя из контекста HTTP-запроса. Далее производится попытка найти канал по переданному идентификатору с предварительной загрузкой связанных с ним видеозаписей. Если канал не найден, выбрасывается исключение с кодом 404. В случае если идентификатор владельца канала не совпадает с идентификатором текущего пользователя, выбрасывается исключение с кодом 403, указывающее на отсутствие прав на выполнение операции. Таким образом могут удалять видео только владельцы соответствующего канала.

Листинг с реализацией метода «Delete» предоставлена в приложении В.

* + 1. Редактирование канала

Для редактирования канала в рамках контроллера «ChannelController» был разработан метод «Put». Метод помечен атрибутом авторизации и доступен только для аутентифицированных пользователей. Он принимает идентификатор канала в формате строки через параметры запроса и модель «EditChannelModel» используя форму запроса.

Реализация метода «Put» предоставлена в листинге 3.10.

[HttpPut, Authorize]

public async Task<IResult> Put([FromForm] EditChannelModel model, [FromQuery] Guid id) {

var authData = AuthorizeData.FromContext(HttpContext);

Channel? channel = await \_db.Channels.FirstOrDefaultAsync(c => c.Id == guid);

if (channel == null) throw new ServerException("Канал не найден!", 404);

if (channel.UserId != authData.Id)

throw new ServerException("Канал вам не пренадлежит!", 403);

channel.Name = model.Name;

channel.Description = model.Description;

\_db.Channels.Update(channel);

await \_db.SaveChangesAsync();

await \_localDataManager.SaveChannelFiles(guid, model.IconFile, model.BannerFile, true);

return Results.Ok(); } }

Листинг 3.10 – Реализация метола «Put» контроллера «ChannelController»

Внутри метода выполняется извлечение авторизационных данных из контекста запроса. Затем происходит попытка преобразовать строковый идентификатор в формат GUID. Если преобразование не удалось, генерируется исключение с сообщением о некорректном идентификаторе. После этого осуществляется поиск канала в базе данных по заданному GUID.

Если канал не найден, генерируется исключение с кодом 404. В случае если пользователь, выполняющий запрос, не является владельцем канала, генерируется исключение с кодом 403.

* + 1. Создание канала

Для создания канала в рамка контроллера «ChannelController» был разработан метод «Post». При вызове метода происходит создание новой записи канала в базе данных. Канал автоматически связывается с идентификатором текущего пользователя, полученного из контекста авторизации. Для канала устанавливается текущая дата и время создания в формате UTC. После успешного сохранения канала в базу данных метод загружает переданные файлы (иконку и баннер) в файловую систему с помощью сервиса \_localDataManager.

В случае успешного выполнения метод возвращает HTTP-статус 200 OK. Если в процессе выполнения возникает ошибка, она перехватывается блоком catch, и клиент получает ответ с описанием проблемы. Этот метод обеспечивает безопасное создание канала, проверяя авторизацию пользователя и корректно обрабатывая возможные исключения. Реализация метода «Post» предоставлена в листинге 3.11.

[HttpPost, Authorize]

public async Task<IResult> Post([FromForm] CreateChannelModel model){

try{

var authData = AuthorizeData.FromContext(HttpContext);

Channel cur = (await \_db.Channels.AddAsync(new Channel()

{

Name = model.Name,

Description = model.Description,

Created = DateTime.UtcNow,

UserId = authData.Id,

})).Entity;

await \_db.SaveChangesAsync();

await \_localDataManager.SaveChannelFiles(cur.Id, model.IconFile!, model.BannerFile!);

return Results.Ok(); }

catch (Exception err){

return Results.Problem(err.Message);} }

Листинг 3.11 – Реализация метола «Post» контроллера «ChannelController»

Таким образом реализована создание канала клиентом.

* + 1. Загрузка видео

Для загрузки видео в рамках контроллера VideoController реализован метод Post, отвечающий за прием и сохранение видеофайлов, а также сопутствующих метаданных. Метод принимает модель CreateVideoModel, передаваемую в формате формы (multipart/form-data), содержащую информацию о загружаемом видео, такую как название, описание, теги и файлы (видео и его превью). Дополнительно передается параметр channelGuid, указывающий, на какой канал пользователя необходимо загрузить видео.

В начале работы метод выполняет проверку: действительно ли указанный канал принадлежит текущему авторизованному пользователю. Это важно для обеспечения безопасности и предотвращения загрузки видео на чужие каналы. Если проверка проходит успешно, создается новая запись в базе данных, содержащая основную информацию о видео, но еще без физических путей к файлам. Реализация метода для загрузки видео на сервер предоставлен в листинге 3.12.

public async Task SaveVideoFiles(Guid id, IFormFile previewFile, IFormFile videoFile) {

try {

if (!Directory.Exists($"{VideosPath}/{id}"))

Directory.CreateDirectory($"{VideosPath}/{id}");

string previewEx = previewFile.FileName.Split('.').Last();

string videoEx = videoFile.FileName.Split('.').Last();

string previewPath = $"{VideosPath}/{id}/preview.{previewEx}";

string videoPath = $"{VideosPath}/{id}/video.{videoEx}";

using (var stream = new FileStream(previewPath, FileMode.Create))

{

await previewFile.CopyToAsync(stream)!;

}

using (var stream = new FileStream(videoPath, FileMode.Create))

{

await videoFile.CopyToAsync(stream)!;

}

SetVideoData(id, new VideoData()

{

PreviewExtention = previewEx,

VideoExtention = videoEx,

});

}

Catch {

throw new ServerException("Ошибка при сохранение файлов видео", 500);

} }

Листинг 3.12 – Реализация метода для загрузки видео

Если на любом из этапов возникает ошибка (например, при записи файла или создании директории), выбрасывается исключение ServerException с сообщением "Ошибка при сохранении файлов видео" и HTTP-статусом 500, что позволяет клиентской части корректно обработать сбой.

* + 1. Редактирование данных видео

Для редактирования данных видео в рамках контроллера «VideoController» был реализован метод «Put». Метод обрабатывает PUT-запрос и принимает модель «EditVideoModel» через форму, а также идентификаторы видео и канала через параметры запроса.

В начале выполнения метода из HTTP-контекста извлекаются авторизационные данные текущего пользователя. Затем происходит попытка найти видео по переданным идентификаторам видео и канала. Если видео не найдено, генерируется исключение с кодом 404 и соответствующим сообщением. Далее проверяется, принадлежит ли видео пользователю. Если авторизованный пользователь не является владельцем канала, к которому принадлежит видео, возбуждается исключение с кодом 403, указывающее на отсутствие прав на редактирование.

Если все проверки пройдены, значения полей видео обновляются на основе переданной модели: изменяются заголовок, описание, теги, тип доступа и флаг "только для взрослых". В случае, если вместе с моделью был отправлен файл превью, он сохраняется с помощью соответствующего сервиса работы с файлами.

Реализация метода предоставлена в листинге 3.13.

public async Task<IResult> Put([FromForm] EditVideoModel model, [FromQuery] Guid id, [FromQuery] Guid channelId) {

try {

var authData = AuthorizeData.FromContext(HttpContext);

Video video = await \_repository.FindVideoWithChannel(id, channelId)

?? throw new ServerException("Такого видео не существует", 404);

if (await \_dbContext.Channels.AnyAsync(channel => channel.UserId == authData.Id))

throw new ServerException("Видео вам не пренадлежит", 403);

video.Title = model.Title;

video.Description = model.Description;

video.Tags = model.Tags;

video.VideoAccess = model.VideoAccess;

video.ForAdults = model.ForAdults;

if (model.PreviewFile != null)

await \_localData.SaveVideoFiles(video.Id, model.PreviewFile!);

\_dbContext.Videos.Update(video);

await \_dbContext.SaveChangesAsync();

return Results.Ok(); }

catch (ServerException srvErr) {

return Results.Json(srvErr.GetModel(), statusCode: srvErr.Code); } }

Листинг 3.13 – Реализация метода «Put» контроллера «VideoController»

После обновления всех данных объект видео сохраняется в базе данных, и в случае успешного завершения операции клиенту возвращается ответ с кодом 200 OK. При возникновении ошибок возвращается JSON-ответ с кодом ошибки и описанием проблемы.

* + 1. Удаление видео

Для удаления видео в рамках контроллера «VideoController» был разработан метод «Delete». Метод обрабатывает DELETE-запрос и принимает идентификаторы видео и канала через параметры запроса.

Сначала из HTTP-контекста извлекаются данные об авторизованном пользователе. Далее происходит попытка найти видео, соответствующее указанным идентификаторам видео и канала. Если видео не существует, генерируется исключение с сообщением об ошибке и кодом 404.

Затем выполняется проверка прав доступа: метод удостоверяется, что текущий пользователь является владельцем канала, к которому принадлежит видео. В случае несоответствия прав возбуждается исключение с кодом 403, указывающее на то, что пользователь не имеет прав на удаление данного видео.

Реализация метода предоставлена в листинге 3.14.\

[HttpDelete, Authorize]

public async Task<IResult> Delete([FromQuery] Guid id, [FromQuery] Guid channelId)

{

try

{

var authData = AuthorizeData.FromContext(HttpContext);

Video video = await \_repository.FindVideoWithChannel(id, channelId)

?? throw new ServerException("Такого видео не существует", 404);

if (await \_dbContext.Channels.AnyAsync(channel => channel.UserId == authData.Id))

throw new ServerException("Видео вам не пренадлежит", 403);

Directory.Delete($"{LocalDataService.VideosPath}/{id}", true);

\_dbContext.Videos.Remove(video);

await \_dbContext.SaveChangesAsync();

return Results.Ok();

}

catch (ServerException srvErr)

{

return Results.Json(srvErr.GetModel(), statusCode: srvErr.Code);

} }

Листинг 3.14 – Реализация метода «Delete» контроллера «VideoController»

После успешного прохождения проверок происходит удаление связанных с видео файлов из локального хранилища, а сама запись видео удаляется из базы данных. Изменения сохраняются, и клиенту возвращается код 200 OK, подтверждающий успешное удаление. При возникновении исключений клиент получает JSON-ответ с описанием ошибки и кодом ответа.

* + 1. Удаление комментариев под своими видео

Для удаления комментариев под своими видео в рамках контроллера «CommentController» был разработан метод «Delete». Метод обрабатывает DELETE-запрос и принимает идентификатор комментария через параметры запроса.

В начале работы метода из контекста HTTP-запроса извлекаются данные об авторизованном пользователе. Затем выполняется попытка получить комментарий из базы данных вместе с информацией о связанном видео и канале. Если комментарий не найден, генерируется исключение с сообщением об ошибке и кодом 404.

После этого метод проверяет, имеет ли текущий пользователь право на удаление комментария. Удаление разрешается, если пользователь является автором комментария, владельцем канала, под которым размещено видео, либо если он обладает правами администратора. При отсутствии прав возбуждается исключение с кодом 403, сигнализирующее о запрете доступа к операции.

Реализация метода «Delete» предоставлено в листинге 3.15.

[HttpDelete, Authorize]

public async Task<IResult> Delete([FromQuery] Guid id)

{

try

{

var authData = AuthorizeData.FromContext(HttpContext);

Comment comment = await \_commentRepository. GetWithVideoAndChannel(id)

?? throw new ServerException("Комментарий не найден!", 404);

if (comment.UserId != authData.Id

&& authData.Role != Entity.Models.User.RoleType.Admin

&& comment.Video.Channel.UserId != authData.Id)

throw new ServerException("Комментарий вам не пренадлежит", 403);

\_commentRepository.Delete(comment.Id);

await \_commentRepository.SaveChanges();

return Results.Ok();

}

catch (ServerException err)

{

return Results.Json(err.GetModel(), statusCode: err.Code);

}

}

Листинг 3.15 – Реализация метола «Delete» контроллера «CommentController»

Если все проверки пройдены успешно, комментарий удаляется из репозитория, а изменения сохраняются. В случае успешного завершения операции клиенту возвращается ответ с кодом 200 OK. Если в ходе выполнения возникают исключения, клиент получает структурированный JSON-ответ с описанием ошибки и соответствующим статусом.

* + 1. Оставление жалобы под видео

Для отправки жалобы в рамках контроллера «ReportController» был разработан метод «Post». Метод принимает модель «ReportModel», содержащую данные о видеоролике, типе жалобы и ее описании. Внутри метода выполняется проверка корректности переданного идентификатора видео. Если идентификатор невозможно распознать как GUID, выбрасывается исключение «ServerException» с соответствующим сообщением. При успешной валидации создается новый объект жалобы, который сохраняется в базе данных с указанием даты создания.

Реализация метода предоставлена в листинге 3.16.

[HttpPost, Authorize]

public async Task<IResult> Post([FromBody] ReportModel model)

{

try

{

if (!Guid.TryParse(model.VideoGuid, out Guid vguid))

throw new ServerException("id is not correct!");

\_db.Reports.Add(new Report()

{

VideoId = vguid,

Description = model.Description,

Type = model.Type,

Created = DateTime.UtcNow

});

await \_db.SaveChangesAsync();

return Results.Ok();

}

catch (ServerException err)

{

return Results.Json(err.GetModel(), statusCode: err.Code);

}

}

Листинг 3.16 – Реализация метода «Post» контроллера «ReportController»

Метод возвращает код 200 OK в случае успешной обработки жалобы. В случае возникновения ошибки возвращается сериализованный объект ошибки с соответствующим кодом.

* + 1. Оставление комментариев под видео

Для оставления комментариев под видео в рамках контроллера «CommentController» был разработан метод «Post». Метод принимает модель «CommentModel», содержащую текст комментария и идентификатор видеоролика. Внутри метода извлекаются авторизационные данные пользователя из контекста запроса. Затем выполняется проверка корректности идентификатора видео, и при неудаче выбрасывается исключение «ServerException» с соответствующим сообщением. При успешной проверке создается новый комментарий, привязанный к указанному видеоролику и пользователю, после чего сохраняется в базе данных.

Реализация метода предоставлена в листинге 3.17.

[HttpPost, Authorize]

public async Task<IResult> Post([FromBody] CommentModel model)

{

try

{

var authData = AuthorizeData.FromContext(HttpContext);

if (!Guid.TryParse(model.VideoId, out Guid vguid))

throw new ServerException("Video id is not correct!");

await \_commentRepository.CreateAsync(new Comment()

{

Message = model.Message,

VideoId = vguid,

UserId = authData.Id,

Likes = [],

Created = DateTime.Now.ToUniversalTime(),

});

await \_commentRepository.SaveChanges();

return Results.Ok();

}

catch (ServerException err)

{

return Results.Json(err.GetModel(), statusCode: err.Code);

}

}

Листинг 3.17 – Реализация метода «Post» контроллера «CommentController»

Метод возвращает код 200 OK при успешном добавлении комментария. В случае ошибки возвращается сериализованный объект исключения с соответствующим статусом.

* + 1. Удаление своих комментариев

Реализация удаления своих комментариев аналогичная рассмотренной в главе 3.3.15. Возвращает код 200 OK при успешном удалении комментария.

* + 1. Редактирование своих комментариев

Для редактирования своих комментариев в рамках контроллера «CommentController» был разработан метод «Put». Метод принимает модель «CommentModel» с обновленным текстом комментария и идентификатор редактируемого комментария, переданный через параметр запроса. Внутри метода извлекаются авторизационные данные пользователя из контекста HTTP-запроса. Далее происходит попытка получения комментария из базы данных вместе с его связанным видеороликом и каналом. Если комментарий не найден, выбрасывается исключение с кодом 404. В случае, если автор комментария не совпадает с текущим авторизованным пользователем, выбрасывается исключение с кодом 403.

Реализация метода предоставлена в листинге 3.18.

[HttpPut, Authorize]

public async Task<IResult> Put([FromBody] CommentModel model, [FromQuery] Guid id)

{

try

{

var authData = AuthorizeData.FromContext(HttpContext);

Comment comment = await \_commentRepository.GetWithVideoAndChannel(id)

?? throw new ServerException("Комментарий не найден!", 404);

if (comment.UserId != authData.Id)

throw new ServerException("Комментарий вам не пренадлежит", 403);

comment.Message = model.Message;

\_commentRepository.Update(comment);

await \_commentRepository.SaveChanges();

return Results.Ok();

}

catch (ServerException err)

{

return Results.Json(err.GetModel(), statusCode: err.Code);

}

}

Листинг 3.18 – Реализация метода «Put» контроллера «CommentController»

При успешной проверке сообщение комментария обновляется, изменения сохраняются в базе данных, и метод возвращает код 200 OK. При возникновении исключений возвращается сериализованный объект ошибки с соответствующим кодом состояния.

* + 1. Совместный просмотр

Для реализации совместного просмотра был разработан класс хаб «WatchTogetherHub». Данный класс содержит специальные сообщения необходимые для работы совместного просмотра. Класс разработан при помощи библиотеки SignalR предоставляющей функционал для обмена данными при помощью протокола WebSocket. Описание сообщений предоставлено в таблице 3.5.

Таблица 3.5 – Описание сообщений «WatchTogetherHub»

|  |  |
| --- | --- |
| Сообщение | Описание |
| OnDisconnectedAsync | Отключает пользователя от лобби при разрыве соединения. |
| PingTest | Возвращает текущее время в миллисекундах (используется для пинга). |
| Play | Отправляет остальным участникам команду воспроизведения видео. |
| Pause | Отправляет остальным участникам команду паузы видео. |
| Seek | Сообщает остальным участникам о перемотке видео на указанное время. |
| Sync | Синхронизирует видео на указанное время среди участников. |
| RequestSync | Запрашивает у ведущего (мастера) синхронизацию времени воспроизведения. |
| VideoChange | Меняет видео в лобби, если пользователь мастер. |
| SendMessage | Отправляет текстовое сообщение в чат лобби. |
| GetMessages | Отправляет пользователю историю сообщений из чата лобби. |
| JoinToLobby | Присоединяет пользователя к лобби и синхронизирует его с остальными. |
| LeaveTheLobby | Удаляет пользователя из лобби и обновляет данные ведущего. |

Так же был разработан специальный контроллер «WatchTogetherController» для управления комнатами. Описание методов предоставлено в таблице 3.6.

Таблица 3.6 – Описание методов «WatchTogetherController»

|  |  |  |
| --- | --- | --- |
| Метод | URL | Описание |
| GET | /api/WatchTogether/lobbys | Получение списка всех комнат |
| GET | /api/WatchTogether/lobby | Получение конкретного комнаты |
| POST | /api/WatchTogether/lobby | Создание новой комнаты |
| DELETE | /api/WatchTogether/lobby | Удаление комнаты мастером |
| GET | /api/WatchTogether/try | Попытка войти в комнату |
| POST | /api/WatchTogether/pass | Вход в приватное комнату по паролю |

Так же разработан метод компенсации задержки. Клиент посылает сообщение «PingTest» которая является эхо запросом, а также фиксируя время, когда было отправлено сообщение. После получения сообщение сервер отсылает его обратно. Как только клиент получил сообщения обратно он фиксирует время получения сообщения, после чего отнимает от его время отправки получая задержку. Время задержки добавляется к пришедшим командам для компенсации сетевой задержки.

* + 1. Создание плейлистов

Для создания нового плейлиста в рамках контроллера «PlaylistController» был разработан метод «Post». Метод принимает модель «PlaylistModel», содержащую название плейлиста и уровень доступа к нему. Внутри метода извлекаются данные авторизации пользователя из контекста HTTP-запроса. Затем создается новый объект плейлиста, связывающий его с текущим пользователем. После добавления плейлиста в базу данных изменения сохраняются.

Реализация метода предоставлена в листинге 3.19.

[HttpPost, Authorize]

public async Task<IResult> Post([FromBody] PlaylistModel model)

{

try

{

var authData = AuthorizeData.FromContext(HttpContext);

\_db.Playlists.Add(new Playlist()

{

Name = model.Name,

Access = model.Access,

UserId = authData.Id,

});

await \_db.SaveChangesAsync();

return Results.Ok();

}

catch (ServerException err)

{

return Results.Json(err.GetModel(), statusCode: err.Code);

}

}

Листинг 3.19 – Реализация метода «Post» контроллера «PlaylistController»

Метод возвращает код 200 OK при успешном создании плейлиста. В случае возникновения ошибки возвращается сериализованный объект исключения с соответствующим статусом.

* + 1. Удаление видео из плейлиста

Для удаления видео из плейлиста в рамках контроллера «PlaylistController» был разработан метод «RemoveVideoFromPlaylist». Метод принимает два параметра через строку запроса – идентификаторы плейлиста и видео. Внутри метода извлекаются данные авторизации пользователя из контекста HTTP-запроса. Затем выполняется поиск плейлиста и видео в базе данных. Если один из объектов не найден, выбрасывается исключение с кодом 404. Также проверяется, что плейлист принадлежит текущему пользователю, в противном случае генерируется исключение с кодом 403. Далее производится поиск элемента плейлиста, который связывает видео и плейлист. Если такого элемента нет, выбрасывается исключение с кодом 404.

Реализация метода предоставлена в листинге 3.20.

[HttpPost, Authorize]

public async Task<IResult> Post([FromBody] PlaylistModel model)

{

try

{

var authData = AuthorizeData.FromContext(HttpContext);

\_db.Playlists.Add(new Playlist()

{

Name = model.Name,

Access = model.Access,

UserId = authData.Id,

});

await \_db.SaveChangesAsync();

return Results.Ok();

}

catch (ServerException err)

{

return Results.Json(err.GetModel(), statusCode: err.Code);

}

}

Листинг 3.20 – Реализация метода «RemoveVideoFromPlaylist» контроллера «PlaylistController»

При успешной проверке элемент удаляется из базы данных, и изменения сохраняются. Метод возвращает код 200 OK при успешном удалении. В случае ошибок возвращается JSON объект исключения с соответствующим статусом.

* + 1. Добавление видео в плейлиста

Для добавления видео в плейлист в рамках контроллера «PlaylistController» был разработан метод «AddVideoToPlaylist». Метод принимает через параметры запроса идентификаторы плейлиста и видео. Внутри метода извлекаются данные авторизации пользователя из контекста HTTP-запроса. Затем производится поиск плейлиста и видео в базе данных. Если один из объектов не найден, генерируется исключение с кодом 404. Также проверяется, что плейлист принадлежит текущему пользователю, в противном случае выбрасывается исключение с кодом 403. Далее метод получает все элементы плейлиста, чтобы определить порядковый номер нового видео, который будет добавлен последним. После этого создается новая запись, связывающая плейлист и видео, с рассчитанным порядком. Изменения сохраняются в базе данных.

Реализация метода предоставлена в листинге 3.20.

[HttpPost("add"), Authorize]

public async Task<IResult> AddVideoToPlaylist([FromQuery] Guid id, [FromQuery] Guid vid {

try {

var authData = AuthorizeData.FromContext(HttpContext);

Playlist? playlist = await \_db.Playlists.FindAsync(id);

Video? video = await \_db.Videos.FindAsync(vid);

if (playlist == null || video == null)

throw new ServerException("Видео или плейлист не найден", 404);

if (playlist.UserId != authData.Id)

throw new ServerException("Плейлист вам не принадлежит", 403);

var playlistItems = await \_db.PlaylistItems

.Where(PlaylistItem => PlaylistItem.PlaylistId == id)

.ToArrayAsync();

int order = 0;

if (playlistItems.Length > 0)

order = playlistItems.Select(PlaylistItem => PlaylistItem.Order).Max() + 1;

\_db.PlaylistItems.Add(new PlaylistItem()

{

PlaylistId = playlist.Id,

VideoId = video.Id,

Order = order,

});

await \_db.SaveChangesAsync();

return Results.Ok();

}

catch (ServerException err) {

return Results.Json(err.GetModel(), statusCode: err.Code);

} }

Листинг 3.20 – Реализация метода « AddVideoToPlaylist» контроллера «PlaylistController»

Метод возвращает код 200 OK при успешном добавлении видео. В случае ошибок возвращается JSON объект исключения с соответствующим кодом состояния.

* + 1. Воспроизведение плейлиста

Для воспроизведения плейлиста в рамках контроллера «VideoController» был разработан метод «GetPlaylistVideos». Метод принимает идентификатор плейлиста через параметр запроса. Внутри метода извлекаются данные авторизации пользователя из контекста HTTP-запроса. Затем выполняется поиск плейлиста вместе с его элементами, видео и каналами в базе данных. Если плейлист не найден, выбрасывается исключение с кодом 404. В случае, если плейлист имеет приватный доступ, проверяется, что пользователь авторизован и является владельцем плейлиста. При отсутствии авторизации или при попытке доступа не владельцем возвращаются соответствующие ошибки с кодами 401 или 403.

Реализация метода предоставлена в листинге 3.21.

[HttpGet("playlist")]

public async Task<IResult> GetPlaylistVideos([FromQuery] Guid playlistId)

{

try

{

var authData = AuthorizeData.FromContext(HttpContext);

Playlist playlist = await \_repository.GetPlaylistWithVideo (playlistId)

?? throw new ServerException("Плейлист не найден!", 404);

if (playlist.Access == Playlist.AccessType.Private)

{

if (authData.IsAutorize)

{

if (authData.Id != playlist.UserId)

throw new ServerException("Видео плейлиста не доступены!", 403);

}

else

throw new ServerException("Не авторизован!", 401);

}

return Results.Json(playlist.PlaylistItems.Select(item => item.Video).Select(video => { ... }));

}

catch (ServerException err)

{

return Results.Json(err.GetModel(), statusCode: err.Code);

} }

Листинг 3.21 – Реализация метода «GetPlaylistVideos» контроллера «VideoController»

После успешной проверки метод формирует и возвращает в ответе JSON-объект с данными всех видео плейлиста, включая информацию о просмотрах, описании, продолжительности и канале каждого видео.

* + 1. Удаление плейлиста

Для удаления видео в рамках контроллера «PlaylistController» был разработан метод «Delete». Метод принимает идентификаторы видео и канала через параметры запроса. Внутри метода извлекаются данные авторизации пользователя из контекста HTTP-запроса. Затем осуществляется попытка найти видео по его идентификатору и принадлежности к указанному каналу. Если видео не найдено, выбрасывается исключение с кодом 404. Далее проверяется, принадлежит ли канал, в котором размещено видео, текущему пользователю. В случае, если пользователь не является владельцем канала, генерируется исключение с кодом 403. Если все проверки пройдены, физические файлы видео удаляются с диска, после чего соответствующая запись удаляется из базы данных и изменения сохраняются.

Реализация метода предоставлена в листинге 3.22.

[HttpDelete, Authorize]

public async Task<IResult> Delete([FromQuery] Guid id, [FromQuery] Guid channelId)

{

try

{

var authData = AuthorizeData.FromContext(HttpContext);

Video video = await \_repository.FindVideoWithChannel(id, channelId)

?? throw new ServerException("Такого видео не существует", 404);

if (await \_dbContext.Channels.AnyAsync(channel => channel.UserId == authData.Id))

throw new ServerException("Видео вам не пренадлежит", 403);

Directory.Delete($"{LocalDataService.VideosPath}/{id}", true);

\_dbContext.Videos.Remove(video);

await \_dbContext.SaveChangesAsync();

return Results.Ok();

}

catch (ServerException srvErr)

{

return Results.Json(srvErr.GetModel(), statusCode: srvErr.Code);

}

}

Листинг 3.22 – Реализация метода «Delete» контроллера «PlaylistController»

При успешном выполнении возвращается код 200 OK. В случае ошибок возвращается JSON объект исключения с соответствующим кодом состояния.

* + 1. Удаление любых комментариев администратором

Реализация удаления любых комментариев администратором аналогичная рассмотренной в главе 3.3.15. Возвращает код 200 OK при успешном удалении комментария.

* + 1. Скрытия видео администратором

Для скрытия видео администратором в рамках контроллера «VideoController» был разработан метод «ChangeStatusByAdmin». Метод принимает идентификатор видео и новое состояние из перечисления «Status» через параметры запроса. В начале метода извлекаются данные авторизации из контекста запроса. Затем проверяется, обладает ли пользователь правами администратора. В случае отсутствия прав администратора выбрасывается исключение с кодом 403. После этого осуществляется поиск видео по заданному идентификатору, и при отсутствии результата генерируется исключение с кодом 404. Если видео найдено, его статус обновляется на переданный в запросе.

Реализация метода предоставлена в листинге 3.23.

[HttpPut("status"), Authorize(Roles = "Admin")]

public async Task<IResult> ChangeStatusByAdmin([FromQuery] Guid id, [FromQuery] Video.Status status)

{

try

{

var authData = AuthorizeData.FromContext(HttpContext);

if (authData.Role != Entity.Models.User.RoleType.Admin)

throw new ServerException("Вы не администратор!", 403);

var video = await \_dbContext.Videos.FindAsync(id)

?? throw new ServerException("Видео не найдена!", 404);

video.VideoStatus = status;

\_dbContext.Videos.Update(video);

await \_dbContext.SaveChangesAsync();

return Results.Ok();

}

catch (ServerException err)

{

return Results.Json(err.GetModel(), statusCode: err.Code);

}

}

Листинг 3.23 – Реализация метода «ChangeStatusByAdmin» контроллера «VideoController»

Изменения сохраняются в базе данных, и при успешной операции возвращается код 200 OK. В случае возникновения исключений возвращается JSON сообщение об ошибке с соответствующим кодом состояния.

* + 1. Удаление видео администратором

Для удаления видео администратором в рамках контроллера «VideoController» был разработан метод «DeleteByAdmin». Метод принимает идентификатор видео через параметр запроса. В начале метода извлекаются данные авторизации из текущего контекста запроса. После этого выполняется проверка роли пользователя. Если пользователь не является администратором, выбрасывается исключение с кодом 403. Далее осуществляется поиск видео с указанным идентификатором, включая связанную информацию о канале. В случае отсутствия видео генерируется исключение с кодом 404.

Реализация метода предоставлена в листинге 3.24.

[HttpDelete("delete"), Authorize(Roles = "Admin")]

public async Task<IResult> DeleteByAdmin([FromQuery] Guid id)

{

try

{

var authData = AuthorizeData.FromContext(HttpContext);

if (authData.Role != Entity.Models.User.RoleType.Admin)

throw new ServerException("Вы не администратор!", 403);

var video = await \_dbContext.Videos

.Include(i => i.Channel)

.FirstOrDefaultAsync(x => x.Id == id)

?? throw new ServerException("Видео не найдена!", 404);

Directory.Delete($"{LocalDataService.VideosPath}/{id}", true);

\_dbContext.Videos.Remove(video);

await \_dbContext.SaveChangesAsync();

return Results.Ok();

}

catch (ServerException err)

{

return Results.Json(err.GetModel(), statusCode: err.Code);

}

}

Листинг 3.24 – Реализация метода «DeleteByAdmin» контроллера «VideoController»

Если видео найдено, из файловой системы удаляется соответствующая директория, содержащая видеофайлы. Затем видео удаляется из базы данных, и изменения сохраняются. При успешном завершении операции возвращается ответ с кодом 200 OK. В случае ошибок возвращается JSON сообщение об исключении с соответствующим кодом состояния.

* + 1. Просмотр жалоб пользователей администратором

Для просмотра жалоб пользователей в рамках контроллера «ReportController» был разработан метод «GetVideoReports». Метод принимает строковое представление идентификатора видео в качестве параметра запроса. Сначала выполняется проверка корректности переданного идентификатора, и в случае ошибки выбрасывается исключение с кодом 400. Если идентификатор корректен, из базы данных извлекаются все жалобы, связанные с указанным видео. Каждая жалоба преобразуется в модель представления с указанием ее идентификатора, описания, типа, идентификатора видео и времени создания.

Реализация метода предоставлена в листинге 3.25.

[HttpGet("video"), Authorize]

public async Task<IResult> GetVideoReports([FromQuery] string vid)

{

try

{

if (!Guid.TryParse(vid, out Guid guid))

throw new ServerException("id is not correct!");

var reports = await \_db.Reports.Where(r => r.VideoId == guid).ToArrayAsync();

return Results.Json(reports.Select(report => new ReportModel()

{

Id = report.Id.ToString(),

Description = report.Description,

Type = report.Type,

VideoGuid = report.VideoId.ToString(),

Created = report.Created,

}));

}

catch (ServerException err)

{

return Results.Json(err.GetModel(), statusCode: err.Code);

}

}

Листинг 3.25 – Реализация метода « GetVideoReports» контроллера «ReportController»

В результате выполнения метода клиент получает JSON-массив с информацией о найденных жалобах. В случае возникновения исключения возвращается соответствующий ответ с описанием ошибки и кодом состояния.

* + 1. Блокировка канала администратором

Для блокировки канала администратором в рамках контроллера «ChannelController» был разработан метод «SetStatusByAdmin». Метод принимает идентификатор канала в формате guid и новый статус канала типа «Channel.ActiveStatus» через параметры запроса. В процессе выполнения метод извлекает канал из базы данных по переданному идентификатору. Если канал не найден, выбрасывается исключение с сообщением об ошибке. При успешной проверке статус канала изменяется на переданный, после чего изменения сохраняются в базе данных.

Реализация метода предоставлена в листинге 3.26.

[HttpPut("statusbyadmin"), Authorize(Roles = "Admin")]

public async Task<IResult> SetStatusByAdmin([FromQuery] Guid id, [FromQuery] Channel.ActiveStatus status)

{

try

{

var channel = await \_db.Channels.FindAsync(id)

?? throw new ServerException("Канал не найден!");

channel.Status = status;

await \_db.SaveChangesAsync();

return Results.Ok();

}

catch (ServerException err)

{

return Results.Json(err.GetModel(), statusCode: err.Code);

}

}

Листинг 3.26 – Реализация метода «SetStatusByAdmin» контроллера «ChannelController»

Метод доступен только для пользователей с ролью администратора. В случае успешного выполнения возвращается код 200 OK.

* + 1. Удаление канала администратором

Для удаления канала администратором в рамках контроллера «VideoController» был разработан метод «DeleteByAdmin». Данный метод позволяет администратору полностью удалить канал пользователя вместе со всеми его видеоматериалами и файловыми данными. Метод принимает на вход идентификатор канала в формате guid, который передается через параметр запроса.

В начале работы метод осуществляет попытку найти канал в базе данных по указанному идентификатору. При этом производится загрузка связанных с каналом видео, что необходимо для их последующего удаления. Если по переданному идентификатору канал не обнаружен, метод выбрасывает исключение с сообщением «Канал не найден!», которое перехватывается и возвращается клиенту в виде JSON-ответа с соответствующим кодом ошибки.

В случае успешного нахождения канала метод переходит к этапу удаления связанных ресурсов. Для каждого видео, принадлежащего каналу, происходит удаление соответствующего каталога и всех связанных файлов с сервера. Далее информация о видео удаляется из базы данных. После этого метод удаляет директорию самого канала из файловой системы, обеспечивая тем самым полное физическое удаление данных пользователя с сервера.

Реализация метода предоставлена в листинге 3.27.

[HttpDelete("deletebyadmin"), Authorize(Roles = "Admin")]

public async Task<IResult> DeleteByAdmin([FromQuery] Guid id)

{

try

{

var channel = await \_db.Channels

.Include(channel => channel.Videos)

.FirstOrDefaultAsync(channel => channel.Id == id)

?? throw new ServerException("Канал не найден!");

foreach (var video in channel.Videos)

{

Directory.Delete($"{LocalDataService.VideosPath}/{video.Id}", true);

\_db.Videos.Remove(video);

}

Directory.Delete($"{LocalDataService.ChannelsPath}/{id}", true);

\_db.Channels.Remove(channel);

await \_db.SaveChangesAsync();

return Results.Ok();

}

catch (ServerException err)

{

return Results.Json(err.GetModel(), statusCode: err.Code);

}

}

Листинг 3.27 – Реализация метода «DeleteByAdmin» контроллера «VideoController»

На заключительном этапе метод удаляет сам объект канала из базы данных и сохраняет изменения. Таким образом, обеспечивается полная и необратимая очистка всех данных, связанных с удаляемым каналом. Метод защищен авторизацией и доступен исключительно пользователям с ролью администратора. При успешном выполнении операции клиенту возвращается стандартный HTTP-ответ с кодом 200 OK, сигнализирующий об успешном завершении удаления.

* 1. Маршруты для контролеров

Все маршруты контроллеров указаны в таблице, которая предоставлена в приложении Г.

* 1. Выводы по разделу

Таким образом в рамках реализации веб-приложения была разработана база данных с использованием СУБД Postgres 15, обеспечивающая хранение и эффективную обработку данных всех сущностей приложения, включая пользователей, видео, комментарии, плейлисты и другие связанные объекты.

Использован необходимый стек технологий, который состоит из 11 библиотек серверной части и 14 библиотек клиентской части, обеспечивающих надежную архитектуру, масштабируемость и высокую производительность приложения.

Реализован весь основной функционал приложения, такой как: просмотр публичных видео, поиск по видео, регистрация и авторизация пользователей, работа с каналами (создание, просмотр, редактирование), загрузка и управление видео, формирование и управление пользовательскими плейлистами. Также был внедрен функционал для совместного просмотра видео, позволяющий нескольким пользователям синхронно смотреть одно и то же видео с возможностью чата и взаимодействия в реальном времени.

Реализованы все необходимые компоненты на стороне клиента, включая адаптивный интерфейс для различных устройств, навигационные элементы, модальные окна, формы и страницы для работы с контентом, а также компоненты для реализации функций взаимодействия с сервером через REST API.

1. Тестирование веб-приложения

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *ДП 04.00.ПЗ* | | | | | |
|  |  |  |  |  |
|  |  | Ф.И.О | Подпись | Дата |
| Разраб | | Окулич Д.Ю. |  |  | *4 Тестирование веб-приложения* | Лит. | | | Лист | Листов |
| Пров. | | Белодед Н.И. |  |  |  | У |  | 1 | 4 |
|  | |  |  |  | БГТУ 1-40 01 01, 2025 | | | | |
| Н. контр. | | Белодед Н.И. |  |  |
| Утв. | | Смелов В.В. |  |  |

* 1. Функциональное тестирование

Для тестирования функционала были разработаны специальные тесты, которые предоставлены в таблице 4.1.

Таблица 4.1 – Функциональные тесты

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название функции | Описание теста | Ожидаемый результат | Результат |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Регистрация | На главной странице нажать «Вход». Ввести корректные данные. Нажать «Подтвердить». | В таблице БД появилась новая запись. | Успешно |
| 2 | Авторизация | На странице авторизации ввести e-mail и пароль ранее созданного пользователя. Нажать "Подтвердить". | Вход осуществлен. | Успешно |
| 3 | Просмотр публичных видео | Зайти на главную страницу как гость. Убедиться, что отображаются доступные видео. Нажать на видео для просмотра. | Воспроизведение видео | Успешно |
| 4 | Просмотр комментариев под видео | Зайти на страницу любого виде с комментариями. Раскрыть вкладку «комментарии». | Комментарии отобразились | Успешно |
| 5 | Поиск публичных видео | Ввести запрос в строку поиска как гость. Нажать «Поиск». | Результаты поиска соответствуют запросу. | Успешно |
| 6 | Поиск по названию | Ввести название видео в строку поиска как гость. Нажать «Поиск». | Результаты поиска соответствуют запросу. | Успешно |
| 7 | Сортировка видео | Ввести видео в строку поиска как гость. Выбрать вид сортировки. Нажать «Поиск». | Результаты поиска соответствуют запросу. | Успешно |
| 8 | Поиск по тегам | Ввести теги в строку поиска как гость. Нажать «Поиск». | Результаты поиска соответствуют запросу. | Успешно |
| 9 | Удаление канала | Авторизоваться. Зайти на страницу редактирования канала. Нажать кнопку «Удалить». | Канал и его видео были удалены. | Успешно |

Продолжение таблиц 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 10 | Редактирование канала | Авторизоваться. Зайти на страницу редактирования канала. Изменить информацию о канале. Нажать кнопку «Подтвердить». | Информация о канале изменена. | Успешно |
| 11 | Создание канала | Авторизоваться. Нажать кнопку «создать канал». Заполнить необходимую информацию (Название, Описание, иконка и тд.) | Канал создан. | Успешно |
| 12 | Загрузка видео | Авторизоваться. Перейти в «Студию». Нажать «Загрузить видео». Выбрать файл, ввести метаданные (название, описание, теги). Нажать «Подтвердить». | Видео появилось в списке ваших видео. | Успешно |
| 13 | Редактирование данных видео | Перейти в "Студию". Выбрать видео. Нажать «Редактировать». Изменить название/описание/теги. Нажать «Сохранить». | Изменения вступили в силу. | Успешно |
| 14 | Удаление видео | Перейти в "Студию". Выбрать видео. Нажать "Редактирование". Нажать «Удалить». | Видео удалено.ы | Успешно |
| 15 | Удаление комментариев под своими видео | Зайти на страницу своего видео. Выбрать комментарий. Нажать «Удалить». Подтвердить действие. | Комментарий удален. | Успешно |
| 16 | Оставление жалобы под видео | На странице видео нажать «Пожаловаться». Ввести причину. Нажать «Подтвердить». | Ошибок не возникло. | Успешно |
| 17 | Оставление комментариев под видео | Перейти в "Студию". Выбрать видео. Нажать «Редактирование». Нажать «Удалить». | Видео удалено. | Успешно |
| 18 | Удаление своих  комментариев | Зайти на страницу своего видео. Выбрать комментарий. Нажать «Удалить». Подтвердить действие. | Комментарий удален. | Успешно |
| 19 | Редактирование своих комментариев | Зайти на страницу своего видео. Выбрать комментарий. Нажать «Редактировать». Изменить текст. Нажать | Комментарий обновлен. | Успешно |
| 20 | Совместный просмотр | Нажать кнопку совместный просмотр. Заполнить необходимые данные. Нажать кнопку кнопку «Создать комнату». | Комната создана. Другие пользователи могут подключаться. | Успешно |

Продолжение таблиц 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 21 | Создание плейлистов | Зайти в "Плейлисты". Нажать «Создать плейлист». Ввести название. Нажать «Подтвердить». | Плейлист появился в списке плейлистов. | Успешно |
| 22 | Удаление видео из  плейлиста | Зайти в "Плейлисты". Выбрать плейлист, нажать кнопку «Удалить» на выбранном видео. | Видео было удалено из плейлиста. | Успешно |
| 23 | Добавление видео в  плейлиста | Выбрать видео. Нажать кнопку «Добавить в плейлист». Выбрать плейлист. | Видео добавлено в плейлист. | Успешно |
| 24 | Воспроизведение  плейлиста | Зайти в "Плейлисты". Выбрать видео. | Плейлист начал воспроизводиться с выбранного видео. | Успешно |
| 25 | Удаление плейлиста | Зайти в «Плейлисты». Выбрать плейлист. Нажать «Удалить». | Плейлист удален. | Успешно |
| 26 | Удаление любых комментариев администратором | Зайти в админ-панель. Выбрать видео. Перейти в список комментариев. Выбрать комментарий. Нажать «Удалить». | Комментарий удален. | Успешно |
| 27 | Скрытия видео администратором | Зайти в админ-панель. Выбрать видео. Нажать «Скрыть». | Видео не доступно для просмотра | Успешно |
| 28 | Удаление видео администратором | Зайти в админ-панель. Выбрать видео. Нажать «Удалить». Подтвердить действие. | Видео удалено. | Успешно |
| 29 | Просмотр жалоб  пользователей администратором | Зайти в админ-панель. Выбрать видео. Перейти в раздел жалоб. | Жалобы отображаются. | Успешно |
| 30 | Блокировка канала администратором | Зайти в админ-панель. Выбрать канал. Нажать «Скрыть». | Канал и его видео не доступны для просмотра | Успешно |
| 31 | Удаление канала администратором | Зайти в админ-панель. Выбрать канал. Нажать «Удалить». Подтвердить действие. | Канал и его видео удалены. | Успешно |

Таким образом был покрыт весь основной функционал.

* 1. Автоматизированное тестирование

Для автоматизированного теста были разработаны специальные тесты, которые предоставлены в таблице 4.2.

Таблица 4.2 – Автоматизированные тесты

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название теста | Описание | Тест пройден |
| 1 | PasswordHasherTest0 | Тестирование класса PasswordHasher на корректные данные. | Да |
| 2 | PasswordHasherTest1 | Тестирование класса PasswordHasher на корректные данные. | Да |
| 3 | JwtManagerTest0 | Тестирование генерации исправного access токена. | Да |
| 4 | JwtManagerTest1 | Тестирование генерации исправного refresh токена. | Да |
| 5 | VideoMediaServiceTest | Тестирование работоспособности утилиты ffmpeg. | Да |

Таким образом был протестирован компоненты веб-приложения. Листинг тестов предоставлен в приложении Д.

* 1. Нагрузочное тестирования

Для проверки стабильности сервера были разработаны специальные нагрузочные тесты, которые предоставлены в таблице 4.3.

Таблица 4.3 – Нагрузочные тесты

|  |  |  |
| --- | --- | --- |
| № | Описание | Результат |
| 1 | Массовое воспроизведение видео. Проверить производительность системы при одновременном просмотре видео большим числом пользователей. | Никаких сбоев |
| 2 | Массовое добавление комментариев к видео. Проверить устойчивость при большом количестве комментариев. | Никаких сбоев |
| 3 | Одновременный просмотр и модерация администратором. Проверить производительность системы при одновременном использовании функций просмотра и модерации. | Никаких сбоев |

Таким образом веб-приложение было успешно протестировано на стрессоустойчивость.

* 1. Выводы по разделу

В результате тестирования веб-приложение BYTUBE показало высокий уровень надежности, устойчивости и готовности к работе в реальных условиях эксплуатации. Выявленные дефекты были оперативно устранены, а результаты нагрузочного тестирования доказали способность системы справляться с большим числом пользователей и операций одновременною. Таким образом покрытость тестами составила около 80%. Это обеспечивает уверенность в том, что приложение удовлетворит ожидания пользователей, обеспечивая стабильность и удобство работы.

1. Руководство пользователя

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *ДП 05.00.ПЗ* | | | | | |
|  |  |  |  |  |
|  |  | Ф.И.О | Подпись | Дата |
| Разраб | | Окулич Д.Ю. |  |  | *5 Руководство пользователя* | Лит. | | | Лист | Листов |
| Пров. | | Белодед Н.И. |  |  |  | У |  | 1 | 14 |
|  | |  |  |  | БГТУ 1-40 01 01, 2025 | | | | |
| Н. контр. | | Белодед Н.И. |  |  |
| Утв. | | Смелов В.В. |  |  |

* 1. Регистрация

Что бы перейдите на форму регистрации требуется нажать кнопку «Регистрация» на главной странице веб-приложения. Скриншот главной страницы приведен в ДП 06.00.ГЧ.

Для создание учетной записи необходимо заполнить форму регистрации. Форма представлена на рисунке 5.1.

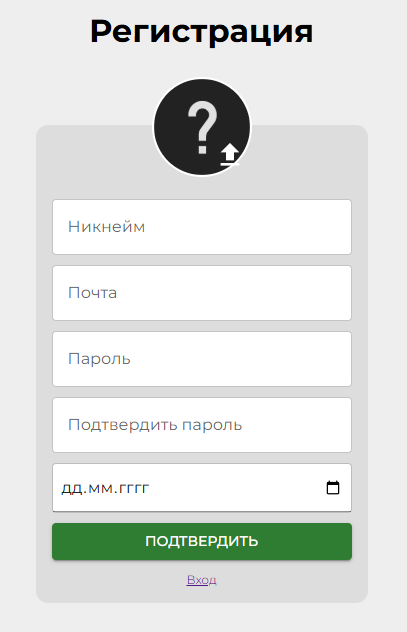


Рисунок 5.1 – Форма регистрации

На данной форме необходимо заполнить следующие поля:

* никнейм – название пользователя в веб-приложении;
* почта – почта пользователя;
* пароль – придумайте надежный пароль, который будите использовать для входа в свой аккаунт;
* подтвердить пароль – повторите пароль, написанный в предыдущем поле;
* дата рождения – укажите вашу дату рождения что бы система понимала доступен ли вам некоторый контент.

Так же есть возможность указать свои иконку профиля, которая будет отображаться у других пользователей.

После заполнения нужно нажать кнопку «ПОДТВЕРДИТЬ». Если все данные введены корректно, то произойдет регистрация в веб-приложения.

* 1. Авторизация

Чтобы перейдите на форму регистрации требуется нажать кнопку «Регистрация» на главной странице веб-приложения. Скриншот главной страницы приведен в ДП 06.00.ГЧ.

Для авторизации нужно заполнить форму авторизации, которая состоит из двух частей. Часть формы с регистрацией предоставлена на рисунке 5.2.

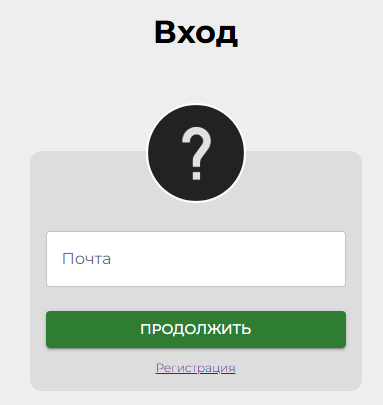


Рисунок 5.2 – Первая часть формы авторизации

На данной части формы необходимо заполнить поле «Почта» после чего нажать кнопку «ПРОДОЛЖИТЬ». Если данные введены верно, то вас переведет на вторую часть формы. Вторая часть формы регистрации предоставлена на рисунке 5.3.

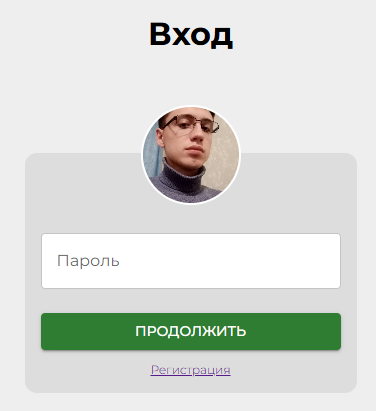


Рисунок 5.3 – Вторая часть формы авторизации

На данной части формы необходимо заполнить поле «Пароль» после чего нажать кнопку «ПРОДОЛЖИТЬ». Если данные введены верно, то авторизация будет произведена и вас перенаправить на главную страницу.

* 1. Просмотр публичных видео.

Чтобы посмотреть публичное видео требуется выбрать любое видео на главной странице. Скриншот главной страницы приведен в ДП 06.00.ГЧ. После чего вас перенаправит на страницу выбранного видео. Скриншот страницы видео предоставлена на рисунке 5.4.

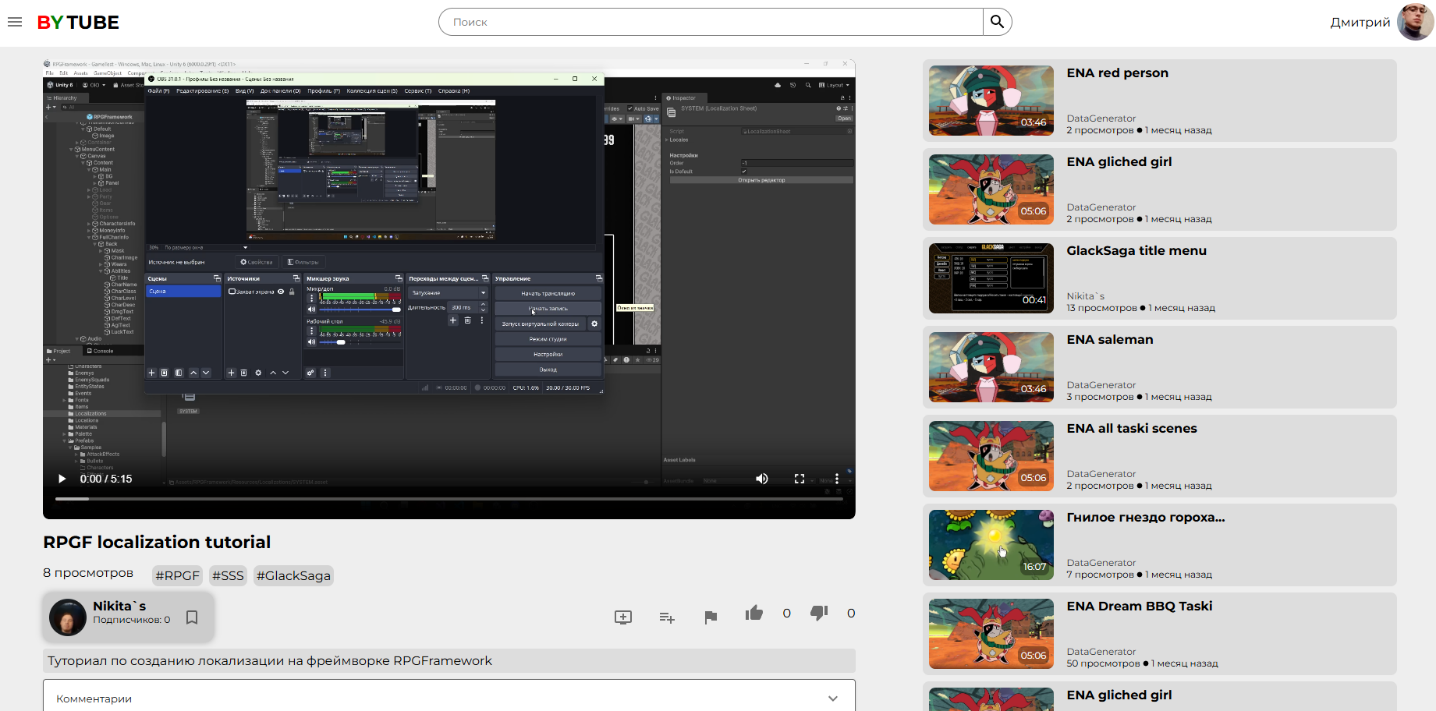


Рисунок 5.4 – Страница для просмотра видео

На данной странице вам предоставлен весь необходимый функционал для просмотра видео.

* 1. Просмотр комментариев под видео

Чтобы посмотреть комментарии под видео требуется перейти на страницу видео. Скриншот страницы видео предоставлена на рисунке 5.4. Далее требуется раскрыть меню «Комментарии», скриншот предоставлен на рисунке 5.5

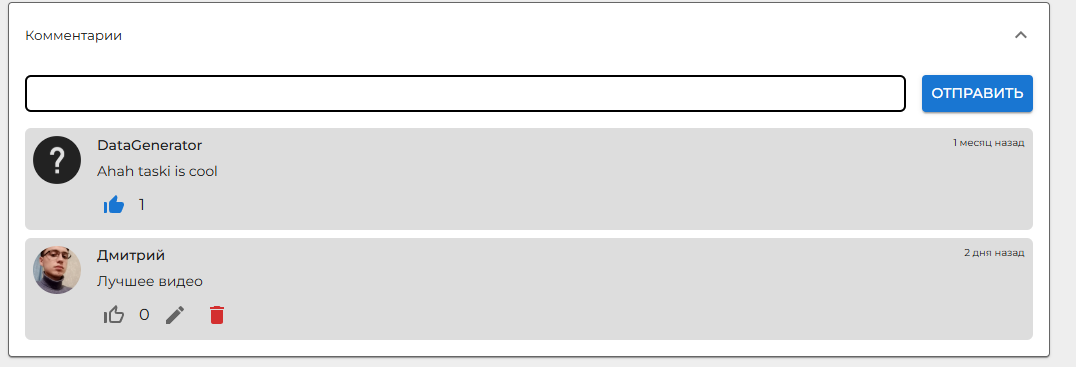


Рисунок 5.5 – Меню с комментариями

В данном меню отображаются все комментарии, которые написаны под выбранным видео.

* 1. Поиск публичных видео

Чтобы искать видео требуется на главной странице заполнить поле «Поиск», вы можете ввести либо название, либо теги. Скриншот главной страницы приведен в ДП 06.00.ГЧ.

* 1. Поиск по названию

Чтобы искать видео по названию вы должны заполнить поле «Поиск» на главной странице названием желаемого видео. Скриншот главной страницы приведен в ДП 06.00.ГЧ.

* 1. Сортировка видео

Что бы сортировать видео вы должны произвести поиск что разморено в главе 5.6. После поиска вас перенаправит на страницу с результатом поиск. Скриншот страницы с результатом поиска предоставлен на рисунке 5.6.

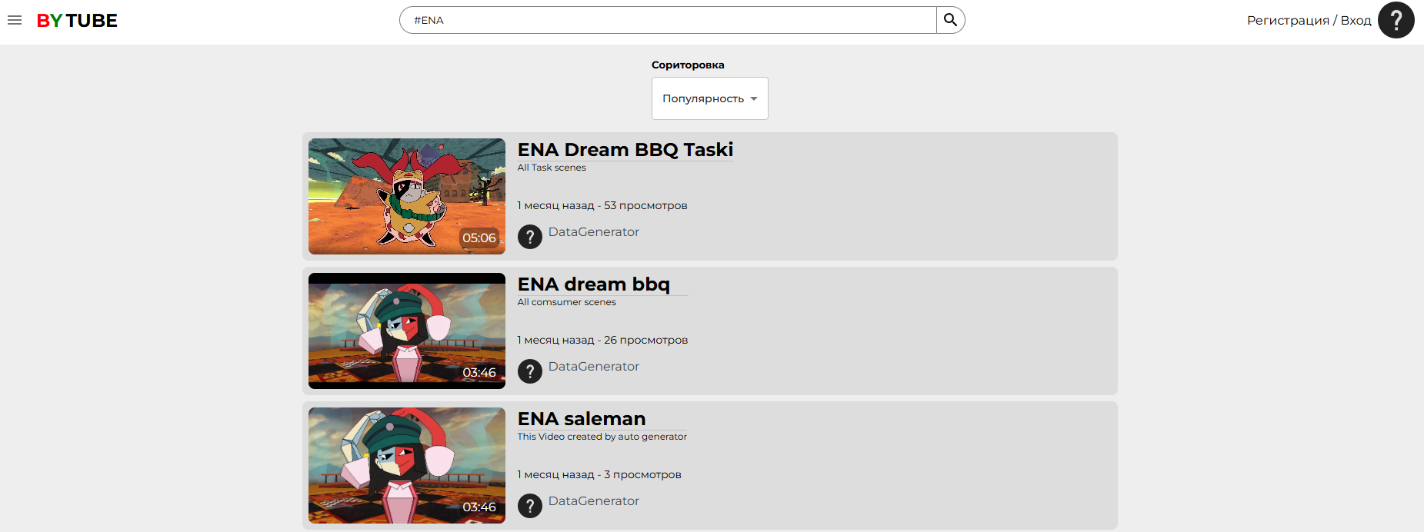


Рисунок 5.6 – Страница с результатом поиска

Чтобы изменить вариант сортировки требуется изменить значение поля «Сортировка».

* 1. Поиск по тегам

Чтобы искать видео по тегу вы должны заполнить поле «Поиск» на главной странице тегами для поиска. Все теги начинаются с символа решетки (#) и разделяются пробелами. Скриншот главной страницы приведен в ДП 06.00.ГЧ.

* 1. Удаление канала

Чтобы удалить канала требуется быть авторизованным в веб-приложении, а также иметь существующий канал, который вам принадлежит. Требуется перейти из главной страницы на страницу студии в настройки вашего канала. Скриншот страницы настроек канала предоставлен на скриншоте 5.7.

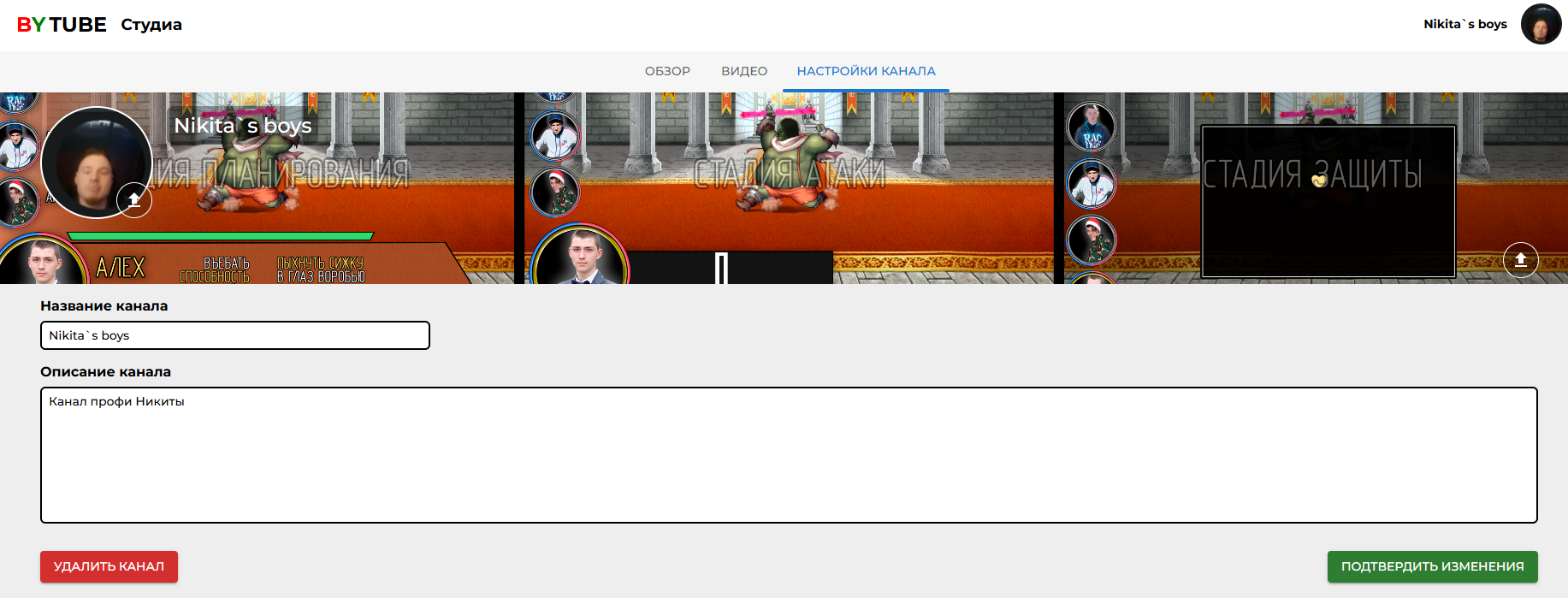


Рисунок 5.7 – Страница с настройками канала

Для того что бы удалить канала требуется нажать кнопку «УДАЛИТЬ КАНАЛ», после чего все данные о вашем канала и сам канал будут удалены.

* 1. Редактирование канала

Чтобы редактировать канал требуется быть авторизованным в веб-приложении, а также иметь существующий канал, который вам принадлежит. Требуется перейти из главной страницы на страницу студии в настройки вашего канала. Скриншот страницы настроек канала предоставлен на скриншоте 5.7.

Вы можете редактировать следующие параметры канала: «Название канала», «Описание канала», «Иконку», «Шапку». Редактируются они через предоставленные одноименные поля.

После редактирование выбранных пунктов надо нажать кнопку «ПОДТВЕРДИТЬ ИЗМЕНЕНИЯ». Если данные корректные, то данные обновятся.

* 1. Создание канала

Чтобы редактировать канал требуется быть авторизованным в веб-приложении. На главной странице требуется открыть левое меню нажав на кнопку с тремя полосками. Скриншот главной страницы приведен в ДП 06.00.ГЧ. После чего нажать кнопку «Создать канал», которая откроет форму для создания канала. Скриншот формы создания канала предоставлена на рисунке 5.8

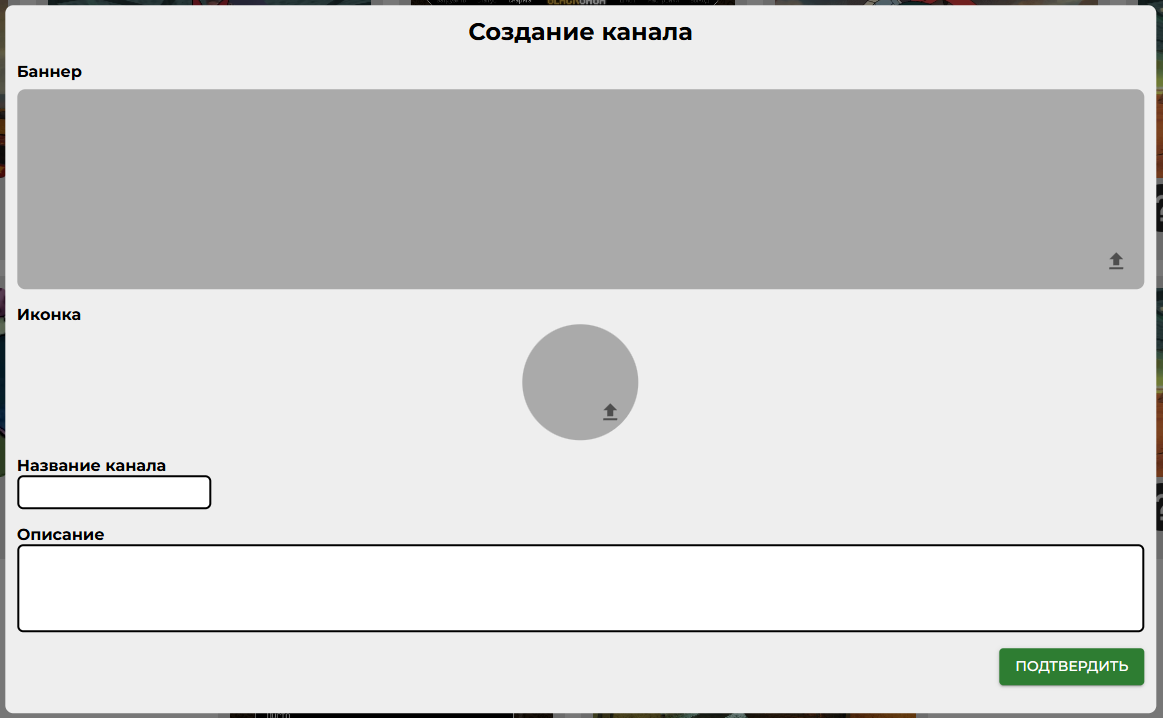


Рисунок 5.8 – Форма создания канала

На этой форме необходимо заполнить следующие поля:

* баннер (Шапка) – изображение фона канала;
* иконка – отображаемая иконка канала;
* название канала – отображаемое название канала;
* описание – дополнительная информация о канале.

После заполнения данных требуется нажать кнопку «ПОДТВЕРДИТЬ». Если данные были введены корректно, то канал будет создан.

* 1. Загрузка видео

Для загрузки видео требуется быть авторизованным в веб-приложении, а также иметь существующий канал. Требуется перейти из главной страницы на страницу студии в список видео вашего канала. Скриншот страницы со списком видео канала предоставлено на рисунке 5.9.

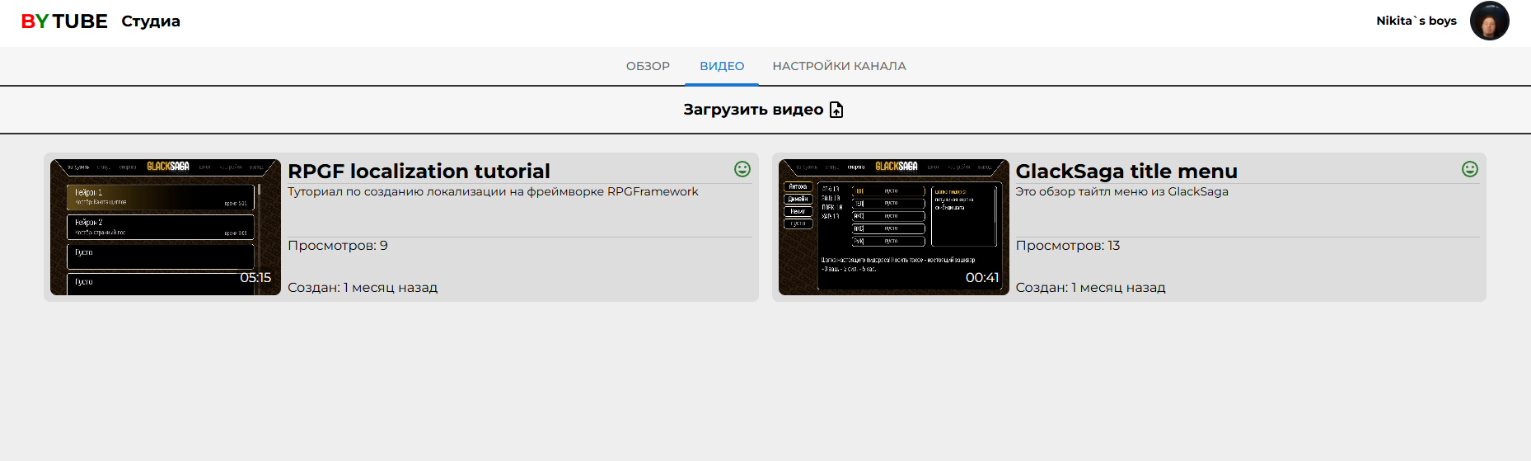


Рисунок 5.9 – Страница со списком видео канала

Чтобы перейти на страницу загрузки видео требуется нажать кнопку «Загрузить видео». Скриншот формы загрузки видео предоставлена на рисунке 5.10.

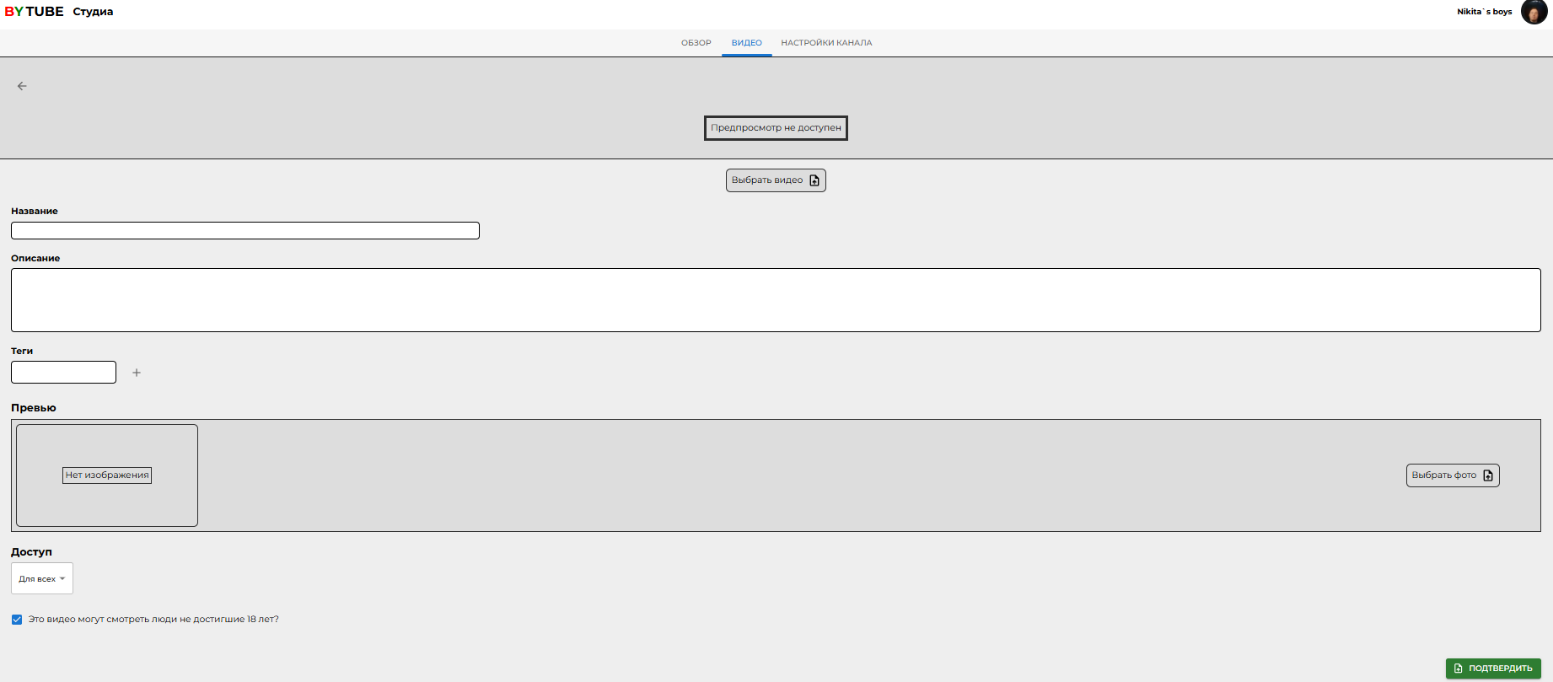


Рисунок 5.10 – Страница со формой загрузки видео

На этой форме необходимо заполнить следующие поля:

* название – отображаемое название видео;
* описание – отображаемое описание видео;
* теги – набор тегов для улучшенного поиска видео;
* превью – изображение обложка;
* доступ – возможность ограничить доступ к видео;

Обязательным является указать видео файл, который будет загружен на сервер. Также можно указать возвратное ограничение. Если все данные введены корректно тогда видео будет загружено на сервер.

* 1. Редактирование данных видео

Для редактирования данных видео требуется быть авторизованным в веб-приложении, а также иметь существующий канал. Требуется перейти из главной страницы на страницу студии в список видео вашего канала. Скриншот страницы со списком видео канала предоставлено на рисунке 5.9.

Требуется выбрать видео, которые будет редактироваться. После чего выбрать вкладку «Редактирование», где будет форма редактирования данных видео. Скриншот формы редактирования предоставлен на рисунке 5.11.



Рисунок 5.11 – Страница со формой редактирования видео

Форма аналогичная форме загрузки видео, рассмотренной в главе 5.12. После заполнения данных требуется нажать кнопку «ПОДТВЕРДИТЬ». Если данные введены верно, то данные видео будут изменены.

* 1. Удаление видео

Для редактирования данных видео требуется быть авторизованным в веб-приложении, а также иметь существующий канал. Требуется перейти из главной страницы на страницу студии в список видео вашего канала. Скриншот страницы со списком видео канала предоставлено на рисунке 5.9.

Требуется выбрать видео, которые будет редактироваться. После чего выбрать вкладку «Редактирование», где будет форма редактирования данных видео. Скриншот формы редактирования предоставлен на рисунке 5.11.

Что бы удалить выбранное видео требуется нажать кнопку «УДАЛИТЬ», после чего видео будет удалено.

* 1. Удаление комментариев под своими видео

Для удаления комментариев под своими видео требуется быть авторизованным в веб-приложении, а также иметь существующий канал. Требуется перейти из главной страницы на страницу вашего видео. Требуется раскрыть меню с комментариями, скриншот предоставлен на рисунке 5.5. Что бы удалить желаемый комментарий требуется нажать кнопку «Удалить» на комментарии. После чего комментарий будет удален.

* 1. Оставление жалобы под видео

Для оставления жалобы под видео требуется быть авторизованным в веб-приложении и требуется перейти на страницу выбранного видео. Пример со страницей видео предоставлен на рисунке 5.4. Что бы оставить жалобу надо нажать кнопку «Пожаловаться» и выбрать причину жалобы.

* 1. Оставление комментариев под видео

Для удаления своих комментариев под видео требуется быть авторизованным в веб-приложении. Требуется перейти из главной страницы на страницу видео. Требуется раскрыть меню с комментариями, скриншот предоставлен на рисунке 5.5. Что бы оставить комментарий требуется заполнить полей «Комментарий» после чего нажать кнопку «ОТПРАВИТЬ». Если данные введены верно, то комментарий будет добавлен.

* 1. Удаление своих комментариев

Для удаления своих комментариев под видео требуется быть авторизованным в веб-приложении. Требуется перейти из главной страницы на страницу видео. Требуется раскрыть меню с комментариями, скриншот предоставлен на рисунке 5.5. Что бы удалить желаемый комментарий требуется нажать кнопку «Удалить» на свой комментарии. После чего комментарий будет удален.

* 1. Редактирование своих комментариев

Для удаления своих комментариев под видео требуется быть авторизованным в веб-приложении. Требуется перейти из главной страницы на страницу видео. Требуется раскрыть меню с комментариями, скриншот предоставлен на рисунке 5.5. Что бы редактировать свой комментарий требуется нажать кнопку «Редактировать» и отредактировать появившиеся текстовое поле заполнить, после чего нажать кнопку «ПОДТВЕРДИТЬ». Если данные введены верно, то комментарий будет обновлен.

* 1. Совместный просмотр

Для совместного просмотра требуется быть авторизованным в веб-приложении. Для перехода на страницу с списком доступных лобби требуется нажать кнопку «Совместный просмотр» на боковой панели. Скриншот страницы с писком комнат предоставлен на рисунке 5.12.

Вы можете, как и создать свою комнату так и подключиться к уже существующей.

Для создание своей комнаты необходимо заполнить форму слева которая состоит из двух полей: «название комнаты», «пароль». Пароль необходим если требуется ограничить доступ к комнате. После заполнения полей требуется нажать кнопку «СОЗДАТЬ», если данные введены верно, то комната будет создана и вас перенаправит на страницу совместного просмотра.

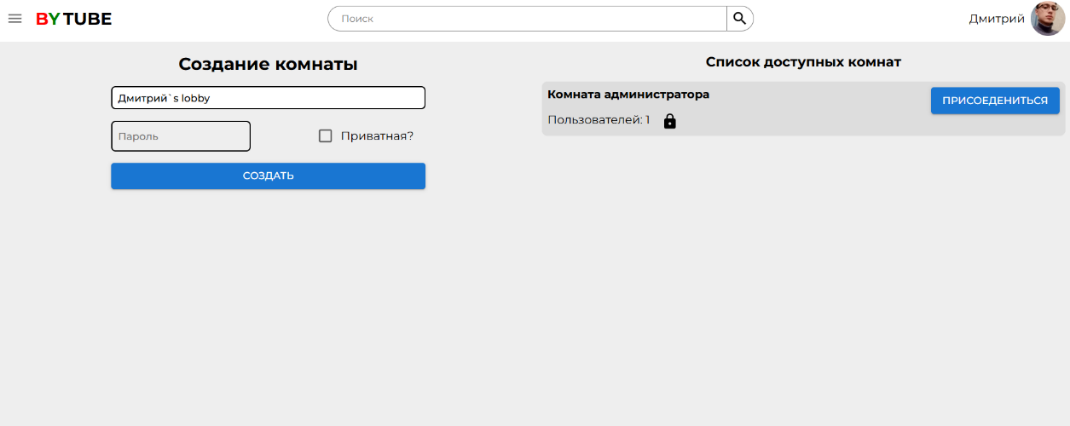


Рисунок 5.12 – Страница со списком комнат

Для подключения к уже существующей комнате требуется нажать кнопку «ПОДКЛЮЧИТЬСЯ» на выбранной комнате, однако если комната приватная, то будет необходимо ввести пароль в появившимся поле. Если данные введены верно, то комната будет создана и вас перенаправит на страницу совместного просмотра.

Скриншот страницы совместного просмотра предоставлена на рисунке 5.13.

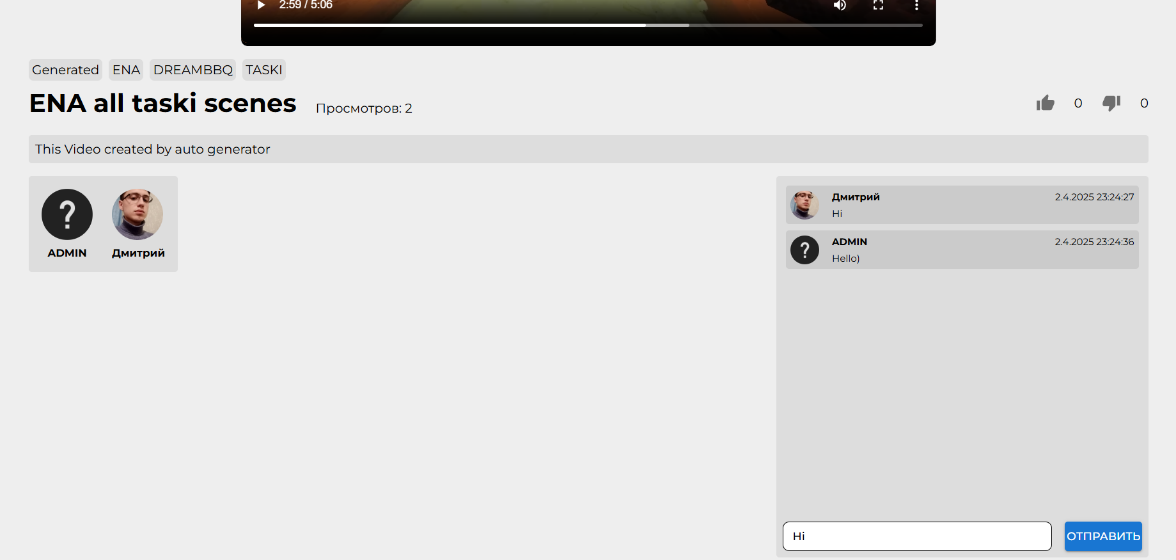


Рисунок 5.13 – Страница совместно просмотра

На странице совместного просмотра вам предоставляется возможность перемотки, воспроизведения, остановки видео, которое отразиться на все, кто находиться в данной комнате. Так же вам доступен чат для того, чтобы общаться с другим пользователями.

* 1. Создание плейлистов

Для создания плейлиста требуется быть авторизованным в веб-приложении. Для перехода на форму создания плейлиста требуется нажат кнопку «Создать плейлист» на боковой панели. Скриншот формы создания плейлиста предоставлен на рисунке 5.14.

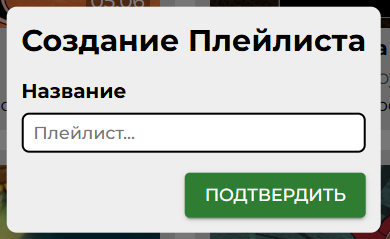


Рисунок 5.14 – Форма создания плейлиста

На данной форме необходимо заполнить название плейлиста, после чего нажать кнопку «ПОДТВЕРДИТЬ». Если данные введены верно, то плейлист будет создан.

* 1. Удаление видео из плейлиста

Для удаления видео из плейлиста требуется быть авторизованным в веб-приложении, а также иметь плейлист. Для перехода в меню плейлиста требуется на боковой панели выбрать желаемый плейлист. Скриншот меню плейлиста предоставлен на рисунке 5.15.

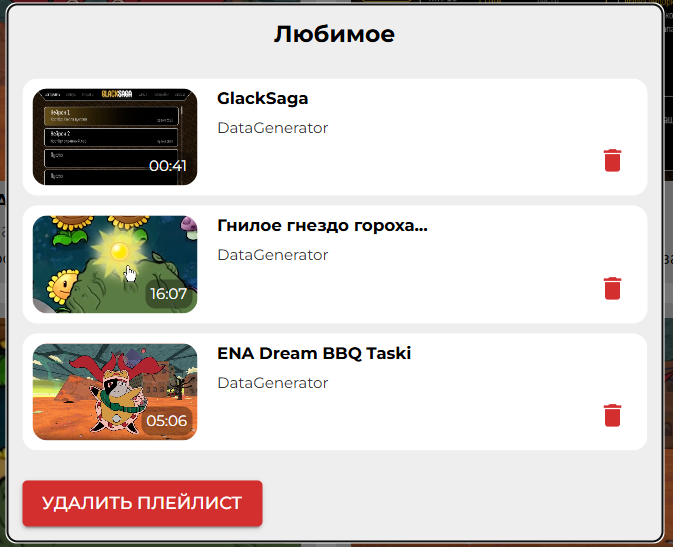


Рисунок 5.15 – Меню плейлиста

Для удаления видео надо нажать кнопку «Удалить» на желаемом видео.

* 1. Добавление видео в плейлиста

Для добавления видео в плейлиста требуется быть авторизованным в веб-приложении, а также иметь плейлист. Перейдите на страницу желаемого видео. Скриншот страницы видео предоставлен на рисунке 5.4. Для добавления видео в плейлист требуется нажать кнопку «Добавить в плейлист» после чего выбрать желаемый плейлист.

* 1. Воспроизведение плейлиста

Для воспроизведения плейлиста требуется быть авторизованным в веб-приложении, а также иметь плейлист. Для перехода в меню плейлиста требуется на боковой панели выбрать желаемый плейлист. Скриншот меню плейлиста предоставлен на рисунке 5.15. Для воспроизведения нажмите на видео, с которого начнется воспроизведение.

* 1. Удаление плейлиста

Для удаления плейлиста требуется быть авторизованным в веб-приложении, а также иметь плейлист. Для перехода в меню плейлиста требуется на боковой панели выбрать желаемый плейлист. Скриншот меню плейлиста предоставлен на рисунке 5.15. Для удаления плейлиста нажмите кнопку «УДАЛИТЬ ПЛЕЙЛИСТ».

* 1. Удаление любых комментариев администратором

Для удаления любых комментариев администратором быть авторизованным в веб-приложении от имени администратора. Необходимо перейти в панель администратора, для этого на боковой панели нажмите кнопку «Панель администратора». Скриншот панели предоставлен на рисунке 5.16.

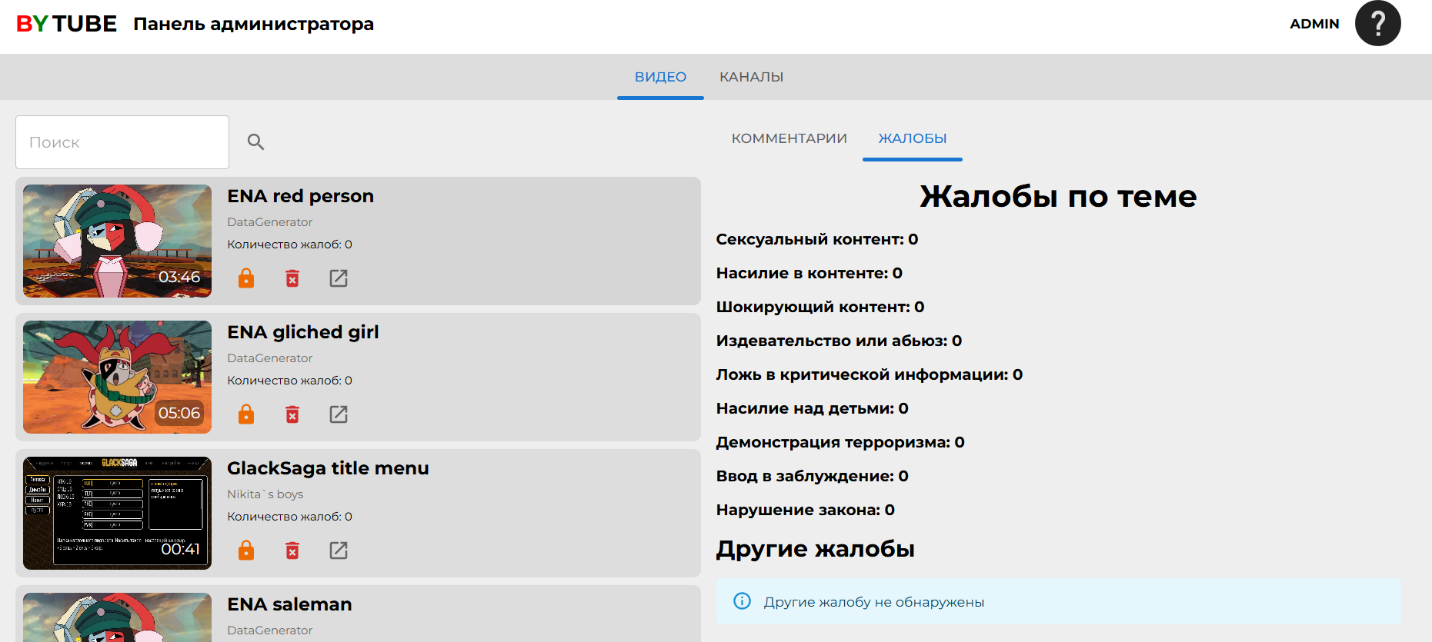


Рисунок 5.16 – Панель администратора (Видео)

Выберите желаемое видео перейдите в локальный раздел «комментарии». Там предоставлен идентичный интерфейс как на рисунке 5.5. Для удаления нажмите кнопку «Удалить» на выбранном комментарии.

* 1. Скрытия видео администратором

Для удаления любых комментариев администратором быть авторизованным в веб-приложении от имени администратора. Необходимо перейти в панель администратора, для этого на боковой панели нажмите кнопку «Панель администратора». Скриншот панели администратора предоставлен на рисунке 5.16.

Нажмите на желаемом видео кнопку «Ограничить», а если требуется блокировка, то кнопку «Заблокировать».

* 1. Удаление видео администратором

Для удаления любых комментариев администратором быть авторизованным в веб-приложении от имени администратора. Необходимо перейти в панель администратора, для этого на боковой панели нажмите кнопку «Панель администратора». Скриншот панели администратора предоставлен на рисунке 5.16.

Для удаления желаемого видео нажмите кнопку «Удалить».

* 1. Просмотр жалоб пользователей администратором

Для удаления любых комментариев администратором быть авторизованным в веб-приложении от имени администратора. Необходимо перейти в панель администратора, для этого на боковой панели нажмите кнопку «Панель администратора». Скриншот панели администратора предоставлен на рисунке 5.16.

Выберите желаемое видео перейдите в локальный раздел «жалобы».

* 1. Блокировка канала администратором

Для удаления любых комментариев администратором быть авторизованным в веб-приложении от имени администратора. Необходимо перейти в панель администратора, для этого на боковой панели нажмите кнопку «Панель администратора». Скриншот панели администратора предоставлен на рисунке 5.17.

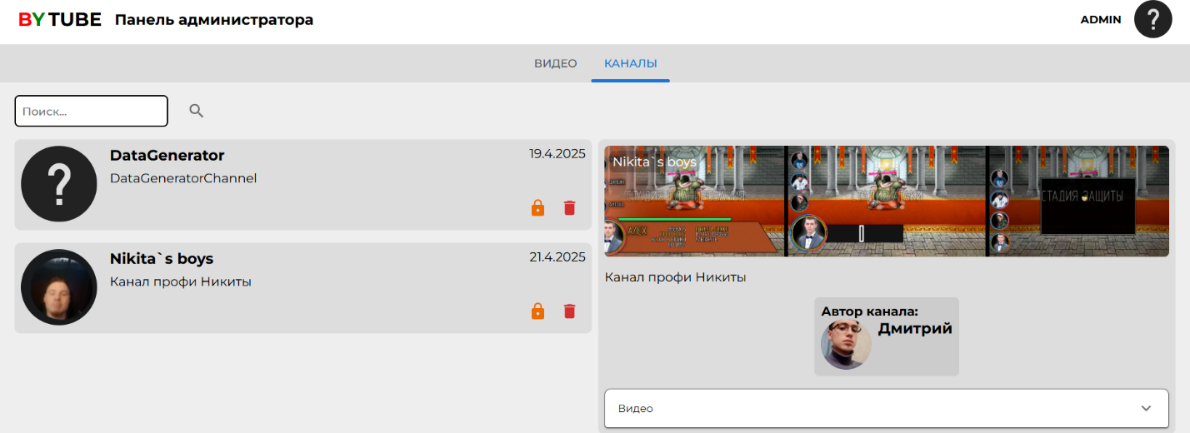


Рисунок 5.17 – Панель администратора (Каналы)

Перейдите в раздел «Каналы», нажмите на выбранном канале кнопку «Ограничить», а для полной блокировки кнопку «Заблокировать».

* 1. Удаление канала администратором

Для удаления любых комментариев администратором быть авторизованным в веб-приложении от имени администратора. Необходимо перейти в панель администратора, для этого на боковой панели нажмите кнопку «Панель администратора». Скриншот панели администратора предоставлен на рисунке 5.17.

Перейдите в раздел «Каналы», нажмите на выбранном канале кнопку «Удалить».

* 1. Вывод по разделу

Раздел охватывает описание основных функций веб-приложения, все действия описаны структурировано, что облегчает пользование, даже неподготовленным пользователям.

В разделе уделено внимание в том числе обработке ошибок, возникающих при использовании системы, в основном это ошибки, касающееся валидации полей ввода в формах.

Описана каждая вкладка, используемая в приложении, что позволит понять по отдельности каждую часть приложения. Это включает описание пользовательского интерфейса, назначение каждой страницы, доступный функционал, а также роли, для которых та или иная вкладка доступна. Такой подход способствует более четкому восприятию структуры веб-приложения и упрощает навигацию внутри него.

Включены инструкции по началу работы с веб-приложением, что помогает новым пользователям быстро освоиться с системой.

Приведены примеры использования веб-приложения в различных сценариях, чтобы показать, как разные функции могут быть применены в реальных ситуациях.

1. Технико-экономическое обоснования проекта

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *ДП 06.00.ПЗ* | | | | | |
|  |  |  |  |  |
|  |  | Ф.И.О | Подпись | Дата |
| Разраб | | Окулич Д.Ю. |  |  | *6 Технико-экономическое обоснование проекта* | Лит. | | | Лист | Листов |
| Пров. | | Белодед Н.И. |  |  |  | У |  | 1 | 7 |
| Консульт. | | Познякова Л.С. |  |  | БГТУ 1-40 01 01, 2025 | | | | |
| Н. контр. | | Белодед Н.И. |  |  |
| Утв. | | Смелов В.В. |  |  |

* 1. Общая характеристика разрабатываемого программного средства

При выполнении данного проекта было разработано веб-приложение BYTUBE, предназначенное для размещения и просмотра видеоконтента. Целью приложения было создание удобной и открытой платформы, где пользователи могут взаимодействовать с видео, каналами и другими участниками, в зависимости от своих ролей.

Пользователи могут сортировать видео, искать их по названию или тегам, а также просматривать публичные ролики и комментарии под ними. Пользователям предлагается гибкая система ролей, каждая из которых открывает дополнительные возможности. Гости могут зарегистрироваться или авторизоваться, чтобы получить доступ к расширенному функционалу. Клиенты управляют своими каналами, загружают и редактируют видео, создают плейлисты, оставляют комментарии и жалобы, а также используют функцию совместного просмотра. Администраторы следят за порядком на платформе, скрывая или удаляя нежелательный контент, блокируя каналы и обрабатывая жалобы.

Во время разработки дипломного проекта использовались технологии ASP.NET Core для серверной части, SignalR для реализации совместного просмотра, EF Core для работы с базой данных, React и MobX для создания динамичного интерфейса, а также PostgreSQL в качестве надежной СУБД.

Разработанное программное решение имеет следующие преимущества перед рассмотренными в главе 1 аналогичными образцами:

* простота использования приложения;
* открытость платформы для всех категорий пользователей;
* совместный просмотра видео в реальном времени.

Стратегия монетизации не предполагается, так как приложение разрабатывается в социальных целях и предоставляется пользователям полностью бесплатно. Предполагается продвижение приложения через социальные сети, тематические сообщества и форумы, ориентированные на просмотр и создание видеоконтента.

* 1. Исходные данные для проведения расчетов и маркетинговый анализ

Источниками исходных данных для данных расчетов выступают действующие нормативные правовые акты.

Исходные данные для расчета приведены в таблице 6.1.

Таблица 6.1 – Исходные данные для расчета

|  |  |  |
| --- | --- | --- |
| Наименование показателя | Условные обозначения | Норматив |
| Норматив дополнительной заработной платы, % | Ндз | 9 |
| Ставка отчислений в Фонд социальной защиты населения, % | Нфсзн | 34 |
| Ставка отчислений по обязательному страхованию в БРУСП «Белгосстрах», % | Нбгс | 0,6 |
| Норматив прочих прямых затрат, % | Нпз | 25 |
| Норматив накладных расходов, % | Нобп,обх | 50 |
| Норматив расходов на сопровождение и адаптацию, % | Нрса | 10 |
| Ставка НДС, % | ННДС | 20 |
| Налог на прибыль, % | Нп | 20 |

В ходе маркетингового анализа была определена стоимость разработки аналогичных веб-приложений, ориентированных на видео контент. Результаты анализа представлены в таблице 6.2.

Таблица 6.2 – Анализ стоимости разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Продукты-аналоги | Источник | Стоимость, руб | Примечание |
| YouTube | https://youtube.com | 40 000 | Информация за 2006 год. |
| Vk Video | https://vkvideo.ru | 22 000 | Примерная стоимость на основе стартового функционала и инфраструктуры что уже была. |
| RUTUBE | https://rutube.ru | 27 000 | Примерная стоимость на основе стартового функционала. |

В ходе проведения маркетингового анализа, была определена стоимость разработки аналогичного программного продукта видео-хостинга. Средняя цена разработки аналогичного продукта составляет 25000-33000 рублей. Так же стоит учитывать модуль совместного просмотра, который отдельно стоит 2000 руб. Таким образом, общая стоимость разработки данного программного средства, выбранного в качестве базы сравнения ставит 26000 рублей без НДС. С НДС цена составляет 31000 рублей.

* 1. Обоснование цены программного средства
     1. Расчет затрат рабочего времени на разработку программного средства

В таблице 6.3 в укрупненном виде указаны работы, которые необходимо выполнить для создания указанного в дипломной работе программного средства, исполнители по данным работам и трудозатраты по каждой работе.

Таблица 6.3 – Затраты рабочего времени на разработку ПС

|  |  |  |
| --- | --- | --- |
| Содержание работ | Исполнитель | Трудозатраты,  чел-часов |
| Руководство проектом | Проектный менеджер | 68 |
| Проектирование архитектуры приложения и структуры базы данных | Бизнес-аналитик | 56 |
| Разработка серверной части приложения | Бэкенд-разработчик | 160 |
| Верстка интерфейса на основе макетов | Фронтенд-разработчик | 112 |
| Функциональное тестирование | Тестировщик | 96 |
| Сопровождение приложения | Системный администратор | 48 |
| Проектирование UI/UX и создание макетов | Дизайнер | 64 |
| Всего |  | 604 |

Таким образом суммарно на разработку будет затрачено 604 часа.

* + 1. Расчет основной заработной платы

Для определения величины основной заработной платы, было проведено исследование величин заработных плат для специалистов в сфере разработки и определение их часовых ставок. Источником данных служили открытые веб-порталы, различные форумы, официальная отчетность, а также общий средний уровень заработка в сфере информационных технологий в Республике Беларусь.

После определения часовых ставок и трудозатрат исполнителей определяются заработные платы всех исполнителей. Заработная плата отдельного специалиста рассчитывается по формуле 6.1. Результаты подсчетов представлены в таблице 6.4.

|  |  |
| --- | --- |
|  | (6.1) |

где Соз – основная заработная плата, руб.;

Траз – трудоемкость, чел./час.;

Сзп – средняя часовая ставка, руб./час.

Таблица 6.4 – Расчет основной заработной платы специалистов

|  |  |  |  |
| --- | --- | --- | --- |
| Исполнитель | Затраты рабочего времени, часов | Средняя часовая ставка, руб./час | Основная заработная плата, руб. |
| Проектный менеджер | 68 | 20 | 1360 |
| Бизнес-аналитик | 56 | 18 | 1008 |
| Бэкенд-разработчик | 160 | 16 | 2560 |
| Фронтенд-разработчик | 112 | 15 | 1680 |
| Тестировщик | 96 | 13 | 1248 |
| Системный администратор | 48 | 15 | 720 |
| Дизайнер | 64 | 12 | 768 |
| Всего | 604 |  | 9344 |

Суммарная основная заработная плата всех специалистов веб-приложения составит 9344 рублей.

* + 1. Расчет дополнительной заработной платы

Дополнительная заработная плата на конкретное программное средство включает выплаты, предусмотренные законодательством о труде, и определяется по нормативу в процентах к основной заработной плате по формуле 6.2.

|  |  |
| --- | --- |
|  | (6.2) |

где Соз – основная заработная плата, руб.;

Ндз – норматив дополнительной заработной платы, %.

Сдз = 9344 ⋅ 9 / 100 = 840,96 руб.

Таким образом дополнительная заработная плата составила 840,96 рублей.

* + 1. Расчет отчислений в Фонд социальной защиты населения и по обязательному страхованию

Отчисления в Фонд социальной защиты населения (ФСЗН) и по обязательному страхованию от несчастных случаев на производстве, и профессиональных заболеваний в БРУСП «Белгосстрах» определяются в соответствии с действующими законодательными актами по нормативу в процентном отношении к фонду основной и дополнительной зарплаты исполнителей и вычисляются по формуле 6.3.

|  |  |  |
| --- | --- | --- |
|  |  | (6.3) |

где Соз – основная заработная плата, руб.;

Сдз – дополнительная заработная плата на конкретное ПС, руб.;

Нфсзн – норматив отчислений в Фонд социальной защиты, %.

Отчисления в БРУСП «Белгосстрах» вычисляются по формуле 6.4.

|  |  |
| --- | --- |
|  | (6.4) |

Таким образом, общие отчисления в БРУСП «Белгосстрах» составили 61,11 руб., а в фонд социальной защиты населения – 3 462,89 руб.

* + 1. Расчет суммы прочих прямых затрат

Расходы на конкретное программное средство Спз включают расходы на приобретение и подготовку специальной технической информации, платных сервисов тестирования и прочие операционные издержки, прямо относимые на проект, и рассчитываются по формуле 6.5.

|  |  |
| --- | --- |
|  | (6.5) |

где Нпз – норматив прочих затрат в целом по организации, %.

Спз = 9344 ⋅ 25 / 100 = 2 336 руб.

Таким образом, сумма прочих прямых затраты при разработке веб-приложения составила 2 336 рублей.

* + 1. Расчет суммы накладных расходов

Сумма накладных расходов Собп,обх – произведение основной заработной платы исполнителей на конкретное программное средство Соз на норматив накладных расходов в целом по организации Нобп,обх, по формуле 6.6.

|  |  |
| --- | --- |
|  | (6.6) |

Сумма накладных расходов составит:

Собп,обх = 9344 **.** 50 / 100 = 4 672 руб.

Таким образом, сумма накладных расходов составила 4 672 руб.

* + 1. Сумма расходов на разработку программного средства

Сумма расходов на разработку программного средства Ср определяется как сумма основной и дополнительной заработных плат исполнителей на конкретное программное средство, отчислений на социальные нужды, суммы прочих прямых затрат и суммы накладных расходов, по формуле 6.7.

|  |  |
| --- | --- |
|  | (6.7) |

Все данные необходимые для вычисления есть, поэтому можно определить сумму расходов на разработку программного средства.

Ср = 9 344 + 840,96 + 3 462,89 + 61,11 + 2 336 + 4 672 = 20 716,96 руб.

Сумма расходов на разработку программного средства была вычислена на основе данных, рассчитанных ранее в данном разделе, и того составила 20 716,96 рублей.

* + 1. Расходы на сопровождение и адаптацию

Сумма расходов на сопровождение и адаптацию программного средства Срса определяется как произведение суммы расходов на разработку на норматив расходов на сопровождение и адаптацию Нрса, и находится по формуле 6.8.

|  |  |
| --- | --- |
|  | (6.8) |

Срса = 20 716,96 ⋅ 10 / 100 = 2 071,7 руб.

Получим, что сумма расходов на сопровождение и адаптацию программного средства, определенная по формуле 6.8, составляет 2 071,7 рубля.

* + 1. Расчет полной себестоимости

Полная себестоимость Сп определяется как сумма двух элементов: суммы расходов на разработку Ср и суммы расходов на сопровождение и адаптацию Срса.

Полная себестоимость Сп вычисляется по формуле 6.9

|  |  |
| --- | --- |
|  | (6.9) |

Сп = 20 716,96 + 2 071,7 = 22 778,65 руб.

Получим, что полная себестоимость мобильного приложения равна 22 778,65 рубля.

* 1. Вывод по разделу

В рамках данного раздела были проведены экономические расчеты, на основе которых была определена себестоимость разрабатываемого программного средства, а также прогнозируемая отпускная цена всего продукта. Анализ такого вида позволяет определить целесообразность разработки приложения.

В таблице 6.5 представлены результаты расчетов для основных показателей данной главы в краткой форме.

Таблица 6.5 – Результаты расчетов

|  |  |
| --- | --- |
| Наименование показателя | Значение |
| Время разработки, ч. | 604 |
| Основная заработная плата, руб. | 9344 |
| Дополнительная заработная плата, руб. | 840,96 |
| Отчисления в Фонд социальной защиты населения, руб. | 3462,89 |
| Отчисления в БРУСП «Белгосстрах», руб. | 61,11 |
| Прочие прямые затраты, руб. | 2336 |
| Накладные расходы, руб. | 4976,13 |
| Себестоимость разработки программного средства, руб. | 4672 |
| Расходы на сопровождение и адаптацию, руб. | 2071,7 |
| Полная себестоимость, руб. | 22 778,65 |

Современный видеохостинг сталкивается с высокой конкуренцией и растущими требованиями пользователей к качеству контента, скорости загрузки и персонализации рекомендаций. Существующие платформы часто не учитывают региональные особенности, испытывают проблемы с модерацией и предлагают шаблонные алгоритмы, которые не всегда соответствуют интересам аудитории. BYTUBE призван решить эти проблемы, предлагая гибкую, адаптивную платформу с акцентом на локальный контент и сообщество.

Необходимость разработки программного средства, обусловлена большим количеством рекламы у конкурентов которая пагубно влияет на развитие сообщества, а также отсутствие некоторые функций.

BYTUBE создает уникальный социальный эффект, устраняя раздражающую рекламу и делая просмотр видео более комфортным. Пользователи могут полностью сосредоточиться на контенте, не отвлекаясь на навязчивые ролики или баннеры. Это формирует лояльное сообщество, где ценность контента стоит на первом месте.

Для поддерживания работоспособности программного продукта, потребности, затраты на содержание серверов, администрирование, возмещение которых предполагается за счет добровольных пожертвований пользователей.

Совместный просмотр в реальном времени превращает BYTUBE в платформу для живого общения. Друзья, родственники или даже незнакомцы могут смотреть видео одновременно, комментировать и обсуждать происходящее в чате. Это создает эффект кинотеатра или телевизионного вечера, но в цифровом пространстве.

Доступность распространения контента на BYTUBE стимулирует творчество среди обычных пользователей. Нет жестких алгоритмических ограничений, как в крупных коммерческих платформах, поэтому даже начинающие авторы могут найти свою аудиторию. Это делает платформу более демократичной и открытой для экспериментов.

Отсутствие монетизации через рекламу снижает конкуренцию за клики, что уменьшает количество "мусорного" контента. Это формирует здоровую экосистему, где творчество важнее заработка на просмотрах.

# **Заключение**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *ДП 00.00.ПЗ* | | | | | |
|  |  |  |  |  |
|  |  | Ф.И.О | Подпись | Дата |
| Разраб | | Окулич Д.Ю. |  |  | *Заключение* | Лит. | | | Лист | Листов |
| Пров. | | Белодед Н.И. |  |  |  | У |  | 1 | 1 |
|  | |  |  |  | БГТУ 1-40 01 01, 2025 | | | | |
| Н. контр. | | Белодед Н.И. |  |  |
| Утв. | | Смелов В.В. |  |  |

Цель дипломного проекта разработка видеохостинга «BYTUBE» достигнуто. В результате выполнения дипломного проектирования было разработано веб-приложение видеохостинг «BYTUBE» для просмотра видеоконтента с возможностью совместного просмотра.

Архитектура приложения построена на сервисной модели и включает три основных компонента:

* сервис, для хранения файлов;
* сервис, хранения комнат совместного просмотра;
* сервис, для обработки загружаемых видео.

В рамка работы были рассмотрены 3 аналогичных решение. Разработана схема вариантов использование, веб-приложение поддерживает 3 роли. Разработана логическая схема база данных, которая включает в себя 10 таблицы. Спроектирована архитектура веб-приложения. Реализована 31 функция, общий объем программного кода – 30 000 строк. Было проведено тестирование веб-приложения, общее количество тестов – 39, степень покрытия кода тестами – 80%. Было разработано руководство пользователя. Было проведено технико-экономическое обоснование проекта, себестоимость составила 22 778,65 руб.

Пользователи имеют возможность регистрироваться и авторизовываться в системе, а также использовать широкий спектр функций, предоставляемых приложением. Администраторы, в свою очередь, получают доступ ко всем возможностям пользователя, дополненным функциями управления видео и канами.

Представленная в рамках веб-приложения система, состоящая из двух ключевых модулей совместного просмотра и взаимодействия, обеспечивает эффективное и последовательное вовлечение пользователей в видеоконтент. Модуль совместного просмотра предоставляет пользователям удобный интерфейс для просмотра видео совместно с другими пользователями, с возможностью обмениваться впечатлениями.

Конечный результат дипломного проекта апробирован на 76 студенческой конференции БГТУ опубликован в сборнике в части 4 в секции «информационные технологии».

Веб-приложение является завершенным продуктом с возможностью дальнейшего расширения, что позволяет масштабировать его функционал. Интерфейс приложения разработан с учетом удобства пользователей, что делает его доступным для людей любого возраста. Программное средство полностью соответствует поставленным задачам и требованиям.

# **Список использованной литературы**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *ДП 00.00.ПЗ* | | | | | |
|  |  |  |  |  |
|  |  | Ф.И.О | Подпись | Дата |
| Разраб | | Окулич Д.Ю. |  |  | *Список использованной литературы* | Лит. | | | Лист | Листов |
| Пров. | | Белодед Н.И. |  |  |  | У |  | 1 | 1 |
|  | |  |  |  | БГТУ 1-40 01 01, 2025 | | | | |
| Н. контр. | | Белодед Н.И. |  |  |
| Утв. | | Смелов В.В. |  |  |

1. ASP.NET Core 8.0 [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/aspnet/core/release-notes/aspnetcore-8.0 – Дата доступа: 28.05.2025
2. React [Электронный ресурс]. – Режим доступа: https://react.dev/blog/2024/12/05/react-19 – Дата доступа: 28.05.2025
3. MOBX [Электронный ресурс]. – Режим доступа: https://mobx.js.org/README.html – Дата доступа: 28.05.2025
4. PostgreSQL [Электронный ресурс]. – Режим доступа: https://www.postgresql.org/docs/ – Дата доступа: 28.05.2025
5. REST API [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/483202/ – Дата доступа: 28.05.2025
6. SignalR [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/aspnet/signalr/ – Дата доступа: 28.05.2025
7. Docker [Электронный ресурс]. – Режим доступа: https://www.docker.com – Дата доступа: 28.05.2025
8. YouTube [Электронный ресурс]. – Режим доступа: www.youtube.com – Дата доступа: 28.05.2025
9. RUTUBE [Электронный ресурс]. – Режим доступа: https://rutube.ru – Дата доступа: 28.05.2025
10. VK Video [Электронный ресурс]. – Режим доступа: https://vkvideo.ru – Дата доступа: 28.05.2025
11. HTTPS RFC [Электронный ресурс]. – Режим доступа: https://datatracker.ietf.org/doc/html/rfc2616 – Дата доступа: 28.05.2025
12. TCP RFC [Электронный ресурс]. – Режим доступа: https://datatracker.ietf.org/doc/html/rfc793 – Дата доступа: 28.05.2025
13. WebSocket RFC [Электронный ресурс]. – Режим доступа: https://datatracker.ietf.org/doc/html/rfc6455 – Дата доступа: 28.05.2025
14. Библиотека Microsoft.AspNetCore [Электронный ресурс]. – Режим доступа: https://www.nuget.org/packages/Microsoft.AspNetCore – Дата доступа: 28.05.2025
15. Библиотека Microsoft.AspNetCore.Session [Электронный ресурс]. – Режим доступа: https://www.nuget.org/packages/Microsoft.AspNetCore.Session – Дата доступа: 28.05.2025
16. Библиотека Microsoft.AspNetCore.StaticFiles [Электронный ресурс]. – Режим доступа: https://www.nuget.org/packages/Microsoft.AspNetCore. StaticFiles – Дата доступа: 28.05.2025
17. Библиотека Microsoft.AspNetCore.Authentication.JwtBearer [Электронный ресурс]. – Режим доступа: https://www.nuget.org/packages/ Microsoft.AspNetCore.Authentication.JwtBearer – Дата доступа: 28.05.2025
18. Библиотека Microsoft.AspNetCore.SignalR.Common [Электронный ресурс]. – Режим доступа: https://www.nuget.org/packages /Microsoft.AspNetCore.SignalR.Common – Дата доступа: 28.05.2025
19. Библиотека EntityFrameworkCore.PostgreSQL [Электронный ресурс]. – Режим доступа: https://www.nuget.org/packages/Npgsql.EntityFrameworkCore. PostgreSQL – Дата доступа: 28.05.2025
20. Библиотека Xabe.FFmpeg [Электронный ресурс]. – Режим доступа: https://www.nuget.org/packages/Xabe.FFmpeg – Дата доступа: 28.05.2025
21. Библиотека NAudio [Электронный ресурс]. – Режим доступа: https://www.nuget.org/packages/NAudio – Дата доступа: 28.05.2025
22. Библиотека Swashbuckle.AspNetCore.SwaggerGen [Электронный ресурс]. – Режим доступа: https://www.nuget.org/packages/Swashbuckle. AspNetCore.SwaggerGen – Дата доступа: 28.05.2025
23. Библиотека System.IdentityModel.Tokens.Jwt [Электронный ресурс]. – Режим доступа: https://www.nuget.org/packages/System.IdentityModel. Tokens.Jwt – Дата доступа: 28.05.2025
24. N-Layer [Электронный ресурс]. – Режим доступа: https://medium.com/design-microservices-architecture-with-p – дата доступа: 28.05.2025
25. Dependency Injection (DI) [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/350068/ – дата доступа: 28.05.2025
26. JSON Web Token (JWT) RFC [Электронный ресурс]. – Режим доступа: https://datatracker.ietf.org/doc/html/rfc7519 – дата доступа: 28.05.2025
27. Паттерн «Repository» [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/248505/ – дата доступа: 28.05.2025
28. JSON RFC [Электронный ресурс]. – Режим доступа: https://datatracker.ietf.org/doc/html/rfc7159.html – дата доступа: 28.05.2025

Диаграмма вариантов использована (ДП 01.00.ГЧ)

Логическая схема базы данных (ДП 02.00.ГЧ)

Диаграмма развертывания (ДП 03.00.ГЧ)

Блок-схема подключения к совместному просмотру (ДП 04.00.ГЧ)

Диаграмма компонентов (ДП 05.00.ГЧ)

Скриншот работы приложения (ДП 06.00.ГЧ)

Приложение А

Таблица Users

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| ID | UUID | Уникальный идентификатор пользователя (PK) |
| name | TEXT | Имя пользователя |
| email | TEXT | Электронная почта |
| password | TEXT | Хешированный пароль |
| birthday | DATE | Дата рождения |
| role | INTEGER | Роль пользователя |
| token | TEXT | Токен авторизации |

Таблица Channels

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| ID | UUID | Уникальный идентификатор канала (PK) |
| name | TEXT | Название канала |
| description | TEXT | Описание канала |
| created | TIMESTAMP | Дата создания канала |
| status | INTEGER | Статус канала |
| userId | UUID | Владелец канала (FK на Users) |

Таблица Videos

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| ID | UUID | Уникальный идентификатор видео (PK) |
| channelId | UUID | Канал-владелец (FK на Channels) |
| title | TEXT | Название видео |
| description | TEXT | Описание видео |
| duration | TEXT | Продолжительность видео |
| forAdults | BOOLEAN | Контент 18+ |
| tags | JSONB | Список тегов |
| created | TIMESTAMP | Дата загрузки |
| videoAccess | INTEGER | Уровень доступа |
| videoStatus | INTEGER | Статус видео |

Таблица Comments

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| ID | UUID | Уникальный идентификатор комментария (PK) |
| message | TEXT | Текст комментария |
| likes | JSONB | Лайки к комментарию |
| created | TIMESTAMP | Дата создания |
| ownerId | UUID | Пользователь, оставивший комментарий (FK на Users) |
| videoId | UUID | Видео, к которому оставлен комментарий (FK на Videos) |

Таблица Playlists

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| ID | UUID | Уникальный идентификатор плейлиста (PK) |
| name | TEXT | Название плейлиста |
| access | INTEGER | Доступ |
| userId | UUID | Владелец плейлиста (FK на Users) |

Таблица PlaylistItems

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| ID | UUID | Уникальный идентификатор элемента (PK) |
| playlistId | UUID | Идентификатор плейлиста (FK на Playlists) |
| videoId | UUID | Идентификатор видео (FK на Videos) |
| order | INTEGER | Порядок отображения в плейлисте |

Таблица Subscriptions

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| ID | UUID | Уникальный идентификатор подписки (PK) |
| userId | UUID | Подписавшийся пользователь (FK на Users) |
| channelId | UUID | Канал, на который подписка (FK на Channels) |

Таблица Reports

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| ID | UUID | Уникальный идентификатор жалобы (PK) |
| type | INTEGER | Тип жалобы |
| message | TEXT | Текст жалобы |
| created | TIMESTAMP | Дата подачи жалобы |
| videoId | UUID | Видео, на которое жалоба (FK на Videos) |

Таблица VideoMarks

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| ID | UUID | Уникальный идентификатор метки (PK) |
| userId | UUID | Пользователь, поставивший метку (FK на Users) |
| videoId | UUID | Видео, к которому метка (FK на Videos) |
| isLike | BOOLEAN | Лайк |
| isDisLike | BOOLEAN | Дизлайк |
| updated | TIMESTAMP | Дата последнего обновления метки |

Таблица VideoViews

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| ID | UUID | Уникальный идентификатор просмотра (PK) |
| userId | UUID | Пользователь, просмотревший видео (FK на Users) |
| videoId | UUID | Видео, которое было просмотрено (FK на Videos) |
| created | TIMESTAMP | Время просмотра |

Приложение Б

CREATE EXTENSION IF NOT EXISTS "uuid-ossp";

CREATE TABLE Users (

ID UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

name TEXT,

email TEXT,

password TEXT,

birthday DATE,

role INTEGER,

token TEXT

);

CREATE TABLE Channels (

ID UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

name TEXT,

description TEXT,

created TIMESTAMP,

status INTEGER,

user\_id UUID REFERENCES Users(ID)

);

CREATE TABLE Videos (

ID UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

channel\_id UUID REFERENCES Channels(ID),

title TEXT,

description TEXT,

duration TEXT,

forAdults BOOLEAN,

tags JSONB,

created TIMESTAMP,

videoAccess INTEGER,

videoStatus INTEGER

);

CREATE TABLE Comments (

ID UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

message TEXT,

likes JSONB,

created TIMESTAMP,

owner\_id UUID REFERENCES Users(ID),

video\_id UUID REFERENCES Videos(ID)

);

CREATE TABLE Playlists (

ID UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

name TEXT,

access INTEGER,

user\_id UUID REFERENCES Users(ID)

);

CREATE TABLE PlaylistItems (

ID UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

playlistId UUID REFERENCES Playlists(ID),

videoId UUID REFERENCES Videos(ID),

order INTEGER

);

CREATE TABLE Subscriptions (

ID UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

user\_id UUID REFERENCES Users(ID),

channel\_id UUID REFERENCES Channels(ID)

);

CREATE TABLE Reports (

ID UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

type INTEGER,

message TEXT,

created TIMESTAMP,

video\_id UUID REFERENCES Videos(ID)

);

CREATE TABLE VideoMarks (

ID UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

user\_id UUID REFERENCES Users(ID),

video\_id UUID REFERENCES Videos(ID),

isLike BOOLEAN,

isDisLike BOOLEAN,

updated TIMESTAMP

);

CREATE TABLE VideoViews (

ID UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

user\_id UUID REFERENCES Users(ID),

video\_id UUID REFERENCES Videos(ID),

created TIMESTAMP

);

Приложение В

[HttpPost("signin")]

public async Task<IResult> SignIn([FromBody] SigninModel model)

{

try

{

var user = await \_db.Users.FirstAsync(i => i.Email == model.Email);

if (!\_passwordHasher.Verify(model.Password, user.Password))

{

throw new ServerException("Пароли не совпадают");

}

HttpContext.Response.Cookies.Append(

"AccessToken",

JwtService.GenerateJwtToken(\_jwtManager.AccessToken, new() { Id = user.Id, Role = user.Role}),

\_jwtManager.JwtCookieOptions

);

string token = JwtService.GenerateJwtToken( \_jwtManager.RefreshToken, new() { Id = user.Id, Role = user.Role });

HttpContext.Response.Cookies.Append(

"RefreshToken",

token,

\_jwtManager.JwtCookieOptions

);

user.Token = token;

\_db.Users.Update(user);

await \_db.SaveChangesAsync();

return Results.Ok();

}

catch (ServerException apperr)

{

return Results.Json(apperr.GetModel(), statusCode: apperr.Code);

}

}

Реализация авторизации

[HttpGet]

public async Task<IResult> Get([FromQuery] Guid id)

{

try

{

var authData = AuthorizeData.FromContext(HttpContext);

Video video = await \_dbContext.Videos.FindAsync(id)

?? throw new ServerException("Видео не найдено", 404);

Channel channel = await \_dbContext.Channels

.Include(i => i.Subscribes)

.FirstAsync(i => i.Id == video.ChannelId);

if (video.VideoAccess == Video.Access.Private)

{

if (!authData.IsAutorize || channel.UserId != authData.Id)

throw new ServerException("Видео вам не доступно", 403);

}

if (video.VideoStatus == Video.Status.Blocked || channel.Status == Channel.ActiveStatus.Blocked)

throw new ServerException("Видео более не доспутно", 403);

var videoLocalData = \_localData.GetVideoData(id);

var channelLocalData = \_localData.GetChannelData(channel.Id);

var views = await \_dbContext.VideoViews.Where(v => v.VideoId == id).CountAsync();

return Results.Json(new VideoFullModel();

);

}

catch (ServerException srverr)

{

return Results.Json(srverr.GetModel(), statusCode: srverr.Code);

}

}

Реализация метода «Get» в «VideoController»

public async Task<IResult> StreamVideo([FromRoute] Guid id) {

try {

var authData = AuthorizeData.FromContext(HttpContext);

string path = $"./Data/videos/{id}/video.mp4";

if (!System.IO.File.Exists(path))

throw new ServerException("Файла больше не существует!", 404);

Video video = await \_dbContext.Videos

.Include(i => i.Channel)

.FirstAsync(i => i.Id == id);

if (video.VideoAccess == Video.Access.Private)

{

if ((authData.IsAutorize && authData.Id != video.Channel.UserId) || !authData.IsAutorize)

throw new ServerException("Видео файл не доступен!", 403);

}

if (video.VideoStatus == Video.Status.Blocked)

throw new ServerException("Видео файл не доступен!", 403);

HttpContext.Response.Headers.Append("Accept-Ranges", "bytes");

var fileStream = new FileStream(path, FileMode.Open, FileAccess.Read, FileShare.Read | FileShare.Delete);

var fileLength = fileStream.Length;

if (Request.Headers.ContainsKey("Range"))

{

var rangeHeader = Request.Headers["Range"].ToString();

var range = rangeHeader.Replace("bytes=", "").Split('-');

var start = long.Parse(range[0]);

var end = range.Length > 1 && !string.IsNullOrEmpty(range[1])

? long.Parse(range[1])

: fileLength - 1;

var chunkSize = end - start + 1;

fileStream.Seek(start, SeekOrigin.Begin);

return Results.File(

fileStream,

"video/mp4",

enableRangeProcessing: true);

}

return Results.File(fileStream, "video/mp4");

}

catch (ServerException srvErr)

{

return Results.Json(srvErr.GetModel(), statusCode: srvErr.Code);

}

}}

Реализация метода «SteamVideo» в «VideoController»

[HttpGet("video")]

public async Task<IResult> GetVideoComments([FromQuery] string vid)

{

try

{

var authData = AuthorizeData.FromContext(HttpContext);

if (!Guid.TryParse(vid, out Guid vguid))

throw new ServerException("vId is not correct!");

Comment[] comments = await \_commentRepository.GetVideoComments(vguid);

return Results.Json(comments.Select(comment =>

{

var usrData = \_localData.GetUserData(comment.User.Id);

bool userIsLikeIt = false;

bool isVideoOwner = false;

if (authData.IsAutorize)

{

userIsLikeIt = comment.Likes.Contains(authData.Id);

isVideoOwner = comment.Video!.Channel!.UserId == authData.Id;

}

return new CommentModel()

{ ... };

}

catch (ServerException err)

{

return Results.Json(err.GetModel(), statusCode: err.Code);

}

}

Реализация метода «GetVideoComments» в «CommentController»

[HttpDelete, Authorize]

public async Task<IResult> Delete([FromQuery] Guid id)

{

try

{

var authData = AuthorizeData.FromContext(HttpContext);

Channel channel = await \_db.Channels

.Include(channel => channel.Videos)

.FirstOrDefaultAsync(c => c.Id == id)

?? throw new ServerException("Канал не найден!", 404);

if (channel.UserId != authData.Id)

throw new ServerException("Канал вам не пренадлежит!", 403);

foreach (var video in \_db.Videos.Where(i => i.ChannelId == channel.Id))

{

Directory.Delete($"{LocalDataService.VideosPath}/{video.Id}", true);

\_db.Videos.Remove(video);

}

Directory.Delete($"{LocalDataService.ChannelsPath}/{id}", true);

\_db.Channels.Remove(channel);

await \_db.SaveChangesAsync();

return Results.Ok();

}

catch (ServerException err)

{

return Results.Json(err.GetModel(), statusCode: err.Code);

}

}

Реализация метода «Delete» в «ChannelController»

Приложение Г

Маршруты контроллеров

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Путь | HTTP-метод | Контроллер | Метод | Описание |
| api/auth/signin | POST | AuthController | SignIn | Аутентификация пользователя. |
| api/auth/signout | GET | AuthController | Logout | Выход из системы. |
| api/auth/register | POST | AuthController | Register | Регистрация нового пользователя. |
| api/video | GET | VideoController | Get | Получить видео по идентификатору. |
| api/video/channel | GET | VideoController | GetChannelVideos | Получить список видео канала по channelId. |
| api/video/playlist | GET | VideoController | GetPlaylistVideos | Получить список видео из плейлиста по playlistId. |
| api/video/select | GET | VideoController | Select | Получить видео по параметрам SelectOptions. |
| api/video | POST | VideoController | Post | Создание нового видео |
| api/video | PUT | VideoController | Put | Редактирование информации о видео с указанным id и channelId. |
| api/video | DELETE | VideoController | Delete | Удаление видео по id и channelId. |
| /data/videos/{id} /video.mp4 | GET | VideoController | StreamVideo | Потоковая передача видео по id. |
| api/video/mark | GET | VideoController | GetMark | Получить оценку видео по id. |
| api/video/mark | POST | VideoController | MarkVideo | Оценить видео по id. |
| api/video/view | POST | VideoController | AddView | Добавить просмотр видео по id. |
| api/video/view | GET | VideoController | GetViews | Получить видео по id. |
| api/video/delete | DELETE | VideoController | DeleteByAdmin | Удаление видео администратором по id. |
| api/video/status | PUT | VideoController | ChangeStatus  ByAdmin | Изменить статус видео администратором по id. |
| api/comment | GET | CommentController | Get | Получить комментарий по его id. |
| api/comment/video | GET | CommentController | GetVideoComments | Получить список комментариев для видео с vid. |
| api/comment | POST | CommentController | Post | Создать новый комментарий |
| api/comment/like | POST | CommentController | LikeComment | Поставить лайк комментария по id |
| api/comment | PUT | CommentController | Put | Редактировать комментарий по id |
| api/comment | DELETE | CommentController | Delete | Удалить комментарий по id |
| api/channel | GET | ChannelController | Get | Получить данные о канале по id. |
| api/channel/check | GET | ChannelController | CheckChannel | Проверить, принадлежит ли текущему пользователю канал с указанным id. |
| api/channel | POST | ChannelController | Post | Создать новый канал. |
| api/channel | PUT | ChannelController | Put | Изменить параметры канала по id. |
| api/channel | DELETE | ChannelController | Delete | Удалить канал по id. |
| api/channel/user | GET | ChannelController | GetUserChannels | Получить список каналов, принадлежащих текущему пользователю. |
| api/channel/subscribe | POST | ChannelController | Subscribe | Подписаться на канал по id. |
| api/channel/subscribe | DELETE | ChannelController | Unsubscribe | Отписаться от канала по id. |
| api/channel/  getchannelsbyadmin | GET | ChannelController | GetAdmin  ControllChannels | Получить список каналов для админ-контроля с фильтрацией по имени. |
| api/channel/  statusbyadmin | PUT | ChannelController | SetStatusByAdmin | Изменить статус активности канала. |
| api/channel/  deletebyadmin | DELETE | ChannelController | DeleteByAdmin | Удалить канал администратором по id. |
| api/playlist | GET | PlaylistController | Get | Получить плейлист по его id. |
| api/playlist/user | GET | PlaylistController | GetByUser | Получить все плейлисты текущего пользователя. |
| api/playlist | POST | PlaylistController | Post | Создать новый плейлист. |
| api/playlist/add | POST | PlaylistController | AddVideoToPlaylist | Добавить видео vid в плейлист id. |
| api/playlist/remove | DELETE | PlaylistController | RemoveVideo  FromPlaylist | Удалить видео vid из плейлиста id. |
| api/playlist | PUT | PlaylistController | Put | Обновить данные плейлиста по id. |
| api/playlist | DELETE | PlaylistController | Delete | Удалить плейлист по id. |
| api/report | GET | ReportController | Get | Получить жалобу по id. |
| api/report | POST | ReportController | Post | Отправить жалобу на видео |
| api/report | DELETE | ReportController | Delete | Удалить жалобу по id. |
| api/report/video | GET | ReportController | GetVideoReports | Получить список жалоб на конкретное видео по vid. |

Приложение Д

public class Tests

{

private readonly JwtManager jwtManager = new JwtManager(new JwtSettings()

{

Audience = "http://localhost:8081/api",

Issuer = "http://localhost:8081",

SecretKey = "N0ek5pYWMgnq-iaCqz811YNsxNFkhTb8oNEFooIyHBg",

ExpiryMinutes = 1,

},

new JwtSettings()

{

Audience = "http://localhost:8081/api",

Issuer = "http://localhost:8081",

SecretKey = "N0ek5pYWMgnq-iaCqz811YNsxNFkhTb8oNEFooIyHBf",

ExpiryMinutes = 2,

});

[Fact]

public void PasswordHasherTest0()

{

PasswordHasher hasher = new PasswordHasher("salt");

string passwordHashed = hasher.Hash("pravoda01");

Assert.True(hasher.Hash("pravoda01") == passwordHashed);

}

[Fact]

public void PasswordHasherTest1()

{

PasswordHasher hasher = new PasswordHasher("salt");

string passwordHashed = hasher.Hash("pravoda01");

Assert.False(hasher.Hash("Pravoda01") == passwordHashed);

}

[Fact]

public void JwtManagerTest0()

{

string token = JwtManager.GenerateJwtToken(jwtManager.AccessToken, new User()

{

Id = 12,

Name = "Test",

Role = User.RoleType.User

});

ClaimsPrincipal claims = JwtManager.ValidateToken(token, JwtManager.GetParameters(jwtManager.AccessToken));

Assert.Equal("12", claims.Claims.First().Value);

}

[Fact]

public void JwtManagerTest1()

{

string token = JwtManager.GenerateJwtToken(jwtManager.RefreshToken, new User()

{

Id = 13,

Name = "Testsad",

Role = User.RoleType.User

});

ClaimsPrincipal claims = JwtManager.ValidateToken(token, JwtManager.GetParameters(jwtManager.RefreshToken));

Assert.Equal("13", claims.Claims.First().Value);

}

[Fact]

public void VideoMediaServiceTest()

{

VideoMediaService videoMedia = new VideoMediaService("C:\\ffmpeg");

var data = videoMedia.GetMediaInfo("D:\\BYData\\GAMBLECORE.mp4");

Assert.NotNull(data);

}

}