

Реферат

Пояснительная записка дипломного проекта содержит 60 страниц пояснительной записки, 1 таблиц, 1 формул, 1 иллюстрации, 1 источник литературы, 1 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

Основной целью дипломного проекта является разработка веб-приложения для турагентства с возможностью гибкого подбора туров.

В рассматриваемом веб-приложении пользователю доступен просмотр туров, их бронирование и оплата. Так же пользователю доступна возможность подбора туров по географическим признакам при помощи фильтров, опоросу и ИИ.

В результате выполненных задач обоснована экономическая целесообразность разработки рассматриваемого веб-приложения. Данные наработки позволят повысить качество и конкурентоспособность выпускаемого программного обеспечения.

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 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 и себестоимость программного продукта.

В заключении приведены результаты проделанной работы.

					ДП 02.00.ПЗ		
		ФИО	Подпись	Дата			
Разраб.		Иванов И.И.			2 Работа редактора над журнальным изданием		
Пров.		Руководитель					
Н. контр.		Нистюк О.А.					
Утв.		Смелов В.В.					
					Лит.	Лист	Листов
						1	8
					БГТУ 1-40 01 01, 2025		

Abstract

The explanatory note of the graduation project contains 60 pages of explanatory notes, 1 table, 1 formula, 1 illustration, 1 literature source, 1 appendix.

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 main goal of the graduation project is to develop a video hosting web application with the ability to watch video together.

In the web application in question, the user can watch videos, create playlists, create channels for uploading their content, and comment on videos. The user also has the opportunity to watch video content together with other users.

As a result of the completed tasks, the economic feasibility of developing the web application in question is substantiated. These developments will improve the quality and competitiveness of the software being produced.

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 the economic parameters and the cost of the software product.

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

					ДП 02.00.ПЗ		
		ФИО	Подпись	Дата			
Разраб.	Иванов И.И.				2 Работа редактора над журнальным изданием		
Пров.	Руководитель						
Н. контр.	Нистюк О.А.						
Утв.	Смелов В.В.						
					Лит.	Лист	Листов
						1	8
					БГТУ 1-40 01 01, 2025		

Содержание

Реферат	1
Abstract	2
Содержание	3
Введение	7
1 Постановка задачи и аналитический обзор аналогичных решений	8
1.1 Постановка задачи	8
1.2 Обзор аналогичных решений	8
1.2.1 Веб-приложение «YouTube»	8
1.2.2 Веб-приложение «RUTUBE»	Ошибка! Закладка не определена.
1.2.3 Веб-приложение «VK Video»	10
1.3 Выводы по разделу	11
2 Проектирование веб-приложения	12
2.1 Функциональность веб-приложения	12
2.2 Логическая схема базы данных	15
2.3 Архитектура веб-приложения	15
2.4 Блок-схема подключения к совместному просмотру	16
2.5 Выводы по разделу	17
3 Разработка веб-приложения	18
3.1 Разработка серверной части веб-приложения	18
3.1.1 Используемые библиотеки	18
3.1.2 Структура серверной части	19
3.1.3 Разработка уровня доступа к данным	20
3.1.4 Разработка уровня бизнес-логики	21
3.1.5 Разработка уровня представления	21
3.1.6 Настройка безопасности	22
3.1.7 Используемые паттерны	23
3.2 Клиентская часть	24
3.2.1 Используемые библиотеки	24
3.2.2 Структура клиентской части	24
3.2.3 Конфигурация проекта	25
3.3 Реализация функций	25
3.3.1 Регистрация	25
3.3.2 Авторизация	26
3.3.3 Просмотр публичных видео	27
3.3.4 Просмотр комментариев под видео	28
3.3.5 Поиск публичных видео	28
3.3.6 Поиск по названию	29
3.3.7 Сортировка видео	29
3.3.8 Поиск по тегам	29

					ДП 02.00.ПЗ		
		ФИО	Подпись	Дата	2 Работа редактора над журнальным изданием		
Разраб.		Иванов И.И.					
Пров.		Руководитель					
Н. контр.		Нистюк О.А.					
Утв.		Смелов В.В.					
					Лит.	Лист	Листов
						1	8
					БГТУ 1-40 01 01, 2025		

3.3.9 Удаление канала	30
3.3.10 Редактирование канала	30
3.3.11 Создание канала	31
3.3.12 Загрузка видео	32
3.3.13 Редактирование данных видео	33
3.3.14 Удаление видео.....	34
3.3.15 Удаление комментариев под своими видео	35
3.3.16 Оставление жалобы под видео	36
3.3.17 Оставление комментариев под видео	37
3.3.18 Удаление своих комментариев	38
3.3.19 Редактирование своих комментариев	38
3.3.20 Совместный просмотр	39
3.3.21 Создание плейлистов	40
3.3.22 Удаление видео из плейлиста	41
3.3.23 Добавление видео в плейлиста	42
3.3.24 Воспроизведение плейлиста	43
3.3.25 Удаление плейлиста.....	45
3.3.26 Удаление любых комментариев администратором.....	46
3.3.27 Скрытия видео администратором	46
3.3.28 Удаление видео администратором	47
3.3.29 Просмотр жалоб пользователей администратором	48
3.3.30 Блокировка канала администратором	49
3.3.31 Удаление канала администратором	50
3.4 Маршруты для контролеров.....	51
3.5 Выводы по разделу.....	52
4 Тестирование веб-приложения	53
4.1 Функциональное тестирование.....	53
4.2 Автоматизированное тестирование.....	56
4.3 Нагрузочное тестирования	56
4.4 Выводы по разделу.....	57
5 Руководство пользователя (руководство по эксплуатации)	57
5.1 Регистрация.....	57
5.2 Авторизация.....	59
5.3 Просмотр публичных видео.....	60
5.4 Просмотр комментариев под видео	60
5.5 Поиск публичных видео	61
5.6 Поиск по названию	61
5.7 Сортировка видео.....	61
5.8 Поиск по тегам	62
5.9 Удаление канала.....	62
5.10 Редактирование канала	62
5.11 Создание канала	63
5.12 Загрузка видео	64
5.13 Редактирование данных видео.....	65

5.14 Удаление видео.....	65
5.15 Удаление комментариев под своими видео	66
5.16 Оставление жалобы под видео	66
5.17 Оставление комментариев под видео	66
5.18 Удаление своих комментариев	66
5.19 Редактирование своих комментариев	66
5.20 Совместный просмотр	67
5.21 Создание плейлистов	68
5.22 Удаление видео из плейлиста	68
5.23 Добавление видео в плейлиста	69
5.24 Воспроизведение плейлиста	69
5.25 Удаление плейлиста.....	69
5.26 Удаление любых комментариев администратором.....	69
5.27 Скрытия видео администратором	70
5.28 Удаление видео администратором	70
5.29 Просмотр жалоб пользователей администратором	71
5.30 Блокировка канала администратором	71
5.31 Удаление канала администратором	71
6 Технико-экономическое обоснования проекта	73
6.1 Общая характеристика разрабатываемого программного средства	73
6.2 Исходные данные для проведения расчётов и маркетинговый анализ	73
6.3 Обоснование цены программного средства	74
6.3.1 Расчёт затрат рабочего времени на разработку программного средства	74
6.3.2 Расчет основной заработной платы.....	75
6.3.3 Расчет дополнительной заработной платы.....	76
6.3.4 Расчет отчислений в Фонд социальной защиты населения и по обязательному страхованию	76
6.3.5 Расчет суммы прочих прямых затрат.....	77
6.3.6 Расчет суммы накладных расходов.....	77
6.3.7 Сумма расходов на разработку программного средства.....	78
6.3.8 Расходы на сопровождение и адаптацию	78
6.3.9 Расчет полной себестоимости.....	78
6.4 Вывод по разделу	79
Заключение	82
Список используемых источников	83
Диаграмма вариантов использования (ДП 01.00.ГЧ)	84
Логическая схема базы данных (ДП 02.00.ГЧ)	85
Диаграмма развертывания (ДП 03.00.ГЧ)	86
Блок-схема подключения к совместному просмотру (ДП 04.00.ГЧ)	87
Диаграмма компонентов (ДП 05.00.ГЧ)	88
Скриншот работы приложения (ДП 06.00.ГЧ)	89
Приложение А	90
Приложение Б	93

Приложение В.....	95
Приложение Г	100
Приложение Д.....	103

Введение

Путешествия всегда привлекали людей — это способ узнать для себя что-то новое, получить впечатления или просто отдохнуть. В современном мире появляется всё больше возможностей для путешествий. Это связано с увеличением количества и качества представления туристических услуг. Многие сталкиваются с необходимостью в помощи подбора направлений, которые соответствуют личным интересам и предпочтениям, без необходимости самостоятельного изучения географических характеристик различных мест.

Цель дипломного проекта — разработать веб-приложение, позволяющее гибко подбирать туры по географическим признакам и их бронировать.

Серверная часть реализована на платформе ASP.NET Core 8.0 [1], клиентская часть — с использованием библиотеки React [2]. В качестве системы управления базами данных используется MariaDB [4], с подключением через Entity Framework Core [5]. Для обеспечения возможности оплаты используется Stripe [6]. Для отображения месторасположения объектов на карте используется OpenStreetMap [7]. Для подборов туров с помощью ИИ используется Gemini [8]. Для передачи данных используется REST API [9]. Приложение развёрнуто в Docker [10].

Основные этапы работы:

- постановка задачи и аналитический обзор аналогичных решений;
- проектирование веб-приложения;
- разработка веб-приложения;
- тестирования веб-приложения;
- руководство пользователя (руководство по эксплуатации);
- технико-экономическое обоснование проекта.

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

Приложение предназначено для широкого круга пользователей, включая:

- любителей путешествовать, которые хотят найти предпочитаемый тур и комфортно его забронировать;
- владельцев отелей транспортных компаний, которые могут продвигать свои услуги через приложение;
- менеджеров, обеспечивающие предоставление туристических услуг и организацию туров.

Приложение будет доступно в сети интернет и будет работать через браузер.

					ДП 02.00.ПЗ		
		ФИО	Подпись	Дата			
Разраб.		Иванов И.И.			2 Работа редактора над журнальным изданием		
Пров.		Руководитель					
Н. контр.		Нистюк О.А.					
Утв.		Смелов В.В.					
					Лит.	Лист	Листов
						1	8
					БГТУ 1-40 01 01, 2025		

1 Постановка задачи и аналитический обзор аналогичных решений

1.1 Постановка задачи

Необходимо разработать веб-приложение, которое поможет людям в организации путешествий. Веб-приложение должно предоставлять пользователям туры для бронирования. В бронь тура будет включена бронь номеров в отеле и транспорта. Пользователи должны иметь возможность удобно подбирать туры по различным критериям и географическим признакам. Необходимо предоставить возможность менеджерам в добавлении, удалении и редактировании туров, отелей, пунктов отправлений, городов и стран. Так же они должны подтверждать заявки на бронирование туров. Необходима модерация, чтобы следить за, тем, чтобы отзывы пользователей не были оскорбительными и неприличными, а также следить за тем, чтобы пользователи не спамили бронями. Для этого необходимы администраторы, которые смогут удалять неприемлемые отзывы, а также блокировать или удалять учётные записи пользователей.

1.2 Обзор аналогичных решений

1.2.1 Веб-приложение «People Travel»

Веб-приложение People Travel [11] предоставляет пользователям широкий спектр возможностей для подбора нужного тура. В первую очередь, подбор туров происходит по странам, которые предоставляются пользователям на главной странице в виде флажков и названий, затем уже пользователи более подробно настраивают себе отображение туров в фильтрах. Есть несколько видов туров. При просмотре отдельных туров, пользователю предоставляется очень подробная информация о нём.

Но при этом у дизайна сайта есть ряд проблем. При заходе на сайт, сразу почти на весь экран отображается реклама. Для взаимодействия с основным функционалом, необходимо совершить ряд действий, такие как прокрутка вниз и выбор страны. Так же интерфейс не единообразный, то есть у разных стран, разные виды туров, и при их выборе, настройки отображения туров будут разными, что создаёт путаницу.

Кроме того, при бронировании тура, нельзя указать какую-либо информацию, кроме своих контактных данных. Это всё создаёт ряд сложностей тем, что менеджерам приходится самостоятельно узнавать у пользователей эту информацию.

					ДП 02.00.ПЗ							
		ФИО	Подпись	Дата								
Разраб.	Иванов И.И.				2 Работа редактора над журнальным изданием			Лит.	Лист	Листов		
Пров.	Руководитель									1	8	
								БГТУ 1-40 01 01, 2025				
Н. контр.	Нистюк О.А.											
Утв.	Смелов В.В.											

Интерфейс веб-приложения представлен на рисунке 1.1.

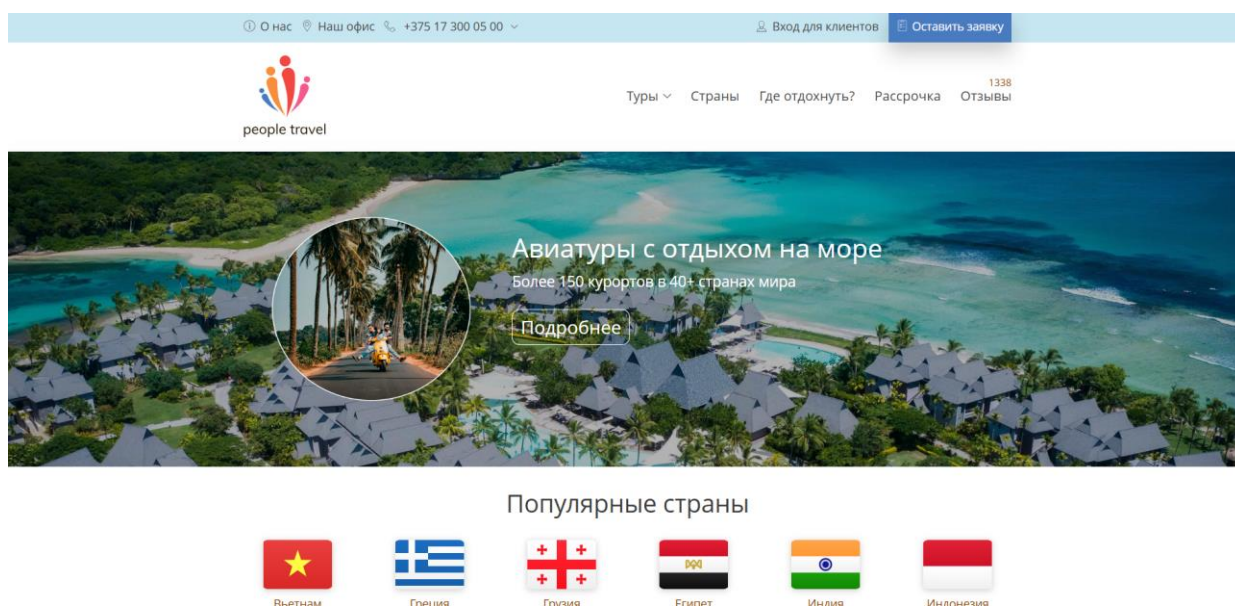


Рисунок 1.1 – Веб-приложение «People Travel»

Проанализировав веб-приложение, было решено сделать похожим дизайн интерфейса подбора стран.

1.2.2 Веб-приложение «Coral Travel»

Coral Traver [12] предоставляет возможность подбора туров, рейсов и отелей. Заказать можно как пакетный тур, в который входит бронь транспорта и отеля, так и просто отель. Фильтрация технически реализована удобно – есть множество возможностей для подбора туров, к примеру, можно указать количество детей и их возраст для подбора туров с отелями, в которых есть возможность размещения таких детей, а также предоставляют для них скидки. Есть история подборов туров. Однако у фильтров есть проблемы в дизайне, и заключаются они в том, что в большинстве случаев это сплошной текст. Например, при выборе направлений, города и страны предоставляются в виде списка, а большое их количество, приводит к тому, что их неудобно выбирать.

При просмотре отдельного тура, можно увидеть подробную информацию о нём. При чём, описание многих характеристик представлено визуально в виде значков или фотографий с отеля. При бронировании тура, можно выбрать конкретный тип номера. Однако нельзя заказать несколько номеров. Таким образом, забронировать мест в туре можно столько, сколько людей помещается в номер, что означает, что, если будет больше людей, значит придётся делать несколько заказов.

Важной особенностью является возможность сравнивать отели. После добавления отелей в сравнение, можно просмотреть, в чём их отличия по конкретным характеристикам.

Интерфейс веб-приложения представлен на рисунке 1.2.

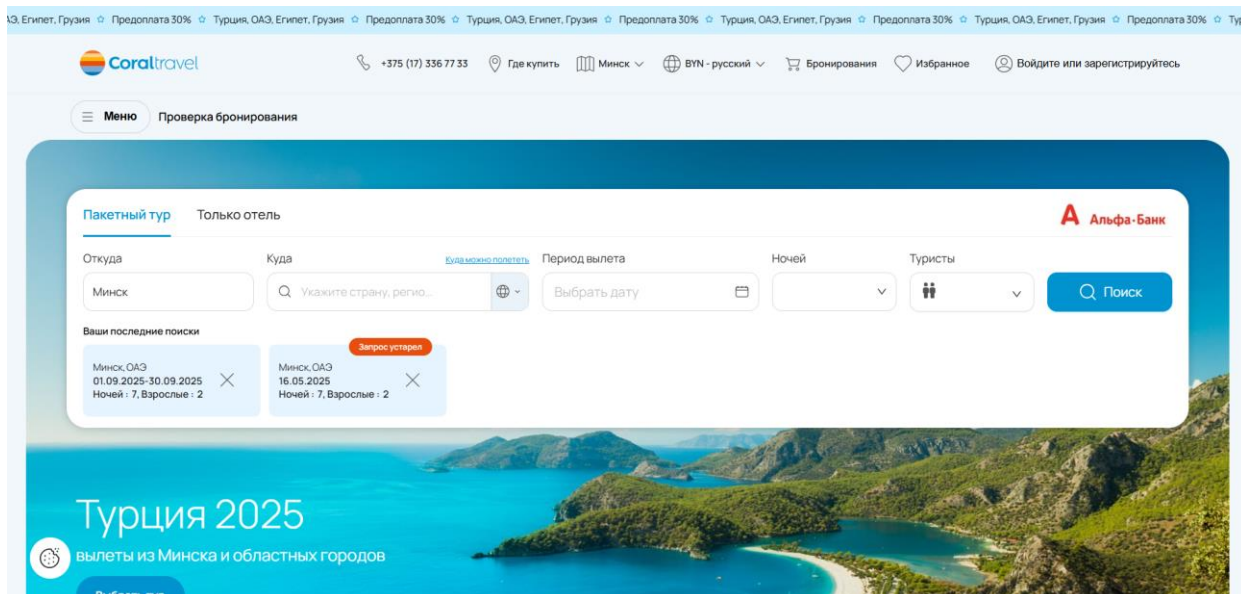


Рисунок 1.2 – Веб-приложение «Coral Travel»

Проанализировав, веб-приложение было решено делать схожую страницу для тура.

1.2.3 Веб-приложение «Tez Tour»

Веб-приложение «Tez Tour» [13] на главной странице сайта сразу предоставляет возможность удобно настраивать отображение туров по основным критериям. На сайте можно посмотреть информацию о деталях туров, таких как отели, транспорт, документы т.д. Очень подробную информацию можно узнать о странах, такую как экскурсии в ней, советы по получению виз, достопримечательности, климат и т.д. Самые популярные страны представлены на главной странице в виде фотографий их достопримечательностей. Ниже стран представлены популярные туры. Туры разделены на типы. Эти типы удобно отображаются в виде значков и текстов.

При просмотре отдельного тура, можно увидеть довольно подробную информацию о нём, однако она расположена не удобно и представляет с собой набор хаотично разбросанных характеристик. При бронировании необходимо выбрать тип номера и рейс. Рейсы отображаются на странице тура под каждым типом номера, что не очень удобно, ведь ко всем типам номеров и так одинаковые рейсы, тем самым они просто дублируются и заполняют место ненужной информацией.

Данное веб-приложение повторяет проблему предыдущего, что в нём нельзя заказать несколько номеров в отеле.

В целом подбор туров довольно удобный визуально и технически, однако сама страница тура переполнена ненужной информацией, которая при этом плохо визуально воспринимается.

Интерфейс веб-приложения представлен на рисунке 1.3.

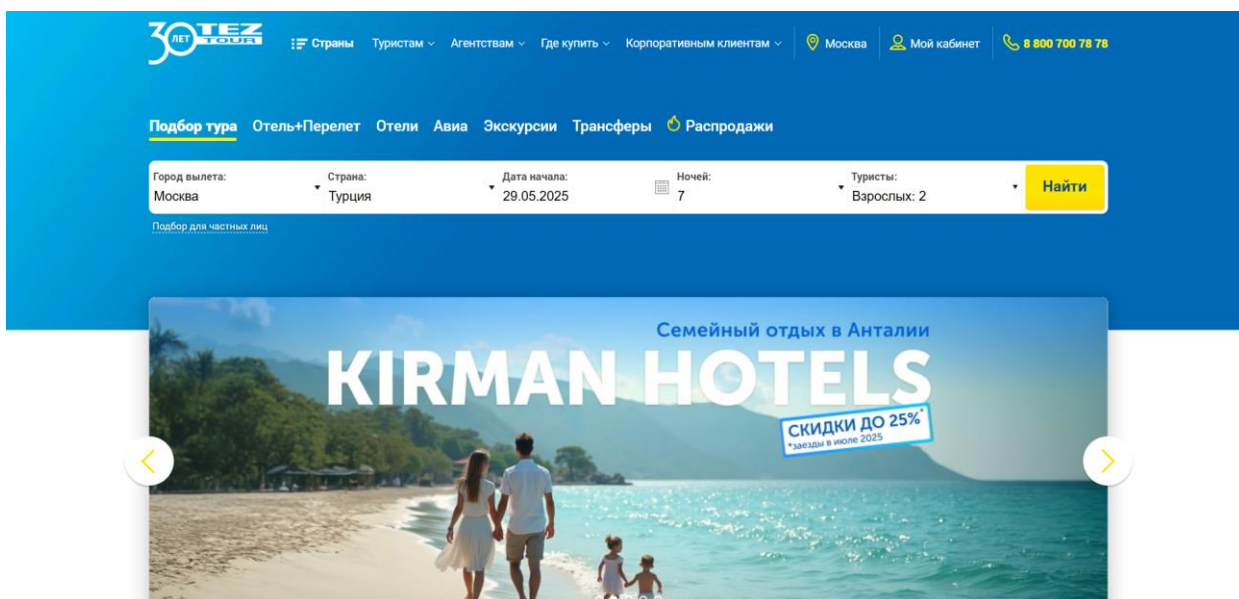


Рисунок 1.3 – Веб-приложение «Tez Tour»

Проанализировав веб-приложение, было решено взять дизайн разделения туров на типы.

1.3 Выводы по разделу

В рамках данной главы было выполнено:

1. Поставлены необходимые задачи для реализации веб-приложения;
2. Разобраны три аналогичных решения: «People Travel», «Coral Travel», «Tez Tour». Из «People Travel» взят интерфейс подбора туров по странам, из «Coral Travel» взят дизайн страницы тура, а из «Tez Tour» взято разделение туров на типы.
3. Было отмечено, что ни в одном из аналогов не имеется возможность подбора туров по географическим признакам, а также выбора нескольких номеров в отеле при бронировании.

2 Проектирование веб-приложения

2.1 Функциональность веб-приложения

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

Описание ролей пользователей web-приложения представлены в таблице 2.1.

Таблица 2.1 – Описание ролей

Роль	Описание
Гость	Пользователь, не прошедший авторизацию. Имеет доступ только к просмотру туров.
Клиент	Зарегистрированный пользователь, имеющий возможность взаимодействовать с контентом.
Менеджер	Может добавлять, удалять, редактировать туры, отели, пункты отправлений, города и страна, а также подтверждать заявки на бронирование туров.
Администратор	Может удалять отзывы к турам, а также удалять, блокировать пользователей и изменять им роль. Так же может выполнять функции менеджера.

Роли в системе разделены таким образом, чтобы разграничить доступ к функционалу приложения и обеспечить безопасное использование платформы. Каждая роль наделена строго определённым 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	Регистрация	Возможность создания нового аккаунта для получения доступа к расширенному функционалу.	Гость
2	Авторизация	Вход в систему для использования персонализированного функционала.	Гость
3	Просмотр маршрутов	Возможность просмотра маршрутов.	Гость, Пользователь, Менеджер, Администратор

					ДП 02.00.ПЗ		
		ФИО	Подпись	Дата			
Разраб.	Иванов И.И.				2 Работа редактора над журнальным изданием		
Пров.	Руководитель						
Н. контр.	Нистюк О.А.						
Утв.	Смелов В.В.						
					Лит.	Лист	Листов
						1	8
					БГТУ 1-40 01 01, 2025		

Продолжение таблицы 2.2

№	Функция	Описание	Доступно ролям
4	Просмотр отелей	Возможность просмотра отелей.	Гость, Пользователь, Менеджер, Администратор
5	Фильтрация маршрутов	Возможность просмотра списка маршрутов с выбранными характеристиками	Гость, Пользователь, Менеджер, Администратор
6	Просмотр стран	Возможность просмотра стран.	Гость, Пользователь, Менеджер, Администратор
8	Подбор маршрутов по опросу	Возможность просмотра списка маршрутов, соответствующих результатам опроса	Гость, Пользователь, Менеджер, Администратор
9	Подбор маршрутов с помощью ИИ	Возможность просмотра списка маршрутов, соответствующих введённому описанию	Гость, Пользователь, Менеджер, Администратор
10	Оставление отзывов к турам	Возможность оставить отзыв к выбранному туру	Пользователь
11	Подача заявки на бронирование тура	Возможность отправить заявку на бронирование выбранного тура	Пользователь
12	Оплата забронированного тура	Возможность оплатить подтверждённый забронированный тур	Пользователь
13	Просмотр истории бронирований	Возможность просмотра списка броней	Пользователь
14	Отмена заявки на бронирование тура	Возможность отменить бронь тура	Пользователь, Менеджер, Администратор
15	Редактирование профиля	Возможность изменения контактных данных своей учётной записи	Пользователь, Менеджер, Администратор
16	Добавление туров	Возможность создания новых туров	Менеджер, Администратор
17	Редактирование туров	Возможность изменения данные существующих туров.	Менеджер, Администратор
18	Удаление туров	Возможность удаления существующих туров	Менеджер, Администратор
19	Добавление отелей	Возможность создания новых отелей	Менеджер, Администратор
20	Редактирование отелей	Возможность изменение данных существующих отелей.	Менеджер, Администратор
21	Удаление отелей	Возможность удаления существующих отелей	Менеджер, Администратор

Продолжение таблицы 2.2

№	Функция	Описание	Доступно ролям
22	Добавление пунктов отправлений	Возможность создания новых пунктов отправления	Менеджер, Администратор
23	Редактирование пунктов отправлений	Возможность изменение данных существующих пунктов отправлений.	Менеджер, Администратор
24	Удаление пунктов отправлений	Возможность удаления существующих пунктов отправлений	Менеджер, Администратор
25	Добавление стран	Возможность изменение данных существующих стран.	Менеджер, Администратор
26	Редактирование стран	Возможность удалять любые комментарии	Менеджер, Администратор
27	Удаление стран	Возможность удаления существующих стран	Менеджер, Администратор
28	Смена цены забронированного тура	Возможность изменения цены у забронированного тура	Менеджер, Администратор
29	Удаление отзывов к турам	Возможность удаления у туров отзывов, нарушающих правила	Администратор
30	Смена роли пользователей	Возможность блокировки у пользователя всех возможностей, кроме	Администратор
31	Разблокировка пользователей	Возможность восстановления прав пользователям	Администратор
32	Удаление пользователей	Возможность удаление существующих учётных записей	Администратор
33	Блокировка пользователей	Блокировка у пользователя всех возможностей, кроме просмотра туров	Администратор

Функции приложения разделены между ролями таким образом, чтобы обеспечить удобство использования для каждой категории пользователей. Например, гости могут только просматривать и искать туры, что снижает нагрузку на систему, так как им не нужно хранить данные о своих действиях. Пользователи, в свою очередь, обладают более широким спектром возможностей, включая бронирование туров. Менеджеры предоставляют пользователям туры и принимают заявки. Администраторы сосредоточены на обеспечении адекватного поведения пользователей

Диаграмма вариантов использования позволяет визуализировать все основные функции и роли веб-приложения. 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.

2.2 Логическая схема базы данных

Логическая схема базы данных приведена в ДП 02.00.ГЧ. База данных содержит 8 таблиц, описание таблиц базы данных предоставлено в таблице 2.3.

Таблица 2.3 – Описание таблиц базы данных

Таблица	Описание
Roles	Хранит данные о ролях пользователей.
Users	Хранит данные о пользователях.
Tours	Хранит данные о турах.
Routes	Хранит данные о маршрутах.
Bookings	Хранит данные о бронях
TourTypes	Хранит данные о типах туров.
TourCharacteristics	Хранит данные о характеристиках туров.
TourDescriptions	Описывает связь между характеристиками и турами.
NutritionTypes	Хранит данные о типах питания.
Hotels	Хранит данные о отелях.
Cities	Хранит данные о городах
Countries	Хранит данные о странах.
Regions	Хранит данные о регионах мира.
DepartmentDepartures	Хранит данные о пунктах отправления.
Reviews	Хранит данные о отзывах к турам
TransportTypes	Хранит данные о типах транспортов
RoomTypes	Хранит данные о типах номеров отелей.
RoomTypeCharacteristics	Хранит данные о характеристиках типов номеров.
RoomTypeDescriptions	Описывает связь между характеристиками и типов номеров.
HotelCharacteristics	Хранит данные о характеристиках отелей.
HotelDescriptions	Описывает связь между характеристиками и отелями.
BookedRoomTypes	Хранит данные о забронированных типах номеров отелей.
Landmarks	Хранит данные о достопримечательностях.
Climates	Хранит данные о климатах.

Все таблицы разработаны для максимального эффективного хранения данных веб-приложения.

2.3 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 путём использования криптографического протокола TLS.
TCP [12]	Обмен данными между Database Server и Backend Server.

Данная архитектура обеспечивает максимально эффективную работу веб-приложения.

2.4 Блок-схема алгоритма бронирования тура

Блок-схема алгоритма бронирования тура приведена в ДП 04.00.ГЧ.

Данная блок-схема описывает полный процесс бронирования тура от выбора тура до его оплаты. После выбора тура, необходимо указать конкретный маршрут и заполнить форму оформления бронирования, в которой необходимо указать количество заказываемых номеров в отеле, количество людей, отправляющихся, в тур, при необходимости, указать наличие детей и выбрать приоритет рассадки в транспорте. При необходимости, можно оставить свои пожелания в поле комментария. После чего, отправить заявку на бронирование тура. Заявка будет отправлена в том случае, если пользователь авторизован, имеет роль «Пользователь», не заблокирован и данные введены корректно. В случае несоответствия одному из этих условий, выведется сообщение об ошибке. Если всё прошло успешно, то пользователю необходимо дождаться подтверждения брони менеджером. После подтверждения, необходимо выбрать данную бронь и оплатить её. После оплаты, тур будет считаться полностью забронированным.

2.5 Выводы по разделу

Таким образом в рамках проектирования веб-приложения было выполнено.

1. Разработана диаграмма вариантов использования, благодаря которой был определён какой функционал требуется приложению, а также какие роли будут иметь к нему доступ.

2. Спроектирована архитектура веб-приложения. Было определено каким образом будут размещены компоненты веб-приложения и какие протоколы будут использоваться.

3. Спроектирована база данных веб-приложения. Были определены таблицы, а также связи между ими.

4. Спроектирована блок схема алгоритма бронирования тура.

3 Разработка веб-приложения

Программное обеспечение – это больше, чем просто программный код. Программа представляет собой исполняемый код, который выполняет некоторые вычислительные задачи. Программное обеспечение считается коллекцией исполняемого программного кода, связанных с кодом библиотек и документации. Программное обеспечение, если оно изготовлено для конкретного требования, называется программным продуктом.

Разработка любого программного продукта – сложный и трудоемкий процесс, в ходе которого необходимо концентрироваться на большом количестве деталей, чтобы не допустить какую-либо критическую ошибку. Важно также следовать тем правилам, которые были заложены в ходе проектирования системы, ведь приложение, реализованное в соответствии с грамотно спроектированным решением – залог успеха всего проекта.

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

3.1 Разработка серверной части веб-приложения

3.1.1 Используемые библиотеки

Описание библиотек предоставлено в таблице 3.1.

Таблица 3.1 – Серверные зависимости

Библиотека	Описание
Microsoft.AspNetCore	Базовые компоненты фреймворка .NET Core.
Microsoft.AspNetCore.Session,	Библиотека, обеспечивающая работу сессий.
Microsoft.AspNetCore.StaticFiles	Библиотека, обеспечивающая работу с файлами и основные веб-функции.
Microsoft.AspNetCore.Authentication.JwtBearer	Библиотека для аутентификации пользователей с использованием JWT.
System.IdentityModel.Tokens.Jwt	Библиотека для работы с JWT-токенами.
Pomelo.EntityFrameworkCore.MySql	Библиотека ORM Entity Framework Core для взаимодействия с базой данных.
Newtonsoft.Json	Библиотека для работы с данными в JSON формате.
Stripe.net	Библиотека для взаимодействия с платёжной системой.

					ДП 02.00.ПЗ						
		ФИО	Подпись	Дата							
Разраб.		Иванов И.И.			2 Работа редактора над журнальным изданием			Лит.	Лист	Листов	
Пров.		Руководитель								1	8
Н. контр.		Нистюк О.А.						БГТУ 1-40 01 01, 2025			
Утв.		Смелов В.В.									

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

3.1.2 Структура серверной части

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

Для обеспечения упорядоченности серверного кода, было решено отказаться от простейшей монолитной архитектуры в пользу довольно простой N-layer [14] архитектуры, с выделением трех основных слоев: уровень доступа к данным, уровень бизнес-логики, уровень представления.

Структура проекта предоставлена на рисунке 3.1.

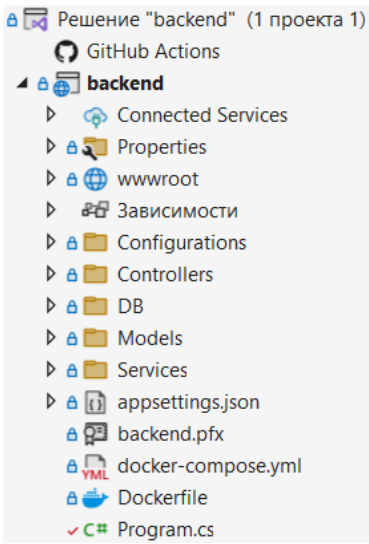


Рисунок 3.1 – Структура проекта

Описание папок входящих в структуру проекта предоставлено в таблице 3.2.

Таблица 3.2 – Папки входящие в проект

Папка	Описание
Configurations	Содержит файлы с конфигурациями, например, для настройки аутентификации с помощью JWT токенов.
Controllers	Содержит контроллеры, которые обрабатывают HTTP-запросы и возвращают ответы клиенту.
DB	Содержит контекст базы данных для Entity Framework.
Models	Содержит все виды моделей: DTO, данные из форм и модели Entity Framework.
Services	Содержит файлы, которые реализуют бизнес-логику для вспомогательных действий в контролерах.
wwwroot	Содержит статические файлы, необходимые для работы клиентской части (bundle.js)

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

3.1.3 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 её свойства – столбцам этой таблицы. В листинге 3.1 предоставлен пример сущности.

```
public class Tour
{
    [Key]
    public int Id { get; set; }
    [Required]
    [MaxLength(255)]
    public string Name { get; set; }
    [MaxLength(255)]
    public string MainDescription { get; set; }
    public int? TourTypeId { get; set; }
    public int? HotelId { get; set; }
    [ForeignKey(nameof(TourTypeId))]
    public TourType TourType { get; set; }
    [ForeignKey(nameof(HotelId))]
    public Hotel Hotel { get; set; }
    public ICollection<TourCharacteristic> Characteristics {
get; set; }
    public ICollection<Route> Routes { get; set; }
    public ICollection<Review> Reviews { get; set; }
}
```

Листинг 3.1 – Сущность Tour

EF автоматически генерирует SQL-запросы на основе моделей данных, обеспечивая их выполнение в базе данных MariaDB. В листинге 3.2 приведён пример получения объектов при помощи Entity Framework Core.

```
List<Tour> tours = await db.Tours
    .Include(t => t.TourType)
    .Include(t => t.Hotel)
    .ThenInclude(h => h.City)
    .ThenInclude(c => c.Country)
    .ToListAsync();
```

Листинг 3.2 – Пример выполнения запроса в базу данных

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.

3.1.4 Разработка уровня бизнес-логики

При разработке бизнес-логики для WebAPI на C# одним из ключевых аспектов является правильное использование сервисов и паттернов внедрения зависимостей.

В основном бизнес-логика реализована в контроллерах. Реализация выполнения некоторых задач, которые выполняются во многих местах, были вынесены в сервисы. Данные сервисы внедряются в контролеры при помощи механизма DI, который автоматически управляет созданием объектов и их зависимостями. Контроллеры могут получать сервисы через конструктора. Пример внедрения сервисов для определения погоды в туре, взаимодействия с ИИ и взаимодействия с контекстом базы данных, предоставлен в листинге 3.4.

```
private AppDbContext db;
private IWeatherService _weatherService;
private IAIService _aiService;

public TourController(AppDbContext context, IWeatherService
weatherService, IAIService aiService)
{
    db = context;
    _weatherService = weatherService;
    _aiService = aiService;
}
```

Листинг 3.4 – пример передачи зависимостей через конструктор

Таким образом было организовано использование бизнес-логики.

3.1.5 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 Controller, который предоставляет доступ к набору стандартных методов для обработки запросов. Каждый контроллер связан с определённым маршрутом, который указывает на URL, к которому привязан данный контроллер.

Пример контроллера предоставлен в листинге 3.5.

```

[Route("api/[controller]")]
[Route("review")]
public class ReviewController : Controller
{
    private AppDbContext db;

    public ReviewController(AppDbContext context)
    {
        db = context;
    }

    [HttpGet("reviews")]
    public async Task<IActionResult> GetReviews([FromQuery] int?
tourId)
    {...}

    [Authorize(Roles = "User")]
    [HttpPost("add")]
    public async Task<IActionResult> AddReview([FromBody]
ReviewForm review)
    {...}

    [Authorize(Roles = "Admin")]
    [HttpDelete("delete")]
    public async Task<IActionResult> DeleteReview([FromQuery]
int? tourId)
    {...}
}

```

Листинг 3.5 – Пример контроллера

Контроллеры могут выполнять бизнес-логику, однако реализация выполнения частых задач, должна быть вынесена в сервисы.

3.1.6 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 токенов.

Создание токенов обычно осуществляется с использованием библиотеки System.IdentityModel.Tokens.Jwt, которая предоставляет все необходимые инструменты для работы с JSON Web Token (JWT) [16]. Access token представляет собой короткоживущий токен, который используется для предоставления доступа к защищённым ресурсам.

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 задаёт ограничение по времени, что делает токен недействительным после периода.

Код создание токенов предоставлен в листинге 3.6.

```
public static string GenerateToken(string email, UserRole role)
{
    List<Claim> claims = new List<Claim> { new
    Claim(ClaimTypes.Email, email), new Claim(ClaimTypes.Role,
    role.ToString()) };

    JwtSecurityToken token = new JwtSecurityToken(
        issuer: AuthOptions.ISSUER,
        audience: AuthOptions.AUDIENCE,
        claims: claims,
        expires: DateTime.UtcNow.Add(TimeSpan.FromHours(8)),
        signingCredentials: new
    SigningCredentials(AuthOptions.GetSymmetricSecurityKey(),
    SecurityAlgorithms.HmacSha256)
    );

    return new JwtSecurityTokenHandler().WriteToken(token);
}
```

Листинг 3.6 – Код создание токенов

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

Таким образом, настройка безопасности через 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.

3.1.7 Используемые сторонние сервисы

Для реализации некоторого функционала на серверной части, использовались API сторонних сервисов:

- Stripe — это сервис, который обеспечивает онлайн-платежи. Он позволяет принимать и обрабатывать платежи по банковским картам, Apple Pay, Google Pay и другим способам через интернет. Взаимодействие с данным сервисом через API, позволяет реализовать оплату бронированных туров. Пользователь вводит данные банковской карты, после чего отправляет их на сервер. Сервер отправляет эти данные сервису Stripe, после чего он выполняет снятие денежных средств с банковской карты пользователя;

– Gemini — это сервис, который предоставляет возможность использования ИИ. Взаимодействие с данным сервисом через API, позволяет реализовать подбор туров при помощи ИИ. Пользователь вводит описание городов или местностей, после чего отправляет его на сервер. Сервер отправляет это описание сервису Gemini, вместе с промптом, указывающий ИИ, что бы он подобрал города, подходящие под описание. ИИ подбирает города, и Gemini высылает их обратно на сервер. Сервер ищет в базе данных туры с полученными города и высылает их пользователю. Подбор туров с помощью ИИ представлен в диаграмме последовательности, которая приведена в ДП 05.00.ГЧ.

3.2 Клиентская часть

3.2.1 Используемые библиотеки

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.
@mui/material, @mui/icons-material, @mui/lab,	Визуальные компоненты и иконки для создания интерфейса.
axios	HTTPS-клиент для общения с сервером
reduxjs/toolkit, react-redux, redux	Библиотеки для управления глобальным состоянием приложения.
stripe/react-stripe-js, stripe/stripe-js	Библиотеки для отображения UI элементов и работы с данными платёжной системы Stripe
dayjs	Библиотека для упрощения работы с датами.
react-leaflet	Библиотека, которая позволяет удобно использовать интерактивные карты

Данные библиотеки полностью закрывают необходимости клиентской части.

3.2.2 Структура клиентской части

Структура проекта организованна таким образом, чтобы все файлы были сгруппированы внутри корневой папки src в директории по их назначению. Структура клиентской части предоставлена на рисунке 3.2.

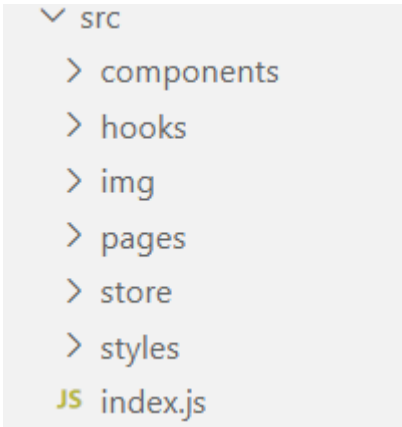


Рисунок 3.2 – Структура клиентской части

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

Таблица 3.4 – Директории входящие в проект

Директория	Описание
components	Содержит компоненты элементов страниц.
hooks	Содержит дополнительные хуки
img	Содержит изображения элементов, используемых на страницах
pages	Содержит компоненты страниц.
store	Содержит глобальные данные, которые могут использоваться во всех компонентах
styles	Содержит файлы стилей

Данная структура обеспечивает удобный доступ ко всем файлам проекта.

3.2.3 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.

3.3 Реализация функций

3.3.1 Регистрация

Метод «Register» отвечает за регистрацию нового пользователя. Сначала он проверяет, существует ли пользователь с переданным в метод email, если существует, то возвращает ответ с 409 статусом. Если нет, то хеширует переданный в метод пароль сохраняет учётные данные пользователя в базу данных. Затем возвращает клиенту токен.

```

[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 существует"]);

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

```

Листинг 3.7 – Реализация регистрации

Таким образом была реализована регистрация в веб-приложении.

3.3.2 Авторизация

Если у пользователя нет токена, ему необходимо авторизоваться. При авторизации, запрос будет отправлен к методу «Login», которые будет искать учётную запись в базе данных, соответствующих переданных в него логину и паролю, и в случае, если запись будет найдена, то пользователю вернётся токен, иначе ответ со статусом 401.

Если у пользователя есть токен, то запрос будет отправлен к методу «Auth», предназначенного для предоставления пользователю его учётных данных по токену. У метода установлен атрибут Authorize, который проверяет корректность полученного токена. Метод получает из базы данных данные пользователя по его email, полученного из токена и возвращает эти данные клиенту. Если пользователь не будет найден, или токен не корректен, то возвращает ответ с 401 статусом.

Листинг реализации метода предоставлен в приложении В.

3.3.3 Просмотр публичных видео

Для просмотра публичных видео в рамках контроллера «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 диапазоновых 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.

3.3.4 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 В.

3.3.5 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
Ignore	Список видео, которые нужно игнорировать
SearchPattern	Паттерн название
Take	Длина одной страницы (нужен для пагинации)
Skip	Сколько элементов пропустить (нужен для пагинации)

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

Возвращает массив моделей видео в формате JSON с 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.

3.3.6 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 с кодом 200 OK.

3.3.7 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 с кодом 200 OK.

3.3.8 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 с кодом 200 OK.

3.3.9 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.

3.3.10 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.

```
[HttpPost, 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 принадлежит!",
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 метода «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.

3.3.11 Создание канала

Для создания канала в рамках 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 метода «Post» контроллера «ChannelController»

Таким образом реализована создание канала клиентом.

3.3.12 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 channelId, 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.

3.3.13 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.

```

[HttpPut, Authorize]
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 принадлежит",
                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 ОК. При возникновении ошибок возвращается JSON-ответ с кодом ошибки и описанием проблемы.

3.3.14 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 принадлежит",
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 ОК, подтверждающий успешное удаление. При возникновении исключений клиент получает JSON-ответ с соответствующим описанием ошибки и кодом ответа.

3.3.15 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 принадлежит",
            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 метода «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.

3.3.16 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 её описании. Внутри метода выполняется проверка корректности переданного идентификатора видео. Если идентификатор невозможно распознать как GUID, выбрасывается

исключение «ServerException» с соответствующим сообщением. При успешной валидации создаётся 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 ОК в случае успешной обработки жалобы. В случае возникновения ошибки возвращается сериализованный объект ошибки с соответствующим кодом.

3.3.17 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 создаётся 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 ОК при успешном добавлении комментария. В случае ошибки возвращается сериализованный объект исключения с соответствующим статусом.

3.3.18 Удаление своих комментариев

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

3.3.19 Редактирование своих комментариев

Для редактирования своих комментариев в рамках контроллера «CommentController» был разработан метод «Put». Метод принимает модель «CommentModel» с обновлённым 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
        принадлежит", 403);

        comment.Message = model.Message;

        _commentRepository.Update(comment);

        await _commentRepository.SaveChangesAsync();

        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 ОК. При возникновении исключений возвращается сериализованный объект ошибки с соответствующим кодом состояния.

3.3.20 Совместный просмотр

Для реализации совместного просмотра был разработан класс «WatchTogetherHub». Данный класс содержит специальные методы

необходимые для корректной работы совместного просмотра. Описание методов предоставлено в таблице 3.5.

Таблица 3.5 – Описание методов «WatchTogetherHub»

Метод	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 командами. 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 конкретной 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 приватную комнату по паролю

Классы «WatchTogetherHub» и «WatchTogetherController» предоставляют весь необходимый функционал для совместного просмотра.

3.3.21 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 создаётся 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 ОК при успешном создании плейлиста. В случае возникновения ошибки возвращается сериализованный объект исключения с соответствующим статусом.

3.3.22 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.

3.3.23 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 создаётся 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.

3.3.24 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 = AuthorizedData.FromContext(HttpContext);
        Playlist playlist = await
_repository.GetPlaylistWithVideo (playlistId)
        ?? throw new ServerException("Плейлист не найден!",
404);
        if (playlist.Access == Playlist.AccessType.Private)
        {
            if (authData.IsAuthorize)
            {
                if (authData.Id != playlist.UserId)
                throw new ServerException("Видео плейлиста не доступны!",
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.

3.3.25 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 ОК. В случае ошибок возвращается JSON объект исключения с соответствующим кодом состояния.

3.3.26 Удаление любых комментариев администратором

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

3.3.27 Скрытие 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 ОК. В случае возникновения исключений возвращается JSON сообщение об ошибке с соответствующим кодом состояния.

3.3.28 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 ОК. В случае ошибок возвращается JSON сообщение об исключении с соответствующим кодом состояния.

3.3.29 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 её 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.

3.3.30 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 ОК.

3.3.31 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 передаётся 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 защищён авторизацией и доступен исключительно пользователям с ролью администратора. При успешном выполнении операции клиенту возвращается стандартный HTTP-ответ с кодом 200 ОК, сигнализирующий об успешном завершении удаления.

3.4 Маршруты для контролеров

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

3.5 Выводы по разделу

Таким образом в рамках реализации веб-приложения было выполнено.

1. Реализована 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;

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

3. 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 внедрён функционал для совместного просмотра видео, позволяющий нескольким пользователям синхронно смотреть одно и то же видео с возможностью чата и взаимодействия в реальном времени;

4. 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.

4 Тестирование веб-приложения

4.1 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	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 осуществлён.	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

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.							

Продолжение таблиц 4.1

№	Название функции	Описание теста	Ожидаемый результат	Результат
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.	Успешно
15	Удаление комментариев под своими видео	Зайти на страницу своего видео. Выбрать комментарий. Нажать «Удалить». Подтвердить действие.	Комментарий удалён.	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 удалён.	Успешно

Продолжение таблиц 4.1

№	Название функции	Описание теста	Ожидаемый результат	Результат
19	Редактирование своих комментариев	Зайти на страницу своего видео. Выбрать комментарий. Нажать «Редактировать». Изменить текст. Нажать	Комментарий обновлён.	Успешно
20	Совместный просмотр	Нажать кнопку совместный просмотр. Заполнить необходимые данные. Нажать кнопку «Создать комнату».	Комната создана. Другие пользователи могут подключаться.	Успешно
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 удалён.	Успешно
26	Удаление любых комментариев администратором	Зайти в админ-панель. Выбрать видео. Перейти в список комментариев. Выбрать комментарий. Нажать «Удалить».	Комментарий удалён.	Успешно
27	Скрытие 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

4.2 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.

4.3 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 протестировано на стрессоустойчивость.

4.4 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 одновременно. Таким образом покрытость тестами составила около 80%. Это обеспечивает уверенность в том, что приложение удовлетворит обеспечивая стабильность и удобство работы.

5 Руководство пользователя (руководство по эксплуатации)

5.1 Регистрация

Что бы перейти 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.

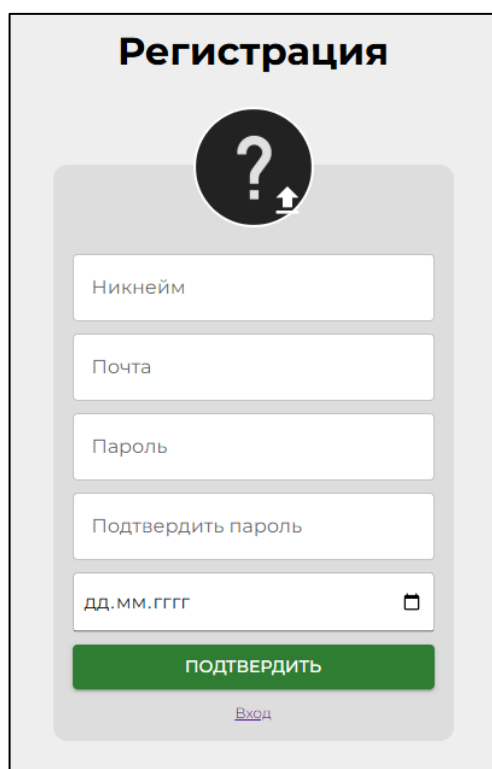


Рисунок 5.1 – Форма регистрации

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 произойдёт регистрация в веб-приложения.

5.2 Авторизация

Что бы перейти 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. Первая часть формы регистрации предоставлена на рисунке 5.2.

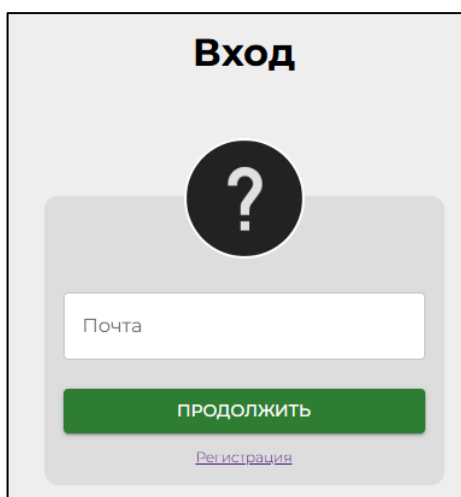


Рисунок 5.2 – Первая часть формы авторизации

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

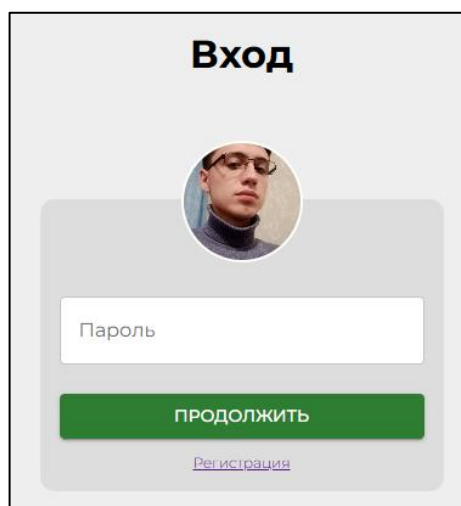


Рисунок 5.3 – Вторая часть формы авторизации

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

5.3 Просмотр публичных видео.

Что бы 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.

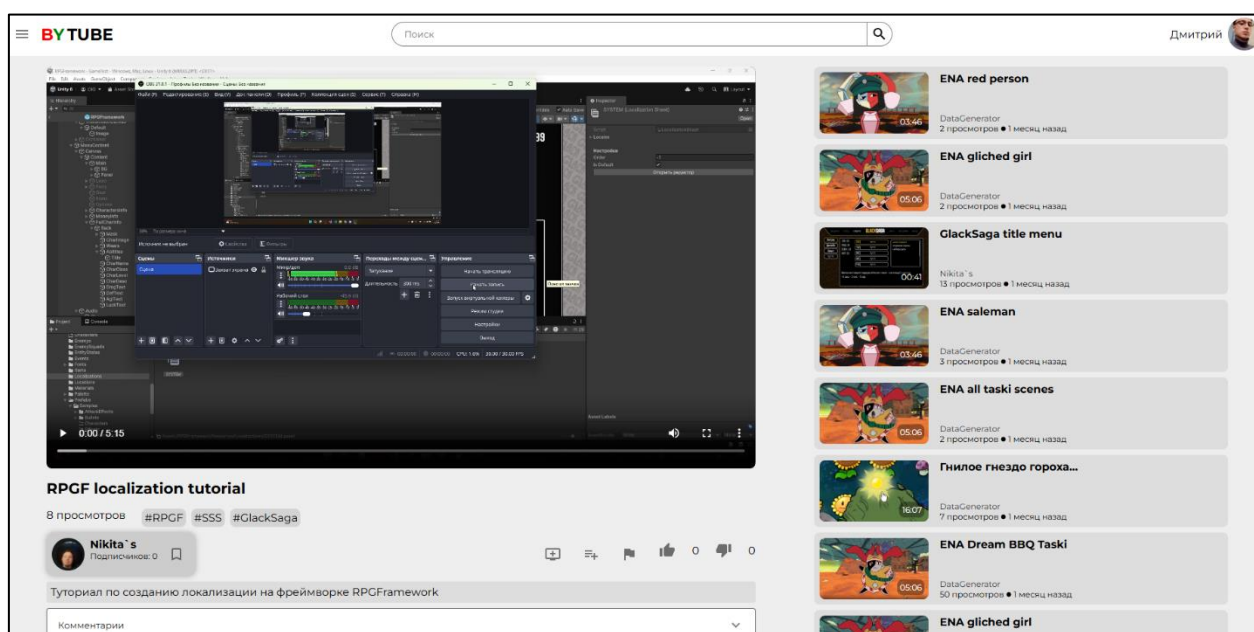


Рисунок 5.4 – Страница для просмотра видео

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

5.4 Просмотр комментариев под видео

Что бы посмотреть комментарии под видео для начала надо перейти на страницу видео. Скриншот со странице видео предоставлена на рисунке 5.4. Далее требуется раскрыть меню «Комментарии», скриншот предоставлен на рисунке 5.5

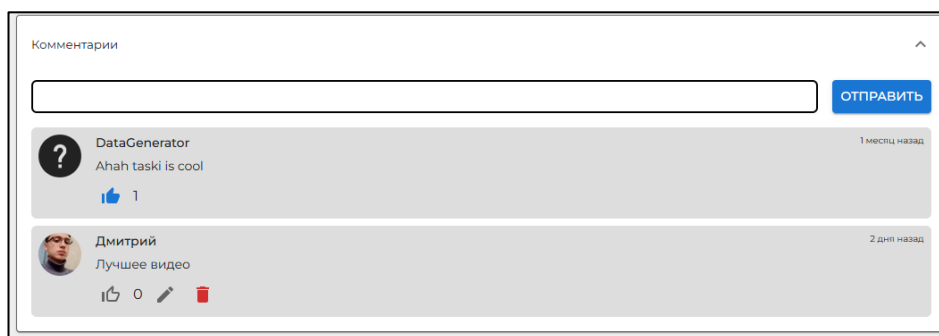


Рисунок 5.5 – Меню с комментариями

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

5.5 Поиск публичных видео

Что бы 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.

5.6 Поиск по названию

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

5.7 Сортировка видео

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

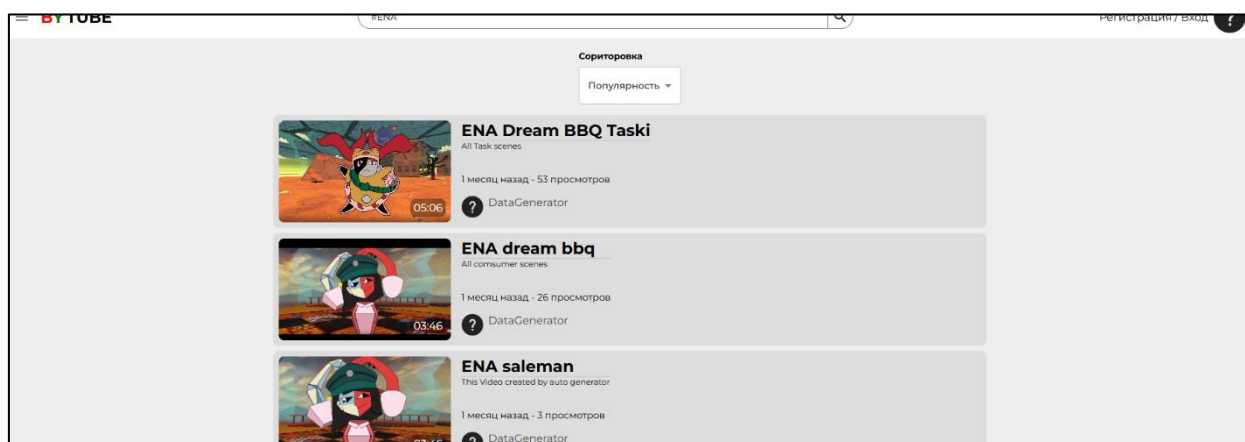


Рисунок 5.6 – Страница с результатом поиска

Что бы изменить вариант сортировки надо изменить значение поля «Сортировка».

5.8 Поиск по тегам

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

5.9 Удаление канала

Что бы 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.

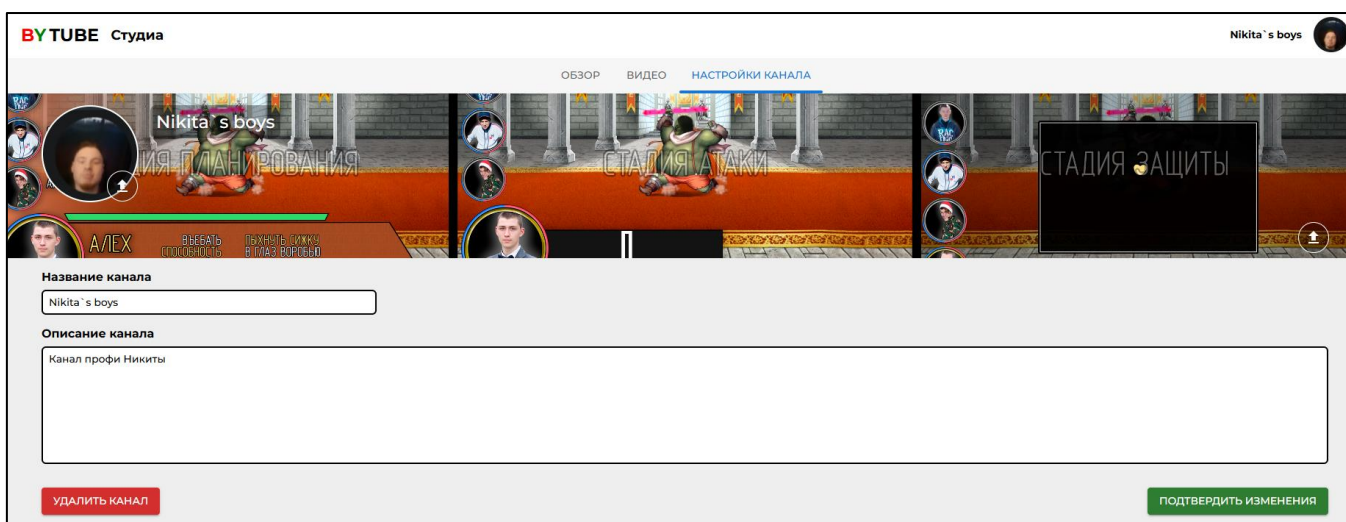


Рисунок 5.7 – Страница с настройками канала

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

5.10 Редактирование канала

Что бы 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 одноимённые поля.

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

5.11 Создание канала

Что бы 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

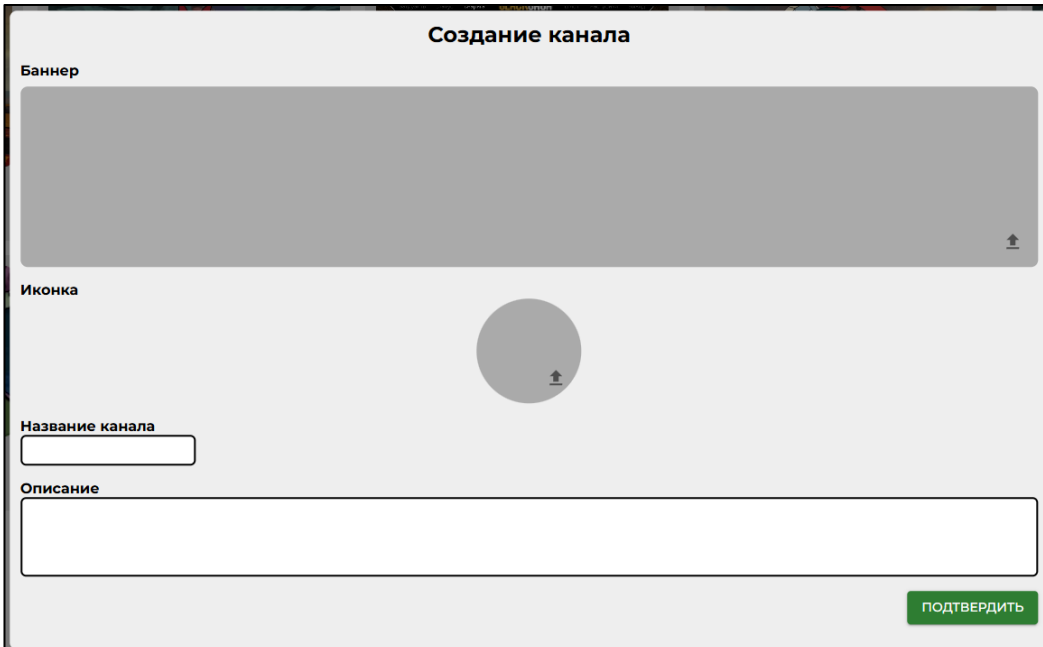


Рисунок 5.8 – Форма создания канала

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.

5.12 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 предоставлен на рисунке 5.9.

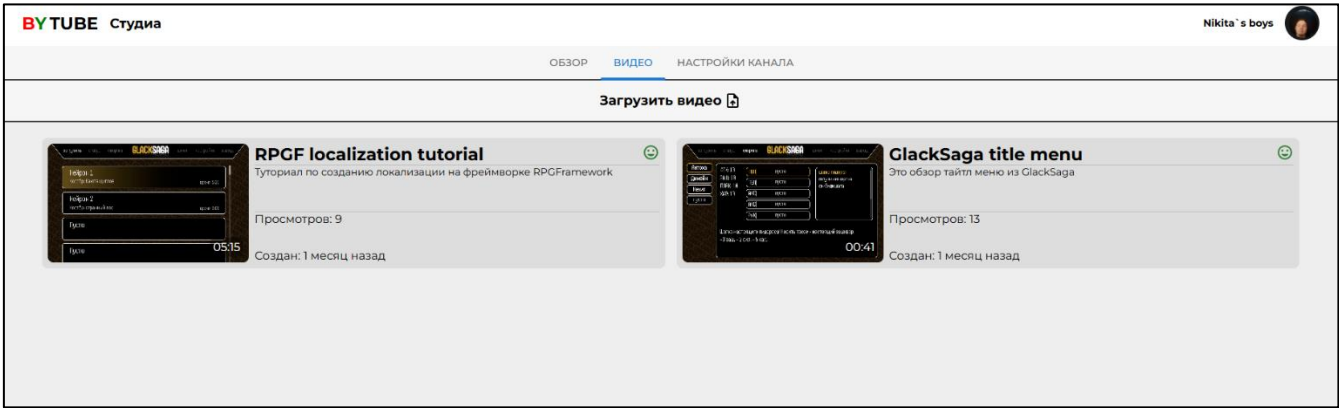


Рисунок 5.9 – Страница со списком видео канала

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

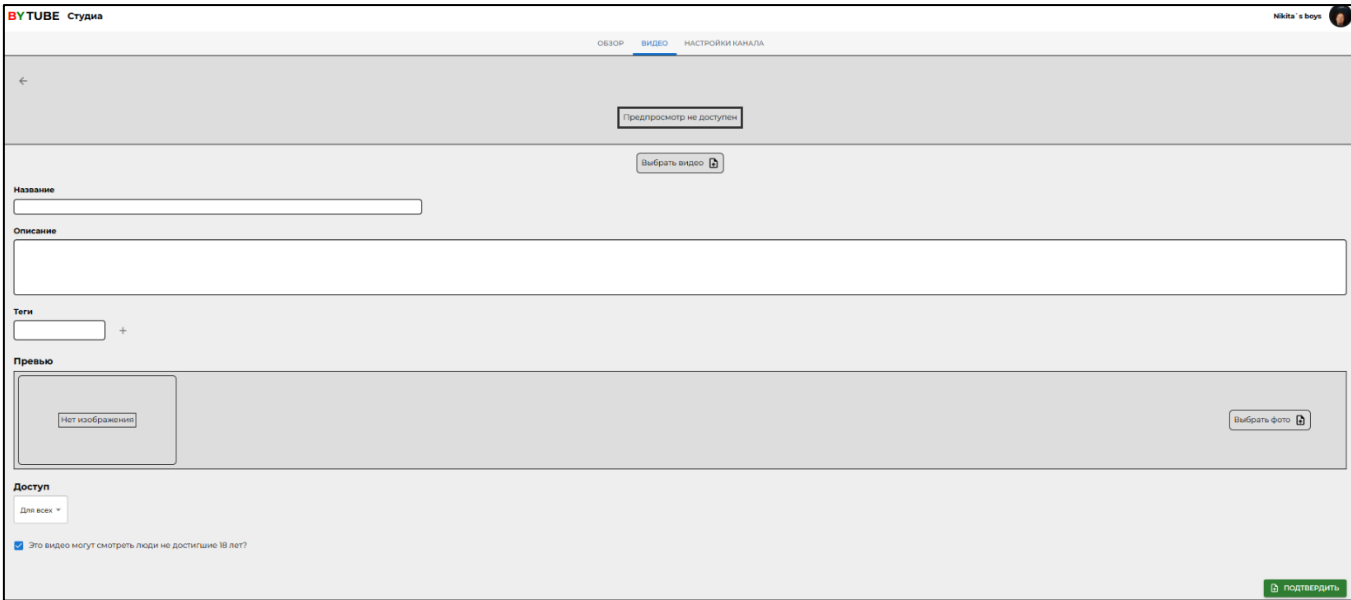


Рисунок 5.10 – Страница со формой загрузки видео

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.

5.13 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.

The screenshot shows a web form for editing video metadata. It contains the following elements:

- Название (Title):** A text input field containing "RPGF localization tutorial".
- Описание (Description):** A larger text area containing "Тutorial по созданию локализации на фреймворке RPGFramework".
- Теги (Tags):** A section with a text input and a "+" button, followed by three tag buttons: "#RPGF", "#SSS", and "#ClackSaga".
- Превью (Preview):** A section with a placeholder image of a game interface and a "Выбрать фото" (Choose photo) button with a camera icon.
- Доступ (Access):** A dropdown menu set to "Для всех" (For all), and a checked checkbox labeled "Это видео могут смотреть люди не достигшие 18 лет?" (This video can be watched by people under 18?).
- Buttons:** A red "УДАЛИТЬ" (Delete) button at the bottom left and a green "ПОДТВЕРДИТЬ" (Confirm) button at the bottom right.

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.

5.14 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.

5.15 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.

5.16 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.

5.17 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 поля «Комментарий» после чего нажать кнопку «ОТПРАВИТЬ». Если данные введены верно, то комментарий будет добавлен.

5.18 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.

5.19 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.

5.20 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 пискком комнат предоставлен на рисунке 5.12.

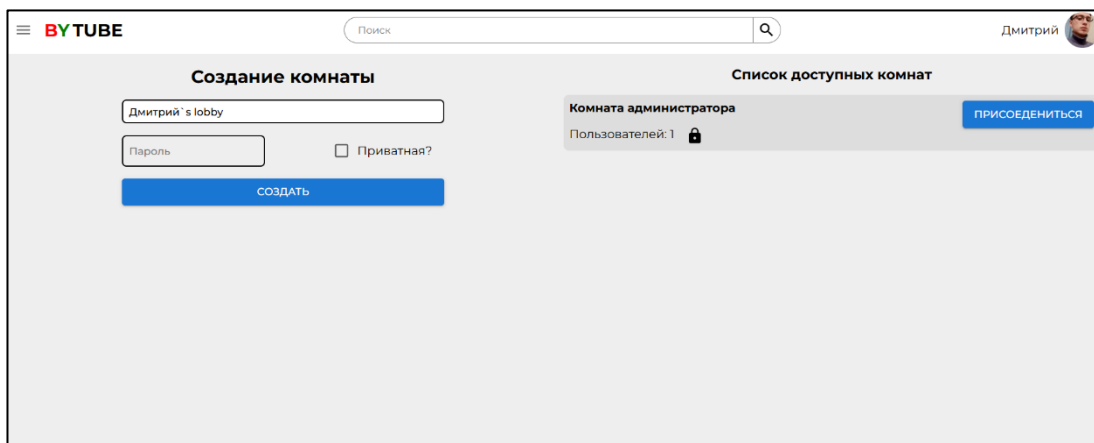


Рисунок 5.12 – Страница со списком комнат

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.

Для подключения к уже существующей комнате требуется нажать кнопку «ПОДКЛЮЧИТЬСЯ» на выбранной комнате, однако если комната приватная, то будет необходимо ввести пароль в появившемся 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.

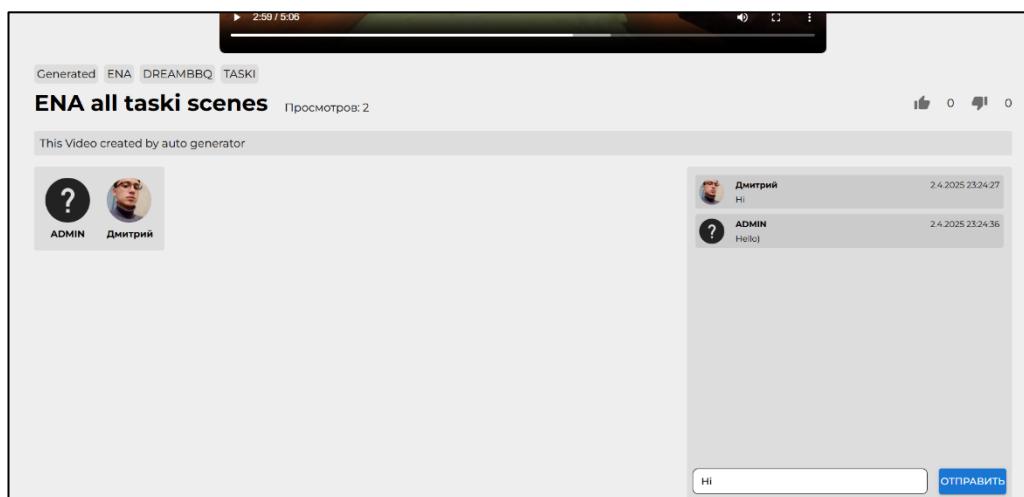


Рисунок 5.13 – Страница совместно просмотра

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

5.21 Создание плейлистов

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

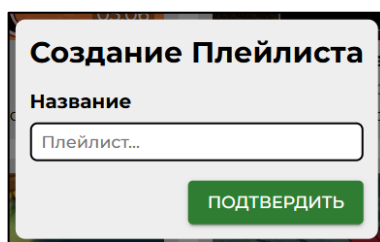


Рисунок 5.14 – Форма создания плейлиста

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

5.22 Удаление видео из плейлиста

Для удаление 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.

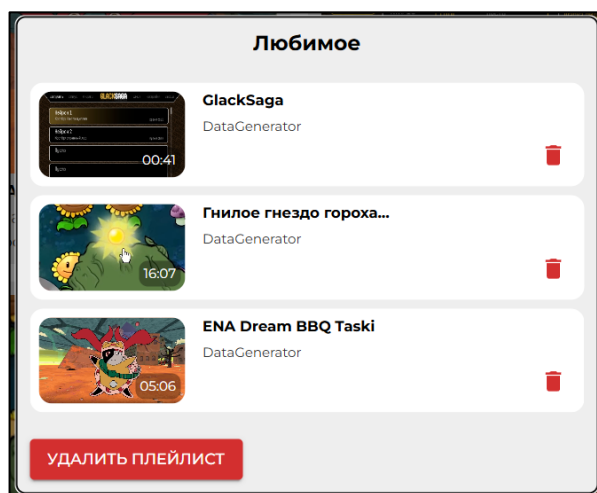


Рисунок 5.15 – Меню плейлиста

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

5.23 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.

5.24 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 начнётся воспроизведение.

5.25 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».

5.26 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 администратора предоставлен на рисунке 5.16.

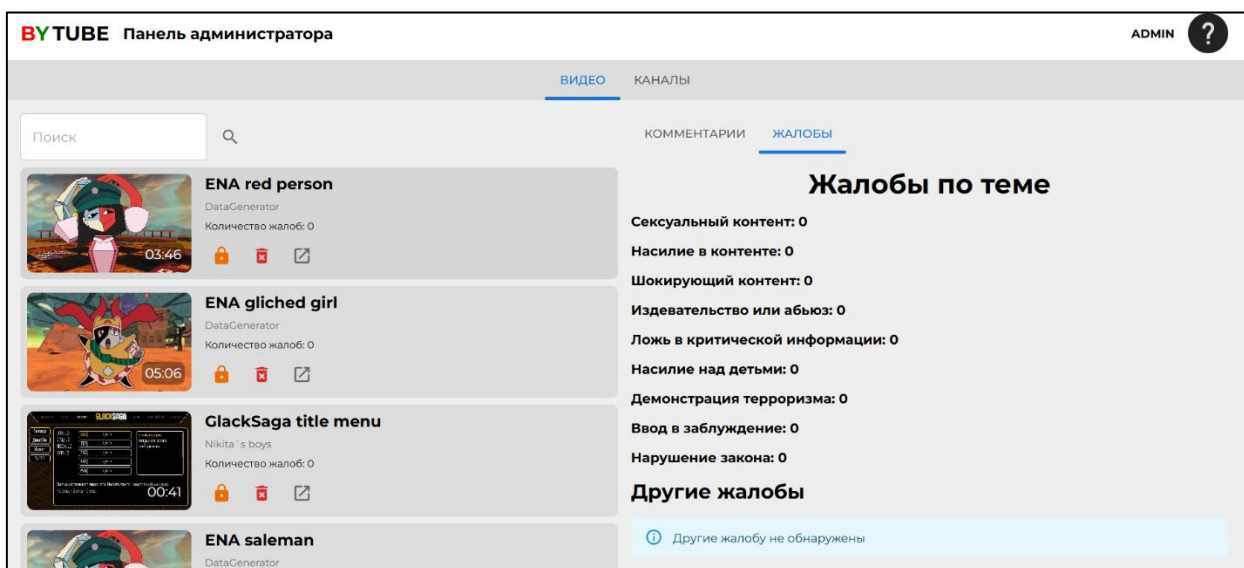


Рисунок 5.16 – Панель администратора (Видео)

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.

5.27 Скрытие 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».

5.28 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».

5.29 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».

5.30 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.

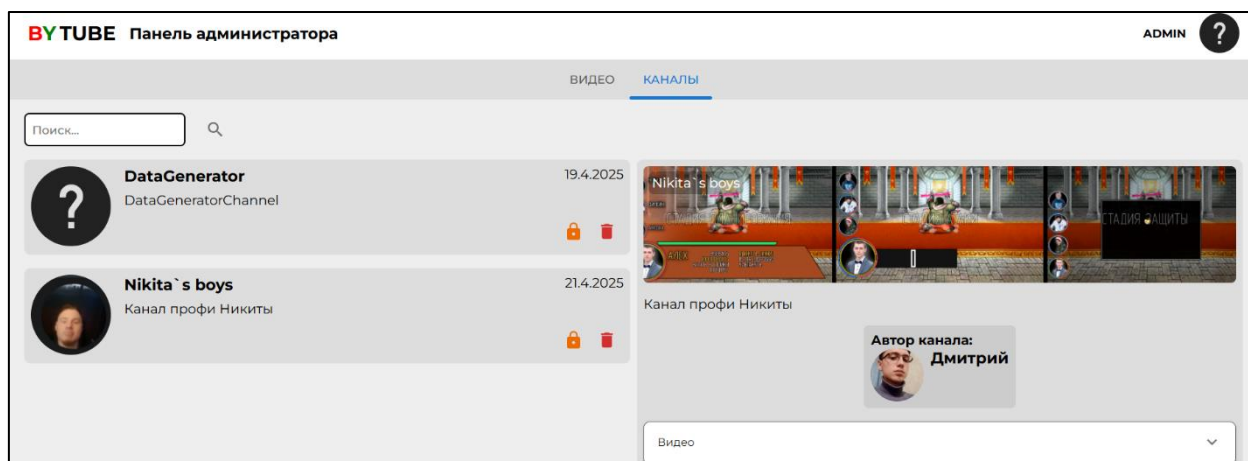


Рисунок 5.17 – Панель администратора (Каналы)

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

5.31 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».

5.32 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, касающиеся валидации полей ввода в формах.

Описана каждая вкладка, используемая в приложении, что позволит понять по отдельности каждую часть приложения.

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.

6 Технико-экономическое обоснования проекта

6.1 Общая характеристика разрабатываемого программного средства

При выполнении данного проекта было разработано веб-приложение YUTUBE, 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 backend-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 просмотр 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.

6.2 Исходные данные для проведения расчётов и маркетинговый анализ

Источниками исходных данных для данных расчетов выступают действующие нормативные правовые акты. Исходные данные для расчета приведены в таблице 6.1.

					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		8	
Консульт.	Познякова Л.С.								БГТУ 1-40 01 01, 2025					
Н. контр.	Белодед Н.И.													
Утв.	Смелов В.В.													

Таблица 6.1 – Исходные данные для расчета

Наименование показателя	Условные обозначения	Норматив
Норматив дополнительной заработной платы, %	$H_{дз}$	9
Ставка отчислений в Фонд социальной защиты населения, %	$H_{фсзн}$	34
Ставка отчислений по обязательному страхованию в БРУСП «Белгосстрах», %	$H_{бгс}$	0,6
Норматив прочих прямых затрат, %	$H_{пз}$	25
Норматив накладных расходов, %	$H_{обп,обх}$	50
Норматив расходов на сопровождение и адаптацию, %	$H_{рса}$	10
Ставка НДС, %	$H_{ндс}$	20
Налог на прибыль, %	$H_{п}$	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.

6.3 Обоснование цены программного средства

6.3.1 Расчёт затрат рабочего времени на разработку программного средства

В таблице 6.3 в укрупнённом 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
Вёрстка 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.

6.3.2 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.

$$C_{\text{оз}} = T_{\text{раз}} \cdot C_{\text{зп}} \quad (6.1)$$

где $C_{\text{оз}}$ – основная заработная плата, руб.;

$T_{\text{раз}}$ – трудоемкость, чел./час.;

$C_{\text{зп}}$ – 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.

6.3.3 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.

$$C_{\text{дз}} = \frac{C_{\text{оз}} \cdot H_{\text{дз}}}{100}, \quad (6.2)$$

где $C_{\text{оз}}$ – основная заработная плата, руб.;

$H_{\text{дз}}$ – норматив дополнительной заработной платы, %.

$$C_{\text{дз}} = 9344 \cdot 9 / 100 = 840,96 \text{ руб.}$$

Таким образом дополнительная заработная плата составила 840,96 рублей.

6.3.4 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 зарплате исполнителей и вычисляются по формуле 6.3.

$$C_{\text{фсзн}} = \frac{(C_{\text{оз}} + C_{\text{дз}}) \cdot H_{\text{фсзн}}}{100}, \quad (6.3)$$

где $C_{\text{оз}}$ – основная заработная плата, руб.;

$C_{\text{дз}}$ – дополнительная заработная плата на конкретное ПС, руб.;

$H_{\text{фсзн}}$ – норматив отчислений в Фонд социальной защиты, %.

Отчисления в БРУСП «Белгосстрах» вычисляются по формуле 6.4.

$$C_{\text{бгс}} = \frac{(C_{\text{оз}} + C_{\text{дз}}) \cdot H_{\text{бгс}}}{100}, \quad (6.4)$$

$$C_{\text{фсзн}} = \frac{(9344 + 840,96) \cdot 34}{100} = 3\,462,89 \text{ руб.}$$

$$C_{\text{бгс}} = \frac{(9344 + 840,96) \cdot 0,6}{100} = 61,11 \text{ руб.}$$

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

6.3.5 Расчет суммы прочих прямых затрат

Расходы на конкретное программное средство $C_{\text{пз}}$ 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.

$$C_{\text{пз}} = \frac{C_{\text{оз}} \cdot H_{\text{пз}}}{100}, \quad (6.5)$$

где $H_{\text{пз}}$ – норматив прочих затрат в целом по организации, %.

$$C_{\text{пз}} = 9344 \cdot 25 / 100 = 2\,336 \text{ руб.}$$

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

6.3.6 Расчет суммы накладных расходов

Сумма накладных расходов $C_{\text{обп,обх}}$ – произведение основной заработной платы исполнителей на конкретное программное средство $C_{\text{оз}}$ на норматив накладных расходов в целом по организации $H_{\text{обп,обх}}$, по формуле 6.6.

$$C_{\text{обп,обх}} = \frac{C_{\text{оз}} \cdot H_{\text{обп,обх}}}{100} \quad (6.6)$$

Сумма накладных расходов составит:

$$C_{\text{обп,обх}} = 9344 \cdot 50 / 100 = 4\,672 \text{ руб.}$$

Таким образом, сумма накладных расходов составила 4 672 руб.

6.3.7 Сумма расходов на разработку программного средства

Сумма расходов на разработку программного средства C_p 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.

$$C_p = C_{оз} + C_{дз} + C_{фсзн} + C_{бгс} + C_{пз} + C_{обп,обх} \quad (6.7)$$

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

$$C_p = 9\,344 + 840,96 + 3\,462,89 + 61,11 + 2\,336 + 4\,672 = 20\,716,96 \text{ руб.}$$

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

6.3.8 Расходы на сопровождение и адаптацию

Сумма расходов на сопровождение и адаптацию программного средства $C_{рса}$ определяется как произведение суммы расходов на разработку на норматив расходов на сопровождение и адаптацию $H_{рса}$, и находится по формуле 6.8.

$$C_{рса} = \frac{C_p \cdot H_{рса}}{100} \quad (6.8)$$

$$C_{рса} = 20\,716,96 \cdot 10 / 100 = 2\,071,7 \text{ руб.}$$

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

6.3.9 Расчет полной себестоимости

Полная себестоимость $C_{п}$ определяется как сумма двух элементов: суммы расходов на разработку C_p и суммы расходов на сопровождение и адаптацию $C_{рса}$.

Полная себестоимость $C_{п}$ вычисляется по формуле 6.9

$$C_{п} = C_p + C_{рса} \quad (6.9)$$

$$C_{п} = 20\,716,96 + 2\,071,7 = 22\,778,65 \text{ руб.}$$

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

6.4 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 создаёт 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 поддержания 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 создаёт эффект кинотеатра или телевизионного вечера, но в цифровом пространстве.

Доступность распространения контента на BYTUBE стимулирует творчество среди обычных пользователей. Нет жёстких 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

В результате выполнения дипломного проектирования было разработано веб-приложение видео-хостинг «BYTUBE» для просмотра видеоконтента с возможностью совместного просмотра.

В рамках работы над проектом был проведен обзор аналогичных решений, выбрана платформа для разработки серверной и клиентской частей программного продукта, спроектирована архитектура приложения.

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.

База данных приложения состоит из семи таблиц, а объем пользовательского кода превышает 11 000 строк.

Для обеспечения качества разработанного продукта было создано 31 тест-кейсов, которые покрыли 100% функционала веб-приложения. 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 каналами.

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. Модуль взаимодействия включает в себя инструменты комментирования, лайков и подписок, плейлистов.

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.

					ДП 00.00 ПЗ			
		Ф.И.О	Подпись	Дата				
Разраб	Окулич Д.Ю.							
Пров.	Белодед Н.И.							
Н. контр.	Белодед Н.И.							
Утв.	Смелов В.В.							
					Лит.		Лист	Листов
					У		1	8

Список используемых источников

- 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 N-Layer [Электронный ресурс]. – Режим доступа: <https://medium.com/design-microservices-architecture-with-patterns/layered-n-layer-architecture-e15ffdb7fa42> – дата доступа: 28.05.2025
- 15 Dependency Injection (DI) [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/articles/350068/> – дата доступа: 28.05.2025
- 16 JSON Web Token (JWT) RFC [Электронный ресурс]. – Режим доступа: <https://datatracker.ietf.org/doc/html/rfc7519> – дата доступа: 28.05.2025
- 17 Паттерн «Repository» [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/articles/248505/> – дата доступа: 28.05.2025
- 18 JSON RFC [Электронный ресурс]. – Режим доступа: <https://datatracker.ietf.org/doc/html/rfc7159.html> – дата доступа: 28.05.2025

						ДП 00.00 ПЗ											
		Ф.И.О		Подпись		Дата											
Разраб		Окулич Д.Ю.						Лит.		Лист		Листов					
Пров.		Белодед Н.И.								у		1					
Н. контр.		Белодед Н.И.															
УТВ		Смелов В В															
						Список используемых источников						БГТУ 1-40 01 01, 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-oss";

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

        if (video.VideoAccess == Video.Access.Private)
        {
            if (!authData.IsAuthorize || 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 доступно",
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.IsAuthorize && authData.Id !=
video.Channel.UserId) || !authData.IsAuthorize)
                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.IsAuthorize)
            {
                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 принадлежит!",
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	RemoveVideoFromPlaylist	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);
    }
}

```