Реферат

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

Пояснительная записка дипломного проекта содержит 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 видео-хостинга с возможностью совместного просмотра видео.

В рассматриваемом веб-приложении пользователю доступен просмотр видео, создание плейлистов, создание каналов для загрузки своего контента, комментирование видео. Так же пользователю доступна возможность совместного просмотра видео контента с другими пользователями.

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.

Abstract

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

The explanatory note of the graduation project contains 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.

Содержание

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

[Реферат 1](#_Toc198858314)

[Abstract 2](#_Toc198858315)

[Содержание 3](#_Toc198858316)

[Введение 6](#_Toc198858317)

[1 Постановка задачи и аналитический обзор аналогичных решений 7](#_Toc198858318)

[1.1 Постановка задачи 7](#_Toc198858319)

[1.2 Обзор аналогичных решений 7](#_Toc198858320)

[1.2.1 Веб-приложение «YouTube» 7](#_Toc198858321)

[1.2.2 Веб-приложение «RUTUBE» 8](#_Toc198858322)

[1.2.3 Веб-приложение «VK Video» 9](#_Toc198858323)

[1.3 Выводы по разделу 10](#_Toc198858324)

[2 Проектирование веб-приложения 11](#_Toc198858325)

[2.1 Функциональность веб-приложения 11](#_Toc198858326)

[2.2 Логическая схема базы данных 13](#_Toc198858327)

[2.3 Архитектура веб-приложения 14](#_Toc198858328)

[2.4 Блок-схема подключения к совместному просмотру 15](#_Toc198858329)

[2.5 Выводы по разделу 15](#_Toc198858330)

[3 Разработка веб-приложения 16](#_Toc198858331)

[3.1 Разработка серверной части веб-приложения 16](#_Toc198858332)

[3.1.1 Используемые библиотеки 16](#_Toc198858333)

[3.1.2 Структура серверной части 17](#_Toc198858334)

[3.1.3 Разработка уровня доступа к данным 18](#_Toc198858335)

[3.1.4 Разработка уровня бизнес-логики 19](#_Toc198858336)

[3.1.5 Разработка уровня представления 20](#_Toc198858337)

[3.1.6 Настройка безопасности 20](#_Toc198858338)

[3.1.7 Используемые паттерны 22](#_Toc198858339)

[3.2 Клиентская часть 22](#_Toc198858340)

[3.2.1 Используемые библиотеки 22](#_Toc198858341)

[3.2.2 Структура клиентской части 22](#_Toc198858342)

[3.2.3 Конфигурация проекта 23](#_Toc198858343)

[3.3 Реализация функций 23](#_Toc198858344)

[3.3.1 Регистрация 23](#_Toc198858345)

[3.3.2 Авторизация 24](#_Toc198858346)

[3.3.3 Просмотр публичных видео 24](#_Toc198858347)

[3.3.4 Просмотр комментариев под видео 25](#_Toc198858348)

[3.3.5 Поиск публичных видео 25](#_Toc198858349)

[3.3.6 Поиск по названию 25](#_Toc198858350)

[3.3.7 Сортировка видео 25](#_Toc198858351)

[3.3.8 Поиск по тегам 26](#_Toc198858352)

[3.3.9 Удаление канала 26](#_Toc198858353)

[3.3.10 Редактирование канала 27](#_Toc198858354)

[3.3.11 Создание канала 27](#_Toc198858355)

[3.3.12 Загрузка видео 27](#_Toc198858356)

[3.3.13 Редактирование видео 27](#_Toc198858357)

[3.3.14 Удаление видео 27](#_Toc198858358)

[3.3.15 Удаление комментариев под своими видео 27](#_Toc198858359)

[3.3.16 Оставление жалобы под видео 27](#_Toc198858360)

[3.3.17 Оставление комментариев под видео 27](#_Toc198858361)

[3.3.18 Удаление своих комментариев 27](#_Toc198858362)

[3.3.19 Редактирование своих комментариев 27](#_Toc198858363)

[3.3.20 Совместный просмотр 27](#_Toc198858364)

[3.3.21 Создание плейлистов 27](#_Toc198858365)

[3.3.22 Удаление видео из плейлиста 27](#_Toc198858366)

[3.3.23 Добавление видео в плейлиста 27](#_Toc198858367)

[3.3.24 Воспроизведение плейлиста 27](#_Toc198858368)

[3.3.25 Удаление плейлиста 27](#_Toc198858369)

[3.3.26 Удаление любых комментариев администратором 27](#_Toc198858370)

[3.3.27 Скрытия видео администратором 27](#_Toc198858371)

[3.3.28 Удаление видео администратором 27](#_Toc198858372)

[3.3.29 Просмотр жалоб пользователей администратором 27](#_Toc198858373)

[3.3.30 Блокировка канала администратором 27](#_Toc198858374)

[3.3.31 Удаление канала администратором 27](#_Toc198858375)

[4 Тестирование веб-приложения 28](#_Toc198858376)

[5 Руководство пользователя (руководство по эксплуатации) 29](#_Toc198858377)

[6 Технико-экономическое обоснования проекта 30](#_Toc198858378)

[6.1 Общая характеристика разрабатываемого программного средства 30](#_Toc198858379)

[6.2 Исходные данные для проведения расчётов и маркетинговый анализ 31](#_Toc198858380)

[6.3 Обоснование цены программного средства 31](#_Toc198858381)

[6.3.1 Расчёт затрат рабочего времени на разработку программного средства 31](#_Toc198858382)

[6.3.2 Расчет основной заработной платы 32](#_Toc198858383)

[6.3.3 Расчет дополнительной заработной платы 33](#_Toc198858384)

[6.3.4 Расчет отчислений в Фонд социальной защиты населения и по обязательному страхованию 33](#_Toc198858385)

[6.3.5 Расчет суммы прочих прямых затрат 34](#_Toc198858386)

[6.3.6 Расчет суммы накладных расходов 34](#_Toc198858387)

[6.3.7 Сумма расходов на разработку программного средства 34](#_Toc198858388)

[6.3.8 Расходы на сопровождение и адаптацию 35](#_Toc198858389)

[6.3.9 Расчет полной себестоимости 35](#_Toc198858390)

[6.4 Вывод по разделу 35](#_Toc198858391)

[Заключение 37](#_Toc198858392)

[Список используемых источников 38](#_Toc198858393)

[Приложение А 40](#_Toc198858394)

[Приложение Б 43](#_Toc198858395)

[Приложение В 45](#_Toc198858396)

# **Введение**

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

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

Цель дипломного проекта — разработать веб-приложение, объединяющее функции видеохостинга и платформы для синхронного просмотра с друзьями. Здесь можно создавать каналы, обсуждать ролики в комментариях и наслаждаться контентом вместе, даже находясь далеко друг от друга.

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

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:

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

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

1. Постановка задачи и аналитический обзор аналогичных решений

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

* 1. Постановка задачи

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

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

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

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

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

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

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

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

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

Интерфейс приложения представлен на рисунке 1.1.

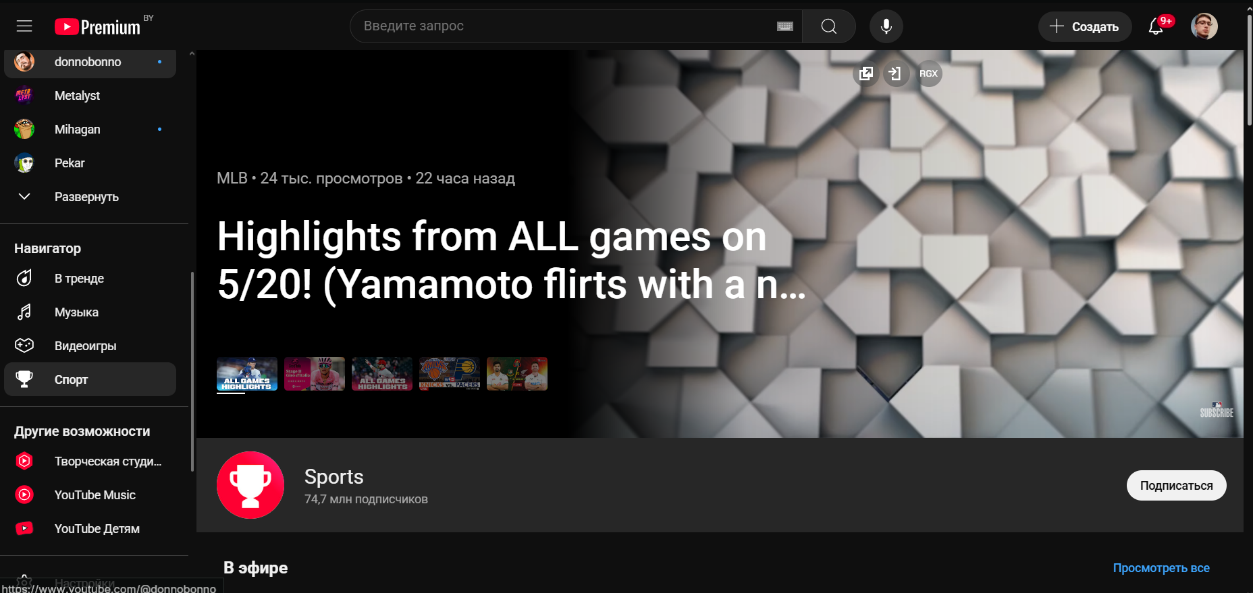


Рисунок 1.1 – Приложение «YouTube»

Проанализировав интерфейс, было принято решение ориентироваться на функционал YouTube в реализации функционала BYTUBE.

* + 1. Веб-приложение «RUTUBE»

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

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

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

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

Интерфейс приложения представлен на рисунке 1.2.

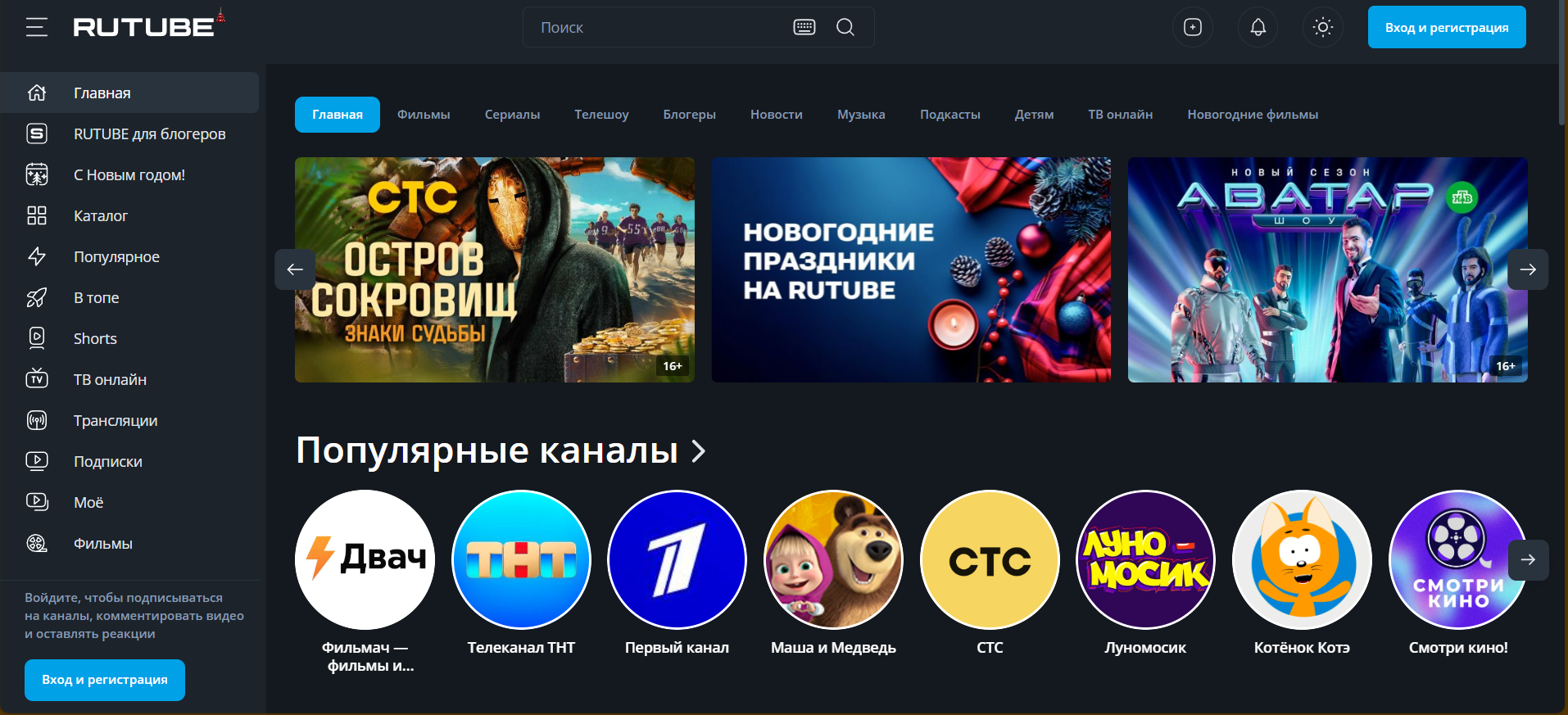


Рисунок 1.2 – Приложение «RUTUBE»

Проанализировав приложение было решено сделать похожим дизайн интерфейса страницы видео.

* + 1. Веб-приложение «VK Video»

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

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

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

Интерфейс приложения представлен на рисунке 1.3.

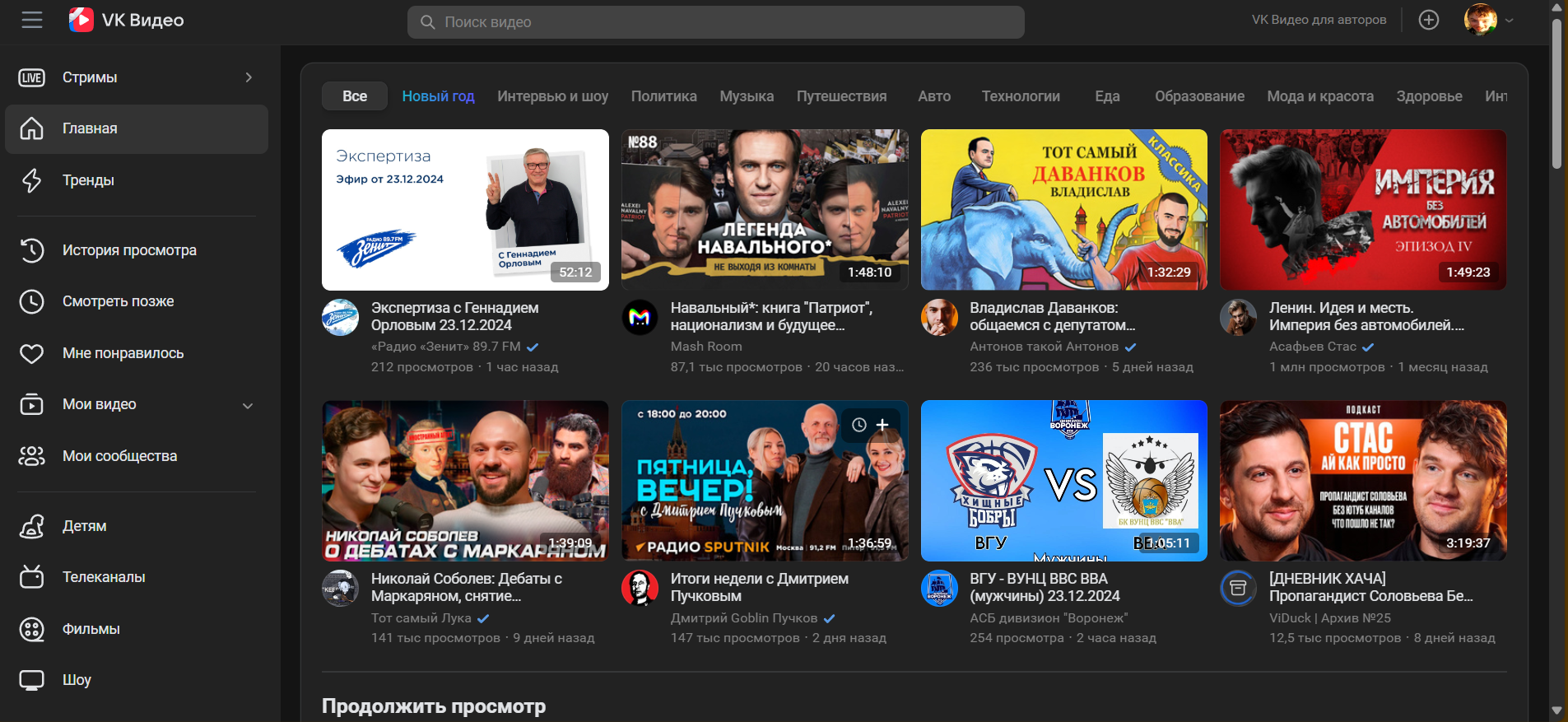


Рисунок 1.3 – Приложение «VK Video»

Проанализировав приложение было решено реализовать похожую боковую панель так как она достаточно интуитивна.

* 1. Выводы по разделу

В рамках данной главы было выполнено:

1. Поставлены необходимые задачи для реализации веб-приложения;

2. Разобраны три аналогичных решения: «YouTube», «RUTUBE», «VK Video». Из «YouTube» взят основной функционал, из “RUTUBE” взят основной дизайн приложения, а из «VK Video» взята боковая панель.

3. Было отмечено, что ни один из аналогов не имеет возможности совместного просмотра видео контента.

1. Проектирование веб-приложения

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *БГТУ 02.00 ПЗ* | | | | | |
|  |  |  |  |  |
|  |  | Ф.И.О | Подпись | Дата |
| Разраб | | Окулич Д.Ю. |  |  | *2 Проектирование веб-приложения* | Лит. | | | Лист | Листов |
| Пров. | | Белодед Н.И. |  |  |  | У |  | 1 | 4 |
|  | |  |  |  | БГТУ 1-40 01 01, 2025 | | | | |
| Н. контр. | | Белодед Н.И. |  |  |
| Утв. | | Смелов В.В. |  |  |

* 1. Функциональность веб-приложения

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

Описание ролей пользователей web-приложения представлены в таблице 2.1.

Таблица 2.1 – Описание ролей

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

Роли в системе разделены таким образом, чтобы разграничить доступ к функционалу приложения и обеспечить безопасное использование платформы. Каждая роль наделена строго 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 публичных видео | Возможность смотреть видео, доступные для всех пользователей. | Гость, Клиент, Администратор |
| 4 | Просмотр комментариев под видео | Возможность просматривать комментарии, доступные под видео для всех пользователей. | Гость, Клиент, Администратор |
| 5 | Поиск публичных видео | Возможность поиска видео, доступных для всех пользователей. | Гость, Клиент, Администратор |
| 6 | Поиск по названию | Возможность поиска видео по названию | Гость, Клиент, Администратор |
| 7 | Сортировка видео | Возможность сортировать  видео по определённым признакам | Гость, Клиент, Администратор |

Продолжение таблицы 2.2

|  |  |  |  |
| --- | --- | --- | --- |
| № | Функция | Описание | Доступно ролям |
| 8 | Поиск по тегам | Возможность поиска видео по тегам | Гость, Клиент, Администратор |
| 9 | Удаление канала | Удаление своего канал пользователем | Клиент, Администратор |
| 10 | Редактирование канала | Редактирование своего канал пользователем | Клиент, Администратор |
| 11 | Создание канала | Создание канал пользователем | Клиент, Администратор |
| 12 | Загрузка видео | Загрузка видео пользователем в рамках канала | Клиент, Администратор |
| 13 | Редактирование данных видео | Редактирование данных видео пользователем в рамках канала | Клиент, Администратор |
| 14 | Удаление видео | Удаление видео пользователем в рамках канала | Клиент, Администратор |
| 15 | Удаление комментариев под своими видео | Удаление любых комментариев автором видео | Клиент, Администратор |
| 16 | Оставление жалобы под видео | Возможность оставить жалобу под видео | Клиент, Администратор |
| 17 | Оставление комментариев под видео | Возможность оставить комментарий под видео | Клиент, Администратор |
| 18 | Удаление своих  комментариев | Возможность удалить свой комментарий | Клиент, Администратор |
| 19 | Редактирование своих комментариев | Возможность редактировать свой комментарий | Клиент, Администратор |
| 20 | Совместный просмотр | Возможность пользователям совместно просматривать видео контент | Клиент, Администратор |
| 21 | Создание плейлистов | Возможность создание плейлиста | Клиент, Администратор |
| 22 | Удаление видео из  плейлиста | Возможность удалять видео из своего плейлиста | Клиент, Администратор |
| 23 | Добавление видео в  плейлиста | Возможность добавлять видео в свой плейлиста | Клиент, Администратор |
| 24 | Воспроизведение  плейлиста | Возможность последовательно воспроизводить видео из  плейлиста | Клиент,  Администратор |
| 25 | Удаление плейлиста | Возможность удаления своего плейлиста | Клиент,  Администратор |
| 26 | Удаление любых  комментариев | Возможность удалять любые комментарии | Администратор |
| 27 | Скрытия видео | Возможность скрыть видео администратором | Администратор |
| 28 | Удаление видео | Возможность удаления видео администратором | Администратор |

Продолжение таблицы 2.2

|  |  |  |  |
| --- | --- | --- | --- |
| № | Функция | Описание | Доступно ролям |
| 29 | Просмотр жалоб  пользователей | Возможность просматривать жалобы пользователей | Администратор |
| 30 | Блокировка канала | Возможность блокировки канала администратором | Администратор |
| 31 | Удаление канала | Возможность удаление канала администратором | Администратор |

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

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

* 1. Логическая схема базы данных

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

Таблица 2.3 – Описание таблиц базы данных

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

* 1. Архитектура веб-приложения

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

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

Таблица 2.4 – Назначение элементов архитектурной схемы веб-приложения

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

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

Таблица 2.5 – Описание используемых протоколов

|  |  |
| --- | --- |
| Протокол | Назначение |
| HTTPS [11] | Обмен данными между Backend Server и Stripe, Backend Server и Amazon AWS, между Stripe и Stripe Listener. Обеспечивает безопасную передачу данных путём использования криптографического протокола TLS. |
| TCP [12] | Обмен данными между Database Server и Backend Server. |
| WebSocket [13] | Используется для поточного просмотра видео и работы комнат совместного просмотра. |

* 1. Блок-схема подключения к совместному просмотру

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

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

* 1. Выводы по разделу

Таким образом в рамках проектирования веб-приложения было выполнено.

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

2. Спроектирована архитектура приложения. Было определено каким образам будут размещены компоненты приложения и какие протоколы будут использоваться.

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

4. Спроектирована блок схема алгоритма подключение к совместному просмотру.

1. Разработка веб-приложения

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

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.

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

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

Описание библиотек предоставлено в таблице 3.1.

Таблица 3.1 – Серверные зависимости

|  |  |
| --- | --- |
| Библиотека | Описание |
| Microsoft.AspNetCore | 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. |
| Microsoft.AspNetCore.SignalR.  Common | Поддержка WebSocket-технологии через SignalR для серверной части. |
| Npgsql.EntityFrameworkCore.  PostgreSQL | Библиотека Entity Framework Core. |
| Xabe.FFmpeg | Обёртка над FFmpeg для обработки видеофайлов. |

Продолжение таблицы 3.1

|  |  |
| --- | --- |
| Зависимости | Описание |
| NAudio | Библиотека для обработки и анализа аудиофайлов. |
| Swashbuckle.AspNetCore.SwaggerGen, SwaggerUI, Annotations | Библиотеки для генерации документации API |
| System.IdentityModel.Tokens.Jwt | Библиотека для работы с JWT-токенами. |

* + 1. Структура серверной части

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

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

Структура проекта предоставлена на рисунке 3.1.

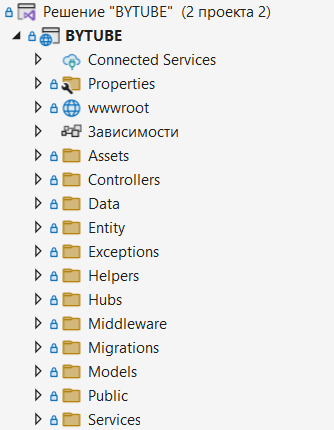


Рисунок 3.1 – Структура проекта

Описание папок входящих в структуру проекта предоставлено в таблице 3.2.

Таблица 3.2 – Папки входящие в проект

|  |  |
| --- | --- |
| Папка | Описание |
| Assets | Статические данные (изображения и тд.) |
| Controllers | Содержит контроллеры |
| Data | Предназначена для хранения тяжелых данных (изображения, видео) |
| Entity | Содержит реализацию базы данных |
| Exceptions | Содержит исключения |
| Helpers | Содержит бизнес логику, которая не поддерживает в DI [15] |
| Hubs | Содержит хаб для совместного просмотра |

Продолжение таблицы 3.2

|  |  |
| --- | --- |
| Папка | Описание |
| Middleware | Содержит необходимые middleware |
| Migrations | Содержит автоматически созданные миграции |
| Models | Содержит модели данных для передачи с клиентом |
| Public | Содержит index.html |
| Services | Содержит бизнес логику с поддержкой DI |
| wwwroot | Содержит статические файлы необходимые для работы клиентской части (bundle.js) |

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

* + 1. Разработка уровня доступа к данным

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

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

public class Comment {

[Key]

public Guid Id { get; set; }

public required string Message { get; set; }

[Column(TypeName = "jsonb")]

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

public required DateTime Created { get; set; }

public required Guid UserId { get; set; }

[ForeignKey(nameof(UserId))]

public User? User { get; set; }

public required Guid VideoId { get; set; }

[ForeignKey(nameof(VideoId))]

public Video? Video { get; set; } }

Листинг 3.1 – Сущность Comment

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

Comment comment = await \_commentRepository.GetCommentWithAuthor(guid)

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

Листинг 3.2 – Пример выполнения запроса в базу данных

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

public interface IRepository<T>

{

public T? Get(Guid guid);

public List<T> GetAll();

public void Create(T value);

public void Update(T value);

public void Delete(Guid guid);

}

Листинг 3.3 – Интерфейс IRepository

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

* + 1. Разработка уровня бизнес-логики

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

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

public WatchTogetherLobbyCleaningService(WatchTogetherLobbyService watchTogetherLobby)

{

\_watchTogetherLobby = watchTogetherLobby;

}

Листинг 3.4 – пример передачи зависимостей через конструктор

Таким образом было организованно использование бизнес-логики.

* + 1. Разработка уровня представления

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

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

Пример контроллера предоставлен в листинге 3.5.

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

[ApiController]

public class VideoController : ControllerBase

{

private readonly PostgresDbContext \_dbContext;

private readonly LocalDataService \_localData;

private readonly VideoMediaService \_videoMedia;

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

{

\_dbContext = db;

\_localData = localData;

\_videoMedia = videoMedia;

}

[HttpGet("view")]

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

{ ... }

}

Листинг 3.5 – Пример контроллера

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

* + 1. Настройка безопасности

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

Создание токенов обычно осуществляется с использованием библиотеки System.IdentityModel.Tokens.Jwt, которая предоставляет все необходимые инструменты для работы с JSON Web Token (JWT) [16]. Access token представляет собой короткоживущий токен, который используется для предоставления доступа к защищённым ресурсам. Refresh token имеет более длительный срок действия и предназначен для обновления access token, когда срок его действия истёк. Это снижает частоту запросов к серверу авторизации и улучшает производительность.

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

Код создание токенов предоставлен в листинге 3.6.

public static string GenerateJwtToken(JwtSettings settings, UserData user)

{

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

var claims = new[] {

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

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

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

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

var tokenDescriptor = new SecurityTokenDescriptor

{

Subject = new ClaimsIdentity(claims),

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

Issuer = settings.Issuer,

Audience = settings.Audience,

SigningCredentials = credentials

};

var tokenHandler = new JwtSecurityTokenHandler();

var token = tokenHandler.CreateToken(tokenDescriptor);

return tokenHandler.WriteToken(token);

}

Листинг 3.6 – Код создание токенов

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

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

* + 1. Используемые паттерны

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

* dependency Injection (DI) — паттерн, который помогает разделить создание зависимостей и их использование. В ASP.NET Core встроена мощная система DI, которая позволяет инжектировать сервисы, такие как репозитории, логеры или контексты баз данных, прямо в контроллеры и другие компоненты через конструктор. Это способствует лучшей модульности, тестируемости и снижению связности между компонентами системы;
* repository [17] — паттерн для абстракции доступа к данным. Он позволяет скрыть детали работы с хранилищем данных (например, базой данных) и предоставляет простые методы для работы с объектами. В ASP.NET Core репозитории используются для инкапсуляции операций CRUD, что упрощает работу с данными и делает код более читаемым и поддерживаемым. Это позволяет легко заменить или изменить реализацию доступа к данным без затрагивания других компонентов.
  1. Клиентская часть
     1. Используемые библиотеки

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

Описание библиотек предоставлено в таблице 3.3.

Таблица 3.3 – Клиентские библиотеки

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

Данные библиотеки полностью закрывают необходимости клиентской части.

* + 1. Структура клиентской части

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

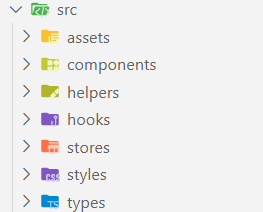


Рисунок 3.2 – Структура клиентской части

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

Таблица 3.4 – Директории входящие в проект

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

Данная структура обеспечивает удобный доступ ко всем файлам проекта.

* + 1. Конфигурация проекта

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

* 1. Реализация функций
     1. Регистрация

Для регистрации пользователя в рамка контроллера «AuthController» был разработан метод «Register». Для регистрации метод принимает имя пользователя, почту, пароль и файл иконки пользователя. Создается новый пользователь с указанным именем, электронной почтой и захешированным паролем. Пользователь сохраняется в базе данных. Если всё прошло хорошо возвращает пользователю код 200 OK. Реализация регистрации предоставлен в листинге 3.7.

[HttpPost("register")]

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

{

try

{

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

{

Name = model.UserName,

Email = model.Email,

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

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

BirthDay = DateOnly.Parse(model.BirthDay)

});

await \_db.SaveChangesAsync();

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

}

catch

{

var errorModel = new ServerErrorModel(400);

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

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

}

return Results.Ok();

}

Листинг 3.7 – Реализация регистрации

Таким образом была реализована регистрация в веб-приложении.

* + 1. Авторизация

Для авторизации в рамках контроллера «AuthController» был разработан метод «SignIn». Метод выполняет вход пользователя с использованием JWT (JSON Веб Token). Принимает в качестве параметров почту и пароль. Метод проверяет существует ли такой пользователь, а также хеширует пришедший пароль и сравнивает с хешом в базе данных. Возвращает 200 OK при успешной авторизации.

* + 1. Просмотр публичных видео

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

Первый метод принимает guid видео и отсылает данные о видео в формате JSON [18] с кодом 200 OK.

Второй метод принимает guid видео и отсылает видео небольшими частями для поточного просмотра, а также что бы не пользователю не пришлось загружать видео целиком.

* + 1. Просмотр комментариев под видео

Для просмотра комментариев под видео в рамках контроллера «CommentController» был разработан метод «GetVideoComments». Данный метод принимает guid видео и проверяет его наличие в базе данных. Возвращает массив комментариев в формате JSON с кодом 200 OK.

* + 1. Поиск публичных видео

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

Таблица 3.4 – Описание параметров метода «Select»

|  |  |
| --- | --- |
| Параметр | Описание |
| Ignore | Список видео, которые нужно игнорировать |
| SearchPattern | Паттерн название |
| Take | Длинна 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 понравившемся 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 c кодом 200 OK.

* + 1. Поиск по названию

Для поиска по названию используется метод «Select» контроллера «VideoController». Требуется указать название видео в параметр «SearchPattern». Возвращает массив моделей видео в формате JSON c кодом 200 OK.

* + 1. Сортировка видео

Для поиска по названию используется метод «Select» контроллера «VideoController». Требуется указать тип сортировки видео в параметр «OrderBy». Всего существует несколько видов сортировки: по новым, по старым, по популярности, по количеству жалоб (для администраторов). Возвращает массив моделей видео в формате JSON c кодом 200 OK.

* + 1. Поиск по тегам

Для поиска по названию используется метод «Select» контроллера «VideoController». Требуется указать теги видео в параметр «SearchPattern». Возвращает массив моделей видео в формате JSON c кодом 200 OK.

* + 1. Удаление канала

Для удаления канала в рамка контроллера «ChannelController» был разработан метод «Delete». Метод принимает guid канала, проверяет является ли клиент владельцем канала, если все верно, то сначала удаляются видео этого канала потом сам канал. Возвращает код 200 OK при успешном удалении канала.

Реализация метода «Delete» предоставлена в листинге 3.8.

[HttpDelete, Authorize]

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

Try {

var authData = AuthorizeData.FromContext(HttpContext);

Channel channel = await \_db.Channels

.Include(channel => channel.Videos)

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

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

if (channel.UserId != authData.Id)

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

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

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

\_db.Videos.Remove(video);

}

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

\_db.Channels.Remove(channel);

await \_db.SaveChangesAsync();

return Results.Ok();

}

catch (ServerException err)

{

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

}

catch (Exception err)

{

return Results.Problem(err.Message);

}

}

Листинг 3.8 – Реализация метола «Delete» контроллера «ChannelController»

Таким образом реализована удаление канала клиентом.

* + 1. Редактирование канала

Для редактирования канала в рамка контроллера «ChannelController» был разработан метод «Put». Метод принимает guid канала и модель «EditChannelModel». Метод проверяет является ли клиент владельцем канала, если все верно, то данные изменяются. Возвращает код 200 OK при успешном редактировании канала.

Реализация метода «Put» предоставлена в листинге 3.9.

[HttpPut, Authorize]

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

Try {

var authData = AuthorizeData.FromContext(HttpContext);

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

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

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

if (channel == null)

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

if (channel.UserId != authData.Id)

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

channel.Name = model.Name;

channel.Description = model.Description;

\_db.Channels.Update(channel);

await \_db.SaveChangesAsync();

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

return Results.Ok();

}

catch (ServerException err)

{

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

}

catch (Exception err)

{

return Results.Problem(err.Message);

}

}

Листинг 3.9 – Реализация метола «Put» контроллера «ChannelController»

Таким образом реализована редактирование канала клиентом.

* + 1. Создание канала

Для создания канала в рамка контроллера «ChannelController» был разработан метод «Post». Метод принимает модель «CreateChannelModel» в виде формы. Метод создаёт канал привязывая его к авторизованному клиенту. Возвращает код 200 OK при успешном создании канала.

Реализация метода «Post» предоставлена в листинге 3.10.

[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 3.10 – Реализация метола «Post» контроллера «ChannelController»

Таким образом реализована создание канала клиентом.

* + 1. Загрузка видео

Для загрузки видео в рамка контроллера «VideoController» был разработан метод «Post». Метод принимает модель «CreateVideoModel» в виде формы, а также параметр «channelGuid» который обозначает на какой канал будет видео загружено. Метод проверяет принадлежит ли данный канал клиенту, после записывает данные о видео в базу данных и начинает загрузки видео на сервер. Для управления медиа файлами используется сервис «LocalDataService».

Реализация метода для загрузки видео на сервер предоставлен в листинге 3.11.

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 3.11 – Реализация метода для загрузки видео

При успешной загрузке видео файла метод «Post» возвращает код 200 OK.

* + 1. Редактирование данных видео

Для редактирования данных видео в рамка контроллера «VideoController» был разработан метод «Put». Метод принимает guid канала, guid видео и модель «EditVideoModel». Метод проверяет является ли клиент владельцем канала и видео, если все верно, то данные изменяются. Возвращает код 200 OK при успешном редактировании данных видео.

* + 1. Удаление видео

Для редактирования данных видео в рамка контроллера «VideoController» был разработан метод «Delete». Метод принимает guid канала, guid. Метод проверяет является ли клиент владельцем канала и видео, если все верно, то видео удаляется. Возвращает код 200 OK при успешном удалении видео.

* + 1. Удаление комментариев под своими видео

Для удаление комментариев под своими видео в рамка контроллера «CommentController» был разработан метод «Delete». Метод принимает guid комментария. Метод проверяет является ли клиент владельцем канала, владельцем комментария или же администратором, если все верно, то комментарий удаляется. Реализация метода «Delete» предоставлено в листинге 3.12.

[HttpDelete, Authorize]

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

try{

var authData = AuthorizeData.FromContext(HttpContext);

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

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

Comment? comment = await \_db.Comments

.Include(i => i.Video)

.Include(i => i.Video!.Channel)

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

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

if (comment.UserId != authData.Id

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

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

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

\_db.Comments.Remove(comment);

await \_db.SaveChangesAsync();

return Results.Ok();

}

catch (ServerException err)

{

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

}

}

Листинг 3.12 – Реализация метола «Delete» контроллера «CommentController»

Возвращает код 200 OK при успешном редактировании данных видео.

* + 1. Оставление жалобы под видео
    2. Оставление комментариев под видео
    3. Удаление своих комментариев
    4. Редактирование своих комментариев
    5. Совместный просмотр
    6. Создание плейлистов
    7. Удаление видео из плейлиста
    8. Добавление видео в плейлиста
    9. Воспроизведение плейлиста
    10. Удаление плейлиста
    11. Удаление любых комментариев администратором
    12. Скрытия видео администратором
    13. Удаление видео администратором
    14. Просмотр жалоб пользователей администратором
    15. Блокировка канала администратором
    16. Удаление канала администратором

1. Тестирование веб-приложения
2. Руководство пользователя (руководство по эксплуатации)

Пояснительная записка дипломного проекта содержит 60 страниц

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *БГТУ 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. |  |  |

1. Технико-экономическое обоснования проекта
   1. Общая характеристика разрабатываемого программного средства

При выполнении данного проекта было разработано веб-приложение BYTUBE, предназначенное для размещения и просмотра видео контента. 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 просмотра видео в реальном времени.

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

* 1. 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.

Таблица 6.1 – Исходные данные для расчета

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

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

Таблица 6.2 – Анализ стоимости разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Продукты-аналоги | Источник | Стоимость, руб | Примечание |
| YouTube | https://youtube.com | 40 000 | Отсутствие совместного просмотра |
| Vk Video | https://vkvideo.ru | 15 000 | Привязка к учетной записи VK. |
| RUTUBE | https://rutube.ru | 27 000 | Устаревший интерфейс. |

В ходе проведения маркетингового анализа, была определена стоимость разработки аналогичного программного продукта по отслеживанию и добавлению отзывов на продукты питания. Средняя цена разработки аналогичного продукта составляет 25000-30000 рублей. Так же стоит учитывать модуль совместного просмотра, который отдельно стоит 2000 руб. Таким образом, общая стоимость разработки данного программного средства, выбранного в качестве базы сравнения ставит 26000 рублей без НДС. С НДС цена составляет 31000 рублей.

* 1. Обоснование цены программного средства
     1. 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,  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.

* + 1. Расчет основной заработной платы

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

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

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

где Соз – основная заработная плата, руб.;

Траз – трудоемкость, чел./час.;

Сзп – средняя часовая ставка, руб./час.

Таблица 6.4 – Расчет основной заработной платы специалистов

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

Суммарная основная заработная плата всех специалистов веб-приложения составит 8576 рублей.

* + 1. Расчет дополнительной заработной платы

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

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

где Соз – основная заработная плата, руб.;

Ндз – норматив дополнительной заработной платы, %.

Сдз = 9344 ⋅ 9 / 100 = 840,96 руб.

Таким образом дополнительная заработная плата составила 840,96 рублей.

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

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

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

где Соз – основная заработная плата, руб.;

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

Нфсзн – норматив отчислений в Фонд социальной защиты, %.

Отчисления в БРУСП «Белгосстрах» вычисляются по формуле 6.4.

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

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

* + 1. Расчет суммы прочих прямых затрат

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

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

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

Спз = 9344 ⋅ 25 / 100 = 2 336 руб.

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

* + 1. Расчет суммы накладных расходов

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

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

Сумма накладных расходов составит:

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

Таким образом, сумма накладных расходов составила 4 672 руб.

* + 1. Сумма расходов на разработку программного средства

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

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

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

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

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

* + 1. Расходы на сопровождение и адаптацию

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

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

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

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

* + 1. Расчет полной себестоимости

Полная себестоимость Сп определяется как сумма двух элементов: суммы

расходов на разработку Ср и суммы расходов на сопровождение и адаптацию Срса.

Полная себестоимость Сп вычисляется по формуле 6.9

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

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

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

* 1. Вывод по разделу

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

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

Таблица 6.5 – Результаты расчетов

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

Современный видеохостинг сталкивается с высокой конкуренцией и растущими требованиями пользователей к качеству контента, скорости загрузки и персонализации рекомендаций. Существующие платформы часто не учитывают региональные особенности, испытывают проблемы с модерацией и предлагают шаблонные алгоритмы, которые не всегда соответствуют интересам аудитории. BUTUBE 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.

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**

# **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: `
2. React [Электронный ресурс]. – Режим доступа: <https://react.dev/blog/2024/12/05/react-19> – Дата доступа: 24.03.2025
3. MOBX [Электронный ресурс]. – Режим доступа: <https://mobx.js.org/README.html> – Дата доступа: 24.03.2025
4. PostgreSQL [Электронный ресурс]. – Режим доступа: <https://www.postgresql.org/docs/> – Дата доступа: 24.03.2025
5. REST API [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/articles/483202/> – Дата доступа: 24.03.2025
6. SignalR [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/ru-ru/aspnet/signalr/> – Дата доступа: 24.03.2025
7. Docker [Электронный ресурс]. – Режим доступа: <https://www.docker.com> – Дата доступа: 24.03.2025
8. YouTube [Электронный ресурс]. – Режим доступа: [www.youtube.com](http://www.youtube.com) – Дата доступа: 24.03.2025
9. RUTUBE [Электронный ресурс]. – Режим доступа: <https://rutube.ru> – Дата доступа: 24.03.2025
10. VK Video [Электронный ресурс]. – Режим доступа: <https://vkvideo.ru> – Дата доступа: 24.03.2025
11. HTTPS RFC [Электронный ресурс]. – Режим доступа: <https://datatracker.ietf.org/doc/html/rfc2616> – Дата доступа: 24.03.2025
12. TCP RFC [Электронный ресурс]. – Режим доступа: <https://datatracker.ietf.org/doc/html/rfc793> – Дата доступа: 24.03.2025
13. WebSocket RFC [Электронный ресурс]. – Режим доступа: <https://datatracker.ietf.org/doc/html/rfc6455> – Дата доступа: 24.03.2025
14. N-Layer
15. Dependency Injection (DI)
16. JSON Web Token (JWT)
17. Паттерн «Repository»
18. JSON
19. A
20. A
21. A

Приложение А

Таблица Users

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

Таблица Channels

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

Таблица Videos

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

Таблица Comments

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

Таблица Playlists

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

Таблица PlaylistItems

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

Таблица Subscriptions

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

Таблица Reports

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

Таблица VideoMarks

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

Таблица VideoViews

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

Приложение Б

CREATE EXTENSION IF NOT EXISTS "uuid-ossp";

CREATE TABLE Users (

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

name TEXT,

email TEXT,

password TEXT,

birthday DATE,

role INTEGER,

token TEXT

);

CREATE TABLE Channels (

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

name TEXT,

description TEXT,

created TIMESTAMP,

status INTEGER,

user\_id UUID REFERENCES Users(ID)

);

CREATE TABLE Videos (

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

channel\_id UUID REFERENCES Channels(ID),

title TEXT,

description TEXT,

duration TEXT,

forAdults BOOLEAN,

tags JSONB,

created TIMESTAMP,

videoAccess INTEGER,

videoStatus INTEGER

);

CREATE TABLE Comments (

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

message TEXT,

likes JSONB,

created TIMESTAMP,

owner\_id UUID REFERENCES Users(ID),

video\_id UUID REFERENCES Videos(ID)

);

CREATE TABLE Playlists (

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

name TEXT,

access INTEGER,

user\_id UUID REFERENCES Users(ID)

);

CREATE TABLE PlaylistItems (

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

playlistId UUID REFERENCES Playlists(ID),

videoId UUID REFERENCES Videos(ID),

order INTEGER

);

CREATE TABLE Subscriptions (

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

user\_id UUID REFERENCES Users(ID),

channel\_id UUID REFERENCES Channels(ID)

);

CREATE TABLE Reports (

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

type INTEGER,

message TEXT,

created TIMESTAMP,

video\_id UUID REFERENCES Videos(ID)

);

CREATE TABLE VideoMarks (

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

user\_id UUID REFERENCES Users(ID),

video\_id UUID REFERENCES Videos(ID),

isLike BOOLEAN,

isDisLike BOOLEAN,

updated TIMESTAMP

);

CREATE TABLE VideoViews (

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

user\_id UUID REFERENCES Users(ID),

video\_id UUID REFERENCES Videos(ID),

created TIMESTAMP

);

Приложение В