МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

по дисциплине «Введение в нереляционные системы управления базами данных»

ТЕМА: ВИДЕОХОСТИНГ

Студенты _____ Анисимов К. 2383

_____ Деметрашвили Д. 2383

_____ Плюсов Д. 2383

_____ Слабнова Д. 2381

Каангу С. 2300

_____

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2025

ЗАДАНИЕ

Студенты

Анисимов К. 2383

Демитрашвили Д. 2383

Плюсов Д. 2383

Слабнова Д. 2381

Каангу С. 2300

Тема работы: Разработка видеохостинга

Исходные данные:

Vue, Java, Spring, TailWind. Docker, Mongo, S3(minio), Typescript

Содержание пояснительной записки:

Содержание

Введение

Макет и сценарии использования

Модель данных

Разработка приложения

Вывод

Список использованных источников

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания: 02.02.2025

Дата сдачи реферата: 29.05.2025

Дата защиты реферата: 29.05.2025

Студенты _____ Анисимов К. 2383

_____ Деметрашвили Д. 2383

_____ Плюсов Д. 2383

_____ Слабнова Д. 2381

_____ Каангу С. 2300

Преподаватель _____ Заславский М.М.

АННОТАЦИЯ

Курсовой проект представляет собой разработку видеохостинга с функционалом авторизации, регистрации, управления видео и пользователями. Основные реализованные функции: многокритериальный фильтр видео, подписки, лайки/дизлайки, комментарии, загрузка и редактирование контента. Для админов добавлены инструменты управления пользователями (блокировка, назначение ролей).

Использованы методы Full-stack разработки:

- Frontend: Vue, TypeScript
- Backend: Java, Spring
- База данных: Mongo, s3
- Деплой: Docker

Результаты:

- Рабочие приложение по типу видеохостинга.
- Разграничение прав для гостей, пользователей и админов.

SUMMARY

The course project is the development of a video hosting service with the functionality of authorization, registration, video management and users. The main implemented functions are a multi-criteria video filter, subscriptions, likes/dislikes, comments, uploading and editing content. User management tools (blocking, role assignment) have been added for admins.

Full-stack development methods are used:

- Frontend: Vue, TypeScript
- Backend: Java, Spring
- Database: Mongo, s3
- Deployment: Docker

Results:

- Working video hosting application type.
- Differentiation of rights for guests, users and admins.

ОГЛАВЛЕНИЕ

1.Введение

Актуальность решаемой проблемы

В современном мире видеохостинги стали неотъемлемой частью интернет-пространства, предоставляя пользователям возможность обмена видеоконтентом, взаимодействия через комментарии и оценки. Однако существующие платформы зачастую избыточны для локальных или специализированных решений, что создает потребность в разработке простых, но функциональных аналогов. Данный проект направлен на создание урезанной версии видеохостинга с базовым набором возможностей, включая загрузку видео, систему лайков/дизлайков, комментарии и управление контентом для разных типов пользователей.

Постановка

Требуется разработать веб-приложение, включающее:

Систему аутентификации и авторизации с ролями «Пользователь» и «Администратор».

Функционал загрузки, просмотра и управления видео (открытый/закрытый доступ).

Механизм взаимодействия с контентом (лайки, дизлайки, комментарии).

Административную панель для управления пользователями и контентом.

Хранение видеофайлов (в MongoDB или S3) и метаданных в нереляционной БД.

Предлагаемое решение

Решение предполагает использование следующего стека технологий:

Backend: Java Spring (REST API), MongoDB (хранение данных), S3 (опционально для видео).

Frontend: Vue/React или чистый HTML + JS/TS с Tailwind, сборка через Vite.

Инфраструктура: Docker для контейнеризации компонентов.

Качественные требования к решению

Масштабируемость: Архитектура должна позволять расширение функционала (например, добавление новых ролей).

Безопасность: Реализация JWT-аутентификации, разделение прав доступа.

Производительность: Оптимизация запросов к БД, кэширование для поиска.

Удобство интерфейса: Интуитивно понятный UI с адаптивным дизайном.

Надежность хранения: Корректная работа с медиафайлами (ограничение размера, обработка ошибок загрузки).

Проект послужит демонстрацией возможностей связки Spring + MongoDB для построения RESTful API и современных фронтенд-технологий для создания интерактивных веб-приложений.

2.Макет и сценарии использования

2.1. UI-макет

Предварительный макет приложения видео-хостинга и этапы его создания (рисунок 1).

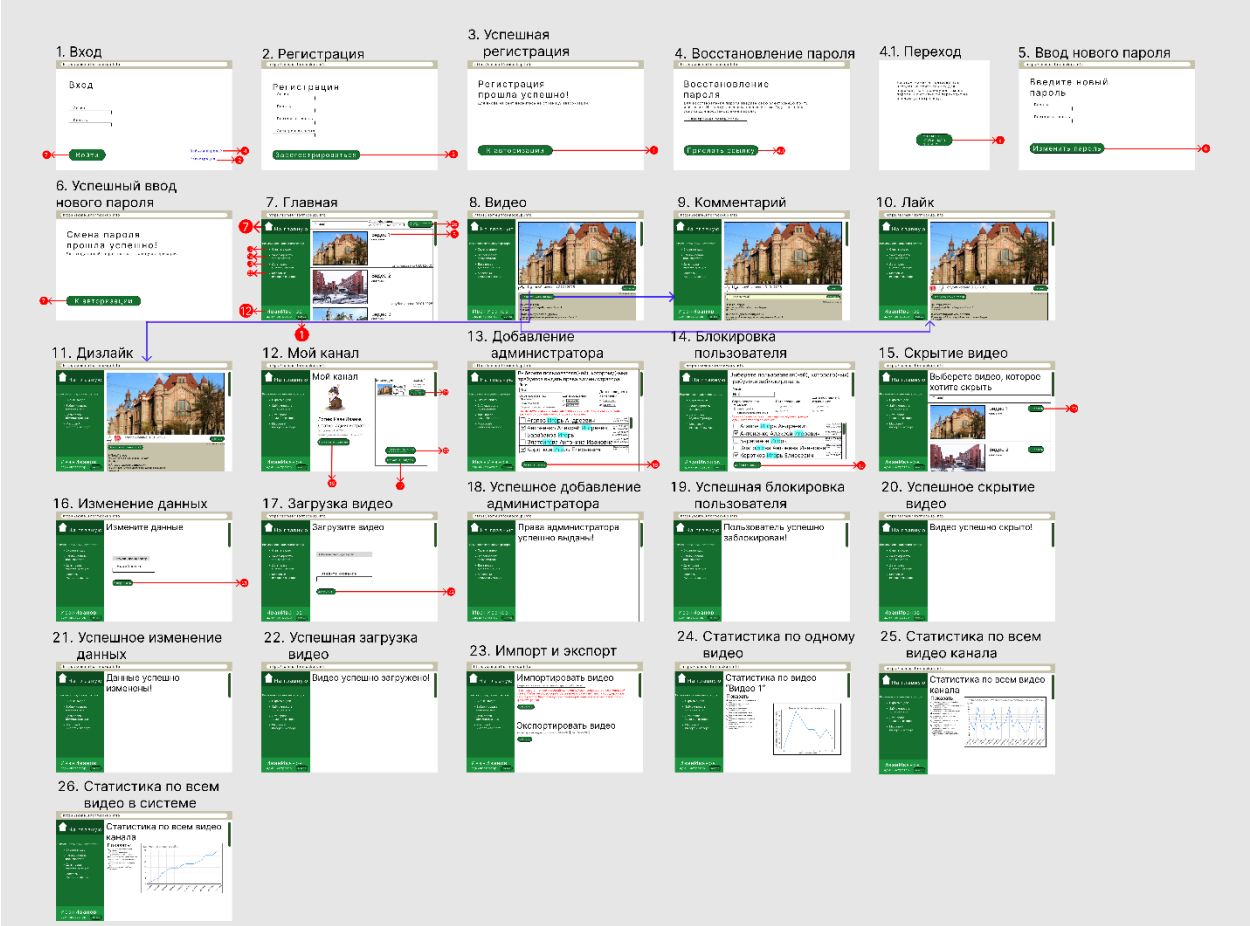

Рисунок 1. UI макет приложения

2.2. Сценарии использования

1. Сценарий: Регистрация нового пользователя

Описание: Пользователь регистрируется в системе, чтобы получить доступ к функционалу.

Основной поток:

1. Пользователь открывает страницу регистрации.
2. Система отображает форму регистрации.
3. Пользователь вводит свои данные (логин, email, пароль).
4. Пользователь нажимает кнопку "Зарегистрироваться".

5. Выполняется переход на страницу с сообщением о том, что регистрация прошла успешно.

Альтернативный поток:

Если логин занят или почта уже зарегистрирована, то после нажатия кнопки переход на страницу с сообщением о том, что регистрация прошла успешно, не происходит. Появляется сообщение о том, что пароль занят.

2. Сценарий: Вход в систему

Описание: Пользователь входит в систему, используя свои учетные данные.

Основной поток:

1. Пользователь открывает страницу входа.
2. Система отображает форму входа.
3. Пользователь вводит логин и пароль.
4. Пользователь нажимает кнопку "Войти".
5. Система проверяет учетные данные.
6. Система предоставляет доступ к личному кабинету и переводит на главную страницу.

Альтернативный поток:

Если данные неверны, система отображает сообщение об ошибке. Перехода на главную страницу не происходит.

3. Сценарий: Просмотр видео с комментированием и реакцией.

Описание: Пользователь включает видео, ставит лайк/дизлайк и оставляет комментарий. Предусловие: пользователь авторизован и находится на главной странице.

Основной поток:

1. Пользователь открывает одно из видео на экране.
2. Система переводит на страницу с видео и отображает его на экране.
3. Во время просмотра пользователь видит список комментариев под видео.
4. Пользователь нажимает на поле "Написать комментарий".

5. Пользователь вводит текст комментария и нажимает кнопку "Отправить".
6. Система добавляет комментарий пользователя в список и обновляет страницу.
7. Пользователь нажимает на кнопку "Лайк" или "Дизлайк" под видео, чтобы выразить свою реакцию.
8. Система обновляет счетчик реакций и сохраняет выбор пользователя.

4. Сценарий: Загрузка видео на аккаунт

Описание: Пользователь загружает видео на свой аккаунт для публикации на платформе. Предусловие: пользователь авторизован.

Основной поток:

1. Пользователь входит в свой канал.
2. Пользователь нажимает кнопку "Загрузить видео".
3. Система открывает страницу загрузки видео.
4. Пользователь выбирает видеофайл(ы) на своем устройстве.
5. Пользователь заполняет название.
6. Пользователь нажимает кнопку "Загрузить".
7. Система загружает видео.
8. Система публикует видео на аккаунте пользователя.

5. Сценарий: Администратор скрывает видео

Описание: Администратор скрывает видео, нарушающее правила платформы. Предусловие: администратор авторизован.

Основной поток:

1. Администратор переходит на страницу с видео для скрытия.
2. Администратор нажимает кнопку "Скрыть видео".
3. Система скрывает видео от публичного доступа. Администратора переводит на страницу с сообщением об успешном выполнении операции.

6. Сценарий: Администратор блокирует пользователя

Описание: Администратор блокирует пользователя. Предусловие: администратор авторизован.

Основной поток:

1. Администратор переходит на страницу блокировки пользователей.
2. Администратор вводит логин блокируемого пользователя.
3. Администратор нажимает кнопку "Заблокировать пользователя".
4. Система блокирует аккаунт пользователя. Администратора переводит на страницу с сообщением об успешном выполнении операции.

Альтернативные потоки:

Если такого пользователя нет, то система сообщает об этом администратора. Перехода на страницу с сообщением об успехе не происходит.

7. Сценарий: Администратор дает права администратора

Описание: Администратор назначает другому пользователю права администратора. Предусловие: администратор авторизован.

Основной поток:

1. Администратор переходит на страницу выдачи прав администратора.
2. Администратор вводит логин пользователя, которому выдаются права.
3. Администратор нажимает кнопку "Выдать права".
4. Система предоставляет пользователю права администратора. Администратора переводит на страницу с сообщением об успешном выполнении операции.

Альтернативные потоки:

Если такого пользователя нет, то система сообщает об этом администратора. Перехода на страницу с сообщением об успехе не происходит.

8. Сценарий: Скачивание видео

Описание: Пользователь скачивает видео с платформы на свое устройство.

Основной поток:

1. Пользователь открывает страницу с видео.
2. Пользователь нажимает кнопку "Скачать".
3. Система начинает загрузку видео на устройство пользователя.

4. Пользователь получает видео в указанной папке.

9. Сценарий: Массовый импорт

Описание: Пользователь-администратор загружает архив видеофайлами поддерживаемых форматов на платформу.

Основной поток:

1. Администратор переходит на страницу "массовый импорт и экспорт" через боковую панель.
2. Пользователь загружает архив с видеофайлами. Извлекаются только видеофайлы, находящиеся непосредственно в корневой папке (корне архива). Файлы не видео форматов, а также вложенные папки и их содержимое игнорируются. Каждое видео сохраняет оригинальное имя файла (без пути и формата файла).
3. Пользователь нажимает кнопку загрузить.
4. Система загружает видео на платформу. Автором видео считается администратор, загрузивший их.

Альтернативные потоки:

Если будет передан файл неверного формата (не архив), то на экране появится об этом сообщение после нажатия кнопки загрузить.

10. Сценарий: Массовый экспорт

Описание: Пользователь-администратор скачивает видеофайлы с платформы.

Основной поток:

1. Администратор переходит на страницу "массовый импорт и экспорт" через боковую панель.
2. Пользователь выбирает период создания видео (от DD.MM.YYYY включительно до DD.MM.YYYY включительно), которые он хочет скачать.
3. Пользователь нажимает кнопку скачать.
4. Видео скачиваются на устройство пользователя в том же формате, в котором хранятся в базе.

Альтернативные потоки:

Если будет передан файл неверного формата (не архив), то на экране появится об этом сообщение.

11. Сценарий: Статистика по одному видео

Описание: Автор видео хочет посмотреть статистику по одному конкретному своему видео. Статистика по видео доступна только автору этого видео.

Основной поток:

1. Автор переходит на страницу канала.
2. Автор нажимает на кнопку "посмотреть статистику по этому видео" рядом с видео, статистику по которому он хочет посмотреть.
3. Выполняется переход на страницу со статистикой.
4. На странице пользователь нажимает кнопку, соответствующую тем данным, которые он хочет посмотреть.

12. Сценарий: Статистика по видео на канале

Описание: Автор видео хочет посмотреть статистику по всем видео канала. Статистика по видео канала доступна только владельцу этого канала.

Основной поток:

1. Автор переходит на страницу канала.
2. Автор нажимает на кнопку "посмотреть статистику по всем видео канала" внизу окна, в котором находятся видео.
3. Выполняется переход на страницу со статистикой по всем видео канала.
4. На странице пользователь нажимает кнопку, соответствующую тем данным, которые он хочет посмотреть.

12. Сценарий: Статистика по видео в системе

Описание: Пользователь видео хочет посмотреть статистику по всем видео в системе.

Основной поток:

1. Автор переходит на главную страницу.
2. Автор нажимает на кнопку "посмотреть статистику по всем видео" вверху экрана.

3. Выполняется переход на страницу со статистикой по всем видео в системе.
4. На странице пользователь нажимает кнопку, соответствующую тем данным, которые он хочет посмотреть.

13. Замечание

Администратор имеет те же возможности, что и обычный пользователь, плюс возможности блокировки пользователей, назначения администраторов, скрытия видео и массового импорта и экспорта данных. Чтобы получить доступ к видео, нужно авторизоваться, т.е. без авторизации видео не видны.

3.Модель данных

3.1 Нереляционная модель

3.1.1 Описание назначений коллекций, типов данных и сущностей

Пользователи (users):

- id: ObjectId (12 байт)
- username: String (до 50 символов, ~50 байт)
- email: String (до 100 символов, ~100 байт)
- password: String (60 байт для bcrypt)
- registrationDate: Date (8 байт)
- subscriberCount: Integer (4 байта)
- blocked: Boolean (1 байта) Итого: ~235 байта на пользователя

Видео (videos):

- id: ObjectId (12 байт)
- userId: ObjectId (12 байт)
- title: String (до 100 символов, ~100 байт)
- description: String (до 5000 символов, ~5000 байт)
- s3Key: String (до 255 символов, ~255 байт)
- views: Integer (4 байта)
- uploadDate: Date (8 байт)
- durationSeconds: Integer (4 байта)
- tags: Array (до 10 элементов по 20 байт каждый, ~200 байт) Итого: ~5595 байт на видео

Комментарии (comments):

- id: ObjectId (12 байт)
- userId: ObjectId (12 байт)
- videoId: ObjectId (12 байт)
- text: String (до 1000 символов, ~1000 байт)
- timestamp: Date (8 байт) Итого: ~1044 байта на комментарий

Реакции(reactions):

- id: ObjectId (12 байт)
- userId: ObjectId (12 байт)
- videoId: ObjectId (12 байт)
- type: String(до 10 символов, “LIKE”/“DISLIKE”, ~10 байт)
- timestamp: Date (8 байт) Итого: 54 байта на лайк

3.1.2 Оценка объема информации, хранимой в модели

Общий объем данных можно выразить через количество пользователей (U), видео (V), комментариев (C) и лайков (L):

TotalSize = 235*U + 5595*V + 1044*C + 54*R + S3Content

На 1 пользователя 10 видео, на 1 видео 5 комментариев и 60 реакций:

TotalSize ≈ 235*U + 5595*(10*U) + 1044*(5*10*U) + 54*(60*10*U)
≈ 235*U + 55950*U + 52200*U + 32400*U
≈ 140,785*U байт ≈ 140.8 KB на пользователя

3.1.3 Избыточность модели

Чистый объем данных (без служебных полей и индексов):

PureSize ≈ **(**50+100+60**)***U + **(**100+5000+255**)***V + **(**1000**)***C + **(**0**)***R
≈ 210*U + 5355*V + 1000*C

Избыточность данных:

Redundancy = **(**TotalSize - PureSize**)** / PureSize
≈ **(**130,384*U - **(**210*U + 5355*10*U + 1000*5*10*U**))** / **(**210*U + 53550*U + 50000*U**)**
≈ **(**130,384 - 103,760**)** / 103,760 ≈ 0.256 или 25.6%

3.1.4 Направление роста модели при увеличении количества объектов каждой сущности

При увеличении количества:

- Пользователей: линейный рост (O(U))

- Видео: линейный рост (O(V)), но в 24 раза больше на единицу, чем пользователи
- Комментариев: линейный рост (O(C)), в 4.5 раза больше на единицу, чем видео
- Реакций: линейный рост (O(L)), но очень компактные записи

3.1.5 Примеры данных

Пользователь:

```
{
  "id": ObjectId("5f8d8a7f4d4e4b2d4c8b4567"),
  "username": "tech_reviewer",
  "email": "reviewer@example.com",
  "password": "$2a$10$N9qo8uLOickgx2ZMRZoMy...",
  "registrationDate": ISODate("2023-01-15T10:00:00Z"),
  "subscriberCount": 25000,
  "blocked": false
}
```

Видео:

```
{
  "id": ObjectId("5f8d8a7f4d4e4b2d4c8b4999"),
  "userId": ObjectId("5f8d8a7f4d4e4b2d4c8b4567"),
  "title": "Обзор",
  "description": "Полный обзор",
  "s3Key": "videos/tech_review_123.mp4",
  "views": 150000,
  "uploadDate": ISODate("2023-05-20T14:30:00Z"),
  "durationSeconds": 720,
  "tags": ["техника", "обзор", "смартфон"]
}
```

Комментарий:

```
{
  "id": ObjectId("5f8d8a7f4d4e4b2d4c8b4777"),
  "userId": ObjectId("5f8d8a7f4d4e4b2d4c8b4000"),
  "videoId": ObjectId("5f8d8a7f4d4e4b2d4c8b4999"),
  "text": "Отличный обзор, спасибо!",
  "timestamp": ISODate("2023-05-21T09:15:00Z")
}
```

Реакция:

```
{
  "id": ObjectId("5f8d8a7f4d4e4b2d4c8b4666"),
  "userId": ObjectId("5f8d8a7f4d4e4b2d4c8b4001"),
  "videoId": ObjectId("5f8d8a7f4d4e4b2d4c8b4999"),
  "type": "LIKE",
  "timestamp": ISODate("2023-05-20T15:45:00Z")
}
```

3.1.6 Примеры запросов

Получить все видео пользователя:

```
db.videos.find({
  userId: ObjectId("5f8d8a7f4d4e4b2d4c8b4567")
}).sort({ uploadDate: -1 })
```

Получить топ-10 самых популярных видео:

```
db.videos.find()
  .sort({ views: -1 })
  .limit(10)
```

Получить видео с информацией об авторе и количеством реакций:

```
const videos = db.videos.aggregate([
  {
    $match: { tags: "техника" }
  },
  {
    $lookup: {
      from: "users",
      localField: "userId",
      foreignField: "id",
      as: "author"
    }
  },
  {
    $lookup: {
      from: "reactions",
      localField: "id",
      foreignField: "videoId",
      as: "reactions"
    }
  },
  {
    $addFields: {
      likeCount: {
```

```
      $size: {
        $filter: {
          input: "$reactions",
          as: "reaction",
          cond: { $eq: ["$$reaction.type", "LIKE"] }
        }
      }
    },
    dislikeCount: {
      $size: {
        $filter: {
          input: "$reactions",
          as: "reaction",
          cond: { $eq: ["$$reaction.type", "DISLIKE"] }
        }
      }
    }
  }
 },
 {
  $project: {
    reactions: 0
  }
 }
])
```

Получить ленту подписок (видео от подписанных авторов):

```
const subscribedVideos = db.videos.aggregate([
 {
  $match: {
    userId: {
      $in: db.subscriptions
        .find({ followerId: currentUserId })
        .map(sub => sub.followingId)
    }
  }
 },
 {
  $sort: { uploadDate: -1 }
 },
 {
  $limit: 20
 },
 {
```

```
    $lookup: {
      from: "users",
      localField: "userId",
      foreignField: "id",
      as: "author"
    }
  }
])
```

3.2 Реляционная модель

3.2.1 Описание назначений коллекций, типов данных и сущностей

USER

- id: INT (4 байта)
- username: VARCHAR(50) (~50 байт)
- email: VARCHAR(100) (~100 байт)
- password: VARCHAR(60) (60 байт)
- registration_date: DATETIME (8 байт)
- subscriber_count: INT (4 байта)
- blocked: BOOLEAN(1 байта) Итого: ~227 байт на пользователя

VIDEO

- id: INT (4 байта)
- user_id: INT (4 байта)
- title: VARCHAR(100) (~100 байт)
- description: TEXT (~5000 байт)
- s3_key: VARCHAR(255) (~255 байт)
- views: INT (4 байта)
- upload_date: DATETIME (8 байт)
- duration_seconds: INT (4 байта) Итого: ~5383 байта на видео

VIDEO_TAG

- video_id: INT (4 байта)
- tag: VARCHAR(20) (~20 байт) Итого: 24 байта на тег

COMMENT

- id: INT (4 байта)
- user_id: INT (4 байта)
- video_id: INT (4 байта)
- comment_text: TEXT (~1000 байт)
- timestamp: DATETIME (8 байт) Итого: ~1020 байт на комментарий

REACTIONS

- user_id: INT (4 байта)
- video_id: INT (4 байта)
- type: VARCHAR(10) (~10 байт)
- timestamp: DATETIME (8 байт) Итого: 26 байт на лайк

3.2.2. Оценка объема информации, хранимой в модели

Общий объем данных можно выразить через количество пользователей (U), видео (V), комментариев (C) и лайков (L):

TotalSizeSQL = 227*U + 5383*V + 24*T + 1020*C + 26*R

10 видео на пользователя, 5 тегов на видео, 5 комментариев на видео, 60 реакций на видео:

TotalSizeSQL ≈ 227*U + 5383*(10*U) + 24*(5*10*U) + 1020*(5*10*U) + 26*(60*10*U)
≈ 227*U + 53830*U + 1200*U + 51000*U + 15600*U
≈ 121,857*U байт ≈ 121.9 KB на пользователя

3.2.3 Избыточность данных

Чистый объем данных:

PureSizeSQL ≈ **(**50+100+60**)***U + **(**100+5000+255**)***V + **(**20**)***T + **(**1000**)***C
≈ 210*U + 5355*V + 20*T + 1000*C

Избыточность:

RedundancySQL = **(**114,256*U - **(**210*U + 5355*10*U + 20*5*10*U + 1000*5*10*U**))** /
(210*U + 53550*U + 1000*U + 50000*U**)**
≈ **(**114,256 - 104,760**)** / 104,760 ≈ 0.091 или 9.1%

3.2.4 Направление роста модели при увеличении количества объектов каждой сущности.

При увеличении количества: Пользователи (USERS):

- Линейный рост O(U)
- Каждый новый пользователь добавляет ~226 байт
- Вторичный рост: каждый пользователь потенциально создает видео, комментарии, реакции

Видео (VIDEOS):

- Линейный рост O(V), но в 24 раза больше на единицу, чем пользователи
- Каждое видео добавляет ~5383 байта + 5 тегов по 24 байта = ~5503 байта
- Основной драйвер роста базы данных

Теги (VIDEO_TAGS):

- Линейный рост O(T), зависит от количества видео
- Обычно 5-10 тегов на видео, каждый добавляет 24 байта

Комментарии (COMMENTS):

- Линейный рост O(C), в среднем 5 комментариев на видео
- Каждый комментарий добавляет ~1020 байт
- Может стать значительной частью базы при активном комментировании

Реакции(REACTIONS):

- Линейный рост O(L), в среднем 60 реакций на видео
- Компактные записи, но при массовых лайках объем может быть значительным

3.2.5 Примеры данных

Пользователь:

```
{
  "users": [
    {
      "id": 1,
      "username": "tech_reviewer",
      "email": "reviewer@example.com",
      "password": "hashed_password_123",
      "registration_date": "2023-01-15T10:00:00Z",
      "subscriber_count": 25000,
      "blocked": false
    }]
}
```

Видео:

```
{
  "videos": [
    {
      "id": 101,
      "user_id": 1,
      "title": "Обзор",
```

```
    "description": "Полный обзор ",
    "s3_key": "videos/tech_review_123.mp4",
    "views": 150000,
    "upload_date": "2023-05-20T14:30:00Z",
    "duration_seconds": 720
  }]
}
```


Комментарий:


```
{
 "comments": [
   {
    "id": 1001,
    "user_id": 2,
    "video_id": 101,
    "comment_text": "Отличный обзор, спасибо!",
    "timestamp": "2023-05-21T09:15:00Z"
   }]
}
```


Реакция:


```
{
 "reactions": [
   {
    "user_id": 2,
    "video_id": 101,
    "type": "LIKE",
    "timestamp": "2023-05-20T15:45:00Z"
   },
   {
    "user_id": 3,
    "video_id": 101,
    "type": "DISLIKE",
    "timestamp": "2023-05-20T16:10:00Z"
   }]
}
```


Подписчики:


```
{
 "subscriptions": [
   {
    "follower_id": 2,
    "following_id": 1,
    "timestamp": "2023-03-01T09:00:00Z"
```

```
   }]
}
```

Тэг:

```
{
  "video_tags": [
    {
      "video_id": 101,
      "tag": "техника"
    }]
}
```

3.2.6 Примеры запросов

Получить все видео пользователя:

```
SELECT * FROM VIDEOS
WHERE user_id = 1
ORDER BY upload_date DESC;
```

Получить топ-10 самых популярных видео:

```
SELECT * FROM VIDEOS
ORDER BY views DESC
LIMIT 10;
```

Получить видео с информацией об авторе и количеством реакций:

```
SELECT
  v.*,
  u.username AS author_name,
  u.subscriber_count AS author_subscribers,
  COUNT(CASE WHEN r.type = 'LIKE' THEN 1 END) AS like_count,
  COUNT(CASE WHEN r.type = 'DISLIKE' THEN 1 END) AS dislike_count
FROM VIDEOS v
JOIN USERS u ON v.user_id = u.id
LEFT JOIN REACTIONS r ON v.id = r.video_id
WHERE EXISTS (
  SELECT 1 FROM VIDEO_TAGS vt
  WHERE vt.video_id = v.id AND vt.tag = 'техника'
)
GROUP BY v.id, u.username, u.subscriber_count
ORDER BY v.views DESC;
```

Получить ленту подписок (видео от подписанных авторов):

```
SELECT
  v.*,
```

```
  u.username AS author_name,
  u.subscriber_count AS author_subscribers
FROM VIDEOS v
JOIN USERS u ON v.user_id = u.id
WHERE v.user_id IN (
  SELECT following_id FROM SUBSCRIPTIONS
  WHERE follower_id = :currentUserId
)
ORDER BY v.upload_date DESC
LIMIT 20;
```

3.3 Сравнение моделей

A. Удельный объем информации

- NoSQL: ~140.8 KB на пользователя
- SQL: ~121.9 KB на пользователя

SQL модель более компактна (на ~13.4% меньше объема) благодаря:

- Отсутствию избыточных ObjectId (12 байт vs 4 байта INT)
- Нормализованным структурам (отдельная таблица для тегов)
- Отсутствию служебных полей MongoDB

B. Запросы по юзкейсам

1. Просмотр видео с комментированием и реакцией:
- NoSQL: 1 запрос (агрегация) (Требуется понимание pipeline-агрегаций)
- SQL: 4 запроса (можно сократить до 2-3 с джойнами)
2. Лента подписок:
- NoSQL: 1 запрос (агрегация) (Требуется понимание pipeline-агрегаций)
- SQL: 1 запрос (с джойнами)
3. Статистика по одному видео:
- NoSQL: Обычно 3-4 отдельных запроса. Можно кэшировать счетчики в документе видео.
- SQL: 1 запрос с JOIN и агрегацией
4. Получение видео с автором и статистикой реакций:
- NoSQL: 1 запрос (агрегация) (Требуется понимание pipeline-агрегаций)
- SQL: 1 запрос с JOIN
5. Получение ленты видео (подписки пользователя):
- NoSQL: 2 запроса (получение подписок + агрегация)
- SQL: 2 запроса (Получаем ID подписок и Получаем видео подписок с информацией об авторах с JOIN)

Критерий	SQL модель	NoSQL модель
Целостность данных	Строгая (внешние ключи)	Требует дополнительных проверок
Сложность запросов	JOIN-ы могут быть сложными	Агрегации требуют обучения
Производительность	Оптимальна для сложных запросов	Лучше для простых/частых запросов
Масштабируемость	Вертикальное масштабирование	Горизонтальное масштабирование
Гибкость схемы	Требует миграций	Можно добавлять поля без изменений

3.3.1 Вывод

Для видеохостинга NoSQL (MongoDB) модель предпочтительнее по следующим причинам:

1. Гибкость схемы: легко добавлять новые поля к видео или пользователям без миграций
2. Производительность на чтение: Вложенные структуры (комментарии, лайки) можно денормализовать для уменьшения JOIN-ов
3. Масштабируемость: Горизонтальное масштабирование MongoDB лучше подходит для высоконагруженных сервисов
4. Хранение медиа: Лучшая интеграция с S3 через ссылки в документах

SQL модель выигрывает в:

- Целостности данных (внешние ключи)
- Более компактном хранении
- Сложных аналитических запросах (но это не основной сценарий для видеохостинга)

4.Разработка приложения

Приложение состоит из трех модулей: бекенд, для реализации серверной части, фронтенд, для клиентской и докер для сборки этих модулей воедино. Используется многослояная архитектура(трехслойная), интерфейс

пользователя, сервер для обработки запросов-ответов и сервер бд. На бекенде используется монолит, на фронтенде – FSD.

Интерфейс фронтенда состоит из ряда страниц: страницы регистрации (рисунок 2), страницы логина (рисунок 3), главной страницы (рисунок 4), страницы массового импорта и экспорта данных (рисунок 5), страницы профиля (рисунок 6), страницы-списка всех пользователей (рисунок 7), страницы статистики видео (рисунок 8) и страницы восстановления пароля (рисунок 9).

Используемые технологии - Vue, Java, Spring, TailWind. Docker, Mongo, S3(minio), Typescript.

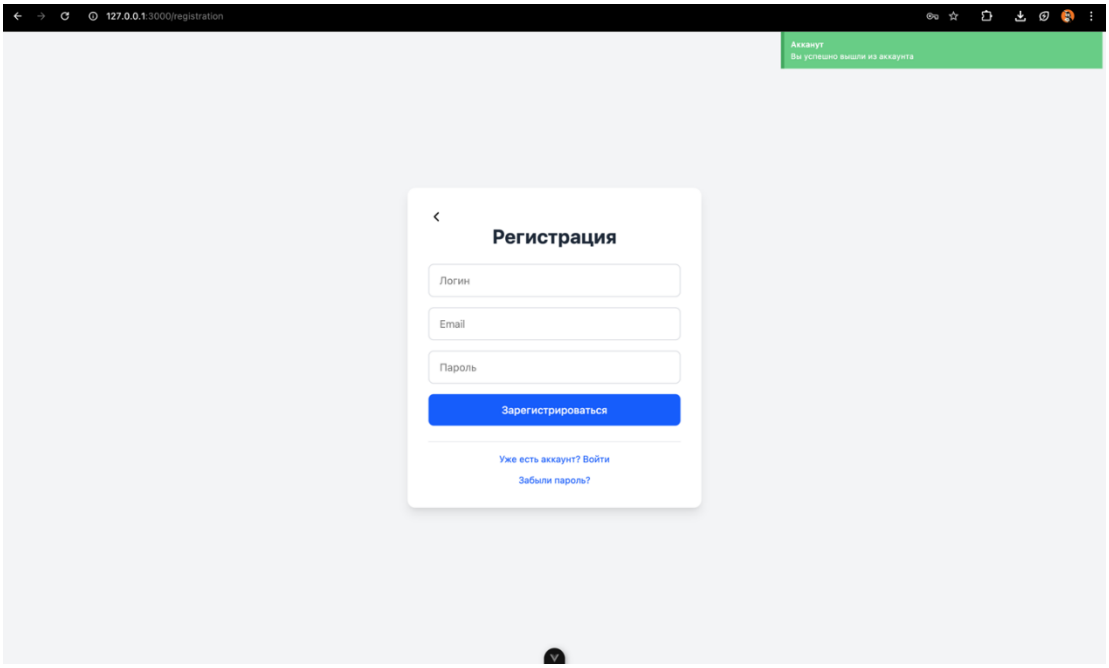

Рисунок 2- Страница регистрации пользователей

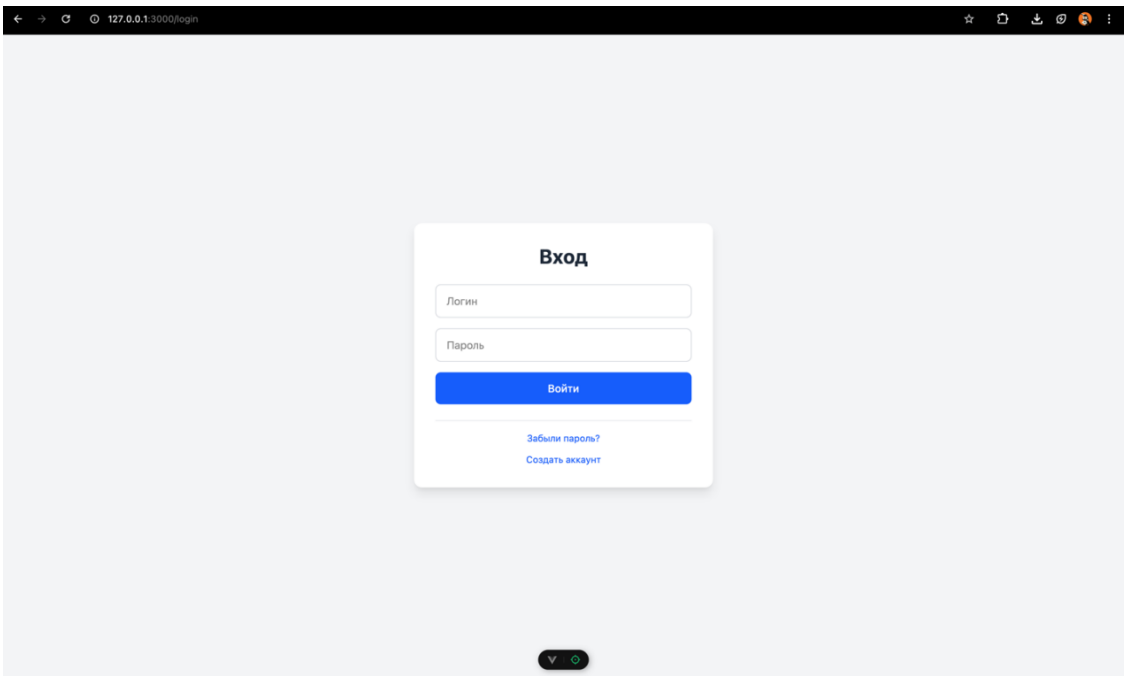

Рисунок 3 - Страница логина

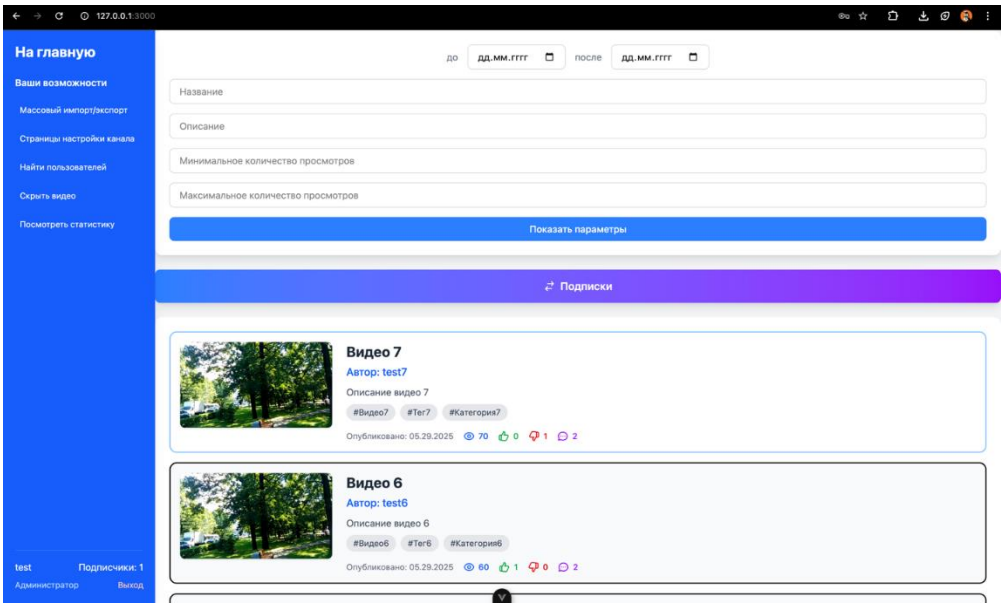

Рисунок 4 - Главная страница

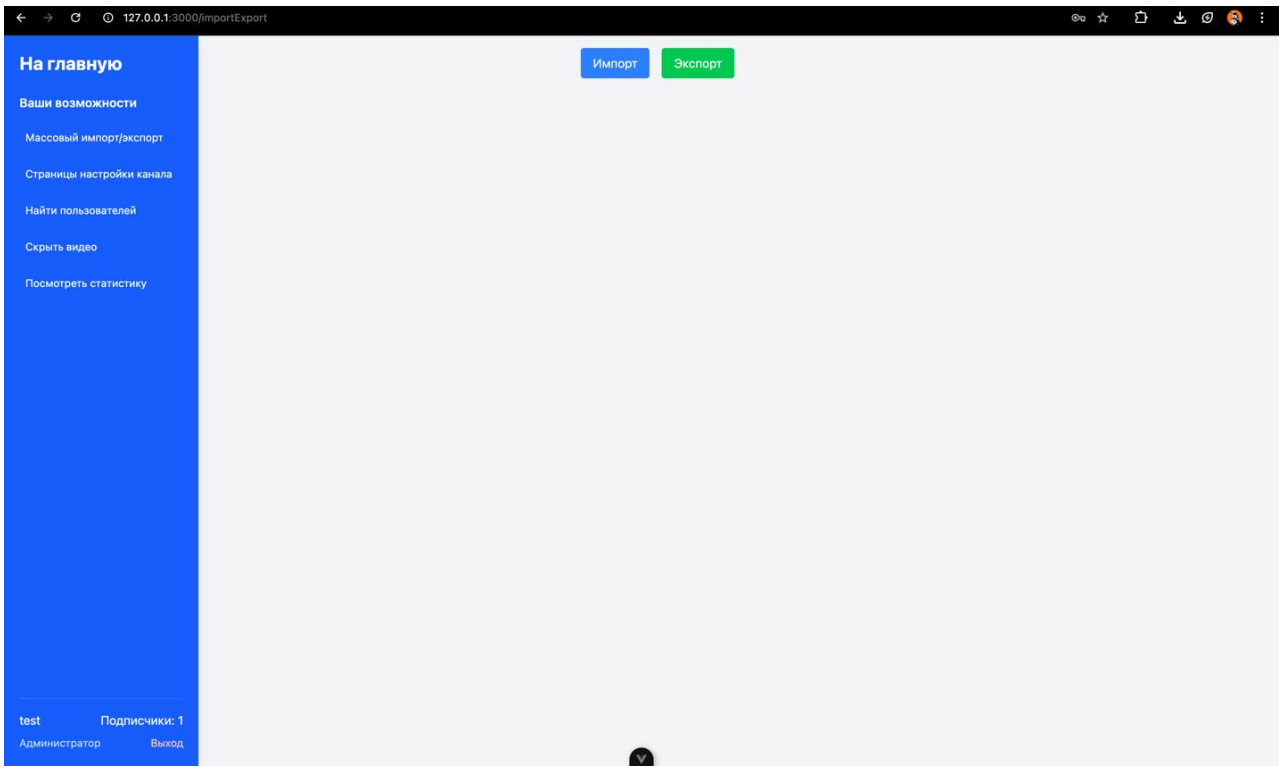

Рисунок 5 - Страница массового импорта и экспорта данных

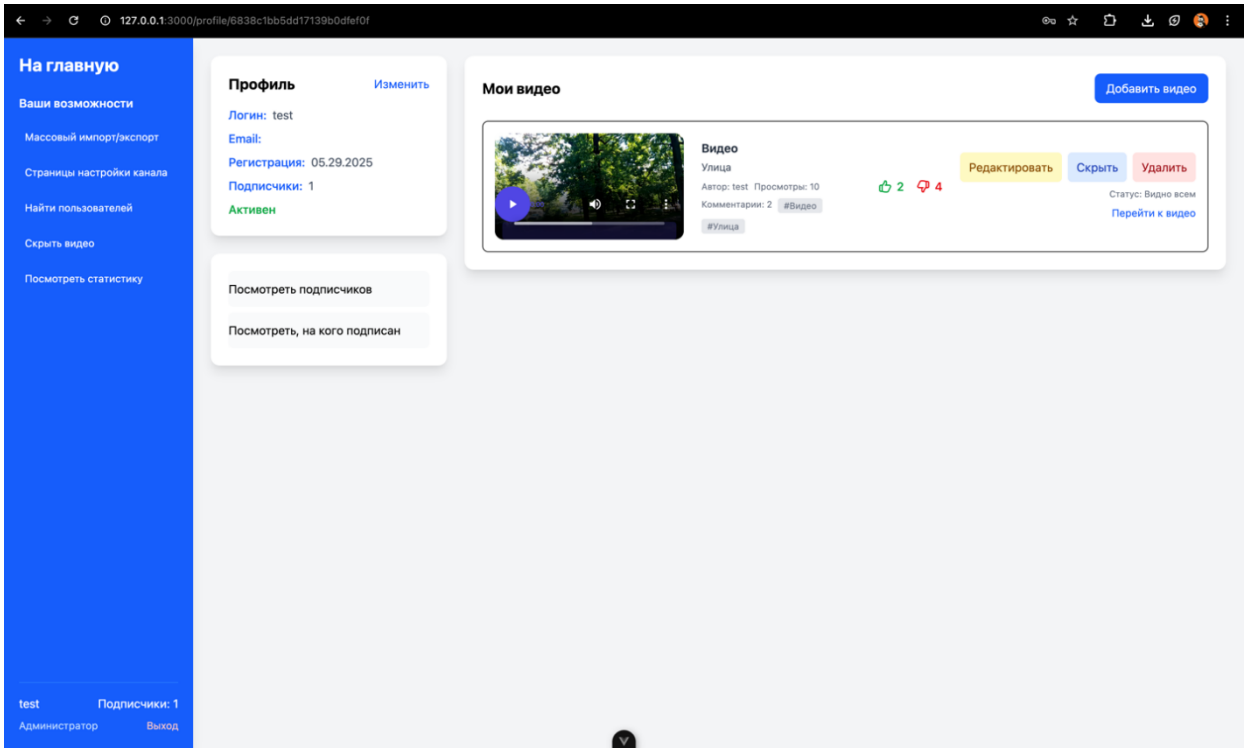

Рисунок 6 - Страница профиля

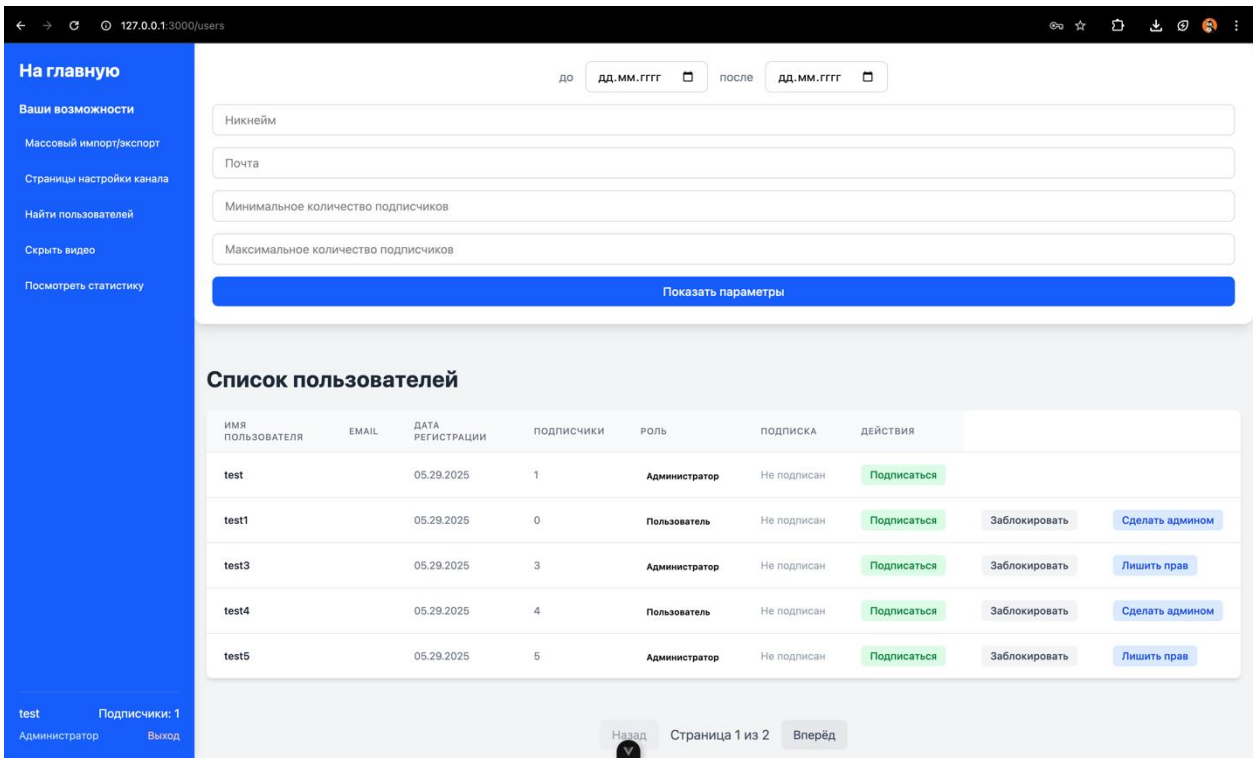

Рисунок 7 - Страница-список всех пользователей

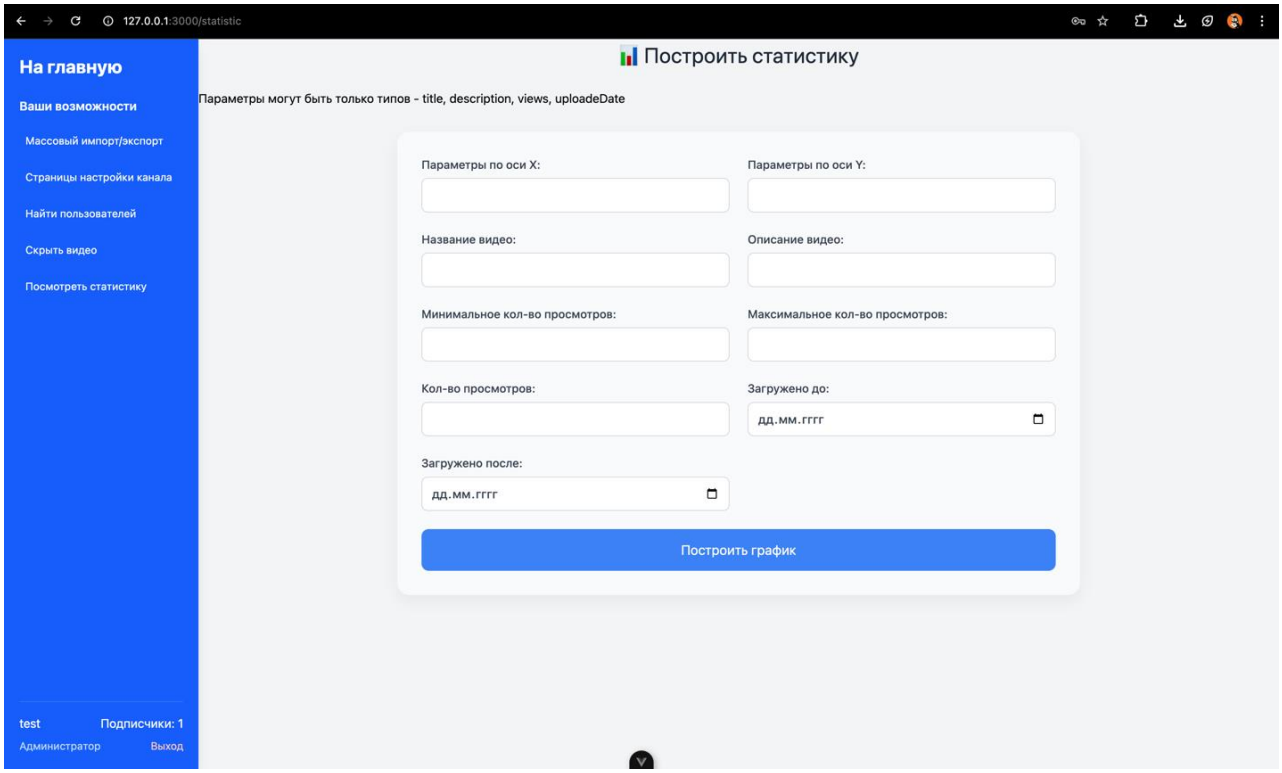

Рисунок 8 - Страница статистики видео

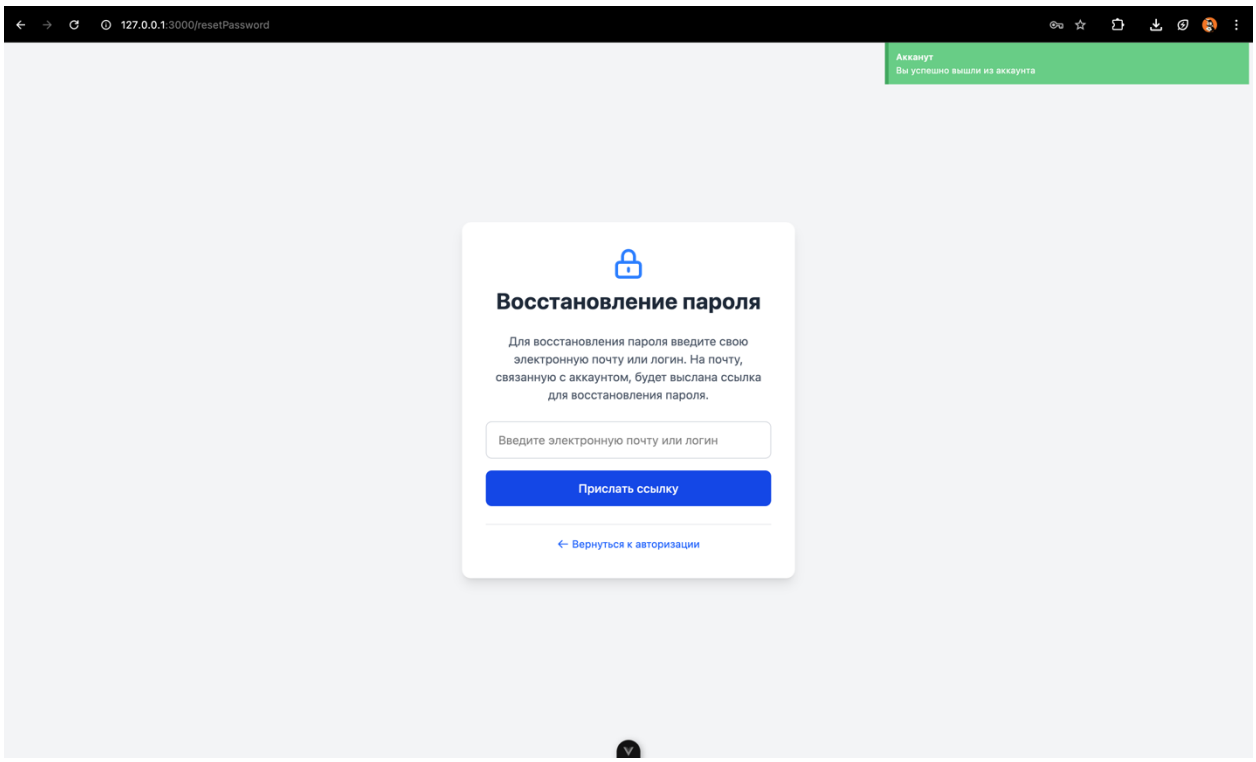

Рисунок 9 - Страница восстановления пароля

5.Вывод

5. 1 Достигнутые результаты

В ходе разработки проекта была успешно реализована урезанная версия видеохостинга с ключевыми функциями:

Система аутентификации и авторизации с ролями «Пользователь» и «Администратор».

Загрузка, хранение и воспроизведение видео с возможностью ограничения доступа (открытые/закрытые ролики).

Механизм взаимодействия с контентом (лайки, дизлайки, комментарии).

Умный поиск с дебаунсом для удобной навигации.

Административная панель для управления пользователями и модерации контента.

Развертывание системы в Docker-контейнерах, обеспечивающее переносимость и масштабируемость.

5.1 Недостатки и пути для улучшения полученного решения

Несмотря на достигнутые результаты, проект имеет ряд ограничений, которые можно устранить в будущем:

Оптимизация хранения видео – при масштабировании лучше полностью перейти на S3-совместимое хранилище вместо хранения в MongoDB.

Расширенная аналитика – добавление статистики просмотров, графиков активности пользователей.

Улучшенная система рекомендаций – внедрение алгоритмов на основе предпочтений пользователей (лайки/дизлайки).

Поддержка стриминга – адаптация под большие видеофайлы с поточным воспроизведением.

Тестирование и отказоустойчивость – добавление unit- и интеграционных тестов, улучшение обработки ошибок.

5.2 Будущее развитие решения

Проект имеет потенциал для превращения в полноценную платформу для видеохостинга. Следующие шаги развития:

Мобильное приложение (React Native/Flutter) для расширения аудитории.

Социальные функции – подписки, плейлисты, уведомления.

Монетизация – рекламные интеграции, платные подписки.

AI-модерация – автоматическое выявление запрещенного контента.

Микросервисная архитектура – разделение сервисов (аутентификация, видео, аналитика) для повышения отказоустойчивости.

Таким образом, текущая реализация является рабочим MVP, который можно развивать в полноценный аналог YouTube с учетом современных технологий и потребностей пользователей.

6. Приложения

Для запуска приложения необходимо использовать docker. Инструкция по эксплуатации описана в README файле, который находится в репозитории (рисунок 10). Также в файле написаны тестовые данные для входа супер-пользователя для наглядной демонстрации полного функционала приложения (рисунок 11).

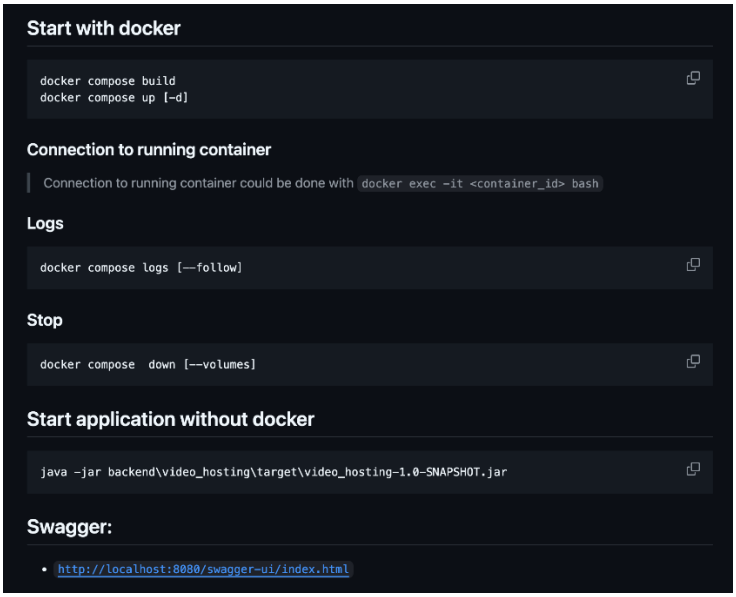

Рисунок 10 – Инструкция по использованию docker

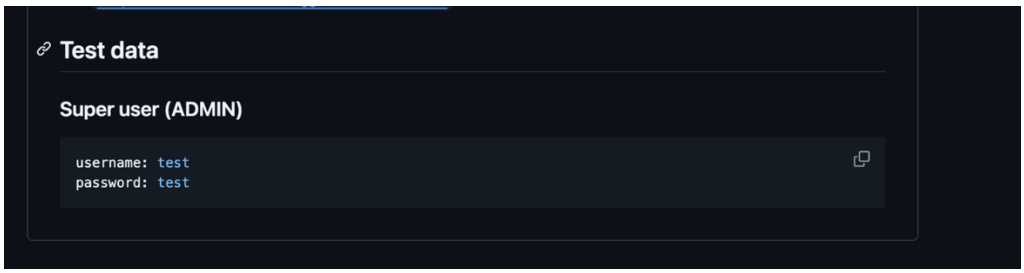

Рисунок 11 – Данные для логина супер-пользователя

7. Литература

1. GitHub // github repo URL: https://github.com/moevm/nosql1h25-videohost

2. Spring Guides // spring URL: https://spring.io/guides

3. Java Platform, Standard Edition (Java SE) // Help Center URL: https://docs.oracle.com/en/java/

4. Introduction // Vue.js url: https://vuejs.org/guide/introduction

5. TypeScript Documentation // TypeScript URL: https://www.typescriptlang.org/docs/

6. DockerDocs // dockerdocs URL: https://docs.docker.com/