Контекст

Вам предстоит разработать архитектуру видеохостинга. Как вам хорошо известно, архитектура — это модель какого-либо процесса или предприятия. Модель можно передать другим участникам команды разными способами: текстом, схемами, в результате серии встреч.

Вам нужно разработать пользовательские истории для двух типов акторов: **владелец видео** и **зритель**. На основании этих пользовательских историй потребуется отрисовать BPMN-диаграммы:
загрузки/редактирования/удаления/просмотра статистики видео со стороны владельца;
просмотра/комментирования/лайка видео со стороны зрителя.
Дополнительные баллы начисляются за диаграммы компонентов, развертывания или взаимодействия, текстовое описание пользовательских историй, разработку собственного хранилища.

Подробное описание видеохостинга располагается под описанием задания.

Задание

1. Разработайте пользовательские истории для владельца видео:
· регистрация;
· вход/выход в личный кабинет;
· загрузка видео, добавление название и описания;
· удаление видео;
· редактирование видео;
· просмотр статистики по видео (просмотри, лайки).
Результати должни сохраняться в документ в формате Word или google-документ.

2. Разработайте пользовательские истории для зрителя:
· поиск видео;
· просмотр видео;
· регистрация
· вход/выход на странице просмотра;
· лайк/отзыв лайка видео;
· комментарий к видео;
· удаление комментария к видео.
Результати должни сохраняться в документ в формате Word или google-документ.

3. Создайте BPMN-диаграмму загрузки видео со стороны владельца видео.

4. Создайте диаграмму компонентов системы.

5. Создайте диаграмму развертивания системы.

6. Создайте диаграмму взаимодействия по загрузке видео в систему.

7. Допускается использование S3-хранилища для видео, но добавление в вышеперечисленние диаграммы реализации собственного хранилища позволит вам получить дополнительные баллы для зачета.

Описание видеохостинга

Ключевые требования к системе

1. доступность
2. бесперебойность
3. производительность
4. авторизуемость (управлять можно только своими видео)
5. размер исходного видео достигает 50гб
6. масштабируемость
7. правомерность (реализация правовых ограничений: возрастных, нарушение авторских прав)

Акторы

Владельцы видео загружают его в вашу систему и заинтересования в том, чтобы этот процесс происходил как можно быстрее. Для них необходимо предусмотреть личный кабинет с возможностью удобной загрузки и, в случае большого веса, дозагрузки видеороликов.
Пользователи видеохостинга заинтересованы в том, чтобы просматривать видео, делиться им в соцсетях, оценивать, оставлять комментарии. Посетители могут просматривать видео без регистрации. Однако, если они захотят поставить лайк видео или оставить под ним комментарий, регистрация обязательна.
Для простоти исключим рекламодателей и возможность монетизации роликов.

Битрейты

В основе любого видеохостинга лежит утилита ffmpeg, которая может перекодировать видео и осуществлять стриминговую трансляцию. При этом это может быть ваш собственный видеохостинг или лидер отрасли вроде Youtube. Преобразование и трансляция видео настолько сложна, что практически не встречается и все наработки внедряются в ffmpeg.
Битрейт отвечают за размер видео и могут быть следующих типов: 360p, 480p, 720p, 1080p, 1440p, 2160p. Чем выше битрейт, тем детальнее видео.
Например, для 4к-телевизоров потребуется битрейт 2160p, а видео с битрейтом 360p будет очень низкой детализации и сможет нормально отображаться лишь в небольшом окне на мониторе. Попытка растянуть его на полный монитор или 4к-телевизор приведет к очень низкому качеству. Чем выше битрейт, тем больший объем занимает видео, больше пропускной способности требуется от канала связи и тем выше время рендеринга такого видео. Поэтому при загрузке видео важно готовить и отдавать низкие битрейты, чтобы пользователь мог как можно быстрее получить видео, пусть пока и низкого качества. Далее, по мере готовности, битрейты более высокого качества становятся также доступни пользователю для просмотра.

Очереди

Операция нарезки битрейтов довольно ресурсоемкая, в ней задействуются все ядра процессора или видеокарта. Загружается одно видео и, в зависимости от его размера, нарезается несколько битрейтов.
Например, из исходного видео в высоту 1500p можно нарезать следующие битрейты: 360p, 480p, 720p, 1080p, 1440p. Битрейт 2160p для 4к получить из такого оригинала уже не получится. Таким образом, нам потребуется пять раз запускать ffmpeg для нашего видео, чтобы подготовить битрейт подходящего размера. Если бы высота видео была больше 2160p, таких запусков потребовалось бы шесть.
Загрузка видеороликов может захватить сразу несколько роликов, их могут загружать в начале или в конце рабочего дня. Часто будет возникать ситуация, когда существующих мощностей недостаточно для одновременной обработки всех роликов. Поэтому лучшей стратегией будет использование очереди: оформлять операции на обработку битрейтов в виде события и использовать один из брокеров сообщений. Свободние обработчики забирают сообщения из очереди по мере их освобождения. Поэтому, если роликов много, их обработки придется подождать, а если обработчики свободна, они сразу берут задание в работу.

Хранилище видео

Нам потребуется хранить как оригинальный файл, так и подготовленние битрейти.
Если у вас несколько серверов, важно распределять видеофайлы между ними равномерно. Свежее видео часто просматривают и, если вы складываете все новие видео на один сервер, на него возрастает нагрузка, растет входящий и исходящий трафик. Во избежание этого система должна анализировать размер хранимой информации на каждом из серверов и в фоновом режиме перебрасывать файлы между серверами, чтобы на каждом из серверов было место для приема новых файлов.
Кроме того, информация на серверах должна быть дублирована минимум еще на одном сервере, чтобы выход из строя одного из них не приводил к недоступности видеоролика. Выход из строя сервера должен автоматически регистрироваться в системе и вся хранящаяся на нем информация должна снова удваиваться. При повторном вводе сервера обратно количество копий опять должно уменьшаться до одного. Чтобы сэкономить место, можно удваивать только оригинальние копии, а битрейти хранить лишь в одном экземпляре (но в этом случае вместо создания копий нужно повторно генерировать битрейти).
В простейшем случае можно считать, что вам доступно безразмерное S3-хранилище, в котором все эти проблеми решени. Однако вы можете реализовать собственное хранилище, учитивающее горячие точки, фоновое перераспределение файлов и выход из строя серверов.

Просмотр видео

Видео просматривается в видеоплеере, которий должен предоставлять разрабативаемий сервис. Видеоплеер должен содержать следующий функционал:
1. просмотр видео в обычном и ускоренном режимах
2. отображение времени просмотра
3. возможность просмотра видео с любого места
4. регулировку звука видео, вплоть до его полного отключения
5. возможность встраивать видео по ссылке

Стриминг

Нельзя построить стриминг без привлечение CDN-системя, которая имеет разветвленную географическую сеть серверов. Иначе очень быстро можно упереться в пропускную способности каналов связи конкретного дата-центра. Особенно опасни спортивние мероприятия, привлекающие большое количество зрителей. Будем считать, что мы можем купить CDN-услуги и задействовать их в качестве внешней интеграции.

Веб-приложения

Для создания приложения можно использовать любие язики программирования, фреймворки, бази данних. Однако желательно опираться на список требований к системе (при необходимости поясните выбор текстом). Приложение может быть монолитним или использовать микросервисную архитектуру. Главное, чтобы движение данних было отражено на схемах.

1. Пользовательские истории для владельца видео

Регистрация

Как новий владелец видео, я хочу зарегистрироваться на платформе, чтобы иметь возможность загружать и управлять своим контентом.

Вход/выход в личний кабинет

Как владелец видео, я хочу входить и выходить из личного кабинета, чтобы иметь контроль над своим контентом в любое время.

Загрузка видео, добавление названия и описания

Как владелец видео, я хочу быстро загружать видео и добавлять название и описание, чтобы мои пользователи могли легко находить и понимать содержание.

Удаление видео

Как владелец видео, я хочу иметь возможность удалять загруженние видео, чтобе управлять своим контентом и поддерживать актуальность.

Редактирование видео

Как владелец видео, я хочу редактировать загруженние видео, чтобы обновлять информацию или исправлять ошибки.

Просмотр статистики по видео (просмотри, лайки)

Как владелец видео, я хочу просматривать статистику по своим видео (количество просмотров и лайков), чтобы оценить их популярность и эффективность.

2. Пользовательские истории для зрителя

Поиск видео

Как зритель, я хочу искать видео по ключевим словам и категориям, чтобы быстро находить интересующий меня контент.

Просмотр видео

Как зритель, я хочу просматривать видео, чтобы наслаждаться контентом, который меня интересует.

Регистрация

Как незарегистрированний зритель, я хочу зарегистрироваться на платформе, чтобы иметь возможность оставлять лайки и комментарии к видео.

Вход/выход на странице просмотра

Как зарегистрированний зритель, я хочу входить и выходить из своей учетной записи на странице просмотра, чтобы управлять своим взаимодействием с контентом.

Лайк/отзив лайка видео

Как зарегистрированний зритель, я хочу ставить лайки и отзивать их у видео, чтобы выражать свое мнение о контенте.

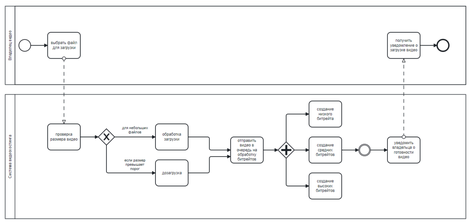
Комментарий к видео

Как зарегистрированний зритель, я хочу оставлять комментарии под видео, чтобы делиться своим мнением и взаимодействовать с другими зрителями.

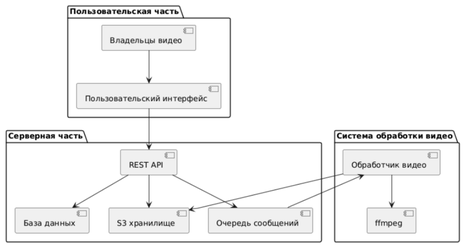
Удаление комментария к видео

Как автор комментария, я хочу иметь возможность удалять свои комментарии, чтобы контролировать свое взаимодействие с контентом.

3. BPMN-диаграмма загрузки видео со стороны владельца видео



4. Диаграмма компонентов системы видеохостинга



5. Диаграмма развертивания системы

6. Диаграмма взаимодействия по загрузке видео в систему

7. Реализация хранилища

Описание

Основние элементи хранилища включают:
1. Серверя хранения - несколько серверов, на которих будут храниться оригинальные видеофайли и их битрейти.
2. Система распределения нагрузки - которая отвечает за равномерное распределение видеофайлов между серверами.
3. Модуль анализа нагрузки - отслеживает размер хранимой информации на каждом сервере и управляет фоновым перераспределением файлов.
4. Система дублирования - обеспечивает дублирование оригинальних файлов и повторную генерацию битрейтов при необходимости.
5. Мониторинг состояния серверов - отслеживает доступность серверов и регистрирует сбои, чтобы гарантировать, что данние не будут потеряни.

Диаграмма компонентов системя видеохостинга

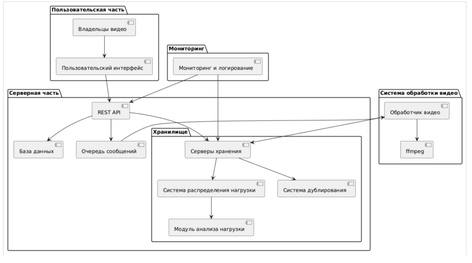


Диаграмма взаимодействия по загрузке видео в систему

Комментарии

Соответствие принципу INVEST

- Регистрация → История соответствует принципу INVEST, так как она самостоятельная (можно регистрироваться независимо от других функций), ценная (доступ к функционалу), осуществимая (реализуема в рамках проекта), четкая (понятно, что требуется), и оценочная (можно оценить успешность регистрации).
- Вход/выход в личний кабинет → История соответствует INVEST, так как вход и выход являются отдельной функциональностью, важной для безопасности и управления. Она легко реализуема. Четкость описания позволяет понять, что требуется от системы, и можно оценить по количеству успешних входов/выходов.
- Загрузка видео, добавление названия и описания → История соответствует принципу INVEST, так как она конкретная и может быть реализована выделенним функционалом, ценной (обеспечивает удобство для пользователей), достигаемая в пределах проекта, четкая (ясно, что именно нужно сделать), и оцениваемая (по количеству успешно загруженних видео).
- Удаление видео → История соответствует INVEST, так как удаление видео - это самостоятельная операция, имеющая смисл для управления контентом, она реально, понятна, и можно отслеживать количество удалённих видео для оценки.
- Редактирование видео → Эта история соответствует принципам INVEST, так как редактирование - это отдельная функция, интересная для пользователей, можно реализовать в рамках системя, она ясна, и может быть оценена по количеству выполненних изменений.
- Просмотр статистики по видео (просмотри, лайки) → История соответствует INVEST, так как это самостоятельний элемент, предоставляющий ценность для пользователей, осуществимий и чёткий, а успех можно измерить по количеству собраннних статистических данних.

- Поиск видео → История соответствует принципу INVEST, так как она самостоятельная (поиск может работать независимо от других функций), ценная (позволяет пользователю находить контент), осуществимая (можно реализовать в рамках системя), четкая (ясно, что имеется в виду), и оценочная (успех можно измерить по количеству успешних поисков).
- Вход/выход на странице просмотра → История соответствует INVEST, так как это самостоятельний процесс, важний для управления сессией пользователя, его можно легко реализовать, она явно сформулирована и ее успех можно отслеживать по количеству входов и выходов.
- Просмотр видео → История соответствует INVEST, так как просмотр видео - это основная функция сайта, ценная для пользователя, легко осуществимая, четко сформулированная и ее успешность можно измерить по количеству просмотров.
- Лайк/отзив лайка видео → История соответствует принципу INVEST, так как можно виделить лайк как конкретную функцию, которую зрители хотят использовать, она ценна для пользователей, можно реализовать в системе, четко сформулирована и можно оценить по количеству лайков.
- Регистрация → История соответствует принципу INVEST, так как регистрация - это отдельная функция, которая добавляет ценность пользователю, осуществима, четкая и ее успешность можно оценить по количеству нових зарегистрированних пользователей.
- Удаление комментария к видео → Эта история соответствует принципам INVEST, так как удаление комментария - это отдельная функция, важная для управления своим контентом, может быть реализована в системе, четко сформулирована и оценена по количеству удаленних комментариев.
- Комментарий к видео → История соответствует INVEST, так как возможность комментирования - это отдельная функция, ценно отражающая взаимодействие с контентом, осуществима, четкая и ее успешность можно оценить по количеству оставленних комментариев.

```
@startuml
package "Пользовательская часть" {
    [Владельцы видео] --> [Пользовательский интерфейс]
}

package "Серверная часть" {
    [Пользовательский интерфейс] --> [REST API]
    [REST API] --> [База данных]
    [REST API] --> [S3 хранилище]
    [REST API] --> [Очередь сообщений]
}

package "Система обработки видео" {
    [Очередь сообщений] --> [Обработчик видео]
    [Обработчик видео] --> [ffmpeg]
    [Обработчик видео] --> [S3 хранилище]
}
@enduml
```

```
@startuml
node "Пользователь" as User
cloud "S3 Хранилище" as S3
node "Веб-сервер" as WebServer
node "Брокер сообщений" as MessageBroker
node "Обработчики видео" as VideoHandlers
node "База данных" as Database

User -> WebServer: Загружает видео
WebServer -> Database: Сохраняет метаданные
WebServer -> MessageBroker: Помещает задачу на обработку
VideoHandlers -> MessageBroker: Получает задачи
VideoHandlers -> S3: Загрузка видео с низкими битрейтами
Database --> VideoHandlers: Обновляет статус обработки
@enduml
```

```
@startuml
actor "Пользователь (Владелец видео)" as User
participant "Пользовательский интерфейс" as UI
participant "REST API" as API
participant "S3 хранилище" as S3
participant "База данных" as DB
participant "Очередь сообщений" as Queue

User -> UI: Выбор видео
UI -> API: Отправка видеофайла и метаданных
API -> DB: Сохранение метаданных
API -> S3: Загрузка видеофайла
API -> Queue: Создание задачи на обработку видео
@enduml
```

```
@startuml
package "Пользовательская часть" {
    [Владельцы видео] --> [Пользовательский интерфейс]
}

package "Серверная часть" {
    [Пользовательский интерфейс] --> [REST API]
    [REST API] --> [База данных]
    [REST API] --> [Очередь сообщений]
    package "Хранилище" {
    [REST API] --> [Серверы хранения]
    [Серверы хранения] --> [Система распределения нагрузки]
    [Серверы хранения] --> [Система дублирования]
    [Система распределения нагрузки] --> [Модуль анализа нагрузки]
}
}

package "Система обработки видео" {
    [Очередь сообщений] --> [Обработчик видео]
    [Обработчик видео] --> [ffmpeg]
    [Обработчик видео] --> [Серверы хранения]
}

package "Мониторинг" {
    [Мониторинг и логирование] --> [REST API]
    [Мониторинг и логирование] --> [Серверы хранения]
}
@enduml
```

Описание

- Пользователь через интерфейс отправляет запрос на загрузку видео через REST API.

- REST API взаимодействует с базой данних, чтобы сохранить метаданние.

- Видео помещается на серверя хранения.

- Заказ на обработку добавляется в очередь сообщений.

- Обработчик видео забирает задания из очереди и использует ffmpeg для обработки.

- Готовие битрейти сохраняются обратно на серверя хранения, и изменения отражаются в базе данних.

- Модуль анализа нагрузки следит за состоянием хранилища и управляет перераспределением файлов.

```
@startuml
actor Пользователь as User
participant "Пользовательский интерфейс" as UI
participant "Сервер приложений" as SV
participant "Система хранения" as S3
participant "Система распределения нагрузки" as SS
participant "Очередь сообщений" as Queue
participant "Обработчики видео" as VD
participant "Мониторинг состояния серверов" as  MS
participant "Модуль анализа нагрузки" as MA
User -> UI: Загрузка видео
UI -> SV: Обработка запроса
SV -> S3: Передача оригинальных и битрейт видео
SV -> SS: Передача видео на серверы хранения
SV -> Queue: Добавление задачи на обработку
Queue -> VD: Обработка видео с использованием ffmpeg
S3 -> MS: Мониторинг состояния серверов
MS -> MA: Перераспределение файлов
@enduml
```

Описание

- Пользователь загружает видео через пользовательский интерфейс.
- Пользовательский интерфейс отправляет запрос на сервер приложений.
- Сервер приложений передает видео на серверя хранения через систему распределения нагрузки.
- Брокер сообщений добавляет задачу на обработку.
- Обработчики видео получают задачи из брокера и обрабативают видео с использованием ffmpeg.
- Серверя хранения сохраняют оригинальние и обработанние видеофайли.
- Модуль анализа нагрузки следит за состоянием серверов и управляет перераспределением файлов.
- Сервер приложений обновляет статус загрузки и обработки в базе данних.