Documento de Arquitectura de Software

Nombre del Software

Autores:

- Samantha Caamal
- Elena Castañeda
- Kirbey García
- Esthefany Mezquita

Introducción

Propósito

Este documento tiene como objetivo proporcionar una descripción general del sistema, así como mantener organizada la arquitectura técnica,
esto a través de una serie de vistas arquitectónicas que representan los distintos aspectos del sistema. Se pretende capturar y
transmitir las decisiones arquitectónicas del sistema para proveer una fuente de referencia a los analistas y a los diseñadores de la aplicación.

Alcance

Este documento tiene como fin, describir el proyecto **Nombre del Proy** y detallar los principales diagramas del mismo, describen las caraterísticas más importantes para la realización del proyecto y está dirigido a los analistas y desarrolladores del proyecto para propósitos informativos y de entendimiento de la arquitectura y componentes del sistema **Nombre del Proy**. Las características mencionadas a continuación pueden ser modificadas, y se pueden agregar más características en el dado caso que sea aprobada una solicitud de cambio.

Documentos de referencia

A continuación se listan los documentos de referencias utilizados para el proyecto:

- ```
  Dailymotion API Documentation
          *https://developer.dailymotion.com/player/
  ```
- ```
  Daylimotion Player API Documentation
          *https://developer.dailymotion.com/player/#player-customisation
  ```
- ```
  Tres
  ```
  ```

Arquitectura

Descripción de la arquitectura utilizada (Capas) (Describir responsabilidad de las capas)

Capa de Presentación
Aquí agrupamos las clases y métodos que conforman el API del servicio web, tales como GET, POST, PUT, DELETE, etc.

- `Controller. Controladores que reciben y procesan las solicitudes del usuario.`

Lógica de Negocio
Se encarga de ejecutar procesamientos y aplicar reglas de negocio. Aquí se contienen las clases que tienen la «sabiduría» o conocimiento
sobre la lógica de lo que hace el servicio web.

- `Capa Service. Componentes encargados de ejecutar procesos complejos de la lógica de negocio.`

Datos

- `Capa de implementación de acceso a datos: ORM Hibernate.`

Adicionalmente tenemos un tipo de objeto que transporta la información entre estas capas de nuestra aplicación, para reducir el acoplamiento. Los Data Transport Object (DTO) o Value Object (VO) que se implementan como objetos planos de java (Plain Old Java Object o POJO), es decir, simples contenedores de datos, sin métodos o comportamiento.

Diagrama de Arquitectura

A continuación se presenta el diagrama de arquitectura con descripción (Arquitectura del proyecto completo)

Diagramas de secuencia:

Enseguida, tenemos 3 diagramas de secuencia que representarán funcionalidades de las partes que consideramos vitales en el sistema:

Diagrama de la Base de Datos

A continuación, la estructura de la base de datos del sistema queda de la siguiente manera:

Entidades

Las entidades del sistema son los siguientes:

- Usuario. Entidad encargada de administrar a los usuarios del sistema.
  El usuario podra tener permiso a las funcionalidades principales del sistema.

- Tipo de Usuario. Entidad que sirve para poder designar permisos y roles de los usuarios del sistema.
  Dependiendo del tipo de usuario será los permisos que tenga el usuario sobre el sistema.

- Cuenta. Entidad que sirve para administrar la cuenta de las proveedor/plataforma que tenga el usuario.
  El usuario puede tener diferentes cuentas en diferentes proveedores.
  La cuenta del usuario debe tener un email y la contraseña ya existente con el proveedor.

- Proveedor. Entidad que sirve para modelar y generar la proveedor/plataforma a la que pertenece la cuenta del usuario.
  El usuario puede tener acceso a diferentes proveedores.

- Comentarios. Entidad que sirve para administrar los comentarios que se realizan a los videos subidos por el usuario.
  Se debe poder visualizar la fecha y el autor del video.

- Interracciones. Entidad que sirve para administrar las interraciones que el usuario tiene con los videos, el usuario puede
  darle like o dislike algún otro video en el sistema.

- ```
  Vídeos. Entidad que sirve para administrar los videos subidos al sistema por el usuario.
  ```

Diagrama entidad relación

A continuación se presenta el diagrama de Entidad-Relación del sistema.

Documentación de la API

En esta sección se cita la documentación de la API

Documentación Individual de cada Endpoint por cada entidad

URL (URI)

Usuario

- GET/usuarios
- GET/usuarios/{idUsuario}
- POST/usuarios
- PUT/usuarios/{idUsuario}
- DELETE/usuarios/{idUsuario}

Proveedores de video

- GET/proveedores
- GET/proveedores/{idProveedor}
- POST/proveedores
- PUT/proveedores/{idProveedor}
- DELETE/proveedores/{idProveedor}

Comentarios

- GET/comentarios
- GET/comentarios/{idComentario}
- POST/comentarios
- PUT/comentarios/{idComentario}
- DELETE/comentarios/{idComentario}

Interacciones

- GET/interacciones
- GET/interacciones/{idInteraccion}
- POST/interacciones
- PUT/interacciones/{idInteraccion}
  ```

- DELETE/interacciones/{idInteraccion}

Videos

- GET/videos
- GET/videos/{idVideo}
- POST/videos
- DELETE/videos/{idVideo}

Descripción

Usuario

Entidad que representa a los usuarios que interactúan con el sistema y hacen uso de este, permite que los usuarios tengan acceso al visualizar, comentar o reaccionar(interactuar) con los videos según su tipo o cuenta. El administrador del sistema, quien se encarga de administrar videos, usuarios y proveedores.

Proveedores de video

Entidad que representa a los proveedores de los videos del sistema. Los proveedores proporcionan contenido obtenido de otras cuentas del usuario externos al sistema.

Comentarios

Entidad que representa a los comentarios que los usuarios pueden realizar en los videos. Se depe administrar la fecha de creación de los comentarios y a que video pertenece el comentario para crear un historial de comentarios.

Interacciones

Entidad que representa los likes o dislikes que los usuarios pueden dejar en los videos. Se debe contabilizar las reacciones(interacciones) que los usuarios realicen a los videos.

Videos

Entidad que representa los videos que los usuarios pueden ver en el sitio. Los videos pueder ser creados por los usuarios u obtenidos por medio de los proveedores.

Campos requeridos

Usuario

idUsuario
nombre
apellido
correo
rol

Proveedores de video

Comentarios

idComentario
idUsuario
idVideo
contenido

Interacciones

idInteraccion
idUsuario
idVideo
Tipo(like o dislike)

Videos

idVideo
video

Validaciones

Usuario

idUsuario --- Que no exista otro usuario conel mismo idUsuario (POST)
correo --- Que cumpla con el formato de correo

Comentarios

contenido --- Que el campo no este vacio

Interacciones

idUsuario | idVideo --- El usuario solo puede reaccionar una vez a un video, por tanto solo puede haber una interacción por cada combinación idVideo e idUsuario.

Videos

video --- Formato valido

Tipo de dato de cada campo

Usuario

idUsuario---[Integer]
nombre------[String]
apellido----[String]
correo------[String]
rol---------[String]

Proveedores de video

Comentarios

idComentario---[Integer]
idUsuario------[Integer]
idVideo--------[Integer]
contenido------[String]

Interacciones

idInteraccion---[Integer]
idUsuario-------[Integer]
idVideo---------[Integer]
Tipo------------[boolean]

Videos

idVideo----[Integer]
video------[File]

Respuesta (Response)

Usuario

- GET/usuarios
  Regresa una lista con todos los usuarios registrados en el sistema.
- GET/usuarios/{idUsuario}
  Regresa al usuario con el idUsuario proporcionado si es que este se encuentra registrado en el sistema.
- POST/usuarios
  Se registra un nuevo usuario.
- PUT/usuarios/{idUsuario}
  Se modifica al usuario con el idUsuario proporcionado si es que este se encuentra registrado en el sistema.
- DELETE/usuarios/{idUsuario}
  Se elimina al usuario con el idUsuario proporcionado si es que este se encuentra registrado en el sistema.

Proveedores de video

- GET/proveedores
  Regresa una lista con todos los proveedores registrados en el sistema.
- GET/proveedores/{idProveedor}
  Regresa al proveedor con el idProveedor proporcionado si es que este se encuentra registrado en el sistema.
- POST/proveedores
  Se registra un nuevo proveedor.

- PUT/proveedores/{idProveedor}
  Se modifica al proveedor con el idProveedor proporcionado si es que este se encuentra registrado en el sistema.
- DELETE/proveedores/{idProveedor}
  Se elimina al proveedor con el idProveedor proporcionado si es que este se encuentra registrado en el sistema.

Comentarios

- GET/comentarios/{idVideo}
  Regresa una lista con todos los comentarios registrados en el sistema para el video con el idVideo proporcionado.
- GET/comentarios/{idComentario}
  Regresa el comentario con el idComentario proporcionado si es que este se encuentra registrado en el sistema.
- POST/comentarios
  Se registra un nuevo comentario.
- PUT/comentarios/{idComentario}
  Se modifica el comentario con el idComentario proporcionado si es que este se encuentra registrado en el sistema.
- DELETE/comentarios/{idComentario}
  Se elimina el comentario con el idComentario proporcionado si es que este se encuentra registrado en el sistema.

Interacciones

- GET/interacciones
  Regresa una lista con todos las interacciones registradas en el sistema para el video con el idVideo proporcionado.
- GET/interacciones/{idInteraccion}
  Regresa la interacción con el idInteraccion proporcionado si es que este se encuentra registrado en el sistema.
- POST/interacciones
  Se registra una nueva interacción.
- PUT/interacciones/{idInteraccion}
  Se modifica la interacción con el idInteraccion proporcionado si es que este se encuentra registrado en el sistema.
- DELETE/interacciones/{idInteraccion} Se elimina la interacción con el idInteraccion proporcionado si es que este se encuentra registrado en el sistema.

Videos

- GET/videos Regresa una lista con todos los videos registrados en el sistema.
- GET/videos/{idVideo}
  Regresa el video con el idVideo proporcionado si es que este se encuentra registrado en el sistema.
- POST/videos
  Se registra un nuevo video.

- DELETE/videos/{idVideo}
  Se elimina el video con el idVideo proporcionado si es que este se encuentra registrado en el sistema.

Ejemplos del Request

Métodos POST para la creación de las entidades del sistema:

Usuario

- ```
  {
    “idUsuario”: 1,
    “nombreUsuario”: “Mariel Castañeda”,
    “username”: “mariel.castañeda”,
    “password”: "dasdjnasduoasndmasi",
    “correo”: “mariel_castañeda@gmail.com”
  }
  ```

Proveedor

- ```
  {
    “idUsuario”: 1,
    “nombreProveedor”: “Dailymotion”,
    “urlProveedor”: “https://www.dailymotion.com/signin?urlback=%2Fmx”
  }
  ```

Video

- ```
  {
    “idVideo”: 1,
    “idCuenta”: 11,
    “idProveedor”: 1,
    “nombreVideo”: "Principales maravillas de Yucatán"
  }
  ```

Cuenta

- ```
  {
    “idCuenta”: 11,
    “idProveedor”: 1,
    “correo”: “mariel_castañeda@gmail.com”,
    “password”: "dasdjnasduoasndmasi"
  }
  ```

Comentario

- ```
  {
    “idComentario”: 10,
    “idVideo”: 1,
    “idCuenta”: 1,
    “comentario”: "Bonito video",
    “fechaComentario”: “10-05-2021”
  }
  ```

Interración

- ```
  {
    “idComentario”: 10,
    “idVideo”: 1,
    “idCuenta”: 1,
    “tipo”: “true”
  }
  ```

Criterios de calidad

En esta sección se describen los criterios de calidad que son correspondientes a la arquitectura. Se indican cada uno de los criterios junto con una descripción clara y precisa. Los criterios de calidad a alcanzar por la arquitectura son los siguientes:

Disponibilidad

Capacidad del sistema de estar operativo y accesible para su uso cuando se requiere.

Usabilidad

Capacidad el sistema que permite al usuario entender adecuadamente sus funcionalidades principales para cubrir las necesidades del cliente.

Tolerancia a fallos

La API deberá ser tolerante a fallos y ser capaz de seguir operando según lo previsto en presencia de fallas de hardware/software.

Integridad

Capacidad de la API para prevenir accesos o modificaciones no autorizados a datos o funciones del sistema.

Seguridad

La información proporcionada al usuario deberá ser íntegra y completa, (es decir la información deberá llegar completa al usuario y sin ninguna modificación). Las información sensible del usuario, como datos de inicio de

sesión (en caso de requerirse) deberá ser resguardada y tratada con un algoritmo de encriptación

Funcionalidad

La API deberá finalizar toda transacción u operación realizada dentro del aplicativo, es decir, no deberá quedar pausada la funcionalidad de la aplicación si la API llegara a fallar. El 85% de las transacciones realizadas deberán ser completadas con éxito.