**Montaje automático de videos**

**(Moovid)**

Arquitectura e Integración de Sistemas Software

Grado de Ingeniería del Software

Curso 2º IS

Daniel Caro Olmedo ([danielcaroolmedo2@gmail.com](mailto:danielcaroolmedo2@gmail.com))

Nicolás De Ory Carmona ([deorynicolas@gmail.com](mailto:deorynicolas@gmail.com))

Antonio González Gómez ([antoniogg696@gmail.com](mailto:antoniogg696@gmail.com))

Ignacio Navarro Blázquez ([inavarroblazquez@gmail.com](mailto:inavarroblazquez@gmail.com))

Tutor: Javier Troya Castilla

Número de grupo:

Enlace de la aplicación: <https://moovid-271019.appspot.com/>

Enlace de proyecto en GitHub: <https://github.com/nicolasdeory/moovid>

Historial de versiones

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Versión | Detalles | Participantes |
| 07/03/2020 | 1.0 | - Incluye introducción, prototipos de las interfaces de usuario y diagramas UML de componentes y despliegue y secuencia de alto nivel. | Daniel Caro  Nicolás De Ory  Antonio González  Ignacio Navarro |
| 03/05/2020 | 2.0 | - Añadidos los diagramas de clase y de secuencia de la arquitectura de la aplicación siguiendo las directrices del patrón MVC. - Añadida la documentación de la API REST en swagger. | Daniel Caro  Nicolás De Ory  Antonio González  Ignacio Navarro |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Índice

[1 Introducción 3](#_Toc41209721)

[1.1 Aplicaciones integradas 4](#_Toc41209722)

[1.2 Evolución del proyecto 5](#_Toc41209723)

[2 Prototipos de interfaz de usuario 6](#_Toc41209724)

[2.1 Vista Inicio 6](#_Toc41209725)

[2.2 Vista Creación Montaje 6](#_Toc41209726)

[2.3 Vista Montaje Finalizado 7](#_Toc41209727)

[3 Arquitectura 8](#_Toc41209728)

[3.1 Diagrama de componentes 8](#_Toc41209729)

[3.2 Diagrama de despliegue 8](#_Toc41209730)

[3.3 Diagrama de secuencia de alto nivel 9](#_Toc41209731)

[3.4 Diagrama de clases 9](#_Toc41209732)

[3.5 Diagramas de secuencia 10](#_Toc41209733)

[4 Pruebas 11](#_Toc41209734)

[5 Manual de usuario 13](#_Toc41209735)

[5.1 API REST 13](#_Toc41209736)

# Introducción

Moovid es una aplicación web que pretende integrar Google Photos junto con Google Drive y Language Understanding Intelligent Service (LUIS – parte de los Cognitive Services de Microsoft).

Su objetivo principal es crear de manera automática videomontajes personalizados a partir de la biblioteca de fotos del usuario. Por medio del lenguaje natural, el usuario tiene la posibilidad de indicarle a la aplicación los parámetros que desea que tenga el montaje, desde la selección de fotos en función de su fecha y lugar de captura, hasta la selección en función del contenido de la propia imagen. Estos parámetros los indica el usuario a través de una interfaz que reconoce el lenguaje natural, gracias a LUIS.

El resultado final se almacena en la nube, dando al usuario la posibilidad de visualizarlo, descargarlo o compartirlo con total flexibilidad, gracias a la integración con Google Drive.

La aplicación tiene como objetivo facilitar en gran medida el trabajo de aquellos usuarios que quieran realizar un montaje con fotos de su biblioteca (abstrayéndoles de cualquier programa de edición), o incluso sorprenderle con recomendaciones personalizadas.

## Aplicaciones integradas

**LUIS**. Recibe los mensajes del usuario expresados en lenguaje natural, y extrae la intencionalidad de estos (por ejemplo – confirmar o cancelar una acción, o petición para realizar un montaje), gracias a un modelo ML entrenado. Además de la intencionalidad del mensaje, LUIS extrae datos clave del mensaje como, cuando el usuario pide realizar un montaje, el rango de fechas que desea el usuario, y la temática del montaje. Asimismo, la API es capaz de discernir entre los mensajes que contienen información relevante y los que no.

**Google Photos**. Proporciona un análisis exhaustivo (Computer Vision) de la biblioteca de fotos del usuario, permitiendo clasificar cada foto en distintas categorías en función de su contenido, ubicación geográfica, y marca de tiempo. Se integra con LUIS para aplicar parámetros en función de lo que desee el usuario, y se utiliza además para ofrecer recomendaciones personalizadas.

**Spotify**. Encargado de, según las especificaciones del cliente, encontrar canciones adecuada para el montaje, esta lista de canciones será dada a la API de Youtube. Es capaz de identificar artistas o estilos de canciones (pausada, bailable, etc...) y devolver las canciones que concuerden con dichos criterios.

**Youtube.** Tras recibir la lista de canciones de Spotify se encargará de buscar las urls de dichas canciones para que puedan ser añadidas al montaje final.

|  |  |
| --- | --- |
| Nombre aplicación | URL documentación API |
| Google Photos | <https://developers.google.com/photos> |
| Youtube | <https://developers.google.com/youtube/v3> |
| LUIS (Language Understanding Intelligent Service) | <https://www.luis.ai/home> |
| Spotify | <https://developer.spotify.com/documentation/web-api/> |

Tabla 1. Aplicación integradas

## Evolución del proyecto

Al principio se pensaba implementar tres APIs, Google Drive, Photos y LUIS, sin embargo tras un estudio detenido de las documentaciones pertinentes decidimos dejar drive ya que Photos ya nos permitía almacenar y descargar las fotos necesarias. Para añadir la complejidad restante al proyecto decidimos implementar música en los montajes en la forma de las APIs de Spotify y Youtube. Adicionalmente, para el montaje del video en cuestión ha sido necesario desarrollar scripts adicionales usando la librería ffmpeg que coge las canciones y las fotos y produce el montaje. Ha supuesto un verdadero reto dada la falta de ejemplos previos.

# Prototipos de interfaz de usuario

## Vista Inicio

Esta vista esta disponible antes de hacer login con Google Photos, se muestra un icono para iniciar sesión y la ventana de conversación con LUIS donde se escribirán las diferentes instrucciones



## Vista Creación Montaje

Se accederá a esta vista una vez el usuario se haya registrado con Google, aparecerán los montajes que se han procesado y las fotos disponibles divididas en categorías. La conversación con LUIS seguirá estando disponible.



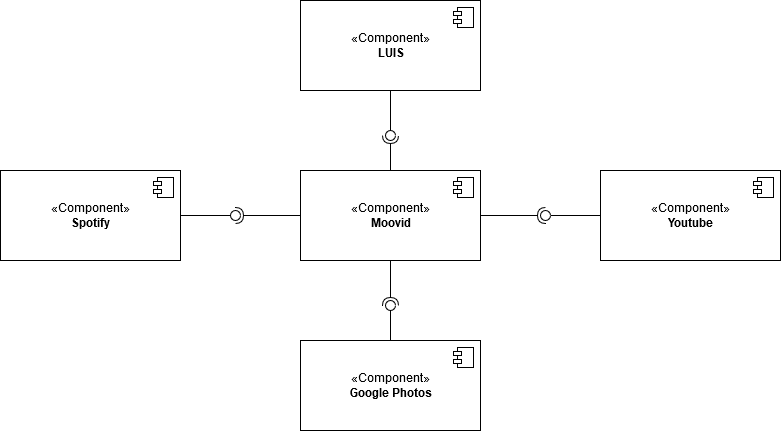
## Vista Montaje Finalizado

Una vez se acabe de procesar un montaje LUIS devolverá un mensaje donde podremos ver una previsualización y algunos enlaces para descargarlo o compartirlo en distintas redes sociales.



# Arquitectura

## Diagrama de componentes

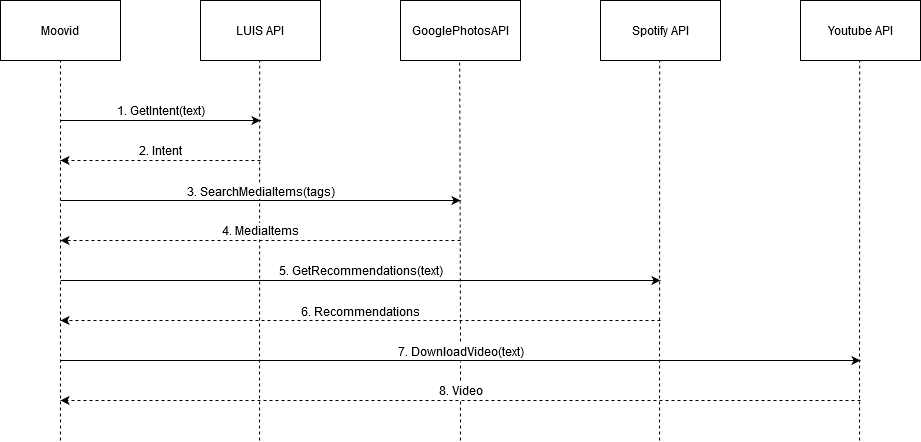


## Diagrama de despliegue

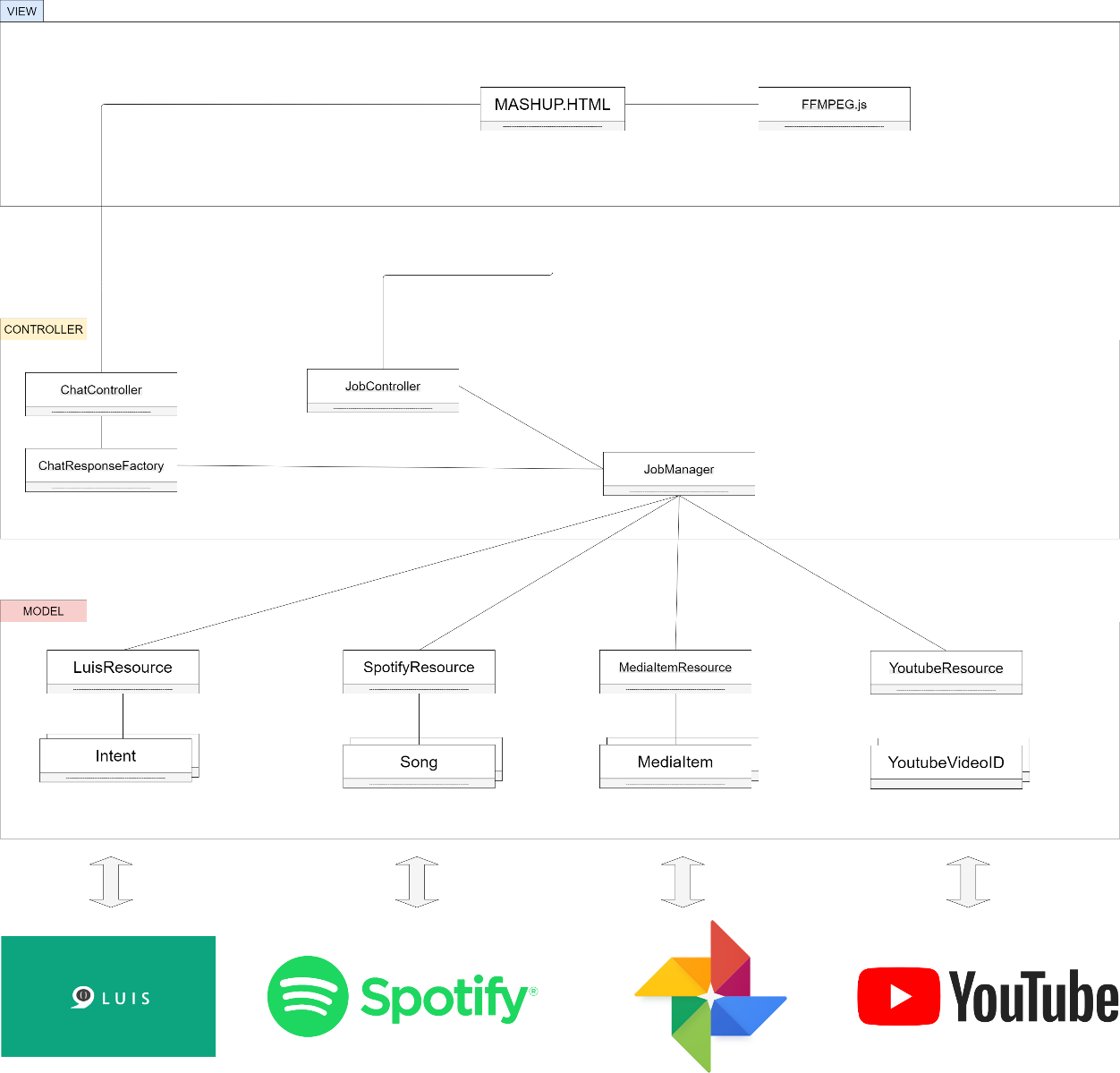
Imagen que contiene captura de pantalla

Descripción generada automáticamente

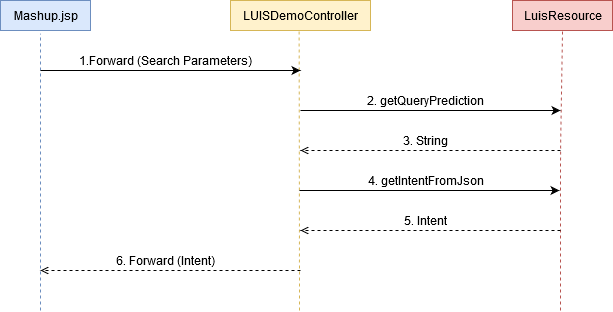
## Diagrama de secuencia de alto nivel

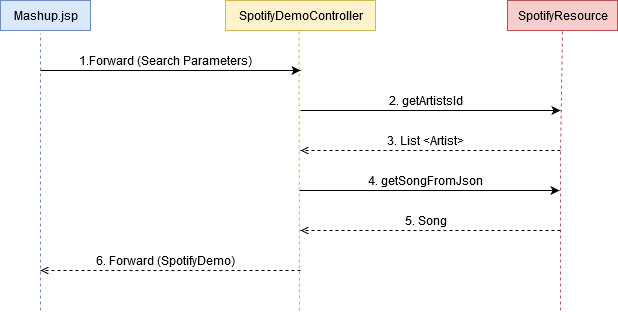


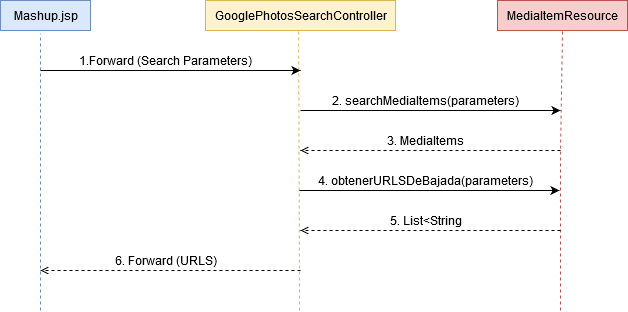
## Diagrama de clases

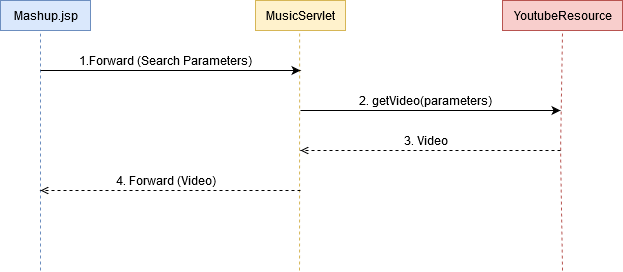


## Diagramas de secuencia









# Implementación

# Pruebas

|  |  |
| --- | --- |
| Resumen |  |
| Número total de pruebas realizadas | 7 |
| Número de pruebas automatizadas | 7 (100%) |

|  |  |
| --- | --- |
| ID | **Generador de String filtro** |
| Descripción | Prueba de conversión en Photos de parámetros a filtro en forma de string. |
| Entrada | Entran Strings que simbolizan los parámetros pertinentes en el filtro. |
| Salida esperada | Se deberán parsear los parámetros individualmente y después devolver un String preparado para ser parseado como Filters. |
| Resultado | **EXITO** |
| Automatizada | Sí |

|  |  |
| --- | --- |
| ID | **Generador de Filtro** |
| Descripción | Prueba de conversión de parámetros a filtro en Photos |
| Entrada | Entran Strings que simbolizan los parámetros pertinentes en el filtro. |
| Salida esperada | Tras parsear los parámetros se obtendrá el filtro en formato string y usando dicho string se formara un objeto Filters. |
| Resultado | **EXITO** |
| Automatizada | Sí |

|  |  |
| --- | --- |
| ID | **Generador de Filtro Vacio** |
| Descripción | Creación de filtro en Photos en caso de que no se pase ningún parámetro |
| Entrada | Los parámetros serán Strings vacíos. |
| Salida esperada | Se generará un objeto tipo Filters con sus valores mínimos |
| Resultado | **EXITO** |
| Automatizada | Sí |

|  |  |
| --- | --- |
| ID | **testGetVideoID** |
| Descripción | Prueba de obtención correcta de la id de un video de Youtube dada una query. |
| Entrada | Parámetro query que identifica el video. |
| Salida esperada | Id del video seleccionado. |
| Resultado | **EXITO** |
| Automatizada | Sí |

|  |  |
| --- | --- |
| ID | **testGetArtistsID** |
| Descripción | Prueba de la obtención de artistas dentro de la API de Spotify. |
| Entrada | Se proporcionan las querys asociadas a los artistas que se testean. |
| Salida esperada | Se devuelve la lista de los artistas (objeto Artist) solicitada. |
| Resultado | **EXITO** |
| Automatizada | Sí |

|  |  |
| --- | --- |
| ID | **testGetRandomMessage** |
| Descripción | Comprobación de obtención de mensaje. |
| Entrada | Se pide un mensaje de despedida a la API LUIS. |
| Salida esperada | Se muestra en pantalla el mensaje devuelto. |
| Resultado | **EXITO** |
| Automatizada | Sí |

|  |  |
| --- | --- |
| ID | **testParseXMLDoc** |
| Descripción | Comprobar el parseo de un documento XML a algo legible por el componente de conversacional. |
| Entrada | Documento en formato XML. |
| Salida esperada | Se muestra en pantalla el parseo del archivo en el formato deseado |
| Resultado | **EXITO** |
| Automatizada | Sí |

# Manual de usuario

## Mashup

## API REST

Documentación interactiva: <https://moovid-271019.appspot.com/docs/index.html>  
  
Fichero YAML: <https://app.swaggerhub.com/apis/moovid/moovid-api/1.0.0#/>