UNIVERSIDAD DE SANTIAGO DE COMPOSTELA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA

Sistema de pregunta-respuesta basado en la aplicación de técnicas de anotación semántica en canales médicos de Youtube.

Autor:
Efrén Rama Maneiro

Directores:
Manuel Lama Penín
Juan Carlos Vidal Aguiar

Grado en Ingeniería Informática

Julio de 2018

Trabajo de Fin de Grao presentado en la Escuela Técnica Superior de Ingeniería de la Universidad de Santiago de Compostela para la obtención del Grado en Ingeniería Informática

D. Manuel Lama Penín, Profesor del Departamento de Electrónica y Computación de la Universidad de Santiago de Compostela, y **D. Juan Carlos Vidal Aguiar**, Profesor del Departamento de Electrónica y Computación de la Universidad de Santiago de Compostela,

INFORMAN:

Que la presente memoria, titulada *Sistema de pregunta-respuesta basado en la aplicación de técnicas de anotación semántica en canales médicos de Youtube*, presentada por **D. Efrén Rama Maneiro** para superar los créditos correspondientes al Trabajo de Fin de Grado de la titulación de Grado en Ingeniería Informática, se realizó bajo nuestra dirección en el Departamento de Electrónica y Computación de la Universidad de Santiago de Compostela.

Y para que así conste a los efectos oportunos, expiden el presente informe en Santiago de Compostela, a Junio de 2018:

O director, O codirector, O alumno,

Manuel Lama Penín Juan Carlos Vidal Aguiar Efrén Rama Maneiro

Índice general

1. Introducción **1**
1.1. Motivación 1
1.2. Objetivos del sistema 4
1.3. Estructura de la memoria 5

2. El algoritmo pregunta-respuesta **7**
2.1. Introducción 7
2.2. Filtrado de vídeos 10
2.3. Comparativa de grafos 11

3. Gestión del proyecto **15**
3.1. Gestión del Alcance 15
3.1.1. Descripción del alcance del producto 15
3.1.2. Criterios de aceptación del producto 15
3.1.3. Entregables del producto 16
3.1.4. Exclusiones del proyecto 16
3.1.5. Restricciones del proyecto 17
3.1.6. Supuestos del proyecto 17
3.2. Gestión de Riesgos 17
3.2.1. Metodología 17
3.2.2. Identificación de riesgos 18
3.2.3. Análisis de riesgos 19
3.2.4. Planificación de riesgos 25
3.2.5. Seguimiento y control 26
3.3. Gestión de Costes 27
3.3.1. Costes directos 27

3.3.2. Costes indirectos 28
3.3.3. Costes totales del proyecto 29
3.4. Gestión de la Configuración 29
3.4.1. Herramientas empleadas 29
3.4.2. Definición del sistema de configuración 29
3.4.3. Estructura del repositorio de documentación 30
3.4.4. Gestión del código fuente 30
3.4.5. Gestión de cambios 31
3.5. Gestión del Tiempo 32
3.5.1. Metodología de desarrollo 32
3.5.2. Estructura de Descomposición del Trabajo 33
3.5.3. Planificación 37

4. Análisis **45**
4.1. Especificación de requisitos 45
4.1.1. Actores 46
4.1.2. Casos de uso 46
4.1.3. Requisitos no funcionales 57
4.1.4. Requisitos de información 60
4.2. Matrices de trazabilidad 62

5. Análisis de tecnologías y herramientas **65**
5.1. Tecnologías de desarrollo 65
5.1.1. Librerías 65
5.1.2. *Frameworks* 66
5.1.3. APIs 68
5.1.4. ADEGA 68
5.1.5. Bases de datos 72
5.1.6. Entornos de desarrollo integrado 74
5.1.7. Lenguajes de programación 74
5.2. Tecnologías de documentación 74
5.2.1. Emacs + LaTeX 74
5.2.2. Git 74
5.2.3. WBSTool 74
5.2.4. Microsoft Project 75

5.2.5. Enterprise Architect 75
5.2.6. Draw.io 75
5.2.7. Lumzy 75

6. Diseño e Implementación **77**
6.1. Arquitectura del sistema 77
6.2. Patrones de arquitectura software 78
6.3. Modelo de datos 81
6.3.1. Diseño del modelo de datos en VoltDB 81
6.3.2. Implementación 83
6.4. Servidor Web *(backend)* 87
6.4.1. Diagrama de paquetes 87
6.4.2. Patrones de diseño software 89
6.4.3. Diagramas de clases 90
6.5. Diagramas de secuencia 104
6.5.1. Anotación de vídeos. 105
6.5.2. Búsqueda de canales y vídeos. 107
6.5.3. Mostrar vídeos anotados 109
6.5.4. Eliminar vídeos anotados 110
6.5.5. Actualizar vídeos anotados 111
6.5.6. Procesamiento de consulta de usuario 112
6.6. Interfaces de usuario (*frontend*) 114
6.6.1. Diseño de la interfaz de usuario 114
6.6.2. Implementación de la interfaz de usuario 116

7. Pruebas y validación **119**
7.1. Pruebas unitarias 119
7.2. Pruebas de integración 124
7.3. Pruebas de validación de requisitos 126
7.3.1. Pruebas de validación de requisitos funcionales. 126
7.3.2. Pruebas de validación de requisitos no funcionales. 131
7.3.3. Matrices de trazabilidad 132
7.4. Validación de la interfaz de usuario. 134
7.4.1. Heurísticos de Nielsen 134
7.4.2. Pruebas de usabilidad 136

8. Conclusiones y trabajo futuro **141**
8.1. Conclusiones 141
8.2. Trabajo futuro 142

A. Manual técnico de despliegue **145**
A.1. Instalación de VoltDB 146
A.2. Instalación de ElasticSearch 147
A.3. Despliegue de la aplicación 147

B. Manual de usuario **149**

Índice de figuras

2.1. Grafo asociado a los subtítulos de un vídeo. 9
2.2. Ejemplo de intersección de dos grafos. 12
2.3. Ejemplo de vecindario y cálculo de la similaridad relacional. . . . 13

3.1. Estructura de carpetas de la documentación del proyecto. 30
3.2. Árbol de directorios de los repositorios de código 31
3.3. Estructura de Descomposición del Trabajo (EDT) (2/2) 34
3.4. Diagrama de gantt resumen del proyecto. 38
3.5. Iniciación del proyecto 39
3.6. Cronograma del incremento 1: búsqueda y almacenamiento de los metadatos de los vídeos de Youtube. 41
3.7. Cronograma del incremento 2: anotador de vídeos seleccionados y almacenamiento de los mismos. 42
3.8. Cronograma del incremento 3: sistema gestor de los vídeos anotados 42
3.9. Cronograma del incremento 4: sistema pregunta-respuesta 43
3.10. Cronograma del cierre del proyecto. 43

4.1. Subsistema de anotación de videos 47
4.2. Subsistema de gestión de vídeos anotados 53
4.3. Subsistema de pregunta-respuesta 56

6.1. Diagrama de la arquitectura del sistema 78
6.2. Aplicación de los patrones de arquitectura software 82
6.3. Modelo Entidad-Relación de la base de datos VoltDB 83
6.4. Diagrama de paquetes del *backend* 88
6.5. Diagrama de clases del paquete Youtube. 91
6.6. Diagrama de clases del paquete DAO 95

6.7. Diagrama de clases del paquete “ADEGA” 97
6.8. Diagrama de clases del paquete “JobQueue” 98
6.9. Diagrama de clases del paquete Management 99
6.10. Diagrama de clases del paquete “QuestionAnswering” 100
6.11. Diagrama de clases para el paquete “Controller” 105
6.12. Diagrama de secuencia: “anotación de vídeos” 106
6.13. Diagrama de secuencia: “búsqueda de canales y vídeos” 108
6.14. Diagrama de secuencia: “Mostrar vídeos anotados” 109
6.15. Diagrama de secuencia: “Eliminación de vídeos anotados” 110
6.16. Diagrama de secuencia: “Actualización de vídeos anotados” 111
6.17. Diagrama de secuencia: procesamiento de consulta de usuario . . 113
6.18. Prototipo de la página principal de la aplicación 115
6.19. Prototipo de la página de mostrar vídeos de un canal 115
6.20. Prototipo de la página de pregunta respuesta. 116
6.21. Implementación de la pantalla principal de la aplicación. 117
6.22. Implementación de la visualización de vídeos de un canal. 117
6.23. Implementación de la interfaz del sistema pregunta-respuesta. . . 118

7.1. Interfaz antigua, antes de las pruebas de usabilidad. 138
7.2. Interfaz nueva, después de las pruebas de usabilidad. 139

B.1. Pantalla principal de la aplicación. 149
B.2. Pantalla de búsqueda de canales. 150
B.3. Pantalla de visualización de vídeos. 150
B.4. Pantalla de éxito. 151
B.5. Pantalla de visualización de trabajos de anotación. 151
B.6. Pantalla de visualización de grafos. 152
B.7. Pantalla de anotación de vídeos en lote. 152
B.8. Pantalla de administración de vídeos anotados. 153
B.9. Pantalla del sistema pregunta respuseta. 153

Índice de cuadros

3.1. Valoración de la probabilidad 18
3.2. Valoración del impacto 18
3.3. Cálculo de la exposición del riesgo a partir del producto de probablidad e impacto. 18
3.4. Registro de riesgos del proyecto 19
3.5. Plantilla de análisis de riesgos 20
3.7. Plantilla de planificación de riesgos 25
3.8. Costes de personal 28

4.9. Matriz: casos de uso - objetivos del sistema 63
4.10. Matriz: casos de uso - entregables del proyecto 64

7.1. Matriz: casos de uso - pruebas de validación. 133
7.2. Plantilla de evaluación de usabilidad. 137

Capítulo 1

Introducción

1.1. Motivación

Las tecnologías de la información están transformando el mundo de la sanidad, ofreciendo nuevas posibilidades a la asistencia médica. Hoy en día son múltiples las aplicaciones que controlan los datos vitales del paciente en tiempo real, su medicación, que miden el azúcar en sangre, etc. La eSalud ("eHealth", en inglés) es ya, en la actualidad, un motor de transformación sanitaria, y en pocos años se integrará de forma natural en nuestro día a día.

Un aspecto importante dentro de la eSalud es que estos sistemas sean capaces de responder a preguntas en tiempo real, de forma que el paciente pueda obtener una respuesta sin necesidad de requerir la intervención de un médico. Para ello es necesario disponer de dos elementos, principalmente:

- Fuentes fiables de información puesto que se debe proporcionar al paciente la información más precisa posible sobre su consulta.
- Fuentes interactivas que faciliten la comprensión de esa información puesto que, en la mayoría de los casos, el paciente no se detendrá a leer un texto que responda a la consulta que ha formulado bien por la extensión de la respuesta o bien por las dificultades que implique su comprensión.

En este Trabajo Fin de Grado (TFG) se propone la implementación de un sistema de pregunta-repuesta (más conocido en inglés por *Question-Answering*)

que permita resolver las dudas de los pacientes a través de vídeos de tal manera que a la consulta en lenguaje natural de un usuario el sistema ofrezca un vídeo que trate su pregunta. Como biblioteca de vídeos se elige la plataforma Youtube por poseer una cantidad de contenido multimedia más que suficiente asociada a temas médicos.

El sistema de pregunta-respuesta se construirá sobre un motor de búsqueda que combina técnicas de recuperación de la información con técnicas de anotación semántica. Dicho motor será capaz de evaluar el grado de adecuación de la consulta del usuario a los vídeos almacenados en el sistema.

El sistema funcionará sobre canales específicos y fiables de la biblioteca de vídeos de Youtube. De cada vídeo se recuperan tanto los metadatos como los subtítulos asociados a los mismos. Sin embargo, únicamente se utilizarán para el proceso de anotación los subtítulos puesto que constituyen la fuente más fiable para conocer el contenido semántico del vídeo. La anotación se realizará a través del anotador semántico ADEGA que utilizará una base de conocimiento (*Knowledge Base*) para enlazar los términos relevantes identificados en los subtítulos con las entidades de la base de conocimiento obteniendo así un grafo. La base seleccionada en el proyecto es MeSH que posee una gran cantidad de información sobre temas específicos de medicina.

Las preguntas al sistema serán realizadas en lenguaje natural por lo que deberán ser procesadas. Para ello se utilizará también el anotador semántico ADEGA por incorporar ya técnicas de procesamiento de lenguaje natural.

Es preciso mencionar que los grafos devueltos por el anotador semántico son *grafos conceptuales*. En dichos grafos coexisten *nodos conceptuales* y *relaciones semánticas*. Los primeros hacen referencia a los distintos conceptos de la base de conocimiento y los segundos relacionan los distintos conceptos mediante relaciones semánticas.

El sistema pregunta-respuesta estará soportado por una plataforma que permita gestionar todo el ciclo de vida del propio sistema, es decir, permitirá la gestión de los vídeos almacenados en el sistema y la anotación de nuevos vídeos

de cualquier canal de la plataforma Youtube (siempre que posea subtítulos) que el administrador desee almacenar. Los vídeos se deben obtener en tiempo real, lo que quiere decir que se utiliza directamente la API de Youtube para obtener esta información.

Para calcular la adecuación de los vídeos anotados en el sistema a la consulta del usuario se implementará un algoritmo de comparación de grafos conceptuales. Dicho algoritmo permitirá obtener un índice de similaridad normalizado (entre 0 y 1) que indica cuanto se adecúa el contenido del vídeo a la pregunta en lenguaje natural del usuario en función de los grafos obtenidos tanto del vídeo como de la consulta en lenguaje natural.

El caso de la comparación de grafos constituye una especialización del problema de isomorfismo de grafos en el que se evalúa si todos los nodos de dos grafos distintos están conectados de la misma forma. La particularidad del problema reside en que no sólo es suficiente con que los nodos conceptuales estén conectados entre si y sean los mismos en ambos grafos si no que también deben estar conectados por la misma relación semántica. Por ejemplo: entre dos nodos denominados España y Mariano Rajoy existen múltiples relaciones semánticas (ex-presidente, ciudadano...). Por lo tanto, se implementará un algoritmo que tenga en cuenta estas particularidades. Por otra parte, lo que interesa saber no es si dos grafos están conectados exactamente igual si no en qué medida están conectados de la misma forma, es decir, cuán similares son.

Además, el algoritmo pregunta-respuesta tiene que ser suficientemente rápido para devolver una respuesta al usuario e, idealmente, escalable de forma independiente del número de vídeos procesados en el sistema. Por lo tanto, no basta con realizar la comparación de grafos con todos los posibles vídeos almacenados en el sistema si no que es preciso realizar un filtrado inicial de los vídeos que hay anotados en el sistema, para, así, agilizar el proceso de respuesta del sistema.

1.2. Objetivos del sistema

El objetivo principal del proyecto es el desarrollo de un sistema software proporcione los mecanismos que den soporte al ciclo de vida del sistema del sistema "pregunta-respuesta", es decir, a la recuperación de las fuentes de información, en nuestro caso vídeos; a su anotación; indexación en el sistema; y recuperación ante una consulta del paciente en lenguaje natural. De modo más preciso, se plantean los siguientes objetivos:

- **OS-1.**: Desarrollo de los componentes que permitan consultar los distintos canales de Youtube, recuperar metadatos y subtítulos de los vídeos, así como anotar e indexar vídeos.

- **OS-2.**: Desarrollo de una interfaz gráfica para la gestión de la anotación de los distintos vídeos. En dicha interfaz será posible seleccionar los distintos vídeos que el administrador del sistema desea anotar e indexar, previsualizar los vídeos que puedan resultar de interés, previsualizar los metadatos que están asociados a cada vídeos, etc.

- **OS-3.**: Desarrollo de los componentes de la arquitectura orientada a servicios que permitan gestionar los vídeos anotados en el sistema, es decir, que permita gestionar aquellos vídeos que puede devolver el sistema pregunta-respuesta como respuesta.

- **OS-4.**: Desarrollo de una interfaz gráfica que permita actualizar la información asociada a los vídeos anotados en el sistema, eliminar aquellos que hayan desaparecido de la plataforma Youtube y consultar aquellos vídeos que hayan sido anotados en el sistema.

- **OS-5.**: Desarrollo de los componentes de la arquitectura orientada a servicios que permitan realizar y procesar consultas en lenguaje natural, así como realizar búsquedas teniendo en cuenta tanto el procesamiento de dicha búsqueda como la base de datos de vídeos anotados semánticamente.

- **OS-6.**: Desarrollo de una interfaz gráfica que permita al usuario del sistema realizar consultas en lenguaje natural y obtener un vídeo relevante que responda a dicha consulta.

1.3. Estructura de la memoria

La memoria está dividida en capítulos que tratan aspectos diferentes del desarrollo del presente proyecto:

- En el **capítulo 1** se expone la introducción al proyecto, describiendo las motivaciones para la realización del mismo, se definen los objetivos del mismo y se exponen conceptos clave para la comprensión de la memoria.
- En el **capítulo 2** se describe el algoritmo pregunta-respuesta que da soporte a todo el trabajo de fin de grado.
- En el **capítulo 3** se describen todos los aspectos relacionados con la gestión del proyecto: gestión de costes, del calendario, de riesgos, de la configuración y del alcance.
- En el **capítulo 4** se corresponde con la fase de análisis del proyecto. Se especifican los casos de uso, requisitos no funcionales y de información.
- En el **capítulo 5** se describen las tecnologías utilizadas en el proyecto así como las herramientas empleadas para desarrollar el proyecto.
- En el **capítulo 6** se describe el diseño y la implementación del software del proyecto.
- En el **capítulo 7** se muestra el diseño del conjunto de pruebas al cual el sistema será sometido y el resultado de las mismas.
- En el **capítulo 8** se describen las conclusiones de la realización del proyecto y se reflexiona sobre posibles mejoras del proyecto.

Capítulo 2

El algoritmo pregunta-respuesta

2.1. Introducción

Como se ha indicado en la introducción, el principal objetivo es la creación de un sistema que permita responder mediante un vídeo a consultas en lenguaje natural de un paciente. Para ello, lo más importante es saber en qué medida se relaciona la consulta del usuario con el contenido del vídeo analizando, en primer lugar, la propia consulta en lenguaje natural del usuario. Este análisis se realiza mediante la anotación semántica de la consulta del usuario, es decir, mediante la generación del **grafo conceptual** asociado a la consulta en lenguaje natural. Este grafo interrelaciona tres elementos principalmente:

- Los términos más relevantes extraídos directamente del análisis del texto a analizar (en este caso la consulta del usuario) representados en un grafo mediante nodos.

- Los conceptos expandidos a partir de los términos más relevantes directamente extraídos. Esta expansión se realiza utilizando una **ontología** concreta, dependiendo así de ella. Estos conceptos no aparecen directamente en el texto, pero están directamente relacionados con ellos.

- Las relaciones entre los dos tipos de conceptos anteriores. Dichas relaciones dependen también de la ontología e indican relaciones semánticas entre conceptos. Por ejemplo, entre un concepto *Napoleón* y un concepto *Francia* existe una relación de *emperador* (entre otras).

Por otra parte, se realiza el mismo análisis sobre los subtítulos de cada uno de los vídeos que manejará el sistema pregunta-respuesta. Se utilizan los subtítulos puesto que es la forma más fiable de conocer el contenido semántico de un vídeo ya que otros metadatos como el título, la descripción o las etiquetas son creados por terceras personas que pueden no entender de forma completa el vídeo o introducir información que no tiene nada que ver con el vídeo.

Como resultado de estos análisis se obtiene un grafo similar al de la figura 2.1 (los tipos de relaciones entre conceptos aparecen omitidas en este grafo para facilitar la visualización del mismo). Dicho grafo corresponde a un vídeo de temática de enfermedades respiratorias. Como se puede observar, los conceptos aparecen interrelacionados entre sí. La expansión se realiza a partir del nodo denominado "pneumothorax", obtenido directamente de los subtítulos hasta alcanzar otras enfermedades respiratorias como "neumonía" o "hemoptisis". Así, una vez se tiene tanto el grafo del vídeo como el grafo de la consulta del usuario, se procede a compararlos mediante un algoritmo de comparación de grafos. La principal ventaja de dicha comparación reside en que es muy precisa puesto que tiene en cuenta todos los nodos y relaciones semánticas tanto de la consulta del usuario como del contenido del vídeo.

Como el proceso de anotación semántica es un proceso lento (para algunos vídeos puede tardar hasta 40 segundos), resulta imperativo almacenar el resultado de todas las anotaciones en una base de datos. Por otra parte, el rendimiento del algoritmo de comparación de grafos es, en ocasiones, lento para los propósitos de la aplicación debido, principalmente, a que el grafo puede tener un número de nodos muy elevado (del orden de centenas de nodos y miles de relaciones).

Variar el primer elemento es posible mediante la variación de la profundidad de los grafos devueltos por el anotador semántico. Sin embargo, no es deseable puesto que los grafos obtenidos serán más pequeños y, por ende, la comparación de la consulta del usuario con los grafos de los vídeos será menos precisa. La solución pasa por hacer que el algoritmo de comparación de grafos tenga como entrada siempre un número constante de grafos. Así, evitamos que el sistema se ralentice demasiado según escala el número de grafos almacenados en la base

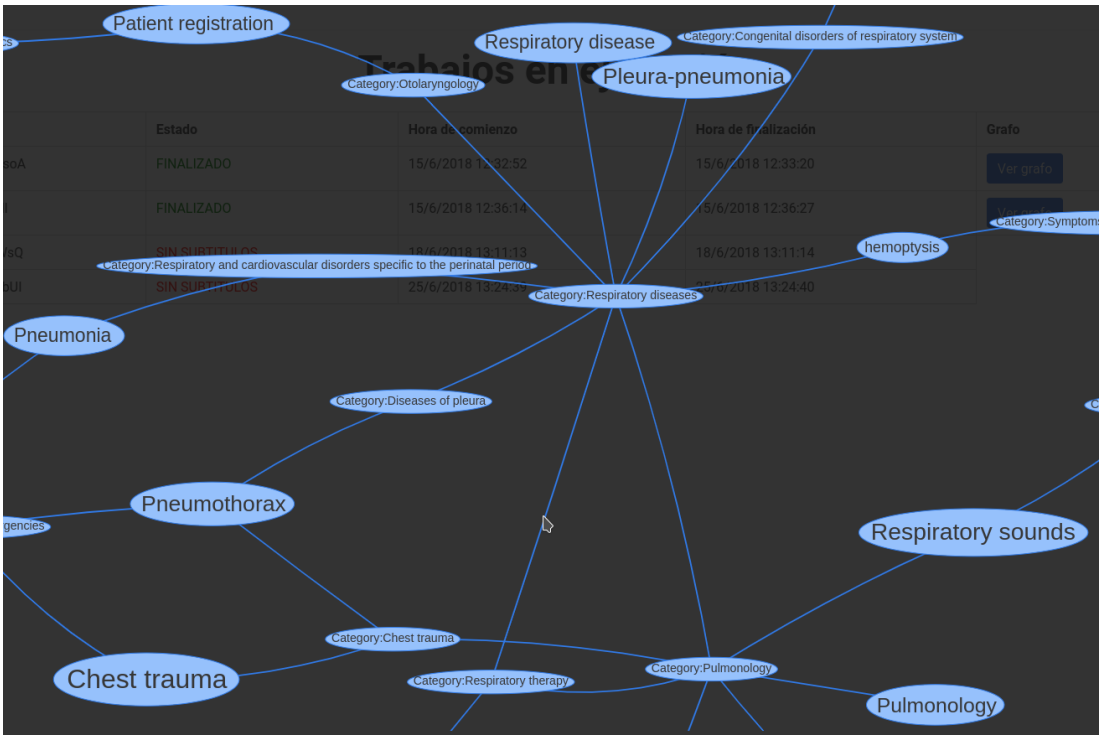

Figura 2.1: Grafo asociado a los subtítulos de un vídeo.

de datos. Por lo tanto, es imperativo realizar un filtrado inicial de los vídeos, descartando el máximo posible de ellos.

En resumen, el proceso de pregunta-respuesta consta de las siguientes fases:

1. Inicialmente, se almacena la anotación semántica de un conjunto de vídeos en una base de datos. Estos vídeos serán los que el sistema devuelva como respuesta.

2. Posteriormente, se anota semánticamente la consulta del usuario, obteniendo un grafo.

3. A continuación, se realiza un filtrado de todos los vídeos almacenados, quedándonos con un número constante de ellos (que serán los más relevantes).

4. Para aquellos vídeos no descartados del filtrado, aplicamos el algoritmo de comparación entre el grafo de la consulta del usuario y los grafos de los vídeos. Devolvemos los vídeos más relevantes.

2.2. Filtrado de vídeos

Como se ha indicado anteriormente, la comparativa de grafos es un proceso relativamente lento, por lo que es preciso hacer un filtrado inicial de los vídeos almacenados en el sistema, en función de su relevancia. Para ello, se utiliza una técnica denominada "búsqueda por palabras" (del inglés, *keyword search*) mediante un motor de búsqueda "*ElasticSearch*" que almacena los nodos asociados a cada uno de los vídeos de la base de datos.

El tipo de consulta se denomina "Match Query" [5] y consiste en una consulta de texto completo, es decir, los términos que se le suministran pasan por un analizador [6] que realiza algunas operaciones básicas de procesamiento de lenguaje natural: tokenización, eliminación de "palabras vacías" (del inglés, *"stopwords"*) y sustitución de letras mayúsculas por minúsculas. Esto permite normalizar la consulta puesto que puede que los términos suministrados no coincidan con los términos del grafo (en caso de que se decida cambiar el algoritmo de búsqueda y obtener los términos sin utilizar ADEGA).

La consulta anterior se aplica sobre el campo "label" de los datos exportados, es decir, sobre cada uno de los conceptos de los grafos de todos los vídeos de la base de datos. Puesto que la consulta se realiza sobre un motor de búsqueda optimizado para tal fin, se realiza más rápido que aplicando el algoritmo de comparativa de grafos directamente.

Para cada término consultado, se obtiene tanto el peso del nodo encontrado (este peso lo devuelve ADEGA como resultado de su anotación) como el índice de similaridad devuelto por ElasticSearch a la hora de realizar la consulta. Para calcular la relevancia del vídeo se realiza el producto del peso del nodo recuperado por la relevancia obtenida mediante la consulta a ElasticSearch y se acumula para cada vídeo por cada término encontrado. Así, si el peso del nodo es nulo, la relevancia para ese término es nula, puesto que el nodo no tiene importancia en el grafo del vídeo. De la misma forma, si la relevancia obtenida de la consulta a ElasticSearch es nula, quiere decir que el término no se corresponde con el nodo.

La lista obtenida anteriormente se ordena según la relevancia calculada. De dicha lista se selecciona cierto número de vídeos que serán los que se comparen utilizando el algoritmo de comparación de grafos. Mediante pruebas empíricas se ha determinado que un número 100 vídeos permite, tanto agilizar en gran medida el procesado mediante la comparación de grafos, como no despreciar vídeos que sí puedan ser relevantes.

2.3. Comparativa de grafos

Para realizar la comparativa de grafos se utiliza un algoritmo [12] que proporciona una medida de la similaridad precisa basada en una medida denominada **coeficiente de Soresen-Dice**. Dicha medida, aplicada al problema de comparativa de grafos, tiene en cuenta específicamente la naturaleza conceptual del grafo, es decir, los tipos de relaciones entre los conceptos que conforman el grafo.

El algoritmo comienza computando la intersección entre el par de grafos a comparar. Dicha intersección está compuesta por dos elementos fundamentalmente: aquellos nodos que estén en ambos grafos y aquellas relaciones que sean del mismo tipo en ambos grafos y relacionen los mismos nodos. En base a estos dos elementos se crea otro grafo, denominado **grafo intersección**. A continuación, se procede a calcular la similaridad entre el grafo intersección y los dos grafos originales. Este componente se subdivide en dos componentes más básicos:

- **Similaridad conceptual**: indica el número de nodos que ambos grafos tienen en común.
- **Similaridad relacional**: indica el número de relaciones que ambos grafos tienen en común, es decir, el número de relaciones del mismo tipo que relacionan nodos iguales.

El cálculo de ambas similaridades está basada en el coeficiente de similaridad de Soresen-Dice. Se escoge este coeficiente por ser muy sencillo de calcular y, por ende, rápido de calcular. Así, la similaridad conceptual se calcula con la fórmula 2.1.

$$s_c = \frac{2n(G_c)}{n(G_1) + n(G_2)} \tag{2.1}$$

Donde:

- s_c denota la similaridad conceptual.
- $n(G)$ es el número de nodos del grafo G.
- G_1 y G_2 son los dos grafos originales a comparar.
- G_c es el **grafo intersección** de G_1 y G_2, es decir, el grafo que tiene los nodos y relaciones comunes a dos grafos originales.

Un ejemplo del anterior proceso se puede ver en la figura 2.2, cuyos grafos iniciales se denominan G_1 y G_2, las relaciones y nodos comunes a ambos grafos se marcan con negrita y son A, B y C y el grafo intersección, G_c, resulta de los nodos y relaciones comunes. De forma análoga, la similaridad relacional se calcula con la fórmula 2.2.

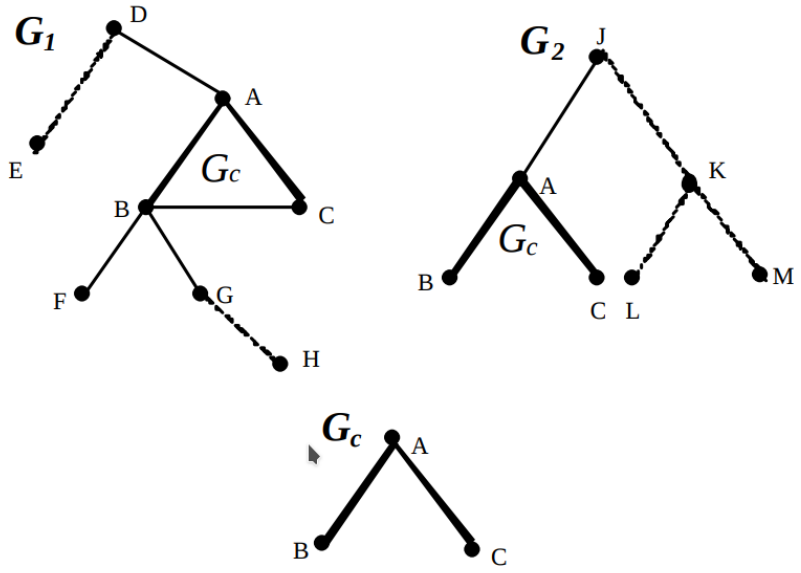

Figura 2.2: Ejemplo de intersección de dos grafos.

$$s_r = \frac{2m(G_c)}{m_{G_c}(G_1) + m_{G_c}(G_2)} \tag{2.2}$$

Donde:

- s_r denota la similaridad relacional.
- $m(G_c)$ el número de relaciones del grafo intersección.
- $m_{G_c}(G_i)$ es el número de relaciones en el **vecindario** del grafo G_i.

Se define como **vecidario** al conjunto de nodos y relaciones del grafo que tienen por lo menos un extremo perteneciente al grafo intersección. Un ejemplo de la anterior fórmula aparece reflejado en la figura 2.3 cuyos grafos son los mismos que los mostrados en la figura 2.2. El vecindario inmediato aparece representado por un sombreado gris e identifica aquellas relaciones y nodos que están en contacto o forman parte del grafo intersección. Las marcas sobre cada una de las relaciones corresponden al número de relaciones que se cuentan de cada grafo que, en el caso del grafo intersección, se cuentan doblemente (puesto que el numerador de la fórmula está multiplicado por dos).

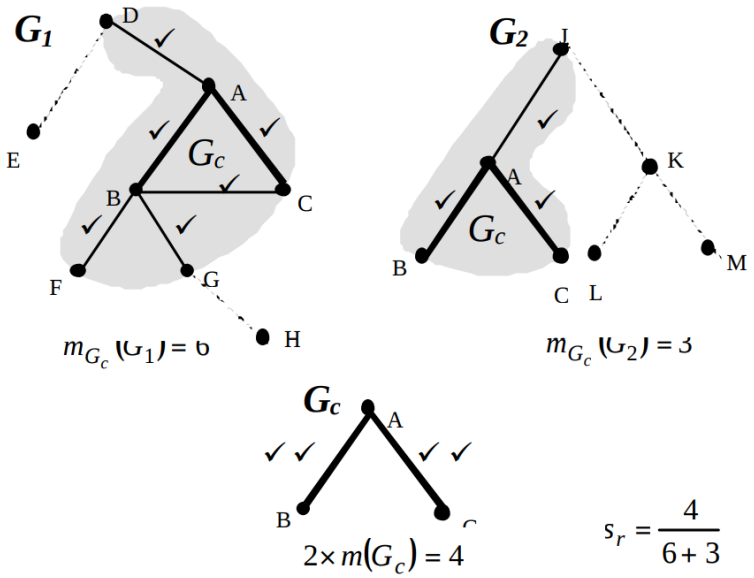

Figura 2.3: Ejemplo de vecindario y cálculo de la similaridad relacional.

Una vez calculada la similaridad conceptual y la similaridad relacional, es preciso combinarlas en una puntuación acumulativa. Para ello, una primera aproximación sería, directamente, multiplicar ambos términos ($s = s_c \cdot s_r$), siendo la puntuación acumulativa proporcional a las dos componentes. Sin embargo, la

similaridad relacional tiene una importancia secundaria, puesto que su existencia depende de la presencia de nodos comunes entre los dos grafos. Así, aún cuando no existan relaciones comunes entre los dos grafos, puede existir similaridad de conceptos, es decir, su valor tiene que ser mayor que 0. Por lo tanto, es preciso modificar la fórmula a la mostrada 2.3:

$$s = s_c \cdot (a + b \cdot s_r) \tag{2.3}$$

Donde:

- s es la puntuación acumulada.
- s_c es la similaridad conceptual.
- s_r es la similaridad relacional.
- a y b son factores de corrección.

De este modo, el coeficiente a expresa el porcentaje de información contenida solo en los nodos y se calcula mediante la expresión 2.4

$$a = \frac{2n(G_c)}{2n(G_c) + m_{G_c}(G_1) + m_{G_c}(G_2)} \tag{2.4}$$

Si la similaridad relacional es cero ($s_r = 0$), entonces la similaridad acumulada es un porcentaje de la similaridad conceptual, es decir, $s = a \cdot s_c$. Por otra parte, cuando los grafos son exactamente iguales, es decir, la similaridad relacional es 1 ($s_r = 1$), se tiene que $a + b \cdot s_r = 1$. Por lo tanto, el coeficiente b es igual a $1 - a$, es decir, $b = 1 - a$.

Capítulo 3

Gestión del proyecto

3.1. Gestión del Alcance

A continuación se incluye una descripción de los procesos necesarios para poder completar el proyecto en tiempo y forma de forma satisfactoria.

3.1.1. Descripción del alcance del producto

La finalidad del proyecto es construir un producto que permita anotar semánticamente aquellos vídeos de la plataforma Youtube que sean relevantes para el usuario. Para ello, se deberá permitir la búsqueda de vídeos, la previsualización de los mismos y la selección para su anotación. Además, deberá gestionar aquellos vídeos ya anotados semánticamente, es decir, deberá permitir la consulta, modificación y borrado de aquellos vídeos que estén almacenados en la base de datos.

Por otra parte, el producto deberá permitir la respuesta mediante un vídeo a consultas en lenguaje natural. Dichas respuestas deberán tener en cuenta la adecuación de la consulta al contenido semántico de los vídeos. Los vídeos propuestos deberán aparecer por orden de mayor a menor relevancia.

3.1.2. Criterios de aceptación del producto

El producto se considerará que está aceptado cuando realice las siguientes funcionalidades correctamente:

- Permite consultar los vídeos y canales de la plataforma Youtube.
- Permite la selección y posterior anotación de vídeos de uno o varios canales de la plataforma Youtube.
- Permite la consulta, actualización y borrado de aquellos vídeos que hayan sido anotados semánticamente.
- Permite responder a consultas en lenguaje natural del usuario mediante un vídeo. Dicha respuesta deberá ser adecuada a la pregunta del usuario y consistirá en uno o más vídeos relacionados con la misma.

3.1.3. Entregables del producto

Los entregables del proyecto se reflejan en la tabla 3.1.3.

Identificador	Entregable
EN-001	Documento de análisis de requisitos software.
EN-002	Diseño preliminar del sistema.
EN-003	Incremento 1: búsqueda y almacenamiento de los metadatos de los vídeos de Youtube.
EN-004	Incremento 2: anotador de vídeos seleccionados y almacenamiento de los mismos.
EN-005	Incremento 3: sistema gestor de los vídeos almacenados.
EN-006	Incremento 4: módulo pregunta-respuesta.
EN-007	Manuales de usuario y técnicos.
EN-008	Memoria del proyecto.

3.1.4. Exclusiones del proyecto

Serán excluidos del proyecto todos aquellos requisitos que, bien por su complejidad o bien por su imposibilidad de realizarlos en el tiempo necesario, pongan en riesgo la finalización del proyecto en los plazos establecidos. No se configurará el sistema para utilizar bases de datos de forma distribuida, es decir, únicamente existirá una base de datos que coexistirá junto con la aplicación en la misma máquina. No se gestionarán límites de acceso para el módulo selector y anotador de vídeos, es decir, no será necesaria la autenticación del usuario para acceder al sistema.

3.1.5. Restricciones del proyecto

El sistema sólo podrá ser ejecutado en plataformas GNU/Linux (tanto x86 como x64). Esta restricción está asociada a la imposibilidad de desplegar una de las bases de datos en otro sistema que no sea GNU/Linux.

3.1.6. Supuestos del proyecto

Para la elaboración del presente proyecto se supondrá que el anotador semántico ADEGA funciona correctamente, es decir, se supone que, a partir de cualquier texto, el anotador devuelve un grafo con las relaciones y términos relevantes.

3.2. Gestión de Riesgos

A continuación se procederá a detallar el plan de riesgos del presente proyecto.

3.2.1. Metodología

El proceso de gestión de riesgos del presente proyecto se ha dividido en las siguientes fases:

- **Identificación de riesgos:** se ha llevado a cabo un análisis que permite identificar los riesgos que pueden afectar al proyecto. Para ello se han realizado tormentas de ideas (*brainstorming*) y revisión de una lista exhaustiva de posibles riesgos.
- **Análisis de riesgos:** en esta fase se analizan los riesgos según su probabilidad e impacto. Para ello, se utilizarán las valoraciones mostradas en las tablas 3.1 3.2. A partir de estos valores se realizará el cálculo de la **exposición** de los riesgos, siendo sus posibles valores aquellos que aparecen en la tabla 3.3.
- **Planificación de riesgos**: una vez se han identificado y analizado los riesgos, se realizará un análisis para evaluar cuál de las siguientes estrategias será necesario aplicar en el caso de ocurrencia del riesgo:
  - *Prevención*: corresponde con estrategias de actuación enfocadas a minimizar la probabilidad de ocurrencia del riesgo.

Ocurrencia del Riesgo	Probabilidad
$\geqslant 80\,\%$ (casi segura)	Alto
Entre $30\,\%$ y $80\,\%$ (muy probable)	Medio
$\leqslant 30\,\%$ (poco probable)	Bajo

Cuadro 3.1: Valoración de la probabilidad

Recurso en Plazo / Esfuerzo / Coste	Impacto
$\geqslant 20\,\%$	Alto
Entre $10\,\%$ y $20\,\%$	Medio
$\leqslant 10\,\%$	Bajo

Cuadro 3.2: Valoración del impacto

- *Minimización*: son estrategias focalizadas en reducir el impacto causado por un riesgo una vez éste se haya producido.
- *Transferencia*: consiste en hacer que el riesgo sea gestionado por un tercero el cual asumirá su control (teniendo un coste asociado).
- *Contingencia*: definen planes de actuación en caso de que un riesgo aparezca.

- **Seguimiento y control**: periódicamente se revisará la lista de riesgos para evaluar si alguno de los valores de probabilidad o impacto se ha visto modificado.

		Probabilidad		
		Alta	**Media**	**Baja**
Impacto	**Alto**	Alto	Alto	Medio
	Medio	Alto	Medio	Bajo
	Bajo	Medio	Bajo	Bajo

Cuadro 3.3: Cálculo de la exposición del riesgo a partir del producto de probablidad e impacto.

3.2.2. Identificación de riesgos

La tabla 3.4 muestra el registro de riesgos identificados mediante las técnicas enunciadas en el apartado anterior. Cada uno de ellos lleva asociado un identificador unívoco que permite referenciarlo a lo largo del documento. Dicho identifi-

cador es de la forma “R-XXX”, donde “XXX” se corresponden con tres números enteros.

Cuadro 3.4: Registro de riesgos del proyecto

Identificador	Nombre
R-001	Planificación optimista, “mejor caso” (en lugar de realista, “caso esperado”).
R-002	Un retraso en una tarea produce retrasos en cascada en las tareas dependientes.
R-003	El ciclo de revisión de los tutores es más lento de lo esperado.
R-004	Las herramientas de desarrollo no funcionan como se esperaba.
R-005	La curva de aprendizaje para las nuevas herramientas de desarrollo es más larga de lo esperado.
R-006	Se añaden requisitos extra.
R-007	No se sigue la gestión de la configuración.
R-008	Reducción de las horas a dedicar al proyecto.
R-009	Las partes del proyecto que no se han especificado claramente consumen más tiempo de los esperado.
R-010	Problemas de integración de las distintas tecnologías.
R-011	El desarrollo de funciones software erróneas requiere su rediseño e reimplementación.
R-012	Los servicios del anotador semántico ADEGA no están disponibles cuando se necesitan
R-013	Alguna de las API utilizadas en el proyecto cambia drásticamente
R-014	Baja del desarrollador del proyecto
R-015	Baja de los tutores

3.2.3. Análisis de riesgos

En este apartado se muestra el análisis de los riesgos previamente identificados en el apartado anterior. Para realizar dicho análisis se ha utilizado la plantilla correspondiente a la tabla 3.5.

Cuadro 3.5: Plantilla de análisis de riesgos

Identificador	R-XXX
Nombre	
Descripción	
Probabilidad	Alta, media o baja.
Impacto	Alto, medio o bajo.
Exposición	Alta, media o baja.
Indicador	Indica si el riesgo se ha manifestado.

Identificador	**R-001**
Nombre	Planificación optimista, "mejor caso" (en lugar de realista, "caso esperado")
Descripción	En el momento de realizar la planificación del proyecto, esta se realiza de forma demasiado optimista asumiendo que no existirá ningún problema de ejecución del plan o que no existirán vueltas a atrás en la planificación.
Probabilidad	Media
Impacto	Alta
Exposición	Alta
Indicador	Existen 5 retrasos consecutivos en todas las tareas una vez iniciado el proyecto

Identificador	**R-002**
Nombre	Un retraso en una tarea produce retrasos en cascada en las tareas dependientes.
Descripción	Debido a la naturaleza de las dependencias entre tareas, es posible que se diera el caso de que el inicio de una tarea se ve retrasado debido a las dependencias de dicha tarea.
Probabilidad	Media
Impacto	Alta
Exposición	Alta
Indicador	El Número de dependencias en cascada es mayor que 4.

Identificador	**R-003**

Nombre	El ciclo de revisión de los tutores es más lento de lo esperado.
Descripción	Debido a circunstancias ajenas al proyecto, los tutores no pueden revisar el estado del mismo justo a la finalización del incremento, produciendo retrasos en la planificación del proyecto.
Probabilidad	Media
Impacto	Medio
Exposición	Media
Indicador.	El ciclo de revisión tarda un 50 % más de lo esperado.

Identificador	**R-004**
Nombre	Las herramientas de desarrollo no funcionan como se esperaba.
Descripción	Puesto que el proyecto a realizar es un proyecto software que utiliza herramientas como *IDE* (entornos de desarrollo integrado) y *frameworks*, es posible que algunas de esas herramientas tengan errores que sea preciso sortear para permitir la finalización del proyecto. El sorteamiento de dichos errores provocaría retrasos en la planificación del proyecto.
Probabilidad	Baja
Impacto	Alto
Exposición	Medio
Indicador	La pérdida de tiempo debida a herramientas defectuosas supera 4 horas.

Identificador	**R-005**
Nombre	La curva de aprendizaje para las nuevas herramientas de desarrollo es más larga de lo esperado.
Descripción	Debido a la inexperiencia en las tecnologías a utilizar en el momento del desarrollo del proyecto, es posible que el tiempo estimado en la planificación para su aprendizaje sea insuficiente con respecto a lo planificado.
Probabilidad	Media
Impacto	Medio
Exposición	Alta

Indicador	La pérdida de tiempo debida a herramientas complejas supera 20 horas.

Identificador	**R-006**
Nombre	Se añaden requisitos extra
Descripción	En el inicio del proyecto se establecen los requisitos que se deben cumplir. Sin embargo, durante el transcurso del proyecto, dichos requisitos se ven afectados de tal manera que se añaden requisitos relevantes que deben ser tenidos en cuenta.
Probabilidad	Baja
Impacto	Alto
Exposición	Media
Indicador	El número de requisitos extras que se añaden es mayor que 5.

Identificador	**R-007**
Nombre	No se sigue la gestión de la configuración
Descripción	No seguir la gestión de la configuración expuesta en la presente memoria podría acarrear resultados desastrosos para el proyecto (como la pérdida de todo el código fuente).
Probabilidad	Baja
Impacto	Alta
Exposición	Alta
Indicador	El tiempo entre "commits" en el repositorio es mayor que una semana.

Identificador	**R-008**
Nombre	Reducción de las horas a dedicar al proyecto.
Descripción	Por motivos ajenos al proyecto, resulta imposible dedicar al proyecto las horas estipuladas en el presente documento durante cierto período de tiempo.
Probabilidad	Media
Impacto	Alta
Exposición	Alta

Indicador	El número de horas que se dedican al proyecto se reduce en 2 o más.

Identificador	**R-009**
Nombre	Las partes del proyecto que no se han especificado claramente consumen más tiempo de los esperado.
Descripción	Debido a la especificación ambigua de alguna de las funcionalidades de un módulo del producto, es necesario realizar tareas de rediseño o reimplementación.
Probabilidad	Media
Impacto	Medio
Exposición	Media
Indicador	El tiempo perdido por reuniones de esclarecimiento de requisitos supera las 3 horas.

Identificador	**R-010**
Nombre	Problemas de integración de las distintas tecnologías.
Descripción	La integración de las distintas tecnologías utilizadas en el proyecto consume más tiempo de lo estipulado en la planificación del proyecto.
Probabilidad	Media
Impacto	Medio
Exposición	Media
Indicador	El tiempo perdido por problemas de integración supera las 10 horas.

Identificador	**R-011**
Nombre	El desarrollo de funciones software erróneas requiere su rediseño e reimplementación.

Descripción	Debido a un diseño o codificación apresurados del producto, es preciso llevar a cabo un trabajo adicional para que estas cumplan correctamente con los requisitos del software. Si este problema no es detectado a tiempo, probablemente será necesario realizar trabajo en cascada que afecte a otros módulos del sistema.
Probabilidad	Baja
Impacto	Medio
Exposición	Baja
Indicador	El número de funcionalidades erróneas supera la 1 unidad.

Identificador	**R-012**
Nombre	Los servicios del anotador semántico ADEGA no están disponibles cuando se necesitan.
Descripción	El anotador semántico deja de funcionar debido a causas inespecificadas lo que evita que se pueda desarrollar utilizando el anotador, que es una base del proyecto.
Probabilidad	Alta
Impacto	Alto
Exposición	Alta
Indicador	El anotador ADEGA deja de funcionar dos días no consecutivos.

Identificador	**R-013**
Nombre	Alguna de las API utilizadas en el proyecto cambia drásticamente.
Descripción	Alguna de las API utilizadas en el proyecto sufre cambios de forma drástica, lo que implica tener que reimplementar parte del código de la aplicación.
Probabilidad	Baja
Impacto	Alto
Exposición	Media
Indicador	El cambio de API provoca errores de compilación o ejecución con la nueva versión.

Identificador	**R-014**
Nombre	Baja del desarrollador del proyecto.
Descripción	Debido a circunstancias físicas o mentales ajenas al proyecto, el desarrollador no podrá estar disponible durante un período de tiempo no despreciable para el proyecto.
Probabilidad	Baja
Impacto	Alto
Exposición	Media
Indicador	El desarrollador no trabaja en el proyecto durante dos semanas consecutivas.

Identificador	**R-015**
Nombre	Baja de los tutores
Descripción	Debido a circunstancias físicas o mentales ajenas al proyecto, alguno de los tutores no puede estar disponible durante un período no despreciable de tiempo.
Probabilidad	Baja
Impacto	Medio
Exposición	Baja
Indicador	El tutor comunica directamente al desarrollador su falta de disponibilidad.

3.2.4. Planificación de riesgos

Para aquellos riesgos con exposición alta mencionados anteriormente, se definirán una serie de medidas que permitirán evitarlos en la medida de lo posible, o, en caso de que ocurran, reducir sus efectos todo lo posible. Para realizar la planificación de riesgos, se utilizará la plantilla definida en la tabla 3.7.

Cuadro 3.7: Plantilla de planificación de riesgos

Identificador	Nombre del riesgo
Tipo de acción	Acción a aplicar

R-001	Planificación optimista, "mejor caso" (en lugar de realista, "caso esperado")
Prevención	Planificar teniendo en cuenta el peor caso posible siempre. Así, se mitiga también la inexperiencia del desarrollador a la hora de planificar proyectos.

R-002	Un retraso en una tarea produce retrasos en cascada en las tareas dependientes.
Minimización	Utilizar un ciclo de vida en incrementos y evitar dependencias en la medida de lo posible

R-007	No se sigue la gestión de la configuración.
Prevención	Realizar revisiones periódicas para comprobar que se esté siguiendo la gestión de la configuración.

R-008	Reducción de las horas a dedicar al proyecto.
Minimización	Sobredimensionar ligeramente la planificación de las tareas del proyecto para que la reducción de horas afecte lo mínimo posible.

R-012	Los servicios del anotador semántico ADEGA no están disponibles cuando se necesitan.
Contingencia	Desplegar una versión local de ADEGA que permita no depender de la versión pública accesible por todo el mundo.

3.2.5. Seguimiento y control

Debe reanalizarse la lista de riesgos identificada al principio del proyecto para comprobar si se modifica la probabilidad e impacto de alguno de los riesgos identificados. En concreto, se hará una revisión con frecuencia semanal para ver si existen cambios en la probabilidad e impacto o si alguno de los indicadores mostrados está a punto de manifestarse. Los riesgos manifestados en el proyecto se indican en el apartado "Planificación'.

3.3. Gestión de Costes

En este apartado se realizará una estimación de los costes asociados a la realización del proyecto. Se dividen los costes en dos tipos: *costes directos* y *costes indirectos*.

3.3.1. Costes directos

Los costes de los equipos informáticos son calculados teniendo en cuenta la amortización y en base al precio total del elemento según la siguiente fórmula:

$$\frac{\text{Precio del producto adquirido}}{12 \cdot \text{Años de duración del equipo}} \cdot \text{Meses de duración del proyecto} \qquad (3.1)$$

Nótese que el denominador de la fracción representa la conversión de años a meses. También se supone que el proyecto comienza el 1 de Marzo de 2018 y termina el 1 de Julio de 2018. Se distinguen los siguientes tipos de costes directos:

- **Equipamiento informático:**
  - Portátil Dell XPS 13 con 16GB RAM, procesador i7-8550U y 512GB de disco SSD valorado en 1480 euros obtenido justo antes del inicio del proyecto. Utilizando la fórmula 3.1 se obtiene un coste de $\frac{1480}{12 \cdot 4} \cdot 6 =$ 185 euros [1].
  - Ratón, teclado mecánico y adaptadores varios necesarios para el portátil, valorados ambos en 50 euros. El coste para el proyecto es de $\frac{50}{12 \cdot 4} \cdot 7 = 7,30$ euros.
  - Monitor de 23 pulgadas cedido por el *Centro Singular de Investigación en Tecnologías de la Información*. No es de reciente adquisición así que el coste se considera despreciable.
- **Software y tecnologías:** se han utilizado versiones gratuitas siempre que ha sido posible. Además, se han utilizado versiones de prueba de "Microsoft Project" y de "Enterprise Architect". Por lo tanto, el coste de este apartado es nulo.

[1] El coste de los equipos incluye IVA.

- **Sistema operativo:** únicamente se ha utilizado Ubuntu 16.04, totalmente gratuito.
- **Recursos humanos:** consultando diversas fuentes [2] se ha estimado el sueldo bruto en 18.000 euros anuales. Para el cálculo del coste real para la empresa se utiliza la calculadora de contratos de la Universidad de Santiago de Compostela [16]. En dicha calculadora suministramos los siguientes parámetros:
  - 18.000 euros distribuidos en 14 pagas. Además, hay que tener en cuenta que se trata de un contrato a tiempo parcial por lo que el salario se divide a la mitad. En resumen: $\frac{18000}{14*2} = 642,85$ euros/mes.
  - *Período del contrato:* del 1/3/2018 al 1/7/2018.
  - *Número de pagas:* 14.
  - *Tipo de contrato:* obra y servicio.
  - *Jornada laboral:* tiempo parcial.
  - *Horas semanales:* 30.
  - *Categoría cotización:* Ingeniero. Investigador en formación.

  Así, se obtienen los siguientes costes:

Bruto	2.592,14€
Pagas extras.	433,86€
Liquidación	99,71€
Contrato	**3.125,71€**
Seguridad Social	1.168€
Total	**4.294€**

Cuadro 3.8: Costes de personal

3.3.2. Costes indirectos

Teniendo en cuenta que, según la Universidad de Santiago de Compostela, es necesario aplicar un 20 % en "concepto de costes indirectos" [10], y que, el coste acumulado hasta el momento de costes directos asciende a un total de 4.486,3€, los costes indirectos del proyecto ascienden a un total de $4.486,3 \cdot 0.20 = 897,26$ €.

3.3.3. Costes totales del proyecto

El coste total del proyecto es la suma de los costes directos más los costes indirectos: 4.294 + 897,26 = 5.191,26 €

3.4. Gestión de la Configuración

El proceso de gestión de la configuración tiene por objetivo controlar y gestionar los cambios sobre los elementos de configuración que conforman el proyecto durante todo el ciclo de vida del mismo. Así, se asegura, por un lado, que siempre se esté trabajando sobre una versión estable y coherente del proyecto y, por otro lado, que las entregas al cliente se realicen siempre sobre la última versión estable del producto.

3.4.1. Herramientas empleadas

En la gestión de la configuración se emplean las siguientes herramientas:

- **Gitlab**: Servicio web de control de versiones basado en Git.
- **Bitbucket**: Servicio web, análogo a Gitlab, basado en Git.
- **Git**: Sistema de control de versiones no centralizado. Su propósito es llevar el registro de cambios sobre cada uno de los ficheros sometidos bajo su supervisión.
- **Maven**: Herramienta para la gestión, compilación y construcción de proyectos Java.

3.4.2. Definición del sistema de configuración

Para el manejo de los repositorios se utilizarán dos opciones:

- *Utilización de la interfaz web:* en el caso que se desee realizar el clonado del repositorio o proponer nuevos cambios. Los motivos de realizar estas tareas por esta vía es la facilidad de realizar dichas tareas mediante la interfaz web (en oposición de la línea de comandos).

- *Linea de comandos:* en el caso de que se deseen realizar las tareas habituales de subida de archivos, actualización de ficheros locales, fusionado y creación de ramas... El motivo de la utilización de la línea de comandos reside, fundamentalmente, en que resulta más rápido, potente, flexible y fácil de automatizar (mediante "alias" o "scripts").

3.4.3. Estructura del repositorio de documentación

Los documentos relativos al proyecto se almacenarán en carpetas distintas siguiendo la siguiente estructura de directorios:

Figura 3.1: Estructura de carpetas de la documentación del proyecto.

En cada carpeta se ubican los ficheros relativos a cada ámbito relevante del desarrollo del proyecto. Nótese que las solicitudes de cambio se almacenan en línea en la plataforma Bitbucket.

3.4.4. Gestión del código fuente

Como ya se ha comentado anteriormente, el código fuente se almacenará en un GitLab. Todos los ficheros correspondientes al código fuente serán sometidos a esta gestión de la configuración. Cada uno de los cambios sobre los ficheros dentro de esta plataforma queda registrado automáticamente. Asimismo, también permite mantener un sistema sofisticado de control de versiones teniendo distintas ramas ("*branches*") sobre un mismo proyecto o volviendo atrás a un cambio anterior en el tiempo. Ambos módulos están integrados en el mismo proyecto Maven por lo que la compilación de los mismos se realiza en un solo paso. La estructura de directorios del proyecto software es la mostrada en el siguiente diagrama:

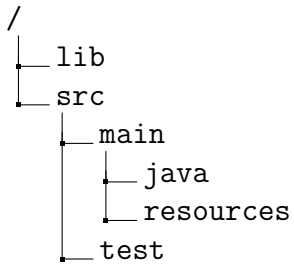

Figura 3.2: Árbol de directorios de los repositorios de código

Donde:

- **lib**: Contiene todas las librerías externas necesarias para compilar el proyecto.
- **src/main/java**: Contiene el código fuente Java del proyecto.
- **src/main/resources**: Contiene el *Front-End* de la aplicación (HTML, Javascripts, CSS)...
- **src/test**: Contienen los test unitarios Java de la aplicación.

Adicionalmente, el fichero `pom.xml` también se ubica en la raíz del repositorio.

3.4.5. Gestión de cambios

Los cambios se almacenarán en el repositorio Bitbucket correspondiente a la documentación del proyecto. Dichos cambios se almacenarán como "problemas" ("*issues*") y deberán seguir la siguiente plantilla (codificada en Markdown para mayor legibilidad):

```
# ISSUE
- **Fuente**: quién origina el cambio
- **Descripción**: descripción del cambio
- **Documentos afectados**: documentos del proyecto
    a los cuales afecta el cambio
- **Estado de aprobación**: aprobado o rechazado
- **Observaciones**: si se desea incluir algo más a mayores.
```

La anterior plantilla se muestra automáticamente en el momento de crear la nueva propuesta de cambio. Aquellos cambios que ya hayan sido implementados deberán ser "cerrados" (*closed*) y los que no aparecerán como "abiertos" (*open*).

3.5. Gestión del Tiempo

3.5.1. Metodología de desarrollo

El ciclo de vida del proyecto elegido para la realización de este proyecto es el **ciclo de vida por incrementos**. En él, se divide el proceso de construcción del software en varios incrementos siendo cada uno de ellos una parte del software final. La motivación de la elección de este ciclo de vida se basa en las siguientes fundamentaciones:

- Una vez finalice un incremento el usuario puede probar el sistema desarrollado hasta el momento, lo que permite una detección temprana de errores de interfaz.
- Los requisitos están bien definidos al principio del proyecto lo que evita los principales problemas del modelo incremental: dificultad de ver si los requisitos son válidos y dificultad en detectar errores en los requisitos de forma temprana.
- Evita construir el sistema de forma monolítica, pues los incrementos corresponden a distintas partes del sistema.

Así el presente proyecto se ha dividido en **4 incrementos** que son los siguientes:

- **Incremento 1**: búsqueda y almacenamiento de los metadatos de los vídeos de Youtube.
- **Incremento 2**: anotador de vídeos seleccionados y almacenamiento de los mismos.
- **Incremento 3**: sistema gestor de los vídeos almacenados (eliminación, actualización y consulta).
- **Incremento 4**: módulo pregunta-respuesta.

Teniendo esto en cuenta, el proyecto se desarrollará en 3 etapas claramente diferenciadas:

- **Etapa inicial**: incluye las tareas relativas a la gestión del proyecto e investigación sobre las tecnologías, siendo estas el análisis de requisitos y la creación del diseño preliminar, así como el establecimiento de un plan de gestión de la configuración e identificación de posibles riesgos.
- **Etapa de desarrollo**: incluye el desarrollo de los anteriores incrementos. En cada incremento se realizan las etapas de diseño, implementación y validación.
- **Etapa de cierre**: incluye las actividades relativas al cierre del proyecto.

3.5.2. Estructura de Descomposición del Trabajo

La Estructura de Descomposición del Trabajo (en adelante, **EDT**), es una representación jerárquica de los entregables y el trabajo del proyecto en componentes más pequeños y más fáciles de manejar. La principal ventaja de crear la EDT es proporcionar una visión estructurada de lo que debe entregar en el proyecto[17]. Así, la estructura de descomposición del trabajo para este proyecto es la reflejada en la figura 3.3.

Los paquetes de trabajo son los siguientes:

- **Iniciación del proyecto**: corresponde a todas las actividades relacionadas con el arranque del proyecto.
  - **Gestión del proyecto**. Estimación temporal: 75 horas horas.
    - Gestión de riesgos: análisis y planificación de los posibles riesgos que pueden surgir durante el transcurso del proyecto.
    - Gestión de la configuración: elaboración y instauración de un plan de control de cambios y control de versiones.
  - **Formación y estudio de tecnologías**. Estimación temporal: 75 horas.
    - ADEGA: estudio del API del anotador semántico ADEGA.

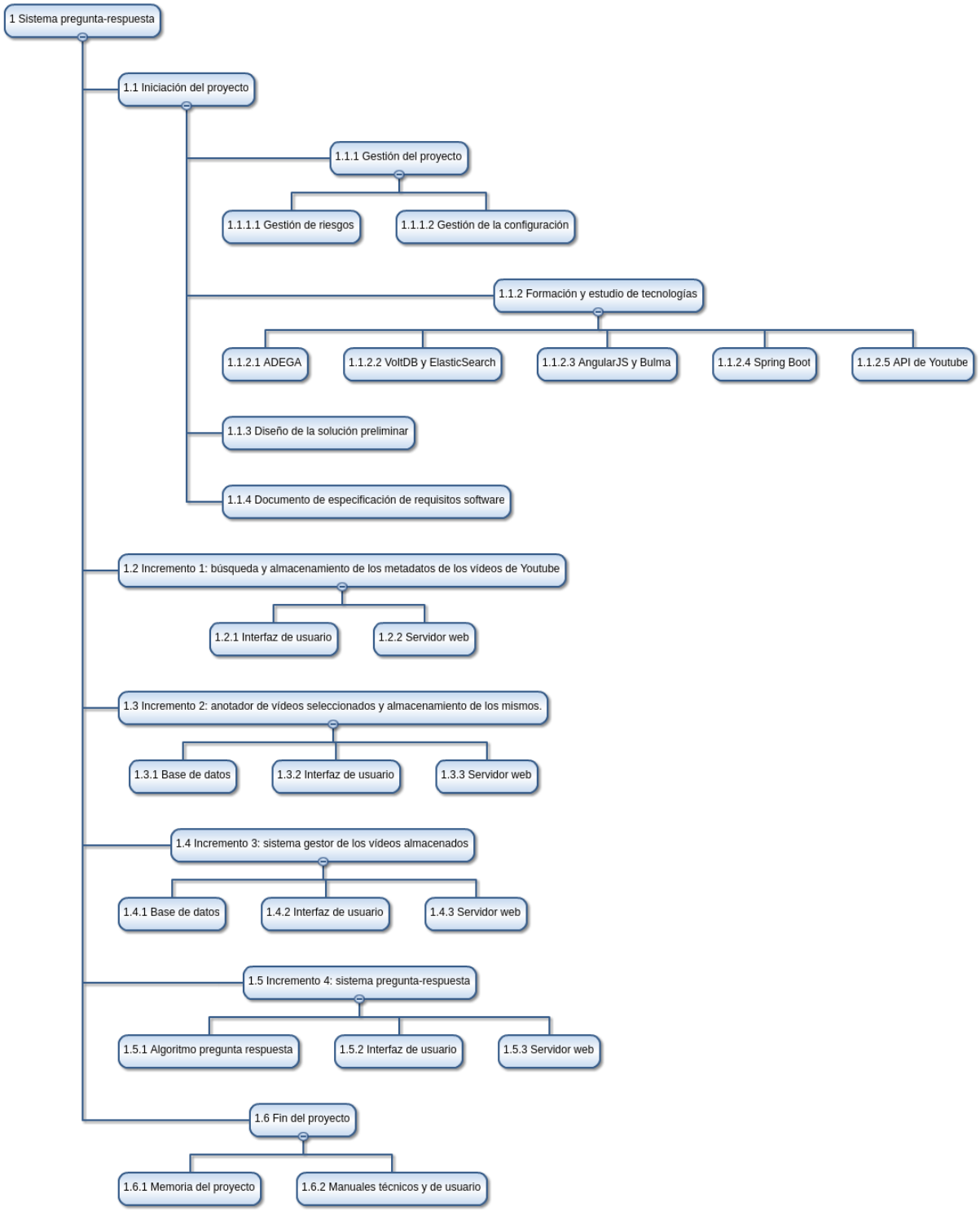

Figura 3.3: Estructura de Descomposición del Trabajo (EDT) (2/2)

    - VoltDB: estudio de la base de datos VoltDB para el almacenamiento de los datos devueltos por ADEGA.
    - ElasticSearch: estudio de la base de datos ElasticSearch para el almacenamiento de algunos de los datos devueltos por ADEGA. Estudio de mecanismos de sincronización entre VoltDB y ElasticSearch.
    - AngularJS: estudio del *framework* AngularJS.
    - Bulma: estudio del *framework* css Bulma para la creación de las interfaces gráficas.
    - Spring Boot: estudio del *framework* Spring Boot para la creación de la capa de servicios.
    - API de Youtube: estudio del API de Youtube para la recuperación de información de dicha plataforma.

  - **Diseño de la solución preliminar**. Diseño de la solución arquitectural a alto nivel del problema. Estimación temporal: 5 horas.
  - **Documento de especificación de requisitos software**. Documento en el que se recogen los requisitos de la aplicación. Estimación tempral: 10 hras.

- **Incremento 1: búsqueda y almacenamiento de los metadatos de los vídeos de Youtube**: implementación del sistema de recuperación de vídeos y canales de Youtube así como de los datos asociados a los mismos.

  - Interfaz de usuario: creación de la interfaz de usuario que permite mostrar los canales de Youtube en función de una consulta y los vídeos asociados a dicho canal.
  - Servidor web: creación de un API REST para la obtención de los vídeos y canales. Esta API será consultada por la interfaz de usuario.

- **Incremento 2: anotador de vídeos seleccionados y almacenamiento de los mismos**: implementación del sistema de recuperación de subtítulos, anotación de los mismos y posterior almacenamiento de los datos obtenidos de ADEGA.

- Base de datos: creación de la base de datos VoltDB y ElasticSearch y sincronización de las mismas. Implementación de las operaciones requeridas sobre las bases de datos.
- Interfaz de usuario: creación de la interfaz de usuario que permite seleccionar los vídeos y añadirlos para su anotación así como mostrar los trabajos de anotación en proceso.
- Servidor web: creación del API REST que permite añadir vídeos para su anotación. Implementación de las llamadas a ADEGA desde el servidor así como del sistema de colas.

- **Incremento 3: sistema gestor de los vídeos anotados**:
  - Base de datos: creación de las consultas que permiten realizar las operaciones de borrado, actualización y consulta de los vídeos anotados.
  - Interfaz de usuario: creación de la interfaz de usuario que permite mostrar los vídeos para su consulta, actualización y borrado.
  - Servidor web: creación del API REST que permite realizar las operaciones sobre la base de datos.

- **Incremento 4: sistema pregunta-respuesta**:
  - Algoritmo pregunta-respuesta: diseño e implementación del algoritmo pregunta-respuesta.
  - Interfaz de usuario: creación de la interfaz que da soporte al algoritmo.
  - Servidor web: creación del API REST que permite enviar consultas en lenguaje natural al algoritmo y recibir los vídeos recomendados.

- **Fin del proyecto**: actividades realizadas al cierre del proyecto.
  - Memoria del proyecto: redacción de la presente memoria.
  - Manuales técnicos y de usuario: redacción de los manuales de operación y soporte del producto.

3.5.3. Planificación

A continuación se muestra la planficación temporal del proyecto. Dicha planificación se ha visto afectada por una serie de circunstancias, que son las siguientes:

- Materialización del riesgo *R-008* ("reducción de las horas a dedicar al proyecto"). Así, el número de horas disponibles diarias se redujo de 5 horas a 4.

- Materialización del riesgo *R-012* ("Los servicios del anotador semántico ADEGa no están disponibles cuando se necesitan"). En este caso se ha aplicado la medida de contingencia descrita en el apartado "Planificación de riesgos": desplegar una versión local de ADEGA para realizar el desarrollo. La medida de sobredimensión de tareas ha ayudado a mitigar el impacto del riesgo *R-008*.

- El tiempo necesario para implementar el procesamiento de lenguaje natural se redujo drásticamente al utilizar directamente ADEGA para tal fin. Ese tiempo ahorrado se utiliza para compensar la reducción de horas a dedicar al proyecto.

Estos dos factores anteriores permiten que la manifestación del riesgo *R-008* se vea aplacada por el ahorro de tiempo asociado al procesamiento de lenguaje natural mediante ADEGA. Además, la medida de contingencia aplicada evita posibles pérdidas de tiempo en esperar a que el anotador semántico esté disponibles. Por lo tanto, se tiene la planificación mostrada en la figura 3.4. En ella se muestran las distintas etapas en las que se desarrolla el proyecto.

Teniendo estas consideraciones en cuenta, el proyecto comienza el 1 de Marzo de 2018 y finaliza el día 26 de Junio de 2018 habiendo dedicado cuatro horas diarias durante 19 semanas y realizando un último esfuerzo de ocho horas diarias durante la última semana siendo así un total de 20 semanas. El número de horas total dedicado es de **420 horas**.

A continuación se explican cada una de las fases de las que consta la planificación, con su respectivo cronograma desplegado.

TFG	85 días	jue 01/03/18	mié 27/06/18
Iniciación del proyecto	22 días	jue 01/03/18	vie 30/03/18
Desarrollo	51 días	lun 02/04/18	lun 11/06/18
Incremento 1: búsqueda y almacenamiento de los metadatos de los vídeos de Youtube	9 días	lun 02/04/18	jue 12/04/18
Incremento 2: anotador de vídeos selecciondos y almacenamiento de los mismos	14 días	vie 13/04/18	mié 02/05/18
Incremento 3: sistema gestor de los vídeos almaceneados (eliminación, actualización...)	10 días	jue 03/05/18	mié 16/05/18
Incremento 4: sistema pregunta-respuesta	18 días	jue 17/05/18	lun 11/06/18
Documentación	12 días	mar 12/06/1	mié 27/06/18
Fin del proyecto	0 días	mié 27/06/1	mié 27/06/18

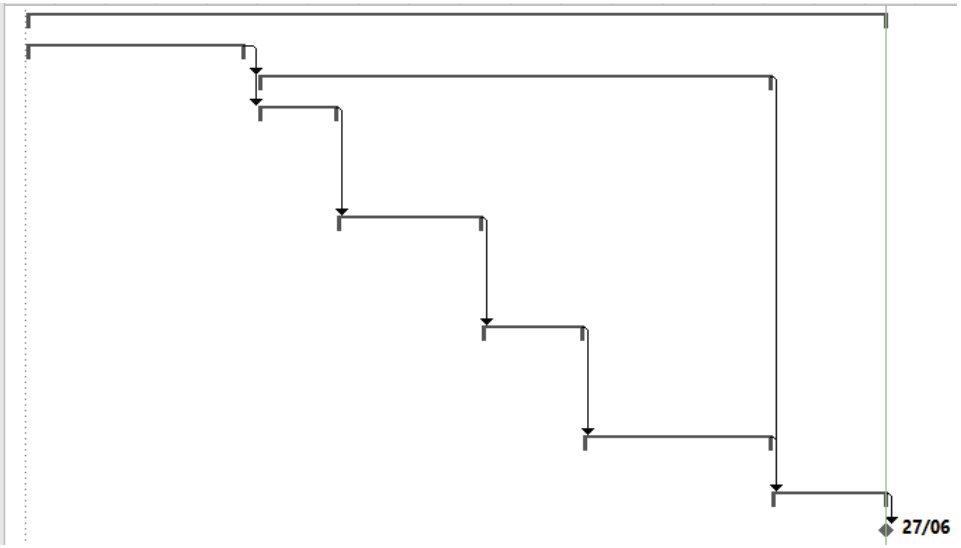

Figura 3.4: Diagrama de gantt resumen del proyecto.

.

Iniciación del proyecto

En la figura 3.5 se muestra el cronograma correspondiente a la fase de iniciación del proyecto. Debido a que el proyecto abarca un campo nuevo para el desarrollador, el conocimiento de este sobre el ámbito en el que va a trabajar es muy escaso. Por este motivo, es necesaria una fase de estudio de tecnologías a utilizar en el proyecto. Es especialmente relevante el tiempo necesario para el estudio de las bases de datos VoltDB y ElasticSearch, por ser tecnologías completamente nuevas para el desarrollador.

Iniciación del proyecto	**22 días**	**jue 01/03/18**	**vie 30/03/18**
Análisis de Requisitos Softwa	**5 días**	**jue 01/03/18**	**mié 07/03/18**
Extracción de requisitos	1 día	jue 01/03/18	jue 01/03/18
Análisis de Requisitos	2 días	vie 02/03/18	lun 05/03/18
Especificación de Requisitc	1 día	mar 06/03/1	mar 06/03/18
Validación de Requisitos	1 día	mié 07/03/1	mié 07/03/18
Gestión del proyecto	**6 días**	**jue 08/03/18**	**jue 15/03/18**
Establecimiento del Plan de Riesgos	2 días	jue 08/03/18	vie 09/03/18
Establecimiento del Sistema de Gestión de la	2 días	lun 12/03/18	mar 13/03/18
Estudio y Selección de Metodología de Desarrollo	2 días	mié 14/03/18	jue 15/03/18
Diseño Preliminar	**1 día**	**vie 16/03/18**	**vie 16/03/18**
Creación del Diagrama de Casos de uso	0,5 días	vie 16/03/18	vie 16/03/18
Creación del diagrama de arquitectura	0,5 días	vie 16/03/18	vie 16/03/18
Diseño preliminar finalizado	0 días	vie 16/03/18	vie 16/03/18
Formación y estudio de las tecnologías	**10 días**	**lun 19/03/18**	**vie 30/03/18**
Estudio de ADEGA	2 días	lun 19/03/18	mar 20/03/18
Estudio de VoltDB	2 días	mié 21/03/1	jue 22/03/18
Estudio de ElasticSearch	2 días	vie 23/03/18	lun 26/03/18
Estudio de AngularJS	1 día	mar 27/03/1	mar 27/03/18
Estudio de Bulma	1 día	mié 28/03/1	mié 28/03/18
Estudio de Spring Boot	1 día	jue 29/03/18	jue 29/03/18
Estudio del API de Youtube	1 día	vie 30/03/18	vie 30/03/18
Iniciación finalizada	0 días	vie 30/03/18	vie 30/03/18

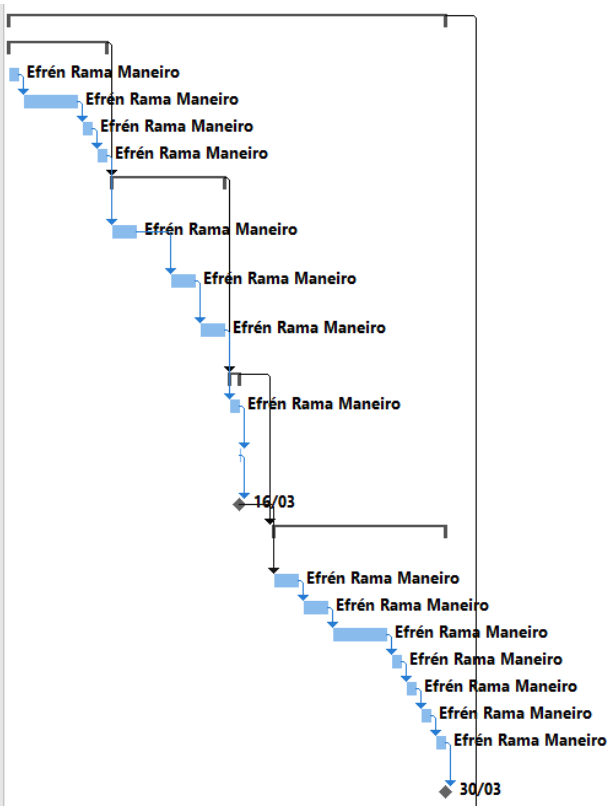

Figura 3.5: Iniciación del proyecto

Como se puede apreciar en el cronograma, esta fase comienza con la extracción de requisitos del software que, a su vez comprende las fases de extracción, análisis, especificación y validación de los distintos requisitos. Posteriormente, se realizan distintas actividades relacionadas con la gestión del proyecto siendo estas el establecimiento de un plan de riesgos, de un sistema de gestión de la configuración y

el estudio de una metodología de desarrollo (en este caso, se opta por un ciclo de vida en incrementos). A continuación, se procede a realizar el diseño preliminar el sistema, en forma de un diagrama de casos de uso y de un diagrama de arquitectura. Por último, se procede a estudiar las distintas tecnologías necesarias para llevar a cabo el proyecto.

Incremento 1: búsqueda y almacenamiento de los metadatos de los vídeos de Youtube

Este incremento, mostrado en la figura 3.6, comprende el diseño, implementación y validación del módulo de almacenamiento de búsqueda y almacenamiento de metadatos de la plataforma Youtube. Nótese que este incremento corresponde a la mitad del subsistema de anotación, descrito en los casos de uso. En primer lugar, se procede a diseñar el diagrama de clases de la aplicación y la interfaz de usuario mediante *wireframes*. Posteriormente, se procede a codificar la abstracción del API de Youtube y la interfaz que permite realizar las búsquedas y almacenamiento de los datos de los vídeos. En este incremento no se codifican los accesos a la base de datos, si no que se deja esa tarea para el siguiente incremento (puesto que es más práctico realizar todo junto). Por último, se valida el sistema diseñando e implementando las pruebas y se redacta el manual de usuario concerniente a esta parte.

Incremento 2: anotador de vídeos seleccionados y almacenamiento de los mismos.

En la figura 3.7 se muestra el cronograma correspondiente al segundo incremento del proyecto. En este incremento, se diseña, implementa y valida el proceso de anotación de vídeos, así como el almacenamiento de los datos resultantes de la anotación. En primer lugar, se diseña la interfaz de usuario, el modelo de datos y el diagrama de clases. A continuación, se crea la base de datos (incluyendo el proceso de sincronización), se codifica el anotador, las interacciones con la base de datos para almacenar el resultado de la anotación y se integra la interfaz de usuario de anotación de los vídeos para que soporte anotación. Por último, se crean y ejecutan las pruebas y se documenta el manual de usuario.

Incremento 1: búsqueda y almacenamiento de los metadatos de los vídeos de Youtube	**9 días**	**lun 02/04/18**	**jue 12/04/18**
Diseño detallado	**2 días**	**lun 02/04/18**	**mar 03/04/18**
Diseño del diagrama de clases	1 día	lun 02/04/18	lun 02/04/18
Diseño de la interfaz de usuario	1 día	mar 03/04/18	mar 03/04/18
Codificación	**5 días**	**mié 04/04/1**	**mar 10/04/18**
Codificación de la abstracción del API de Youtube	3 días	mié 04/04/18	vie 06/04/18
Codificación de la interfaz de usuario	2 días	lun 09/04/18	mar 10/04/18
Ejecución de las pruebas	1 día	mié 11/04/1	mié 11/04/18
Creación del manual de usuario	1 día	jue 12/04/18	jue 12/04/18
Incremento 1 finalizado	0 días	jue 12/04/18	jue 12/04/18

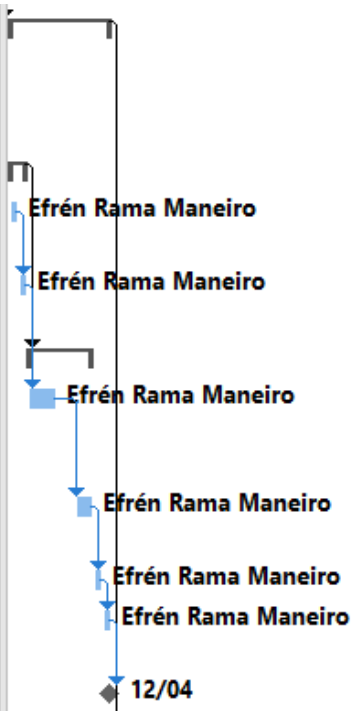

Figura 3.6: Cronograma del incremento 1: búsqueda y almacenamiento de los metadatos de los vídeos de Youtube.

Incremento 3: sistema gestor de los vídeos almacenados.

En la figura 3.8 se muestra el cronograma correspondiente al tercer incremento. En este incremento, se diseña, implementa y valida el módulo que permite gestionar los vídeos que están almacenados en el sistema. De nuevo, se crea el diagrama de clases y el diseño de la interfaz de usuario de este módulo, se codifican los accesos a la base de datos y se crea la interfaz de usuario asociada con el sistema gestor de vídeos almacenados. Por último, se ejecutan las pruebas y se documentan los manuales de usuario.

Incremento 4: sistema pregunta-respuesta.

En la figura 3.9 se muestra el cronograma correspondiente al último incremento. En él, se diseña, implementa y valida el sistema pregunta-respuesta. En primer lugar, se diseña tanto el algoritmo a utilizar para realizar la comparativa de la adecuación entre el contenido de los vídeos y la consulta en lenguaje natural del usuario como el diagrama de clases y la interfaz de usuario correspondiente a

Incremento 2: anotador de vídeos selecciondos y almacenamiento de los mismos	**14 días**	**vie 13/04/18**	**mié 02/05/18**
Diseño detallado	**3 días**	**vie 13/04/18**	**mar 17/04/18**
Diseño del modelo de datos del sistema	1 día	vie 13/04/18	vie 13/04/18
Diseño del diagrama de clases	1 día	lun 16/04/18	lun 16/04/18
Diseño de la interfaz de usuario	1 día	mar 17/04/18	mar 17/04/18
Codificación	**9 días**	**mié 18/04/1**	**lun 30/04/18**
Creación de la base de datos	2 días	mié 18/04/18	jue 19/04/18
Codificación del sistema anotador	3 días	vie 20/04/18	mar 24/04/18
Codificación del almacenamiento de la anotación	2 días	mié 25/04/18	jue 26/04/18
Creación de la interfaz de usuario	2 días	vie 27/04/18	lun 30/04/18
Ejecución de las pruebas	1 día	mar 01/05/1	mar 01/05/18
Creación del manual de usuario	1 día	mié 02/05/18	mié 02/05/18
Incremento 2 finalizado	0 días	mié 02/05/1	mié 02/05/18

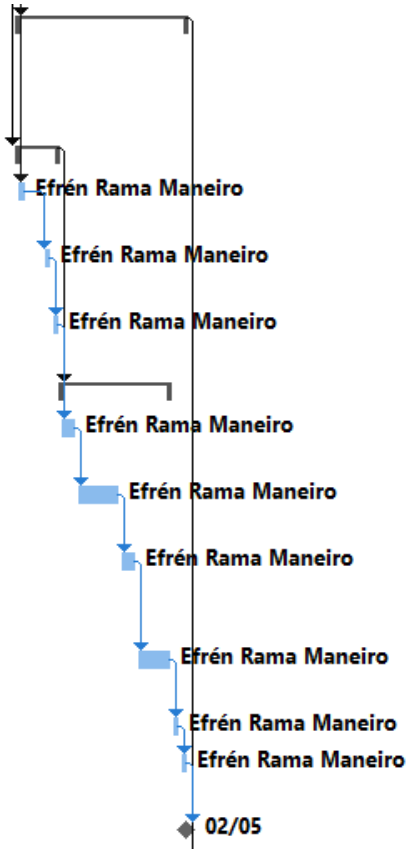

Figura 3.7: Cronograma del incremento 2: anotador de vídeos seleccionados y almacenamiento de los mismos.

Incremento 3: sistema gestor de los vídeos almaceneados (eliminación, actualización...)	**10 días**	**jue 03/05/18**	**mié 16/05/18**
Diseño detallado	**2 días**	**jue 03/05/18**	**vie 04/05/18**
Creación del diagrama de clases	1 día	jue 03/05/18	jue 03/05/18
Diseño de la interfaz de usuario	1 día	vie 04/05/18	vie 04/05/18
Codificación	**6 días**	**lun 07/05/18**	**lun 14/05/18**
Codificación de las operaciones sobre la baes de datos.	4 días	lun 07/05/18	jue 10/05/18
Creación de la interfaz de usuario	2 días	vie 11/05/18	lun 14/05/18
Ejecución de las pruebas	1 día	mar 15/05/18	mar 15/05/18
Creación de los manuales de usuario	1 día	mié 16/05/18	mié 16/05/18
Incremento 3 finalizado	0 días	mié 16/05/1	mié 16/05/18

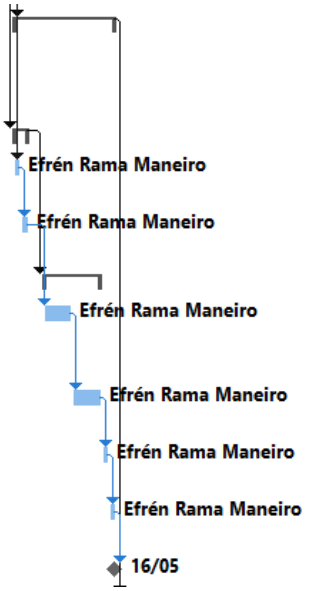

Figura 3.8: Cronograma del incremento 3: sistema gestor de los vídeos anotados

esta parte del sistema. Posteriormente, se codifica el sistema de procesamiento de lenguaje natural del usuario (mediante ADEGA), el sistema de búsqueda por palabras con ElasticSearch, el propio algoritmo de comparación de grafos y la interfaz de usuario. Por último, se ejecutan las pruebas y se documentan los manuales de usuario.

Incremento 4: sistema pregunta-respuesta	**18 días**	**jue 17/05/18**	**lun 11/06/18**
Diseño detallado	**6 días**	**jue 17/05/18**	**jue 24/05/18**
Diseño del algoritmo a utilizar	4 días	jue 17/05/18	mar 22/05/18
Creación del diagrama de clases	1 día	mié 23/05/18	mié 23/05/18
Diseño de la interfaz de usuario	1 día	jue 24/05/18	jue 24/05/18
Codificación	**10 días**	**vie 25/05/18**	**jue 07/06/18**
Procesamiento del lenguaje natural	1 día	vie 25/05/18	vie 25/05/18
Codificación del sistema de búsquedas en ElasticSearch	2 días	lun 28/05/18	mar 29/05/18
Codificación del algoritn	5 días	mié 30/05/1	mar 05/06/18
Codificación de la interfaz de usuario	2 días	mié 06/06/18	jue 07/06/18
Ejecución de las pruebas	1 día	vie 08/06/18	vie 08/06/18
Creación de los manuales de usuario	1 día	lun 11/06/18	lun 11/06/18
Incremento 4 finalizado	0 días	lun 11/06/18	lun 11/06/18

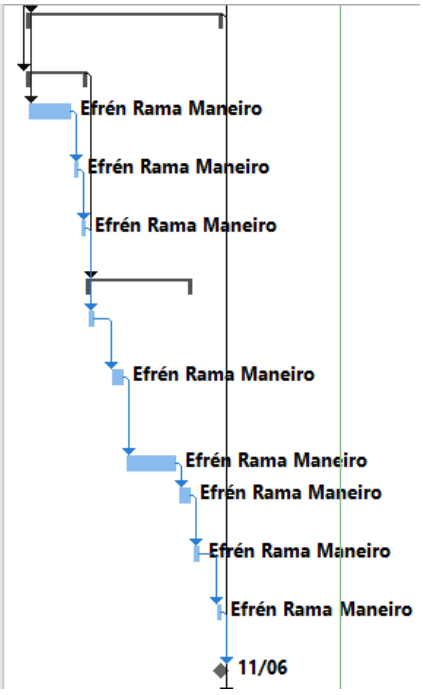

Figura 3.9: Cronograma del incremento 4: sistema pregunta-respuesta

Documentación.

En la figura 3.10 se muestra el cronograma correspondiente a la etapa de cierre del proyecto. En esta etapa, se procede a la creación de la memoria del TFG y la creación del manual técnico del sistema.

Documentación	**12 días**	**mar 12/06/18**	**mié 27/06/18**
Creación de la memoria del T	3,2 sem.	mar 12/06/18	lun 25/06/18
Creación de los manuales técnicos	2 días	mar 26/06/18	mié 27/06/18
Fin del proyecto	0 días	mié 27/06/18	mié 27/06/18

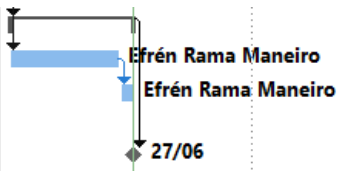

Figura 3.10: Cronograma del cierre del proyecto.

Capítulo 4

Análisis

4.1. Especificación de requisitos

Una especificación de requisitos software es una descripción completa del sistema software a desarrollar en el proyecto. Una especificación de requisitos contiene:

- **Casos de uso**: Conjunto de funcionalidades que realizará el sistema software.
- **Requisitos no funcionales**: Describen las características del sistema a desarrollar desde el punto de vista de la seguridad, el rendimiento...
- **Requisitos de información**: Describe los datos que utilizará el sistema software.

El principal motivo por el que no se incluyen los requisitos funcionales reside en que tanto casos de uso como requisitos funcionales representan funcionalidades del sistema, y, por lo tanto, pueden ser redundantes. En general, se recomienda utilizar casos de uso puesto que tienen un modelo más detallado que facilita una definición más precisa de una función. Sólo es recomendable utilizar ambos cuando se necesita describir funcionalidades con distinto nivel de abstracción, es decir, funcionalidades muy complejas que, en la práctica, necesitarán de varias clases colaborando y funcionalidades muy sencillas que pueden ser implementadas con el método de una clase pero que son importantes para entender lo que se pretende. En este caso, no existen tales funcionalidades por lo que se omite la descripción de requisitos funcionales.

4.1.1. Actores

- **Administrador**: Cualquier usuario que esté autorizado para utilizar el sistema.
- **Paciente**: Cualquier usuario que desee ejecutar consultas al sistema de pregunta-respuesta.

4.1.2. Casos de uso

Para la definición de los casos de uso se utilizará la siguiente plantilla:

CU. ID	Nombre del caso de uso
Descripción	Descripción del propósito del caso de uso
Actores	Enumeración de los actores que participan en el caso de uso
Pre–condiciones	
1. Precondición 1 2. Precondición 2	
Secuencia normal	
1. Paso 1 2. Paso 2	
Curso alternativo	
3.a. Indica cursos alternativos en el caso de uso.	
Importancia	Indica la importancia que tiene el requisito (vital, quedaría bien).
Estabilidad	Indica que estabilidad debe tener el requisito (alta, media o baja).
Validación	Indica los criterios de validación del requisito

Los casos de uso aparecen agrupados en los subsistemas a los que pertenecen siendo estos los siguientes:

- **Subsistema de anotación de vídeos**: gestiona la búsqueda de canales y vídeos de Youtube, así como la anotación de estos últimos.
- **Subsistema de gestión de vídeos anotados**: permite gestionar aquellos vídeos que ya hayan sido anotados (consulta, actualización y borrado).

- **Subsistema pregunta-respuesta**: permite responder a preguntas en lenguaje natural de un usuario.

Subsistema de anotación de vídeos

El subsistema de anotación de vídeos se encarga de habilitar búsquedas de vídeos en Youtube y realizar la anotación semántica a partir de los subtítulos de los vídeos. Los datos obtenidos de este proceso se almacenan en bases de datos para su posterior aprovechamiento. El diagrama de casos de uso que representa este subsistema se muestra en la figura 4.1. La descripción de los casos de uso que conforman este subsistema se presentan a continuación.

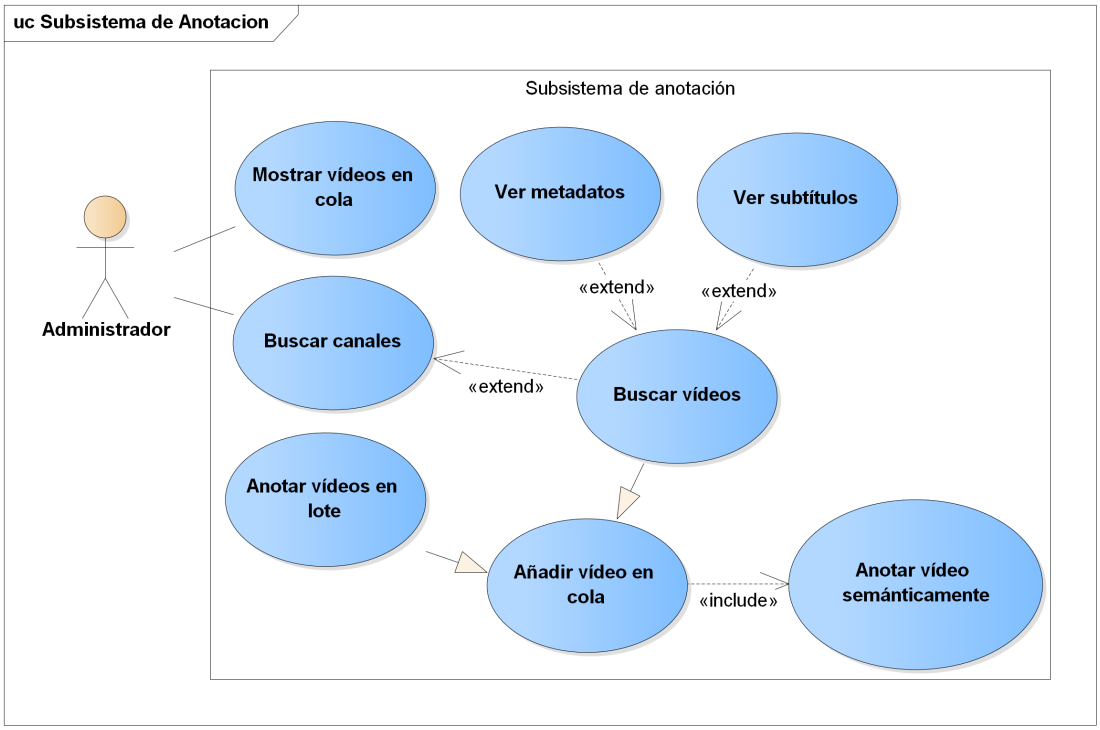

Figura 4.1: Subsistema de anotación de videos

CU. 1	Anotar vídeo semánticamente.
Descripción	Permite anotar semánticamente un vídeo de Youtube.
Actores	Administrador
Pre–condiciones	
1. Se conoce el identificador del vídeo que se desea anotar, ya presente en la cola de anotación.	

Secuencia normal	
1. El sistema descarga y purga los subtítulos para el vídeo en cuestión. 2. El sistema ejecuta el algoritmo de contexto sobre los subtítulos descargados. 3. El sistema ejecuta el algoritmo de nodos raíz sobre los subtítulos descargados. 4. El sistema ejecuta el algoritmo de generación de grafo sobre los subtítulos descargados. 5. El sistema guarda en las bases de datos los resultados de la anotación. 6. El sistema devuelve el grafo generado de los pasos anteriores.	
Curso alternativo 1	
1.a. El sistema no encuentra ningún subtítulo aprovechable del vídeo. 2. El sistema devuelve un código de error indicando que no se han encontrado subtítulos.	
Importancia	Vital
Estabilidad	Alta
Criterio de validación	Se considera el requisito cumplido si es posible realizar una anotación semántica de un vídeo en concreto y almacenar dicha información en las bases de datos.

CU. 2	Anotar vídeos en lote
Descripción	Permite añadir a la cola de anotación un conjunto de vídeos.
Actores	Administrador
Pre–condiciones	
1. No hay	
Secuencia normal	
1. El usuario selecciona la opción “Anotar vídeos en lote” de la interfaz gráfica. 2. El sistema muestra un recuadro de texto donde introducir la lista de vídeos. 3. El usuario introduce la lista de vídeos separada por un espacio en blanco cada uno. 4. El sistema ejecuta el caso de uso “Añadir vídeo en cola” por cada vídeo introducido. 5. El sistema muestra un mensaje indicando que los vídeos han sido añadidos correctamente a la cola.	

Importancia	Vital
Estabilidad	Alta
Criterio de validación	Se considera el requisito cumplido si es posible añadir un conjunto de vídeos a la cola de ejecución.

CU. 3	Añadir vídeo en cola
Descripción	Permite añadir un vídeo a la cola de anotaciones.
Actores	Administrador
Pre–condiciones	
1. Se conoce el vídeo que se desea añadir a la cola. **Secuencia normal**	
1. El sistema añade el vídeo a la cola de anotaciones. 2. El sistema cambia el estado de procesamiento del vídeo a “En espera”. 3. El sistema comprueba que se pueda ejecutar la anotación y cambia el estado de procesamiento del vídeo a “En ejecución”. 4. El sistema ejecuta el caso de uso “Anotar vídeo semánticamente”. 5. El sistema cambia el estado de procesamiento del vídeo a “Finalizado”.	
Curso alternativo 1	
3.a. El sistema no puede ejecutar la anotación porque hay otros vídeos ya ejecutándose con más prioridad. 4. Se vuelve al paso 2.	
Curso alternativo 2	
4.a. La anotación indica que no hay subtítulos para el vídeo. 5. Se cambia el estado de la anotación a “Sin subtítulos”.	
Importancia	Vital
Estabilidad	Alta
Criterio de validación	Se considera el requisito cumplido si es posible añadir y procesar un vídeo a la cola de anotaciones.

CU. 4	Mostrar vídeos en cola
Descripción	Permite mostrar los vídeos que están o han estado en cola de ejecución.
Actores	Administrador

Pre–condiciones	
1. No hay	
Secuencia normal	
1. El usuario selecciona la opción “Mostrar cola de anotaciones”. 2. El sistema muestra una tabla con todos los trabajos presentes en el sistema, junto con su estado de procesamiento.	
Importancia	Vital
Estabilidad	Alta
Criterio de validación	Se considera el requisito cumplido si es posible visualizar los vídeos en cola y su estado en la misma.

CU. 5	Buscar canales
Descripción	Permite buscar canales de la plataforma Youtube.
Actores	Administrador
Pre–condiciones	
1. No hay.	
Secuencia normal	
1. El usuario selecciona la opción “Anotar canal”. 2. El sistema muestra un recuadro de texto que invita a introducir el nombre del canal a buscar. 3. El usuario introduce el nombre del canal y pulsa sobre el botón buscar. 4. El sistema muestra una lista de canales coincidentes con la búsqueda del usuario.	
Importancia	Vital
Estabilidad	Alta
Criterio de validación	Se considera el requisito cumplido si es posible buscar canales de Youtube.

CU.6	Buscar vídeos
Descripción	Permite buscar los vídeos asociados a un canal de Youtube.
Actores	Administrador
Pre–condiciones	

1. Se ha realizado previamente una búsqueda de canales.	
Secuencia normal	
1. El usuario pulsa sobre el botón "Ver vídeos" del canal que le interese. 2. El sistema muestra una lista de vídeos del canal. 3. El usuario marca para anotación los vídeos que le interesen y pulsa "Anotar seleccionados" una vez finalice. 4. El sistema ejecuta el caso de uso "Añadir vídeo en cola" por cada uno de los vídeos seleccionados.	
Importancia	Vital
Estabilidad	Alta
Criterio de validación	El requisito se considera cumplido cuando sea posible ver los vídeos dentro del canal y seleccionar vídeos para su procesamiento.

CU. 7	Ver metadatos
Actores	Administrador
Descripción	Permite ver los metadatos asociados a un vídeo de Youtube ("likes", etiquetas, descripción, número de comentario, etc.).
Pre–condiciones	
1. Se ha realizado previamente una búsqueda de vídeos.	
Secuencia normal	
1. El usuario pulsa sobre el botón "Ver metadatos" asociado a un vídeo de la lista de vídeos buscados 2. El sistema muestra una ventana emergente con los metadatos más relevantes del vídeo. 3. El usuario pulsa el botón cerrar cuando haya finalizado de ver los metadatos. 4. El sistema cierra la ventana emergente.	
Importancia	Vital
Estabilidad	Alta

Criterio de validación	El requisito se considera cumplido cuando sea posible ver los vídeos dentro del canal y seleccionar vídeos para su procesamiento.

CU. 8	Ver subtítulos.
Descripción	Permite ver los subtítulos asociados a un vídeo de Youtube.
Pre–condiciones	
1. Se ha realizado previamente una búsqueda de vídeos.	
Secuencia normal	
1. El usuario pulsa sobre el botón "Ver subtítulos" asociado a la ventana de metadatos de un vídeo. 2. El sistema muestra una nueva ventana con los subtítulos del vídeo de Youtube seleccionado. Dichos subtítulos se muestran tal y como los enseña la plataforma. 3. El usuario pulsa sobre el botón cerrar cuando haya finalizado de ver los subtítulos.	
Importancia	Vital
Estabilidad	Alta
Criterio de validación	El requisito se considera cumplido cuando sea posible ver los subtítulos asociados a un vídeo de la plataforma Youtube.

Subsistema de gestión de vídeos anotados

El subsistema de gestión de vídeos anotados se encarga de permitir visualizar los vídeos que ya han sido almacenados en el sistema. Sobre dichos vídeos es posible eliminarlos completamente del sistema o actualizar los datos del vídeo, incluyendo, en este caso, tanto los datos obtenidos directamente de Youtube como los resultados de la anotación semántica. El diagrama de casos de uso que representa este subsistema se muestra en la figura 4.2. La descripción de los casos de uso que conforman este subsistema se describen a continuación:

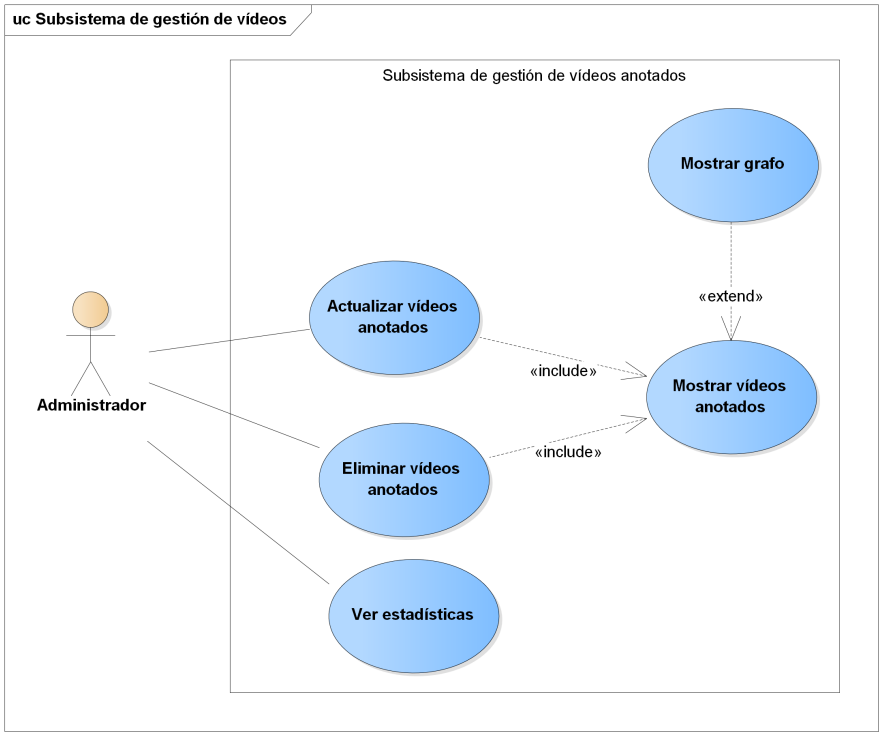

Figura 4.2: Subsistema de gestión de vídeos anotados

CU. 9	Mostrar vídeos anotados
Descripción	Permite mostrar los vídeos almacenados en las bases de datos del sistema
Actores	Administrador
Pre–condiciones	
1. No hay.	
Secuencia normal	
1. El usuario selecciona la opción “Administrar vídeos anotados”. 2. El sistema muestra un recuadro de texto que invita a buscar vídeos. 3. El usuario introduce su consulta en el recuadro de texto. 4. El sistema muestra los vídeos anotados en el sistema teniendo en cuenta la consulta del usuario.	
Importancia	Vital
Estabilidad	Alta

Criterio de validación	Se considera el requisito cumplido si es posible mostrar todos los vídeos anotados en el sistema y si es posible filtrar la lista de vídeos anotados en función de la consulta del usuario.

CU. 10	Eliminar vídeos anotados
Descripción	Permite eliminar un conjunto de vídeos del sistema
Actores	Administrador
Pre–condiciones	
1. No hay.	
Secuencia normal	
1. Se ejecuta el caso de uso "Mostrar vídeos anotados". 2. El usuario selecciona los vídeos que desea eliminar de la lista mostrada y pulsa en "Eliminar vídeos". 3. El sistema elimina los vídeos de la base de datos.	
Importancia	Vital
Estabilidad	Alta
Criterio de validación	Se considera el requisito cumplido si es posible eliminar los vídeos seleccionados por el usuario.

CU. 11	Actualizar vídeos anotados
Descripción	Permite actualizar las anotaciones de un conjunto de vídeos del sistema
Actores	Administrador
Pre–condiciones	
1. No hay.	
Secuencia normal	
1. Se ejecuta el caso de uso "Mostrar vídeos anotados". 2. El usuario selecciona los vídeos que desea eliminar de la lista mostrada y pulsa en "Actualizar vídeos". 3. El sistema actualiza los vídeos realizando las operaciones necesarias (anotación y modificación de los datos).	
Importancia	Vital
Estabilidad	Alta

Criterio de validación	Se considera el requisito cumplido si es posible actualizar los vídeos seleccionados por el usuario.

CU. 12	Mostrar grafo
Descripción	Permite mostrar el grafo de aquellos vídeos que han sido anotados en el sistema
Actores	Administrador
Pre–condiciones	
1. No hay	
Secuencia normal	
1. Se ejecuta el caso de uso "Mostrar vídeos anotados". 2. El usuario pulsa sobre el botón "Ver grafo" adjunto a un vídeo de la lista de vídeos. 3. El sistema muestra una pantalla con el grafo resultante de la anotación de dicho vídeo. 4. El usuario cierra la pantalla cuando haya terminado la operación.	
Importancia	Vital
Estabilidad	Alta
Criterio de validación	Se considera el requisito cumplido si es posible mostrar el grafo asociado a cualquier vídeo anotado del sistema.

CU. 13	Mostrar estadísticas
Descripción	Permite mostrar las estadísticas y el estado del sistema en tiempo real.
Actores	Administrador
Pre–condiciones	
1. No hay	
Secuencia normal	
1. El administrador pulsa sobre la opción del menú "*dashboard*". 2. El sistema muestra el número de vídeos anotados, unos gráficos que muestran el uso de recursos de los servidores de la base de datos y si las máquinas de las bases de datos están funcionando o no.	
Importancia	Quedaría bien

Estabilidad	Alta
Criterio de validación	Se considera el requisito cumplido si es posible mostrar el grafo asociado a cualquier vídeo anotado del sistema.

Subsistema de pregunta-respuesta

El subsistema de pregunta-respuesta se encarga de responder a las preguntas en lenguaje natural del usuario mediante un vídeo. Pese a que está conformado por un único caso de uso, no tiene nada que ver con los casos de uso anteriores por lo que debe especificarse aparte. El diagrama de casos de uso que representa este subsistema se muestra en la figura 4.3. La descripción del caso de uso que conforma este subsistema se describe a continuación:

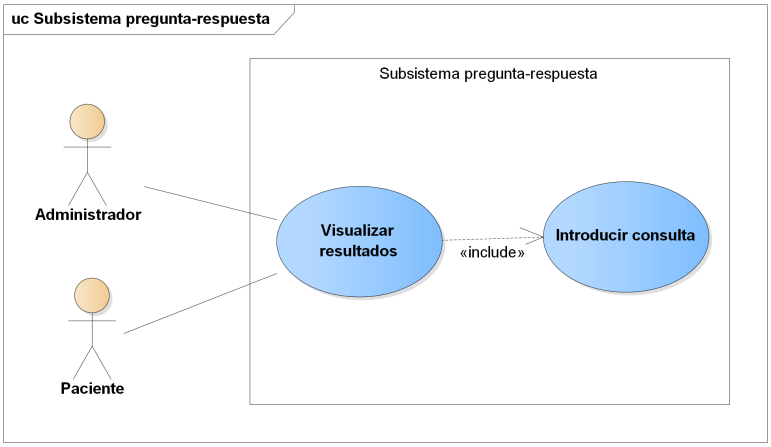

Figura 4.3: Subsistema de pregunta-respuesta

CU. 14	Introducir consulta.
Descripción	Permite introducir una consulta al sistema pregunta-respuesta para su procesamiento.
Actores	Paciente, Administrador
Pre–condiciones	
1. No hay	
Secuencia normal	
1. El usuario introduce en el cuadro de texto su consulta en lenguaje natural.	

2. El sistema captura la consulta y ejecuta el algoritmo sobre la misma.	
Importancia	Vital
Estabilidad	Alta
Criterio de validación	El requisito se considera cumplido si es posible enviar y procesar la consulta del usuario en lenguaje natural y obtener un conjunto de vídeos relevantes para el usuario.

CU. 15	Visualizar resultados
Descripción	Permite ver los resultados del procesamiento de la consulta del usuario.
Actores	Paciente, Administrador
Pre–condiciones	
1. Se ha capturado previamente una consulta del usuario.	
Secuencia normal	
1. El sistema devuelve una lista de vídeos como resultado de procesar la consulta del usuario. 2. El usuario visualiza los resultados devueltos por el sistema.	
Importancia	Vital
Estabilidad	Alta
Criterio de validación	Se considera el requisito validado cuando sea posible ver los resultados devueltos por el sistema pregunta-respuesta.

4.1.3. Requisitos no funcionales

Para la descripción de los requisitos no funcionales se utiliza la siguiente plantilla:

Requisito RQNF-X	
Título:	Nombre del requisito
Descripción:	Descripción del requisito
Importancia:	Importancia del requisito (alta, media o baja)
Urgencia:	Urgencia de la implementación del requisito (alta, media o baja)

Criterio de validación:	Criterio de validación del requisito

La cabecera de la tabla representa el identificador del requisito, que es unívoco a lo largo del documento.

A continuación se especifica el catálogo de requisitos no funcionales del sistema software:

Requisito RQNF-1	
Título:	Rendimiento del sistema de consultas
Descripción:	El sistema de pregunta respuesta no debe tardar más de 10 segundos en responder a una consulta del usuario. El motivo de este requisito reside en que el sistema debe responder a preguntas en tiempo real.
Importancia:	Alta
Urgencia:	Media
Criterio	Las consultas que se realicen tardan todas menos de 10 segundos en ser procesadas.

Requisito RQNF-2	
Título:	Idioma utilizado por el usuario
Descripción:	El idioma utilizado por el usuario para realizar consultas al sistema pregunta-respuesta será el inglés.
Importancia:	Alta
Urgencia:	Media
Criterio de Validación	Las consultas que se realicen responden correctamente al idioma inglés.

Requisito RQNF-3	
Título:	Interfaz usable

Descripción:	La interfaz de usuario diseñada deberá ser usable para facilitar la adaptación de los usuario, cumpliendo con el mayor número de heurísticos de Nielsen posibles y obteniendo la mejor nota posible en los tests de usabilidad.
Importancia:	Alta
Urgencia:	Media
Criterio de Validación	La evaluación de usabilidad supera la evaluación heurística mediante los principios de Nielsen y los tests de usabilidad tienen una nota media mayor que 8.

Requisito RQNF-4	
Título:	Interfaz gráfica independiente de motor.
Descripción:	El sistema pregunta-respuesta debe ser implementado de forma separada a la interfaz, puesto que es posible que en un futuro se cambie la interacción con el paciente.
Importancia:	Alta
Urgencia:	Alta
Criterio de Validación	Se considerará validado este requisito si el sistema pregunta-respuesta se desarrolla independientemente de la interfaz pudiendo ser llamado sin la necesidad de esta última, es decir, si el sistema pregunta-respuesta se expone como un servicio REST.

Requisito RQNF-5	
Título:	Extensiblidad
Descripción:	El código debe estar diseñado de forma que sea posible efectuar, de forma simple, un cambio del sistema pregunta-respuesta sin afectar de forma significativa al resto del código.
Importancia:	Alta
Urgencia:	Alta

4.1.4. Requisitos de información

Para la descripción de los requisitos de información se utiliza la siguiente plantilla:

Requisito RQI-X	
Título:	Nombre del requisito
Descripción:	Descripción del requisito
Importancia:	Importancia del requisito (alta, media o baja)
Estabilidad	Alta, media o baja.
Datos específicos	Datos específicos que maneja el requisito de información.

A continuación se especifica el catálogo de requisitos de información del sistema.

Requisito RQI-1	
Título:	Vídeo
Descripción:	Cada uno de los vídeos recuperados de la plataforma Youtube.
Importancia:	Alta
Estabilidad	Alta
Datos específicos	
Identificador. Título. Descripción. Etiquetas ("*tags*"). Número de visitas. Número de "me gustas" ("*likes*"). Número de "no me gustas" ("*dislikes*"). Número de favoritos. Número de comentarios.	

Requisito RQI-2	
Título:	Término del contexto
Descripción:	Cada uno de los elementos del contexto que se recuperan del proceso de anotación semántica

Importancia:	Alta
Estabilidad	Alta
Datos específicos	
Término. URI. Relevancia. Lista de sinónimos. Posición. Longitud. Tipo de entidad (*posTag*).	

Requisito RQI-3	
Título:	Nodo
Descripción:	Cada uno de los nodos que se recuperan del algoritmo de anotación semántica
Importancia:	Alta
Estabilidad	Alta
Datos específicos	
Identificador. Etiqueta. Recurso. Es categoría. Es desambiguación. Término del contexto asociado.	

Requisito RQI-4	
Título:	Relación
Descripción:	Cada una de las relaciones que se generan a partir del grafo.
Importancia:	Alta
Estabilidad	Alta
Datos específicos	
Identificador de nodo origen. Nombre de la relación que une los nodos.	

Identificador de nodo destino.

Requisito RQI-5	
Título:	Estadísticas
Descripción:	Cada una de las estadísticas que se muestran en el sistema.
Importancia:	Alta
Estabilidad	Alta
Datos específicos	
Número de vídeos anotados. Número de nodos anotados. Número de relaciones anotadas.	

4.2. Matrices de trazabilidad

Las matrices de trazabilidad de requisitos es una herramienta que permite vincular los requisitos del producto, desde su concepción, hasta los entregables del proyecto que los satisfacen. Se distinguen las siguientes tipos de matrices de trazabilidad:

- **Matriz: caso de uso - objetivo del sistema**: permite vincular las funcionalidades del sistema con los objetivos del mismo. Se corresponde con la matriz de la figura 4.9.
- **Matriz: caso de uso - entregable del proyecto**: permite vincular las funcionalidades del sistema con los entregables del proyecto. Se corresponde con la matriz de la figura 4.10.

Cuadro 4.9: Matriz: casos de uso - objetivos del sistema

	OS. 01	OS. 02	OS. 03	OS. 04	OS. 05	OS. 06
CU. 01	↱	↱				
CU. 02	↱	↱				
CU. 03	↱	↱				
CU. 04	↱	↱				
CU. 05	↱	↱				
CU. 06	↱	↱				
CU. 07	↱	↱				
CU. 08	↱	↱				
CU. 09			↱	↱		
CU. 10			↱	↱		
CU. 11			↱	↱		
CU. 12			↱	↱		
CU. 13			↱	↱		
CU. 14					↱	↱
CU. 15					↱	↱

Cuadro 4.10: Matriz: casos de uso - entregables del proyecto

	EN. 01	EN. 02	EN. 03	EN. 04	EN. 05	EN. 06	EN. 07	EN. 08
CU. 01	↰	↰	↰				↰	↰
CU. 02	↰	↰	↰				↰	↰
CU. 03	↰	↰		↰			↰	↰
CU. 04	↰	↰		↰			↰	↰
CU. 05	↰	↰	↰				↰	↰
CU. 06	↰	↰	↰				↰	↰
CU. 07	↰	↰	↰				↰	↰
CU. 08	↰	↰	↰				↰	↰
CU. 09	↰	↰			↰		↰	↰
CU. 10	↰	↰			↰		↰	↰
CU. 11	↰	↰			↰		↰	↰
CU. 12	↰	↰			↰		↰	↰
CU. 13	↰	↰			↰		↰	↰
CU. 14	↰	↰				↰	↰	↰
CU. 15	↰	↰				↰	↰	↰

Capítulo 5

Análisis de tecnologías y herramientas

En este capítulo se realizará un análisis exhaustivo de las tecnologías que se han utilizado en el proyecto, así como su motivación para su utilización.

5.1. Tecnologías de desarrollo

A continuación se procederá a describir las tecnologías que se han utilizado para desarrollar el sistema software del proyecto.

5.1.1. Librerías

Google2SRT

Google2SRT [26] es una aplicación de escritorio que permite realizar diversas manipulaciones con los subtítulos de cualquier vídeo de Youtube. De todas las funcionalidades que provee, únicamente se utiliza la descarga de subtítulos. Como la aplicación se distribuye en un ".jar" es posible incluirlo en el proyecto Java directamente y acceder a todas las funciones públicas de forma sencilla. Es imperativo utilizar esta librería puesto que, pese a que la API de Youtube proporciona métodos para acceder a los subtítulos, no es posible descargar subtítulos autogenerados directamente mediante la API y, en la mayoría de los casos, los únicos subtítulos disponibles son de este tipo.

GSON

GSON [14] es una librería de Google para la serialización y deserialización de Objetos Java a JSON y viceversa. Es preciso utilizar esta librería para crear una abstracción para obtener y enviar datos de ADEGA de forma programática desde Java directamente.

VisJS

VisJS [32] es una librería que permite representar grafos en aplicaciones web de forma sencilla. Se utiliza para representar los grafos de los vídeos anotados en el administrador de manera dinámica.

5.1.2. *Frameworks*

Spring Boot

Spring Boot [25] permite crear aplicaciones web basadas en Spring listas para producción autocontenidas, es decir, el resultado del empaquetamiento de la aplicación es un fichero ejecutable que despliega un servidor web completo. Este framework simplifica la configuración de Spring siempre que sea posible, minimizando al máximo la necesidad de editar ficheros XML para configurar el servidor. Así, uno de los principales pilares de Spring Boot es permitir desplegar aplicaciones lo más fácil y rápido posible. Spring Boot integra un servidor web embebido: Tomcat 7 [7], es decir, está incluido en la propia aplicación Spring Boot y se encarga de exponer tanto los servicios REST como la aplicación web.

Así, este framework permite, por un lado, crear los servicios REST que serán consumidos por la aplicación web y, por otro lado, servir al cliente la propia aplicación web.

Maven

Maven [25] permite gestionar el ciclo de vida de construcción de cualquier aplicación Java. Permite gestionar las dependencias de la aplicación mediante un fichero XML. Esta herramienta descarga automáticamente las dependencias de un repositorio centralizado (MavenCentral o JCenter, por ejemplo) y permite

automatizar la construcción y despliegue de la aplicación así como ejecutar la batería de pruebas software de forma automática.

Así, se evita la dependencia del Entorno de Desarrollo Integrado para realizar la construcción y despliegue de la aplicación, pudiendo realizarla sobre línea de comandos en cualquier máquina si es preciso.

AngularJS

AngularJS [13] es un framework para el desarrollo de *front-end* mantenido por Google mediante el lenguaje de programación Javascript. Una de las mayores fortalezas de este framework es el desarrollo de aplicaciones de una sóla página (*Single Page Application*). Dichas aplicaciones permiten intercambiar dinámicamente el contenido de la página web sin necesidad de refrescar el navegador de forma sencilla. Por otra parte, este framework facilita la creación de una aplicación web basada en el patrón Modelo-Vista-Controlador (MVC) proporcionando herramientas que facilitan la utilización de los servicios web creados mediante el framework Spring Boot.

Bulma

Bulma [30] es un framework CSS para el diseño de aplicaciones web. Los motivos de la elección de este framework frente a otros más populares como *Bootstrap* o *Materialize* son los siguientes:

- Únicamente contiene CSS, es decir, no incluye código Javascript para la generación de elementos como modales. Esto permite integrarlo con cualquier framework de *front-end* de forma sencilla o, incluso, no utilizar ninguna librería Javascript y realizar todo el desarrollo en *Vanilla Javascript*.
- Sistema de columnas sencillo. Únicamente es necesario definir el número de elementos *"div"* que conforman la malla sin especificar su tipo.
- Es modular, lo que implica que es posible importar elementos aislados del framework para su utilización. Por ejemplo, es posible utilizar únicamente el sistema de columnas sin importar el resto de elementos de Bulma (botones, formularios, etc.).

- El diseño se controla mediante variables SASS, lo que permite, mediante la modificación un único fichero de variables, personalizar el estilo de toda la página de forma rápida y sin problemas.

JUnit y MockMVC

JUnit [19] es una librería que habilita la creación de pruebas unitarias para el lenguaje Java. Se complementa con MockMVC, que permite realizar pruebas unitarias sobre servicios REST.

5.1.3. APIs

Youtube V3

El API de Youtube [15] consiste en un conjunto de servicios REST que se consumen para acceder a funcionalidades de búsqueda de canales, vídeos, obtención de metadatos asociados a los vídeos, etc. El acceso a esta API se realiza mediante un API Java que permite evitar el tener que utilizar funciones HTTP para llamar a los servicios directamente. Así, simplemente se utilizan clases Java que abstraen dichas llamadas.

Para utilizar esta API es necesario realizar un proceso de autorización mediante alguna de las dos formas siguientes: mediante un sistema denominado "OAuth2" o mediante una llave API. Se escoge la segunda forma puesto que es totalmente transparente al usuario (la primera requiere pulsar un botón que confirme el acceso a la API).

5.1.4. ADEGA

ADEGA [9] es un anotador semántico que, a partir de un documento de texto realiza las siguientes operaciones:

- Identifica los términos relevantes del texto (también denominados *menciones*). Es preciso destacar que estas menciones pueden ser *ambiguas*, es decir, una misma mención puede hacer referencia a conceptos distintos.
- Selecciona dentro de una ontología un conjunto de entidades (denominadas *entidades candidatas*) para anotar cada una de las menciones.

- Crea un grafo semántico que interconecta estas entidades candidatas.
- Utiliza el grafo anterior para desambiguar colectivamente cuál de las entidades candidatas es la más adecuada para anotar cada una de las menciones.

ADEGA no está ligado a una ontología determinada y, por ello, puede anotar textos de cualquier dominio. La calidad de su anotación depende, en gran medida, de la ontología utilizada para seleccionar las entidades más adecuadas para el proceso de anotación. Dos ejemplos de ontologías muy conocidas son:

- **DBpedia:** es una formalización de la Wikipedia y, por lo tanto, tiene un carácter genérico y multidominio.
- **MeSH:** contiene terminología médica especializada. Es la ontología utilizada en el presente proyecto.

En concreto, las operaciones que es posible realizar mediante la API proporcionada por esta herramienta se describen a continuación.

Autorización

Todas las operaciones del API REST requieren autorización. Para ello, únicamente se precisa el correo electrónico del individuo que está utilizando el API. Si el correo es válido se devuelve un *token* que identifica al usuario en todas las llamadas al API para esa sesión.

Extracción del contexto

Esta operación permite extraer los términos más relevantes del texto. En primer lugar, se extraen las entidades nombradas del texto en cuestión (principalmente sustantivos) y, a partir de dichos entidades, se extraen los términos compuestos (como, por ejemplo, Banco de España).

La extracción de términos compuestos se realiza "probando" todas las combinaciones de hasta longitud 5 de las entidades nombradas del contexto. La "prueba" consiste en buscar el término combinado en un índice Lucene de términos obtenido directamente de MeSH. En caso de que la búsqueda sea fructífera, el término

es compuesto; en otro caso, se descarta la combinación y se prueba la siguiente. Así, esta operación tiene los siguientes parámetros configurables:

- **Texto:** texto a analizar.
- **Ontología:** indica la ontología que se va a utilizar para realizar la búsqueda de los términos compuestos. Las ontologías soportadas actualmente por ADEGA son DBPedia (en inglés) y MeSH.
- **Extractor de términos compuestos:** indica qué extractor de términos compuestos se va a utilizar.
- **Número de elementos del contexto a extraer:** indica el número de elementos del contexto que se devuelven de la llamada al API. En caso de que no se especifique, se devuelven todos los términos encontrados.

Extracción de los nodos raíz

Esta operación permite asignar a cada término del contexto la URI del recurso que lo representa en una ontología determinada. Para ello, de nuevo, se realiza una búsqueda en el índice para comprobar la URI correspondiente a cada término. La URI que más relevancia tenga, es decir, la que "más se parezca", es la URI que se asigna. Hay que tener en cuenta que la relevancia la proporciona directamente Lucene.

En este paso se realiza la desambiguación de los términos del contexto. Se conoce por "desambiguación" al proceso de obtener los nodos a los que se refiere un nodo ambiguo. Por ejemplo, dado el nodo "DBA", que es ambiguo, los nodos de la desambiguación serían: "Database Administrator", "Doctor in Business Administration", etc.

Así, los parámetros configurables en esta operación son los siguientes:

- **Contexto:** contexto extraído del paso anterior en formato JSON.
- **Ontología:** ontología que se va a utilizar para realizar la extracción, de forma análoga al paso anterior.

- **Extractor de nodos raíz:** permite cambiar la estrategia de extracción de los nodos. Se distinguen dos estrategias:
  - **Extractor de nodos simple:** este extractor simplemente realiza la asignación de término a recurso sin intentar desambiguar.
  - **Extractor de nodos con desambiguación:** este extractor intenta desambiguar añadiendo todos los nodos relacionados con el nodo ambiguo.

Generación del grafo

Esta operación permite generar el grafo a partir de los nodos raíz extraídos en el paso anterior. El algoritmo a seguir es el siguiente:

- Inicialmente, es preciso localizar los nodos que se pretende alcanzar desde la raíz. Dichos nodos se denominan "**nodos hoja**". Estos nodos se obtienen consultando al índice Lucene usando el contexto extraído en la operación correspondiente. Se obtiene como resultado los 100 nodos hoja más relevantes.
- Se explora desde los nodos raíz hasta cierta profundidad pesado los nuevos nodos obtenidos del proceso de exploración, se crea el subgrafo los nodos explorados de este paso y se realiza el pesado del grafo completo.
- Se realiza el mismo procedimiento anterior pero esta vez utilizando los nodos hoja.
- Se realiza la intersección de los dos grafos anteriores para obtener el grafo de exploración completo. Este algoritmo de búsqueda se denomina "**búsqueda bidireccional**".

Así, los parámetros que se pueden configurar en esta operación son los siguientes:

- **Contexto:** contexto obtenido de su correspondiente operación en formato JSON.
- **Nodos raíz:** nodos raíz obtenidos de su correspondiente operación en formato JSON.

- **Profundidad:** profundidad máxima a la que se va a explorar.
- **Ontología:** ontología a utilizar para la exploración (de forma análoga a las operaciones anteriores).
- **Método de pesado de los nodos:** algoritmo a utilizar para realizar el pesado de los nodos.
- **Método de pesado del grafo:** algoritmo a utilizar para realizar el pesado del grafo.
- **Extractor de nodos hoja:** indica el algoritmo que se utiliza para extraer los nodos hoja para realizar la exploración.

5.1.5. Bases de datos

ElasticSearch

ElasticSearch es un motor de búsqueda basado en Lucene que se puede utilizar como base de datos NoSQL. Provee las siguientes funcionalidades:

- **Capacidad de búsquedas de textos completos**. A partir de un texto en lenguaje natural es capaz de identificar las palabras de la búsqueda así como la relevancia de las mismas en el texto.
- **Procesamiento de lenguaje natural**. Proporciona herramientas para ejecutar algunas de las operaciones típicas de procesamiento de lenguaje natural como es la tokenización de palabras, división de frases, normalización, etc.
- **Potencia de consultas**. ElasticSearch proporciona un lenguaje de consultas propio denominado *Query DSL*. Este lenguaje permite realizar consultas como búsqueda por términos, búsqueda por términos priorizando unos campos más que otros, búsquedas a texto completo...

VoltDB

Pese a que es posible utilizar ElasticSearch como un sistema de almacenamiento NoSQL, existen problemas derivados de la inserción de datos en dicho motor de

búsqueda [4]. Por lo tanto, se recomienda utilizar un sistema de almacenamiento secundario que permita, por un lado tener un almacenamiento estable y rápido de los datos y, por otro lado, exportar dichos datos para poder consultarlos en ElasticSearch.

La base de datos elegida para tal fin se denomina **VoltDB**. Esta base de datos pertenece a una nueva generación de sistemas de almacenamiento denominados como *NewSQL* [22], combinando, por un lado, las capacidades ACID de las bases de datos relacionales (así como el lenguaje de consultas SQL) y, por otro lado, la escalabilidad de los sistemas NoSQL. En concreto, las características más relevantes de dicha base de datos son las siguientes [35]:

- **Lenguaje de consultas SQL**: utiliza como lenguaje de consultas SQL de forma prácticamente completa. Esto permite agilizar el desarrollo al evitar el aprendizaje de un lenguaje de consultas específico, como, por ejemplo, MongoDB, o al utilizar lenguajes parecidos a SQL, que no soportan todas las funcionalidades, como, por ejemplo, Cassandra.

- **Escalabilidad horizontal**: es posible añadir nodos a una instalación existente para incrementar su capacidad de almacenamiento.

- **Almacenamiento en memoria**: en lugar de almacenar los datos íntegramente en disco, se almacenan en memoria RAM. Esto permite mayores velocidades de acceso y escritura, con la salvedad de que los datos no serían persistentes. Este problema se soluciona mediante *snapshots* periódicas a disco.

- **Módulo de exportación a ElasticSearch**: permite crear una tabla virtual en la cual los datos insertados en ella se envían automáticamente a ElasticSearch. Esto evita los problemas de escritura relacionados con ElasticSearch (al realizar las comprobaciones de la escritura el propio módulo) y facilita la manipulación de los índices ElasticSearch al ser creados y manipulados automáticamente.

5.1.6. Entornos de desarrollo integrado

Intellij IDEA

Se utiliza este editor [18] para la edición del código Java correspondiente a los servicios web. También se utiliza para la programación del cliente web mediante Javascript.

5.1.7. Lenguajes de programación

Se han utilizado los siguientes lenguajes de programación:

- Java.
- Javascript.
- SQL.
- Shell Script.

5.2. Tecnologías de documentación

5.2.1. Emacs + LaTeX

Emacs [27] es un editor de texto de propósito general altamente extensible mediante *plugins*. Uno de esos plugins se denomina **auctex** [11] y permite convertir el editor en un IDE de desarrollo de LaTeX potente y liviano. También se ha utilizado un modo del editor denominado **flyspell** que permite incorporar corrección de errores ortográficos al vuelo.

5.2.2. Git

Se ha utilizado Git [31] para almacenar la memoria en un repositorio así como para controlar los cambios sobre la misma.

5.2.3. WBSTool

WBSTool [29] es un editor online gratuíto de Estructuras de Descomposición del Trabajo. Permite visualizar el EDT mientras se está realizando, su exportación

a distintos formatos y la edición del mismo de forma visual. Se ha utilizado para generar la EDT del presente proyecto.

5.2.4. Microsoft Project

Microsoft Project [21] es un software de ofimática enfocado en la gestión de proyectos. Sirve para gestionar la calendarización del proyecto mediante la generación de diagramas de Gantt, gestión de recursos, etc. Permitiendo así administrar de forma completa la planificación del proyecto.

5.2.5. Enterprise Architect

Enterpise Architect [28] es un software de modelado UML que permite gestionar todo tipo de diagramas: de clases, secuencia, de arquitectura, etc. Se utiliza para gestionar todos los diagramas concernientes al diseño del proyecto así como para la generación de los diagramas de casos de uso.

5.2.6. Draw.io

Draw.io [3] es una aplicación web de dibujo de propósito general. Se utiliza para representar todos aquellos diagramas que no es posible representar mediante el "Enterprise Architect".

5.2.7. Lumzy

Lumzy [20] es una herramienta para la creación de prototipos navegables de interfaces de usuario.

Capítulo 6

Diseño e Implementación

6.1. Arquitectura del sistema

La arquitectura del sistema permite ver a alto nivel como interacciona el sistema software con los distintos componentes externos al mismo. La figura 6.1 muestra un diagrama que representa la arquitectura de forma simplificada. En dicho diagrama se distinguen dos partes claramente diferenciadas; por una parte, aquellos elementos externos al software a desarrollar y sobre los que no tenemos control (denominados "sistemas externos") y, por otra parte, aquellos elementos desarrollados en el proyecto sobre los que sí se tiene control (denominados "elementos internos"). Así, los dos únicos elementos externos son el anotador semántico ADEGA y la API de Youtube y ambos son accedidos a través del servidor web. Dicho servidor alberga los servicios REST que son consumidos por el cliente y sirve de interfaz para realizar las peticiones a estos servicios externos, teniendo así todo centralizado en el servidor. Los clientes web (representados en el diagrama como interfaces gráficas) únicamente consumen los servicios proporcionados por el servidor.

Por otra parte, los accesos a las bases de datos también se realizan a través del servidor y existe un proceso de sincronización entre la base de datos VoltDB y el motor de búsqueda ElasticSearch mediante los mecanismos de exportación proporcionados por VoltDB. El principal motivo por el que existen dos bases de datos reside, principalmente, en que ElasticSearch tiene varios problemas si se utiliza como almacenamiento de datos primario [4], tal y como se ha comenta-

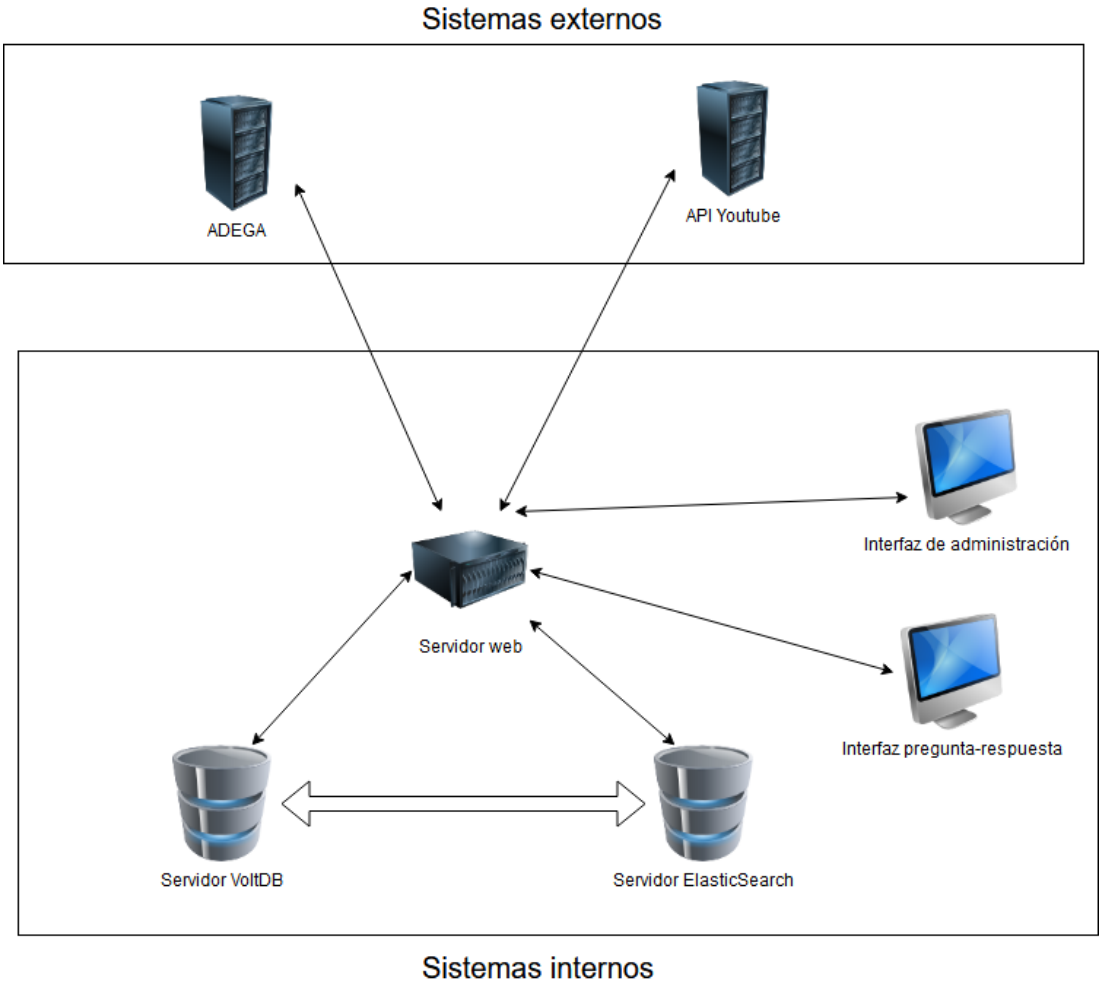

Figura 6.1: Diagrama de la arquitectura del sistema

do en el apartado "Análisis de tecnologías". Por ello, es más seguro exportar a ElasticSearch solo aquellos elementos que constituyen el alcance de la búsqueda y almacenar en VoltDB todo el conjunto de datos.

6.2. Patrones de arquitectura software

Los patrones de arquitectura software ofrecen soluciones a problemas de arquitectura software en el ámbito de la ingeniería del software y expresan un esquema de la organización de la estructura del sistema software. Estos patrones representan un nivel de abstracción mayor que en el caso de los patrones de diseño. De

este modo los patrones de arquitectura aplicados son los siguientes:

- **Cliente-servidor**: es un modelo de software que permite repartir las tareas de un sistema en dos grupos diferenciados. Por un lado, los **clientes** se encargan de realizar peticiones para satisfacer sus objetivos de negocio. Por otro lado, el **servidor** se encarga de recibir las peticiones del cliente y darle respuesta. En el caso del presente proyecto, los clientes son clientes web (una web) y el servidor es una aplicación Java que expone una serie de servicios.

- **Arquitectura orientada a servicios:** este enfoque de arquitectura consiste en exponer todas las funcionalidades de la aplicación como una serie de servicios, es decir, como una serie de actividades cuyo resultado da soporte a una funcionalidad concreta. En el caso del presente proyecto, estos servicios se exponen como una serie de servicios web, de tal manera que para cada funcionalidad de la aplicación existe una dirección web que permite utilizarla. Los servicios únicamente definen una interfaz, es decir, los parámetros que aceptan y el resultado que devuelven, de tal manera que se desacopla la implementación del servicio de su consumo por parte de los clientes.

- **Modelo-Vista-Controlador**: el patrón modelo-vista-controlador propugna la división de una aplicación en tres componentes:

  - **Modelo**: se encarga de representar y gestionar la información del sistema, es decir, gestiona las consultas, actualizaciones, eliminaciones e inserciones de datos en el sistema.

  - **Vista**: muestra los datos del modelo en un formato adecuado para interactuar con el mismo. Normalmente se presenta mediante una interfaz de usuario.

  - **Controlador**: se encarga de responder a eventos e invoca peticiones al modelo para satisfacer las solicitudes de información que se hagan sobre el mismo. También puede enviar comandos a la vista para solicitar que se actualice en función de los cambios que sucedan en el modelo.

Estos tres patrones de arquitectura no aparecen aislados si no que están integrados de forma transparente. En concreto, teniendo en cuenta los patrones anteriores, la arquitectura de la aplicación es la que se muestra en la figura 6.2. Aplicando los patrones antes mencionados, coexisten las siguientes partes diferenciadas:

- **Vista**: constituye la interfaz gráfica de la aplicación. Cada vista se implementa como una página que forma parte de una *Single Page Application*. En este caso, la vista está formada por 7 páginas distintas.
- **Controlador**: constituyen los controladores implementados en AngularJS. Su principal objetivo es llamar a los servicios del modelo, es decir, sirve de enlace entre la vista, proporcionándole los datos obtenidos del modelo, y el modelo, obteniendo los datos mediante llamadas a los servicios REST. En este caso en concreto, cada controlador se comunica directamente con su vista asociada (por ejemplo, el controlador de estadísticas se comunica directamente con la GUI de estadísticas). Además, estos controladores llaman a los servicios que necesiten de la capa de servicios.
- **Modelo**: a su vez, el modelo se compone de:
  - **Capa de servicios**: comprende los distintos servicios REST que posee la aplicación. Estos servicios se agrupan en cuatro grandes bloques: aquellos que sirven de interfaz con el API de Youtube, aquellos que permiten gestionar los vídeos anotados y almacenados en el sistema, aquellos que permiten llamar al sistema pregunta-respuesta y aquellos que permiten gestionar la anotación de nuevos vídeos mediante llamadas al API de ADEGA.
  - **Algoritmos**: incluye los algoritmos utilizados en el sistema pregunta-respuesta, es decir, el algoritmo de búsqueda por palabras y el algoritmo de comparación de grafos conceptuales. Estos algoritmos son llamados por los "servicios de pregunta-respuesta" de la capa de servicios.
  - **Acceso a datos**: el acceso a datos está compuesto por dos capas distintas. Una primera capa implementa la lógica de negocio de la aplicación (accesos al API de Youtube, de ADEGA, administración

de los vídeos anotados y gestión de las colas de anotación). Otra capa implementa los accesos a las distintas bases de datos de la aplicación. Nótese que la sincronización de las inserciones entre la base de datos VoltDB y ElasticSearch se realiza de forma automática, es decir, se gestiona de forma transparente a la aplicación.

6.3. Modelo de datos

Como ya se ha comentado en apartados anteriores, se distinguen dos bases de datos distintas: el motor de búsqueda ElasticSearch y la base de datos *NewSQL* VoltDB. En esta sección se procederá a describir los modelos de datos de ambas bases de datos así como el procedimiento utilizado para realizar la exportación y sincronización.

6.3.1. Diseño del modelo de datos en VoltDB

El modelo entidad relación que representa el almacenamiento en este sistema aparece reflejado en la figura 6.3. Como se puede observar, se distinguen las siguientes tablas:

- **YoutubeVideo**: almacena los metadatos de los vídeos de Youtube. La relación recursiva permite almacenar las etiquetas asociadas a un vídeo sin necesidad de utilizar tipos de datos como "array".
- **ContextNode**: almacena los términos del contexto asociados al vídeo de Youtube en cuestión. Nótese que un mismo nodo puede estar asociado a vídeos distintos por lo que la única restricción que se impone es que el vídeo y el identificador del nodo (un número entero) sean unívocos.
- **RootNodes**: almacena los nodos raíz asociados a los términos del contexto. Dependiendo de la estrategia de obtención que se utilice en ADEGA la relación puede ser 1..1 o 1..*. En este caso, para mayor flexibilidad, se asume que la relación es 1..*. La relación recursiva permite almacenar las relaciones entre nodos de un mismo vídeo. Así, la restricción que se impone es que tanto el identificador del vídeo y el identificador del nodo sean unívocos.

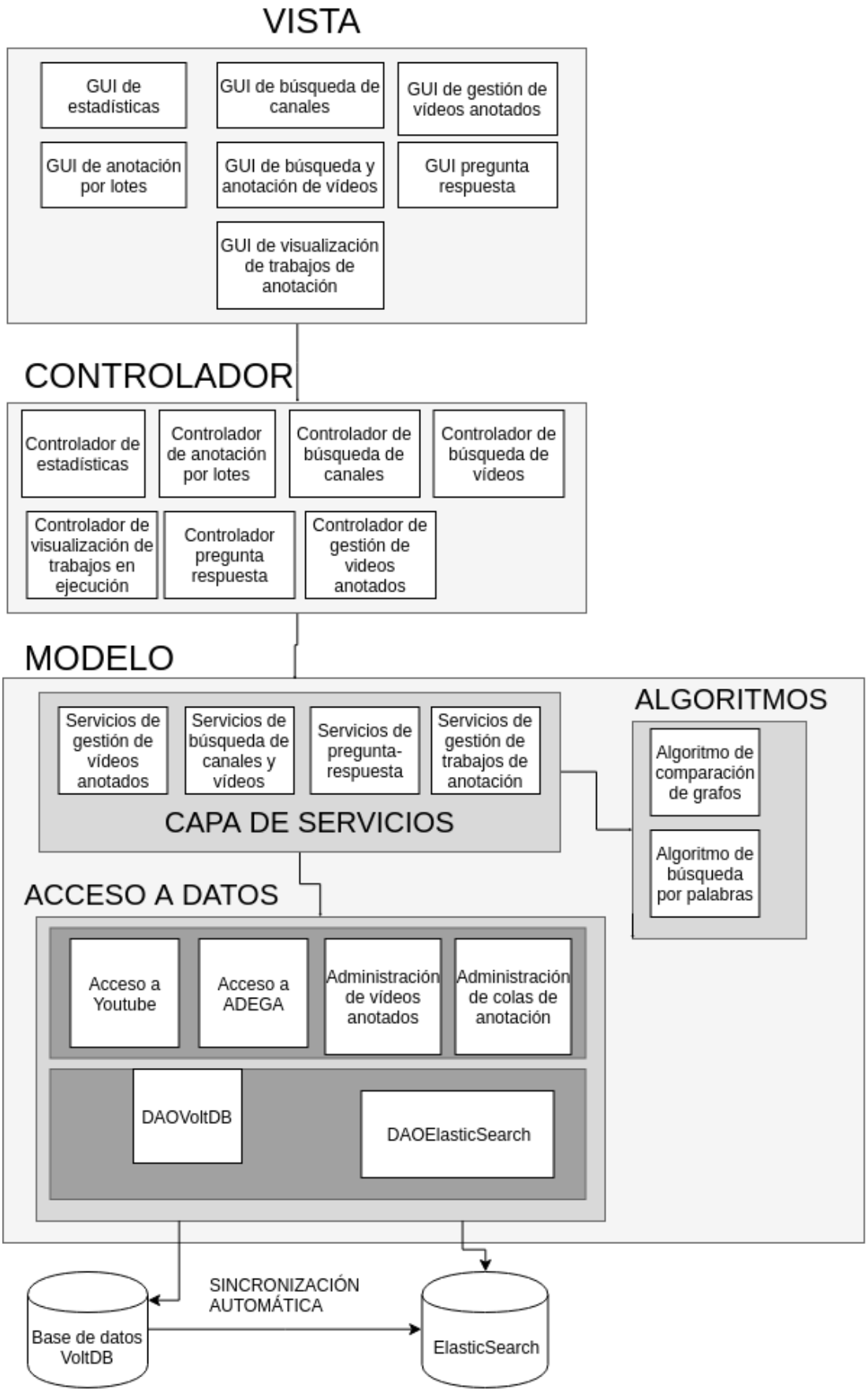

Figura 6.2: Aplicación de los patrones de arquitectura software

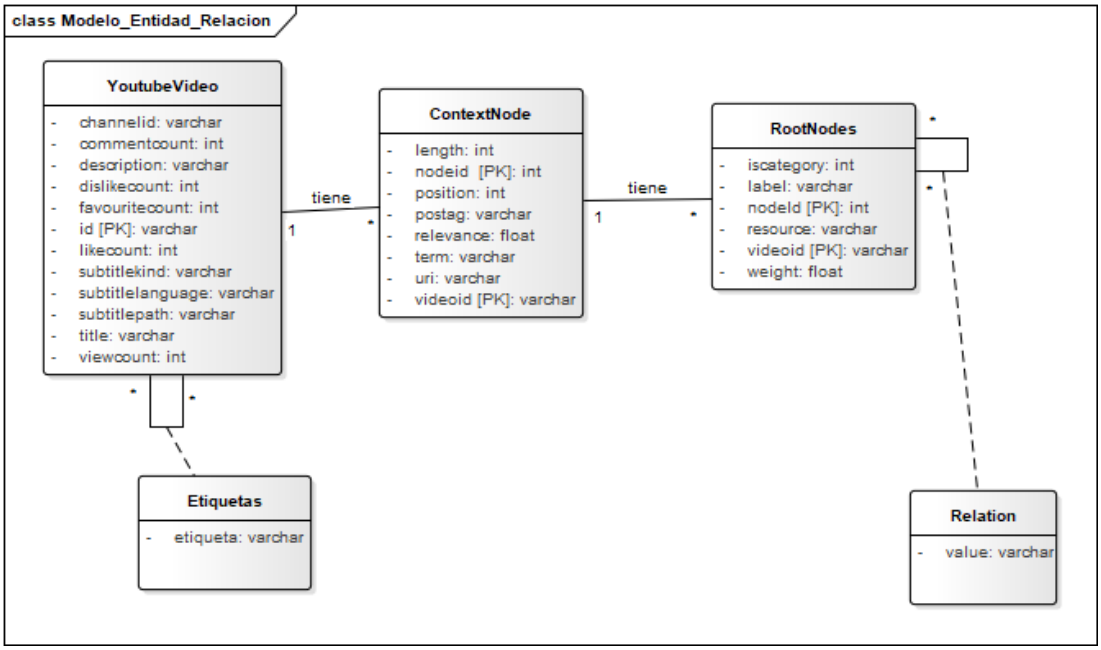

Figura 6.3: Modelo Entidad-Relación de la base de datos VoltDB

6.3.2. Implementación

Implementación del modelo de datos en VoltDB

A la hora de realizar la implementación del modelo de datos de VoltDB uno de los posibles inconvenientes de la base de datos es que no soporta integridad referencial (es decir, no existe la cláusula `foreign key`)[34]. Esto se debe a que la naturaleza distribuida de la base de datos, en la que los datos se almacenan en servidores independientes, hace complicado implementar esta característica desnto de vista de la implementación de la base de datos. Además, este tipo de implementación ralentizaría en gran medida el rendimiento de la misma puesto que la mayoría de estrategias de implementación de integridad referencial asumen que los datos están localmente accesibles, lo cual no es el caso de una base de datos distribuida como VoltDB.

Sin embargo, es posible implementar las claves primarias asociadas a los identificadores de los vídeos ya que esto permite evitar anotaciones repetidas de los vídeos en la base de datos. Esto es especialmente relevante a la hora de actualizar la anotación de un vídeo. Por lo tanto, la implementación del modelo de datos

incluye las siguientes tablas:

- **YoutubeVideo**: almacena los metadatos del vídeo.
- **VideoTags**: almacena las etiquetas de un vídeo.
- **ContextNode**: almacena los nodos del contexto.
- **RootNodes**: almacena los nodos raíz del vídeo.
- **Relations**: almacena las relaciones entre los nodos. Se almacena tanto el vídeo que los relaciona como los identificadores de los nodos involucrados (que son los mismos que los obtenidos en ADEGA). También se almacena el tipo de relación que une a los nodos (persona, lugar de nacimiento, etc.).

Implementación de la exportación en ElasticSearch

ElasticSearch trabaja con una notación similar a las bases de datos relacionales. Para ElasticSearch, un índice tiene la misma consideración que una base de datos. Dentro de un índice, es posible definir distintos "tipos" que, utilizando el simil con una base de datos relacional, se corresponderían con una tabla. Dentro de un tipo existen documentos que, de nuevo, se corresponderían con las distintas filas de una tabla en una base de datos relacional. A diferencia de la mayoría de las bases de datos NoSQL, la definición de un tipo es estática, es decir, una vez definido un tipo no es posible que dos documentos tengan campos distintos. Por lo tanto, aquellos campos que no existan deberán estar a NULL.

Para simplificar la implementación, solo se exportarán los nodos relativos a los nodos raíz y los títulos de los vídeos. Los primeros son necesarios para poder obtener los grafos a la hora de implementar el sistema. Los segundos son necesarios para obtener los vídeos a la hora de administrar la base de datos de vídeos (en la que se realiza una búsqueda por título). Para habilitar la exportación desde VoltDB, es preciso modificar el XML de configuración para añadir la url de acceso de ElasticSearch (*endpoint*). Dicha configuración se modifica en el fichero "`deployment.xml`" de la raíz de la base de datos. Es preciso agregar el siguiente fragmento de XML:

```
<export>
        <configuration target="elasticsearch" enabled="true" type="
    elasticsearch">
            <property name="endpoint">
        http://10.10.0.160:9200/voltdb/adega
      </property>
        </configuration>
</export>
```

En el anterior fragmento de código, especificamos el índice ElasticSearch (voltdb) y el tipo de documento (ADEGA). En caso de que no exista cualquiera de los dos elementos anteriores, este será creado automáticamente por VoltDB. Asimismo, también especificamos el nombre de la exportación, necesario para crear la tabla virtual de inserción. Para crear la tabla virtual de inserción se utiliza el siguiente código[1]:

```
CREATE STREAM ROOTNODES_EXPORTED EXPORT TO TARGET elasticsearch (
    VIDEOID varchar(2048) NOT NULL,
    VIDEOTITLE varchar(2048) NOT NULL,
    NODEID integer NOT NULL,
    LABEL varchar(2048),
    RESOURCE varchar(2048),
    WEIGHT float,
    ISCATEGORY integer
);
```

Así, cada documento del índice está compuesto por el identificador del vídeo y uno de los nodos asociados al grafo obtenido del vídeo.

Implementación de la persistencia en VoltDB

Como se ha comentado en apartados anteriores, VoltDB es una base de datos fundamentalmente en memoria, por lo que en caso de que la base de datos se

[1]El nombre "elasticsearch" se corresponde con el identificador definido en el XML

apague los datos se perderían. Sin embargo, es posible conseguir persistencia mediante varias vías [33]:

- ***Snapshots***: mediante esta característica, se tiene un volcado completo de la base de datos en un momento temporal. Estos volcados se pueden programar para ocurrir a ciertos intervalos.

- ***Command Logging***: mediante esta característica, se tiene un log de las operaciones de cada transacción. Cuando la base de datos se apaga, se recuperan los datos primero del volcado de la base de datos y, posteriormente, del registro de operaciones hasta alcanzar un estado consistente.

- ***K-safety***: en un sistema distribuido, se refiere al número de duplicaciones de las particiones de una base de datos (siendo una partición un fragmento de los datos totales almacenados en la base de datos), de tal manera que la pérdida de un servidor no ocasione una interrupción del servicio. Así, por ejemplo, un valor de *K-safety* de 2 indica que existen dos copias de una partición distribuidas en el sistema.

- ***Database replication***: es similar a *K-safety* solo que, en lugar de copiar una partición, se copia la base de datos entera. Esta funcionalidad está pensada para tener copias de una misma base de datos en lugares geográficos distintos.

Desafortunadamente, las características *Command Logging* y *K-safety* no están disponible en la versión gratuíta de VoltDB (Community). Además, no tiene ningún sentido realizar replicación de la base de datos en un sistema que no es distribuido (tal y como se indicó en las restricciones del proyecto). Por lo tanto, la única estrategia a utilizar es la utilización de *Snapshots*. Configurar esta característica es muy sencillo: basta con agregar la siguiente línea al fichero `deployment.xml`:

```
1 <snapshot frequency="10m" retain="4" prefix="adegaVideoDB"/>
```

La anterior línea permite configurar *Snapshots* con una periodicidad de 10 minutos. Para evitar que las copias llenen el disco duro, instruímos a VoltDB para que únicamente conserve las 4 más recientes. Dichas copias estarán prefijadas con el nombre "adegaVideoDB".

6.4. Servidor Web *(backend)*

6.4.1. Diagrama de paquetes

El servidor web es el elemento del sistema más complejo. Por ello, es preciso una organización de las clases del código en paquetes que faciliten la programación y permitan una mayor extensibilidad en un futuro. Dichos paquetes no se corresponden con un módulo en concreto del sistema, sino que agrupan clases con funcionalidades comunes. Así, el diagrama de paquetes que representa al servidor web es el que se muestra en la figura 6.4. Se distingue la siguiente división en paquetes:

- **Controller**: contiene todos los controladores Spring, es decir, todos los métodos Java que se traducen a servicios REST.
- **QuestionAnswering**: contiene todas las clases relativas a la implementación del sistema pregunta-respuesta.
- **Youtube**: contiene todas las clases que permiten acceder al API de Youtube y obtener los datos necesarios de la misma.
- **Management**: contiene todas las clases que permiten administrar los vídeos del sistema.
- **DAO**: contiene todos las clases DAO que permiten acceder a las distintas bases de datos. También incluye la factoría abstracta que permite instanciar los DAOs.
- **JobQueue**: contiene las clases relativas a crear y manejar la cola de procesamiento de anotaciones de vídeos.
- **Adega**: contiene todas aquellas clases que permiten interaccionar con ADEGA.

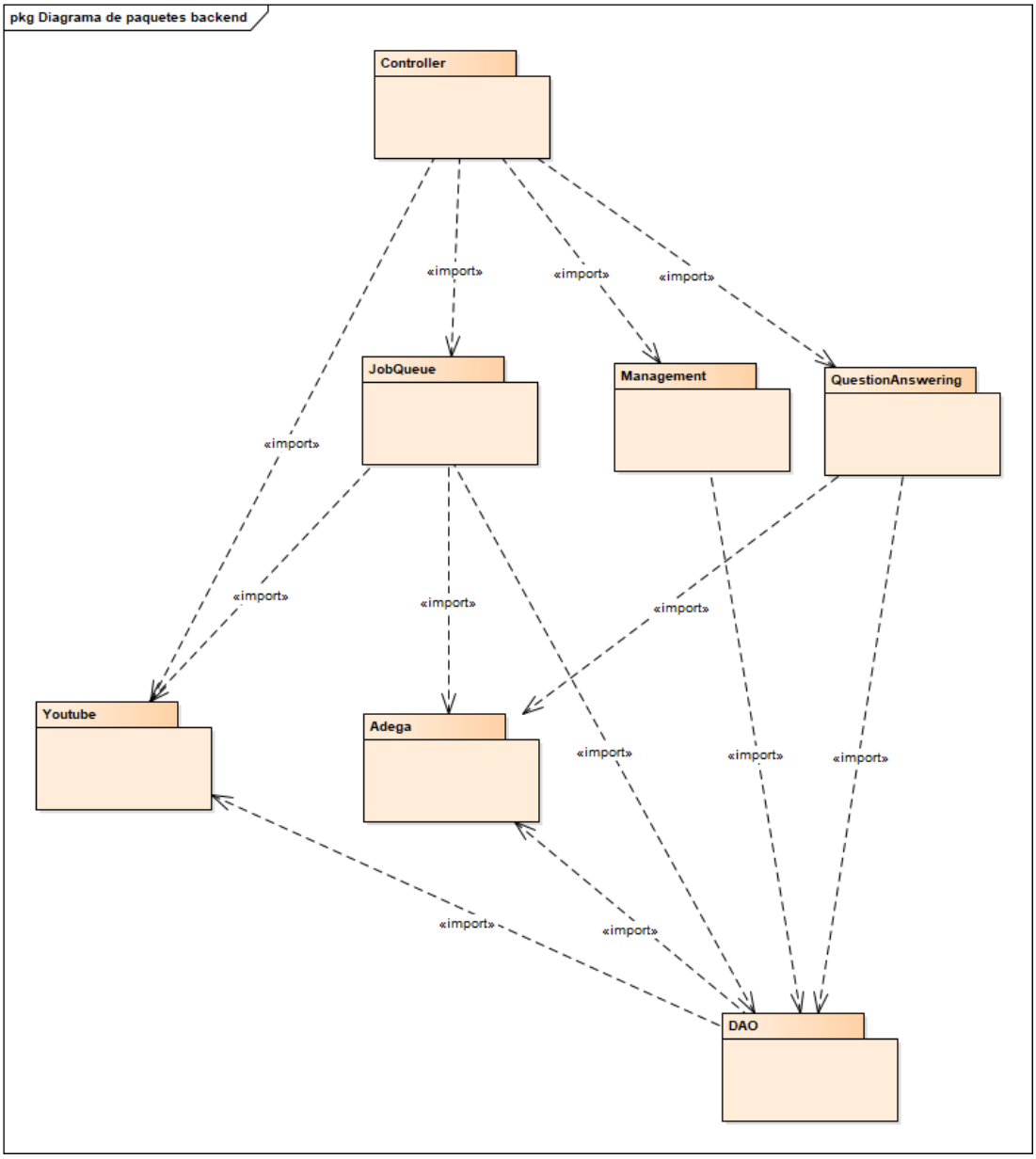

Figura 6.4: Diagrama de paquetes del *backend*

Las dependencias entre paquetes se derivan de los detalles de implementación de las funcionalidades. En concreto son salientables las dependencias siguientes:

- Se puede observar que el propósito del controlador es delegar la responsabilidad de las llamadas a los servicios al paquete que se encargue de ello. Los servicios que únicamente consultan a Youtube (búsqueda de canales y vídeos del canal) se redirigen directamente al paquete “Youtube”.

- El paquete "JobQueue" necesita del paquete Youtube. Esto se debe a que, lo único que se necesita para encolar un vídeo es el identificador del mismo. Esto permite utilizar el mismo método tanto para procesar los vídeos seleccionados por un usuario como procesar los vídeos en lote (que simplemente son cadenas de texto).
- El paquete "QuestionAnswering" precisa del paquete "Adega" debido a que se utiliza en las primeras fases para procesar el lenguaje natural del usuario.

A continuación se procederán a describir los patrones de diseño utilizados y el contenido de los paquetes anteriormente descritos.

6.4.2. Patrones de diseño software

Un patrón de diseño es una técnica utilizada para resolver problemas comunes de diseño de software. Los patrones aquí listados hacen referencia a los patrones GoF (*Gang Of Four*) creados por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides [8]. En concreto, los patrones utilizados en este proyecto son los siguientes:

- **DAO/DTO**: no encajan directamente en la categoría de patrones de diseño puesto que son una forma de estructurar los accesos a los datos. Por una parte un DAO es un objeto que suministra una interfaz común entre la aplicación y los accesos a una fuente de datos (habitualmente una base de datos). Por otra parte, DTO constituye un objeto que permite transportar datos entre distintas partes de la aplicación sin poseer en sí mismo nada más que capacidad para almacenar datos. Así, un DAO realiza consultas a la fuente de datos devolviendo DTOs como resultado. Estos dos tipos de objeto son la base para el patrón *Abstract Factory*.
- **Abstract Factory**: permite controlar la instanciación de clases. En concreto, se utiliza para controlar la creación de los objetos DAO de tal manera que sea posible modificar la base de datos que se esté utilizando de forma sencilla. Así, añadir una nueva base de datos al sistema implica únicamente implementar las interfaces correspondientes al DAO y modificar la factoría para que tenga en cuenta la nueva base de datos a la hora de obtener los correspondientes DAOs.

- **Singleton**: este patrón de diseño permite restringir la instanciación de objetos de una clase. En concreto, se utiliza este patrón para que haya un único objeto DAO instanciado a lo largo del programa, es decir, todos los accesos a los datos pasan por un único objeto. Este patrón se utiliza en combinación con el patrón **Abstract Factory** de tal manera que la factoría controla la instanciación de objetos asegurándose que solo hay un DAO instanciado a la vez.

- **Strategy**: permite mantener un conjunto de algoritmos de tal manera que se pueda elegir el algoritmo más conveniente para cada ocasión. Se utiliza en el sistema pregunta-respuesta para dividir los algoritmos de búsqueda por palabras y comparación de grafos conceptuales en algoritmos distintos, permitiendo así el uso de uno u otro de forma independiente.

- **Facade**: permite reducir la complejidad de un subsistema mediante una interfaz simple a un sistema complejo. Se utiliza en el sistema de anotación para abstraer la complejidad del API de Youtube mediante la ocultación de los tipos devueltos por el API de Youtube así como mediante una simplificación de las operaciones que se pueden realizar a través del API con la definición de métodos claros y sencillos.

6.4.3. Diagramas de clases

Un diagrama de clases es una estructura que permite mostrar las distintas clases que conforman un sistema software así como sus relaciones, atributos y métodos. Los diagramas de clases aquí mostrados no pretenden ser una descripción exhaustiva de las clases del sistema si no que solo muestran los aspectos más importantes de las clases que conforman los distintos paquetes así como las relaciones entre las clases de un mismo paquete.

Paquete "Youtube"

Como ya se ha comentado anteriormente, este paquete se encarga de acceder a la API de Youtube para obtener la información relevante sobre los vídeos y canales. La estructura del paquete se detalla en el diagrama de clases de la figura 6.5.

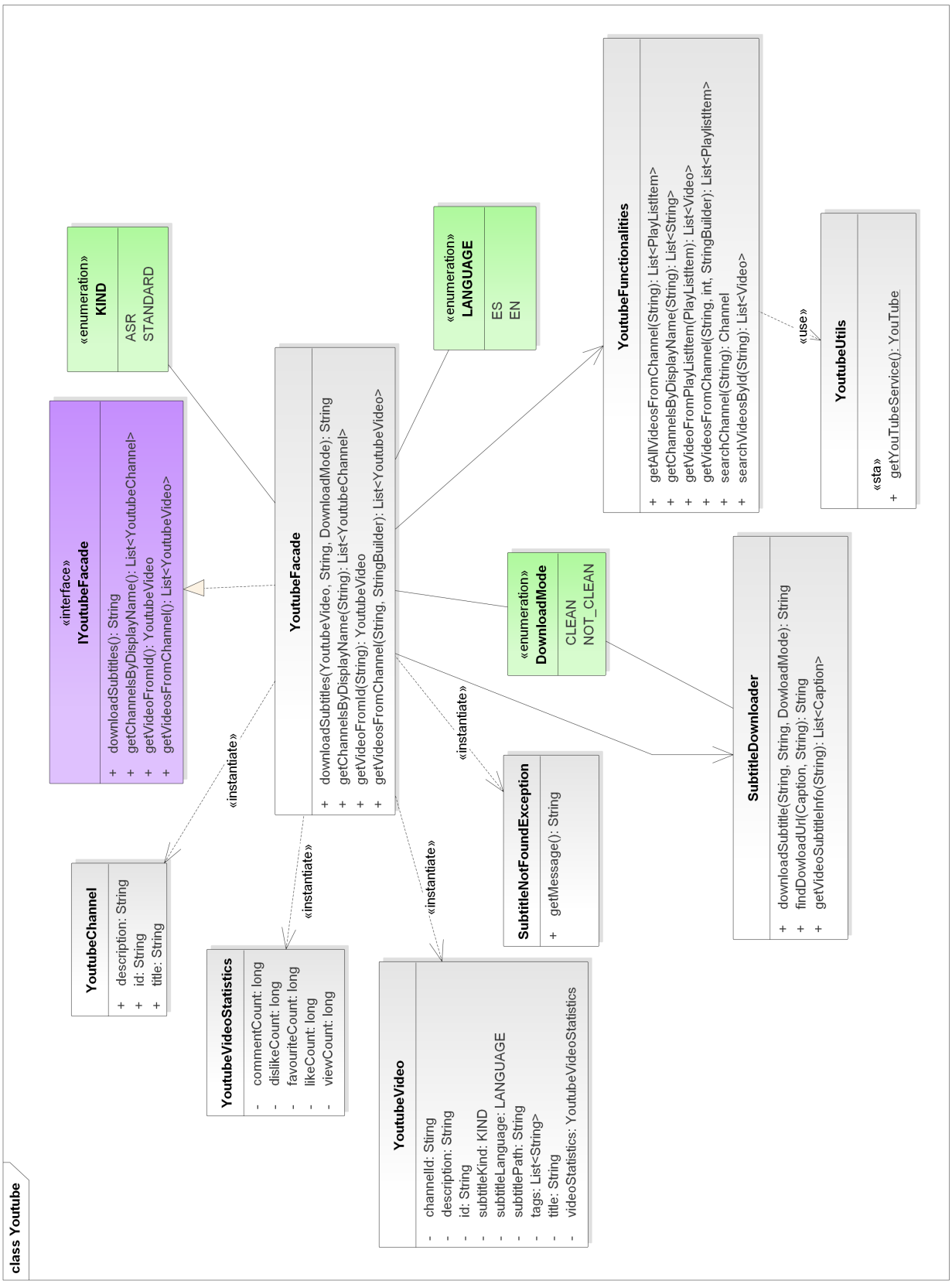

Figura 6.5: Diagrama de clases del paquete Youtube.

En primer lugar, existen tres clases (`YoutubeVideo`, `YoutubeVideoStatistics` y `YoutubeChannel`) que son simplemente contenedores de datos de los vídeos de Youtube, las estadísticas de cada vídeo y los canales de Youtube respectivamente. Se utilizan estas clases en lugar de las proporcionadas por el API de Youtube para simplificar la transmisión de datos entre el cliente web y el servidor. Así, no se transmiten datos que no sean necesarios para el cliente ni para el servidor web.

Como se puede observar, existe una **fachada** (*YoutubeFacade*) que permite abstraer la complejidad de utilización de la API de Youtube mediante la ejecución de métodos que no devuelvan ni reciban como parámetro clases internas del API. Dicha clase tiene como métodos las funcionalidades principales del paquete que son las siguientes:

- **downloadSubtitles**: Se encarga de descargar los subtitulos. Requiere de los datos del vídeo (para saber que lenguaje debe descargarse y de que vídeo), el nombre del fichero donde se guardarán los subtítulos y, por último, el modo de descarga de los subtítulos. Este último parámetro controla si los subtitulos se descargan de forma "limpia" o "no limpia", es decir, si se purgan los elementos que no sean puramente subtítulos textuales o no (puesto que los subtítulos se descargan en formato XML y es preciso eliminar las etiquetas).

- **getChannelsByDisplayName**: Obtiene una lista de canales según el nombre que se muestre al público. Estos nombres pueden no ser unívocos por lo que se devuelve una lista de canales en función de la aproximación de la consulta.

- **getVideoFromId**: a partir de un identificador de un vídeo (que es posible obtener directamente desde la URL del mismo) se obtienen los datos relacionados con ese vídeo.

- **getVideosFromChannel**: obtiene los vídeos de un canal. Recibe como parámetro el identificador del canal y el token del canal. Dicho token permite obtener la lista de vídeos del canal de forma paginada, es decir, en fragmentos de N vídeos de cada vez (en lugar de todos a la vez, que podría ser una operación que tarde mucho tiempo). Se utiliza la clase Java String-

Builder en lugar de la clase String para permitir simular un paso por referencia del token: cada vez que se pida una nueva lista de vídeos con un determinado token, se devuelve el siguiente token a través de este parámetro del método (utilizando los métodos internos para modificar la cadena que contiene el StringBuilder). Esto permite sortear la limitación de Java de devolver un único elemento en una función y tener que crear una clase que albergue tanto la lista de vídeos como el siguiente token de cada vez.

Existe una clase que se encarga específicamente de facilitar las operaciones necesarias para la descarga de los subtítulos (**SubtitleDownloader**). Dicha clase está compuesta de los siguientes métodos:

- **getSubtitleInfo**: consulta el API de Youtube sobre los subtítulos disponibles para un vídeo en concreto. Es preciso mencionar que el API de Youtube no permite descargar cualquier subtítulo: solo permite descargar aquellos que han sido creados por un humano y no aquellos que son autogenerados por el sistema de generación de subtítulos de Youtube.

- **findDownloadUrl**: Este método se encarga de obtener la URL interna de descarga de los subtítulos. El código correspondiente a este método se presenta en el siguiente extracto de código:

```
String magicUrl = v.retrieveMagicURL(videoUrl);
magicUrl += "&";
magicUrl += "name=" + captionInfo.getSnippet().getName() + "&";
magicUrl += "lang=" + captionInfo.getSnippet().getLanguage() +
    "&";
magicUrl += "type=track&";
String kind = captionInfo.getSnippet().getTrackKind().
    toLowerCase();
if(!kind.equals("standard"))
    magicUrl += "kind=" + captionInfo.getSnippet().getTrackKind
    ().toLowerCase();

return magicUrl;
```

En primer lugar, se llama a la API para obtener la URL base de descarga de los subtítulos. Dicha URL se obtiene inspeccionando el código HTML de la página en la que está albergado el vídeo (puesto que la URL debe estar firmada). A partir de esa URL base, se configuran el resto de parámetros: el nombre del subtiulo, el idioma y el tipo de subtítulo (este último elemento solo se añade si el subtítulo es autogenerado).

- **downloadSubtitle** A partir de la URL generada en el método **findDownloadUrl** se descarga el fichero XML asociado a dicha URL y, si se elige el modo de limpieza, se eliminan todas las etiquetas XML del fichero (puesto que el texto de los subtítulos aparece como nodos hoja del XML). Dicha información se almacena en un fichero para su posterior anotación.

La clase **YoutubeFunctionalities** permite, tanto realizar las operaciones definidas en la fachada, como realizar las conversiones entre los distintos tipos de elementos que devuelve el API en cuestión. Como se están realizando llamadas directas al API, es preciso obtener el objeto que permite realizar dichas llamadas. Dicho objeto se obtiene mediante el método estático `getYouTubeService` de la clase **YoutubeUtils**.

Es necesario destacar que, pese a que el método de autenticación recomendado por Google es OAUth 2.0, este método require autorización explícita del usuario al utilizar el API por primera vez (requiere pulsar un botón en una interfaz). Por lo tanto, se utiliza el método de clave API (*API key*), que, a partir de una clave de API obtenida del panel de administración de `Google Developers Console` permite autenticar todas las llamadas al API.

Paquete "DAO"

Este paquete tiene la responsabilidad de centralizar los accesos a las bases de datos. El diagrama de clases de este paquete aparece reflejado en la figura 6.6. Se ha implementado un patrón *"Abstract Factory"* que permite controlar la instanciación de los DAO en función de la base de datos que se desee utilizar. En concreto, se distinguen las siguientes clases:

- **DAOFactory**: es la factoría propiamente dicha. Controla la instanciación

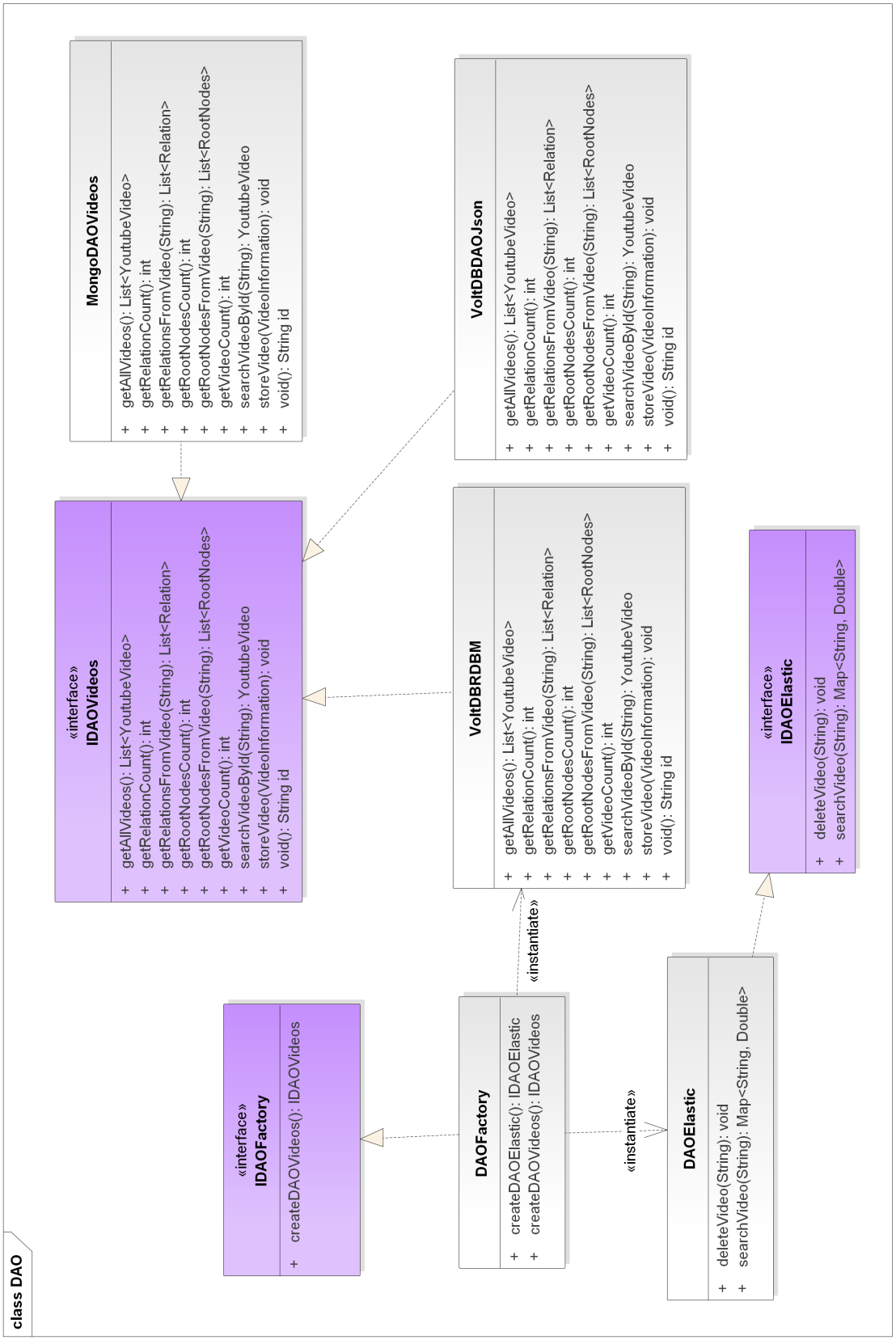

Figura 6.6: Diagrama de clases del paquete DAO

de los DAO de la base de datos de almacenamiento persistente (en este caso VoltDB) y la instanciación del DAO de ElasticSearch. Este último DAO posee una interfaz genérica que permite intercambiar la implementación de ElasticSearch por otra como *"Apache Solr"* o *"Lucene"* directamente.

- **VoltDBRDBM**, **VoltDBDAOJSON**, **MongoDAOVideos**: se corresponden con las distintas implementaciones de bases de datos para el almacenamiento de los datos de los vídeos. En concreto:

  - **MongoDAOVideos**: se corresponde con la implementación en MongoDB del almacenamiento de los datos. Pese a que MongoDB es una base de datos NoSQL escalable para este problema y que los datos de los grafos de los vídeos se obtienen directamente en formato JSON, no se ha utilizado ya que no existen exportadores fiables de MongoDB a ElasticSearch e implementar uno es inviable por lo comentado en apartados anteriores.
  - **VoltDBDAOJSON**: se corresponde con la implementación en VoltDB del almacenamiento de los datos como JSONs directamente. Se ha descartado por ser poco eficiente.
  - **VoltDBRDBM**: se corresponde con la implementación en VoltDB del modelo relacional descrito en apartados anteriores de este mismo capítulo. Es la implementación que utiliza el sistema en la actualidad.

Paquete "ADEGA"

El diagrama de clases para este paquete se muestra en la figura 6.7. Este paquete centraliza las operaciones que se pueden hacer mediante el API ADEGA. Como se puede observar, contiene dos subpaquetes:

- **ADEGA_CONSTANTS**: contiene constantes requisito de las llamadas al API de ADEGA. Permite definir, por ejemplo, la ontología a utilizar.

- **ADEGA_CONTAINERS**: contiene clases contenedoras de datos que devuelve ADEGA.

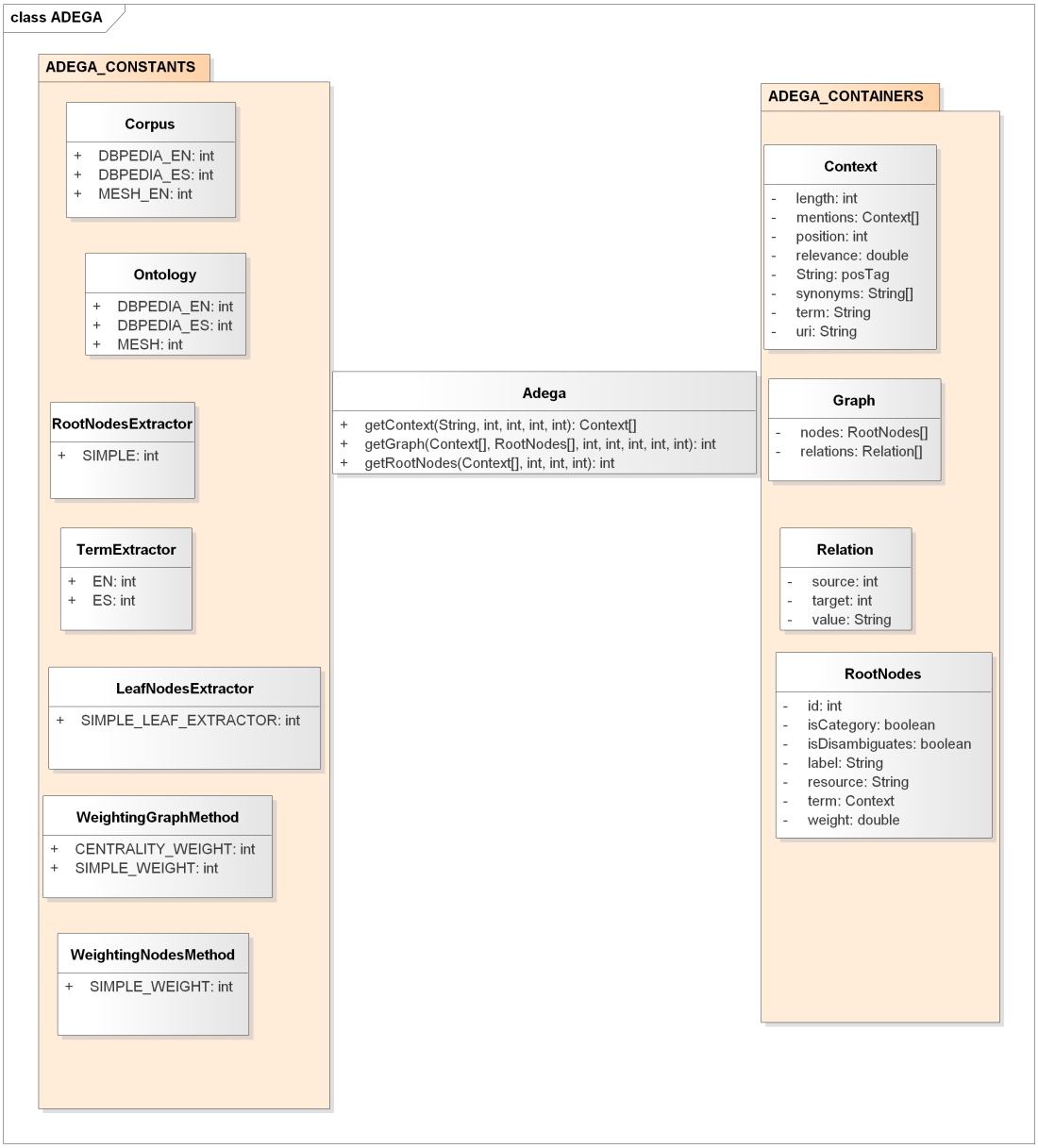

Figura 6.7: Diagrama de clases del paquete “ADEGA”

Adicionalmente, se incluye la clase **AdegaHandler** que permite realizar las llamadas a los servicios de ADEGA mediante métodos distintos. En concreto, permite realizar las llamadas a los servicios de obtención del contexto, nodos raíz y grafo respectivamente.

Paquete "JobQueue"

Este paquete contiene las clases que permiten gestionar la cola de anotaciones. El diagrama de clases de este paquete aparece reflejado en la figura 6.8. En concreto, se distinguen las siguientes clases:

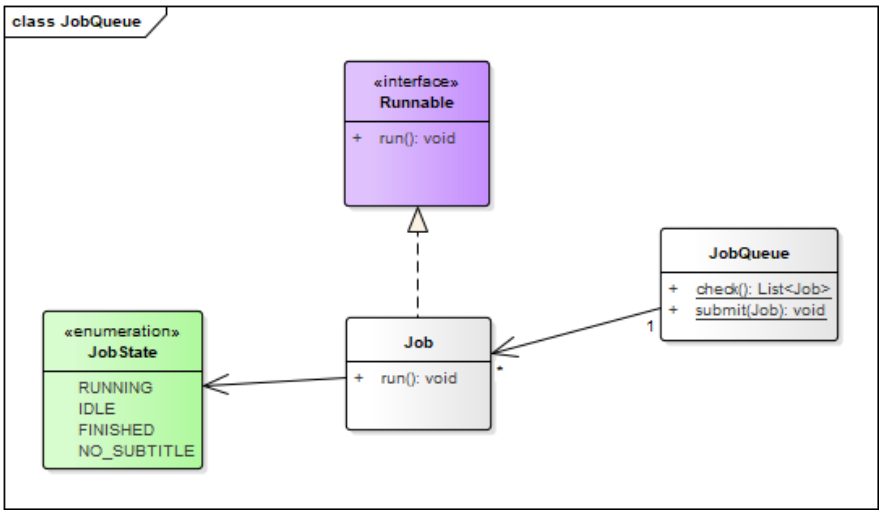

Figura 6.8: Diagrama de clases del paquete "JobQueue"

- **JobState**: esta enumeración permite representar los posibles estados que puede estar un trabajo: en ejecución, en espera, finalizado o sin subtítulos.
- **Job**: esta clase incluye la implementación de la anotación mediante trabajos encolados. La anotación es multihilo, de ahí que sea necesario la implementación de la interfaz `Runnable`. Dentro del método run, se modifica el estado del trabajo en función de si está en ejecución o no.
- **JobQueue**: esta clase contiene la cola de ejecución en sí misma. La cola se implementa mediante una clase java denominada como `ThreadPoolExecutor`. Dicha clase permite ejecutar como máximo un número de hilos predeterminado (en la implementación son tres). El resto de hilos son mantenidos en espera.

Paquete "Management"

El diagrama de clases de este paquete se muestra en la figura 6.9. Como se puede observar, este paquete únicamente una clase (`Management`) con la implementación

de algunos métodos necesarios para los servicios de administración de vídeos. Los métodos que contiene dicha clase son los siguientes:

- **getGraph(String videoId)**: devuelve un grafo de un vídeo en concreto.
- **delete(String videoList)**: elimina un vídeo de las bases de datos.
- **update(String videoList)**: actualiza un vídeo de las bases de datos.
- **getAllVideos(String userQuery)**: obtiene los vídeos de la base de datos que se correspondan con la búsqueda del usuario. Se corresponde con la búsqueda de vídeos para su administración.

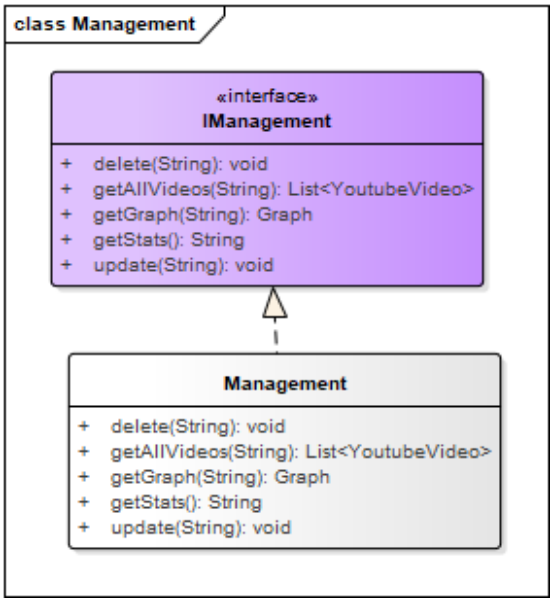

Figura 6.9: Diagrama de clases del paquete Management

Paquete "QuestionAnswering"

El diagrama de clases que representa este paquete se muestra en la figura 6.10. La clase UserQueryProcessor se encarga de realizar la llamada a ADEGA para obtener el grafo de la consulta del usuario. Después, delega la obtención de los vídeos en el algoritmo o algoritmos que corresponda enviando tanto el grafo de

la consulta, los términos asociados a la consulta (que realmente se obtienen del grafo, pero así es más sencillo) y el histórico del paciente[2].

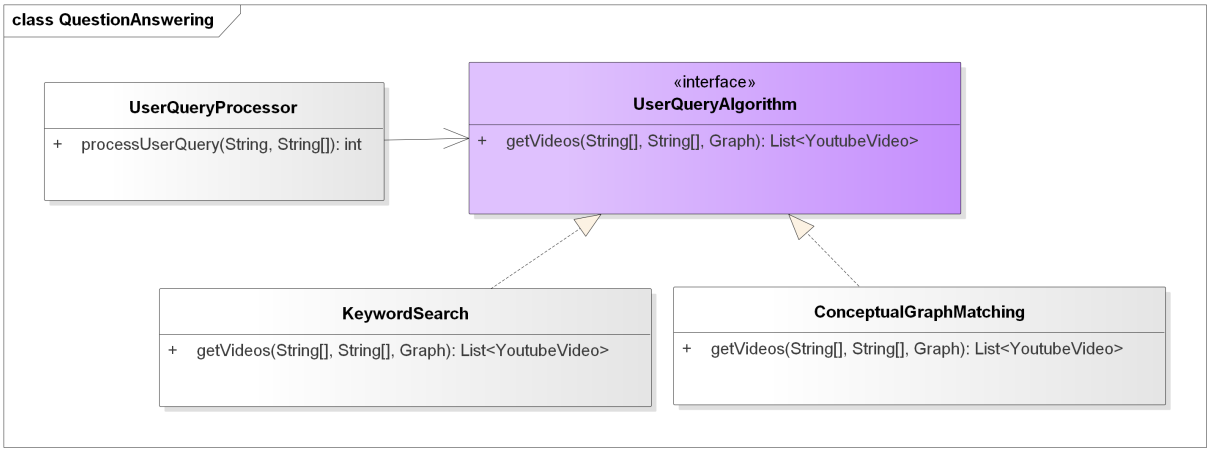

Figura 6.10: Diagrama de clases del paquete "QuestionAnswering"

Paquete "Controller"

El diagrama de clases para este paquete se muestra en la figura 6.11. Este paquete contiene la implementación de todos los servicios REST de la aplicación. Como se puede observar en el diagrama de paquetes, simplemente delega la responsabilidad de resolver la llamada en el paquete correspondiente al servicio. Los controladores existentes en la aplicación y su funcionalidad son los siguientes:

- **VideoController:** controlador asociado a la recuperación de vídeos y canales. Así, este controlador invoca a los siguientes servicios REST:

Controlador:	VideoController
Servicio:	[GET]/searchChannels
Descripción:	Busca los canales de Youtube que se correspondan con la consulta del usuario
Parámetros:	
`displayName`: consulta del usuario a buscar	
Respuesta:	

[2]En el caso del estado actual del proyecto no se tiene en cuenta. Se añade con previsión de una posible ampliación.

`200 (OK)`: código de respuesta estándar HTTP que indica que la operación se ha llevado a cabo con éxito. se devuelve un resultado en formato json `500 (INTERNAL SERVER ERROR)`: código de respuesta estándar HTTP que indica que ha habido un fallo al procesar la petición.

Controlador:	VideoController
Servicio:	[GET]/searchVideos
Descripción:	Busca los vídeos asociados a un canal.
Parámetros:	
`channelID`: identificador del canal. `token`: token que permite paginar la recuperación de los vídeos	
Respuesta:	
`200 (OK)`: código de respuesta estándar HTTP que indica que la operación se ha llevado a cabo con éxito. se devuelve un resultado en formato json `500 (INTERNAL SERVER ERROR)`: código de respuesta estándar HTTP que indica que ha habido un fallo al procesar la petición.	

- **AnotationController:** controlador asociado a la anotación de vídeos, es decir, a la creación de trabajos en cola. Así, este controlador invoca a los siguientes servicios REST:

Controlador:	AnotationController
Servicio:	[POST]/anotate
Descripción:	Añade tantos trabajos en cola como vídeos se le pasen como parámetro.
Parámetros:	
`listVideo`: lista de vídeos a anotar separados por una coma cada uno.	
Respuesta:	
`200 (OK)`: código de respuesta estándar HTTP que indica que la operación se ha llevado a cabo con éxito. se devuelve un resultado en formato json `500 (INTERNAL SERVER ERROR)`: código de respuesta estándar HTTP que indica que ha habido un fallo al procesar la petición.	

Controlador:	AnotationController
Servicio:	[GET]/getJobs
Descripción:	Obtén la lista de trabajos actuales del sistema junto con el estado de los mismos. No tiene parámetros.
Respuesta:	
`200 (OK)`: código de respuesta estándar HTTP que indica que la operación se ha llevado a cabo con éxito. se devuelve un resultado en formato json `500 (INTERNAL SERVER ERROR)`: código de respuesta estándar HTTP que indica que ha habido un fallo al procesar la petición.	

- **ManagementController:** controlador asociado a la administración de los vídeos anotados en el sistema. Así, este controlador invoca a los siguientes servicios REST:

Controlador:	ManagementController
Servicio:	[GET]/getVideoCount
Descripción:	Permite obtener estadísticas de los vídeos almacenados en la base de datos. No tiene parámetros.
Respuesta:	
`200 (OK)`: código de respuesta estándar HTTP que indica que la operación se ha llevado a cabo con éxito. se devuelve un resultado en formato json `500 (INTERNAL SERVER ERROR)`: código de respuesta estándar HTTP que indica que ha habido un fallo al procesar la petición.	

Controlador:	ManagementController
Servicio:	[GET]/getAllVideos
Descripción:	Obtén todos los vídeos de la base de datos relacionados con la consulta del usuario.
Parámetros:	
`query`: consulta del usuario para obtener los vídeos de la base de datos	
Respuesta:	
`200 (OK)`: código de respuesta estándar HTTP que indica que la operación se ha llevado a cabo con éxito. se devuelve un resultado en formato json	

`500 (INTERNAL SERVER ERROR)`: código de respuesta estándar HTTP que indica que ha habido un fallo al procesar la petición.

Controlador:	ManagementController
Servicio:	[GET]/delete
Descripción:	Elimina un conjunto de vídeos del sistema.
Parámetros:	
`videoList`: lista de identificadores de los vídeos a eliminar del sistema (separados por una coma cada uno).	
Respuesta:	
`200 (OK)`: código de respuesta estándar HTTP que indica que la operación se ha llevado a cabo con éxito. se devuelve un resultado en formato json `500 (INTERNAL SERVER ERROR)`: código de respuesta estándar HTTP que indica que ha habido un fallo al procesar la petición.	

Controlador:	ManagementController
Servicio:	[GET]/update
Descripción:	Actualiza un conjunto de vídeos del sistema.
Parámetros:	
`videoList`: lista de identificadores de los vídeos a actualizar del sistema (separados por una coma cada uno.	
Respuesta:	
`200 (OK)`: código de respuesta estándar HTTP que indica que la operación se ha llevado a cabo con éxito. se devuelve un resultado en formato json `500 (INTERNAL SERVER ERROR)`: código de respuesta estándar HTTP que indica que ha habido un fallo al procesar la petición.	

Controlador:	ManagementController
Servicio:	[GET]/getGraph
Descripción:	Obtiene un grafo asociado a un vídeo en concreto.
Parámetros:	
`videoId`: identificador del vídeo del cual se debe recuperar el grafo.	
Respuesta:	

`200 (OK)`: código de respuesta estándar HTTP que indica que la operación se ha llevado a cabo con éxito. se devuelve un resultado en formato json `500 (INTERNAL SERVER ERROR)`: código de respuesta estándar HTTP que indica que ha habido un fallo al procesar la petición.

- **UserQueryController:** controlador encargado de procesar las consultas del usuario del sistema pregunta-respuesta. Así, este controlador invoca a los siguientes servicios REST:

Controlador:	UserQueryController
Servicio:	[POST]/processQuery
Descripción:	A partir de una consulta en lenguaje natural del usuario se obtiene una lista de vídeo relevantes. Se diferencia del servicio de obtención de vídeos de "ManagementController" en que la búsqueda que se realiza aquí es mucho más precisa (puesto que utiliza el algoritmo pregunta-respuesta).
Parámetros:	
`query`: consulta en lenguaje natural del paciente	
Respuesta:	
`200 (OK)`: código de respuesta estándar HTTP que indica que la operación se ha llevado a cabo con éxito. se devuelve un resultado en formato json `500 (INTERNAL SERVER ERROR)`: código de respuesta estándar HTTP que indica que ha habido un fallo al procesar la petición.	

6.5. Diagramas de secuencia

Los diagramas de secuencia permiten mostrar las interacciones entre las distintas instancias de clases y los actores. Se omiten aquellos diagramas de secuencia que sean demasiado sencillos como para aportar información relevante. En concreto, se omiten los diagramas de secuencia de los casos de uso **CU. 7** ("Ver metadatos"), **CU. 8** ("Ver subtítulos"), **CU. 12** ("Mostrar grafo"), **CU. 13**

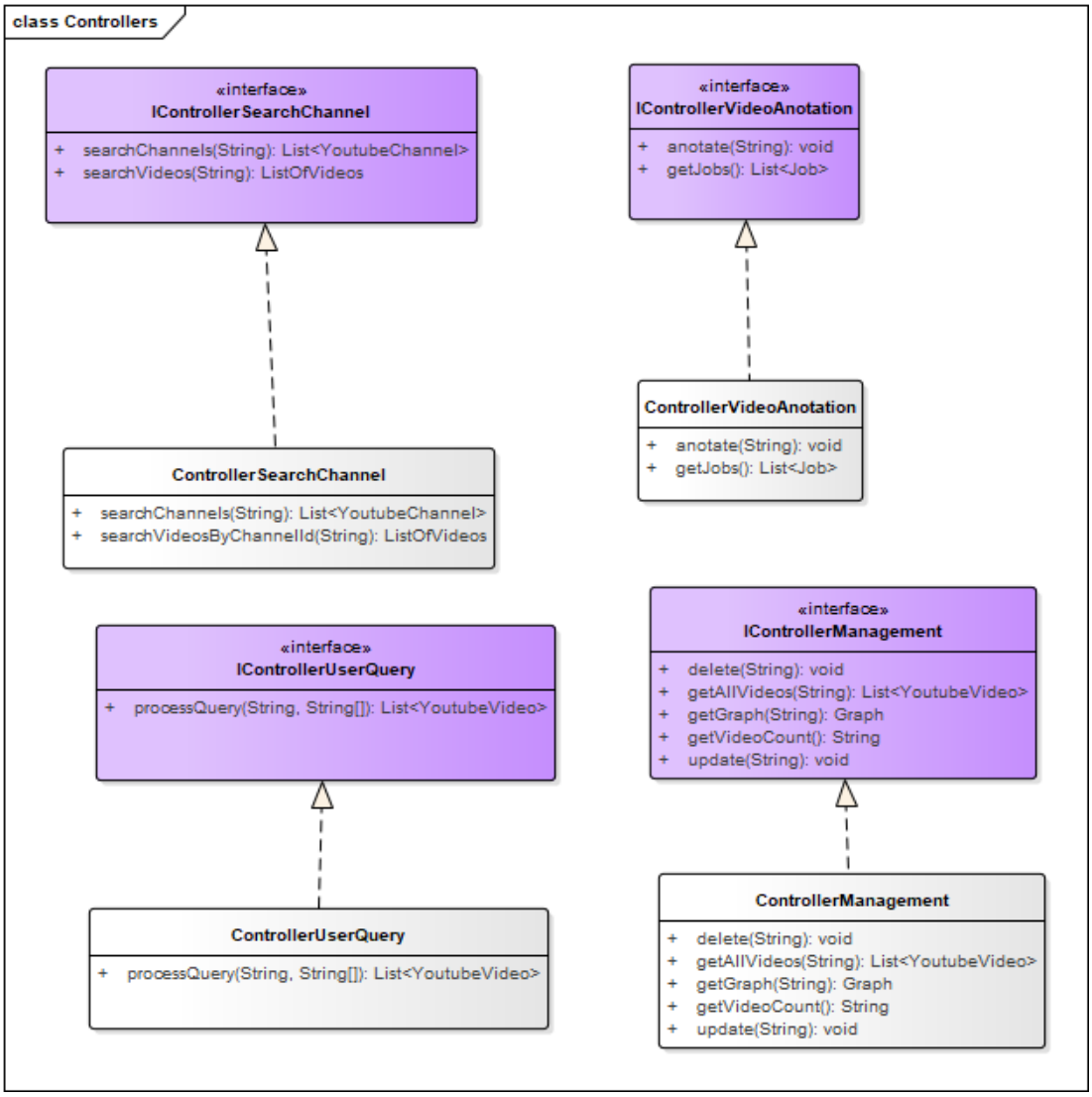

Figura 6.11: Diagrama de clases para el paquete “Controller”

(“Mostrar estadísticas”).

6.5.1. Anotación de vídeos.

El diagrama de secuencia de la figura 6.12 muestra las interacciones del administrador con el sistema para realizar la anotación de los vídeos, es decir, obtener el grafo asociado a un conjunto de vídeos de la plataforma Youtube y almacenarlos en el sistema. Este diagrama corresponde al caso de uso **CU. 1** (“Anotar vídeo semánticamente”) y **CU. 3** (“Añadir vídeo en cola”).

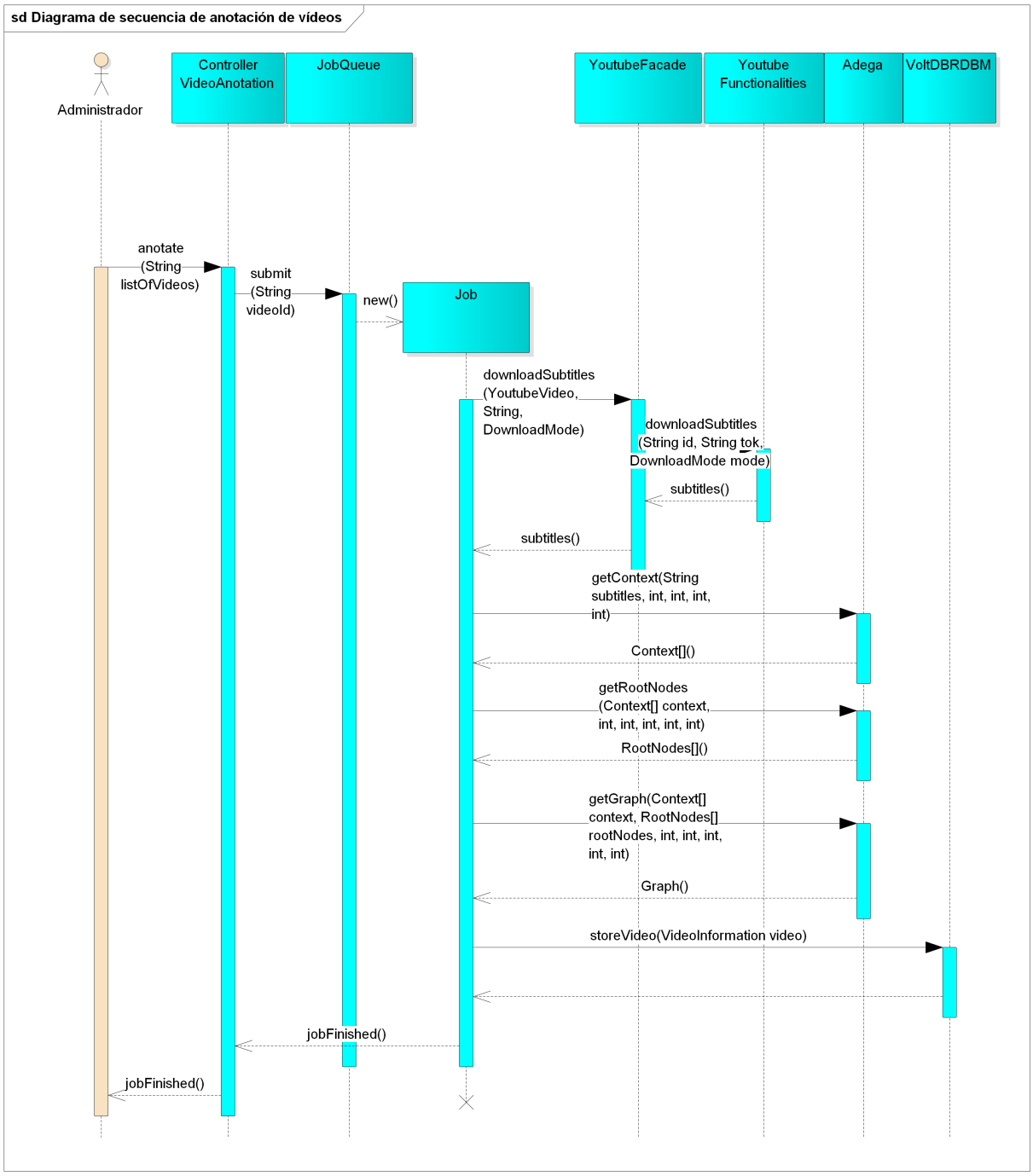

Figura 6.12: Diagrama de secuencia: “anotación de vídeos”

La interacción se inicia con la llamada al servicio de anotación pasando una lista de identificadores de vídeo como parámetros. Por cada vídeo, se envía un nuevo trabajo en cola, identificado por la instanciación de la línea de vida "Job". Cada trabajo realiza las siguientes operaciones:

- Descarga los subtítulos asociados a un vídeo. No se entra en las interioridades de la descarga de subtítulos puesto que depende enormemente del modo de descarga de los mismos.
- Realiza una llamada a ADEGA para obtener el contexto asociado a los subtitulos descargados.
- Realiza una llamada a ADEGA para obtener los nodos raíz del contexto obtenido anteriormente.
- Realiza otra llamada a ADEGA para obtener el grafo asociado al contexto y a los nodos raíz obtenidos en los dos pasos anteriores.
- Toda esta información (la obtenida de las llamadas a ADEGA junto con la información de los vídeos) se almacena en las bases de datos. El mismo método de almacenamiento realiza la sincronización entre las bases de datos.
- Se notifica al administrador de que el trabajo ha finalizado exitosamente.

6.5.2. Búsqueda de canales y vídeos.

El diagrama de secuencia de la figura 6.13 muestra las interacciones del administrador con el sistema cuando desea realizar una búsqueda de canales y vídeos de la plataforma Youtube. Este diagrama corresponde a los casos de uso **CU. 5** ("Buscar canales") y **CU.6** ("Buscar vídeos").

En primer lugar, la interacción se inicia con la búsqueda de un canal en función de su nombre. De ahí, se devuelve una lista de canales, junto con el identificador de cada canal (representado por la clase "YoutubeChannel"). Posteriormente, se obtienen los vídeos asociados a un canal en concreto mediante su identificador. Para cada vídeo (representado por un PlayListItem) se obtienen los metadatos del vídeo asociado (representado por el bucle). Se devuelve al usuario una lista de vídeos del canal seleccionado.

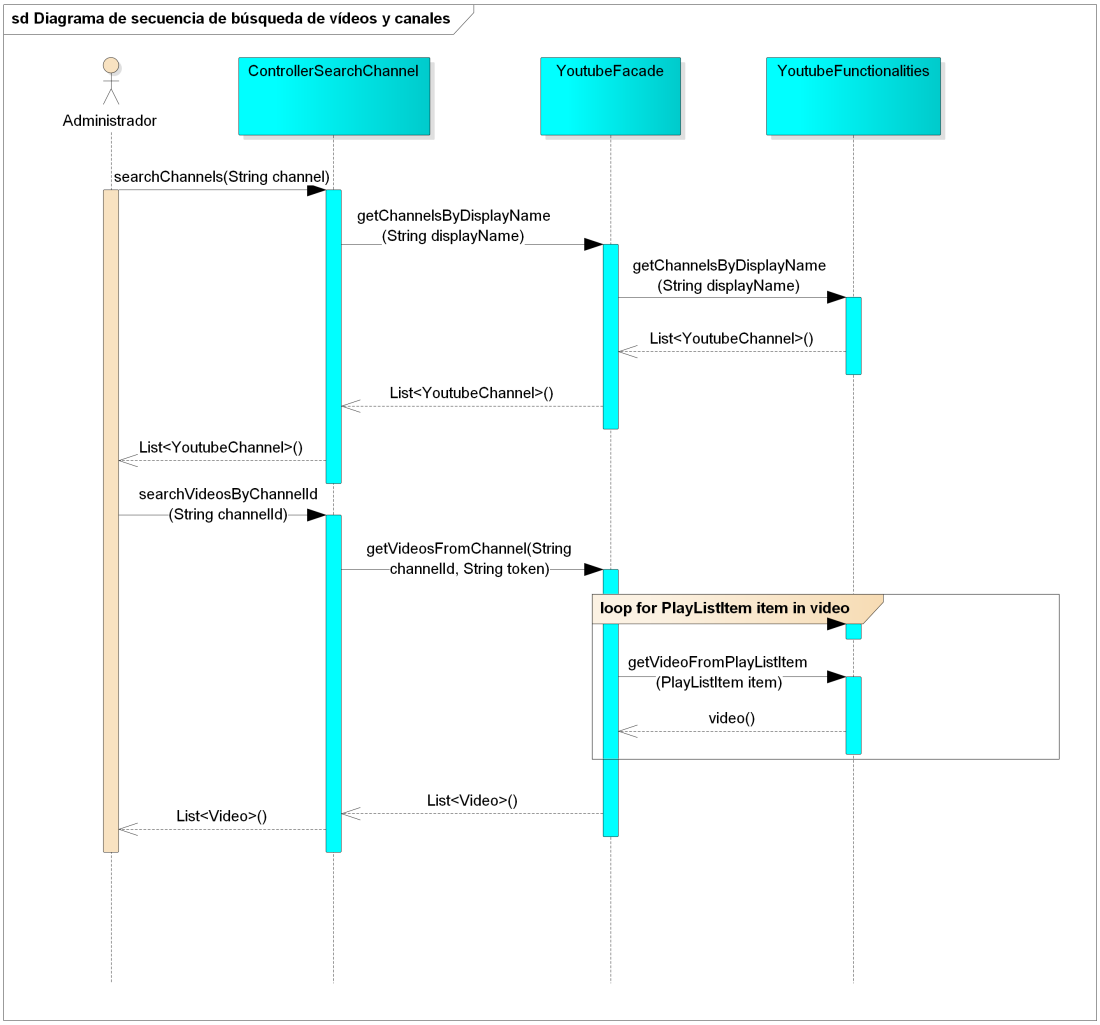

Figura 6.13: Diagrama de secuencia: “búsqueda de canales y vídeos”

6.5.3. Mostrar vídeos anotados

El diagrama de secuencia de la figura 6.14 muestra las interacciones del administrador con el sistema cuando desea ver los vídeos que hay anotados en el sistema. Este diagrama se corresponde con el caso de uso **CU. 9** ("Mostrar vídeos anotados").

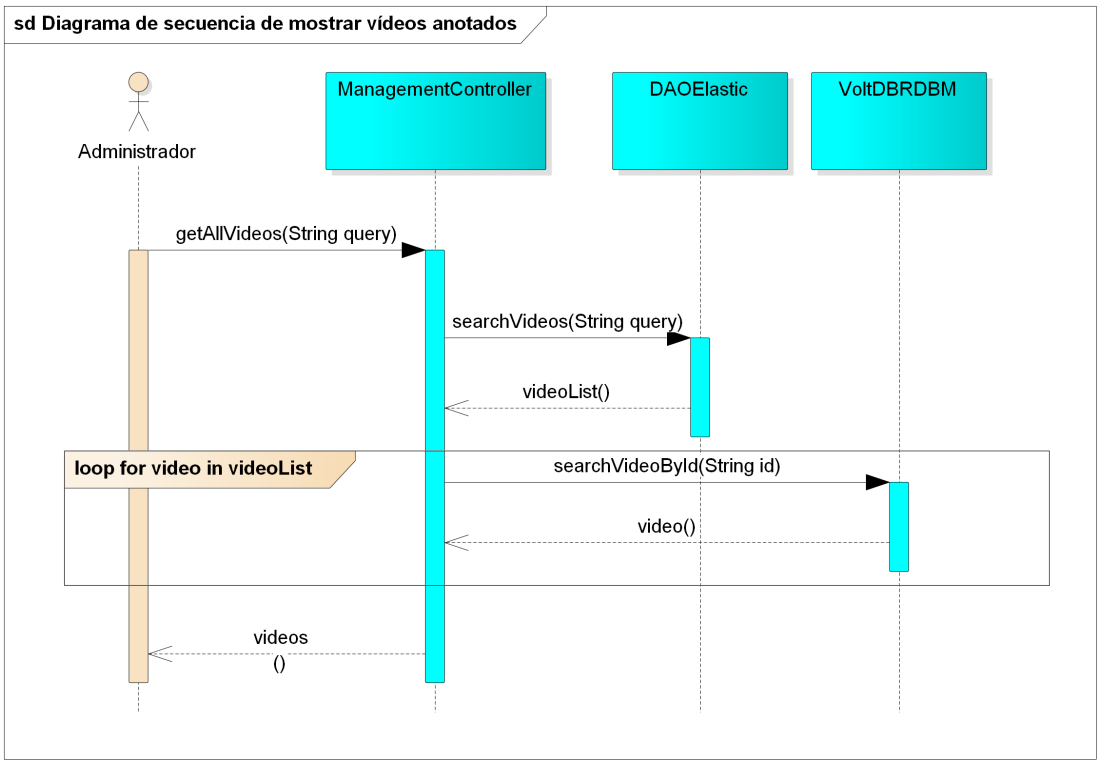

Figura 6.14: Diagrama de secuencia: "Mostrar vídeos anotados"

La interacción se inicia con la llamada al servicio de obtención de vídeos en función de una consulta. Primeramente, se consulta a ElasticSearch acerca de los vídeos disponibles en el sistema. Esto se realiza así debido a que ElasticSearch permite una mayor flexibilidad a la hora de realizar búsquedas en función de campos de texto. De la consulta se obtiene una lista de vídeos asociados, y, a su vez, para cada vídeo de la lista se obtienen los metadatos más relevantes (título y descripción, principalmente) de VoltDB. Estos datos se devuelven al administrador para su visualización.

6.5.4. Eliminar vídeos anotados

El diagrama de secuencia de la figura 6.15 permite ver las interacciones del administrador con el sistema cuando este desea eliminar un conjunto de vídeos del sistema. Este diagrama de secuencia se corresponde con el caso de uso **CU. 10** ("Eliminar vídeos anotados").

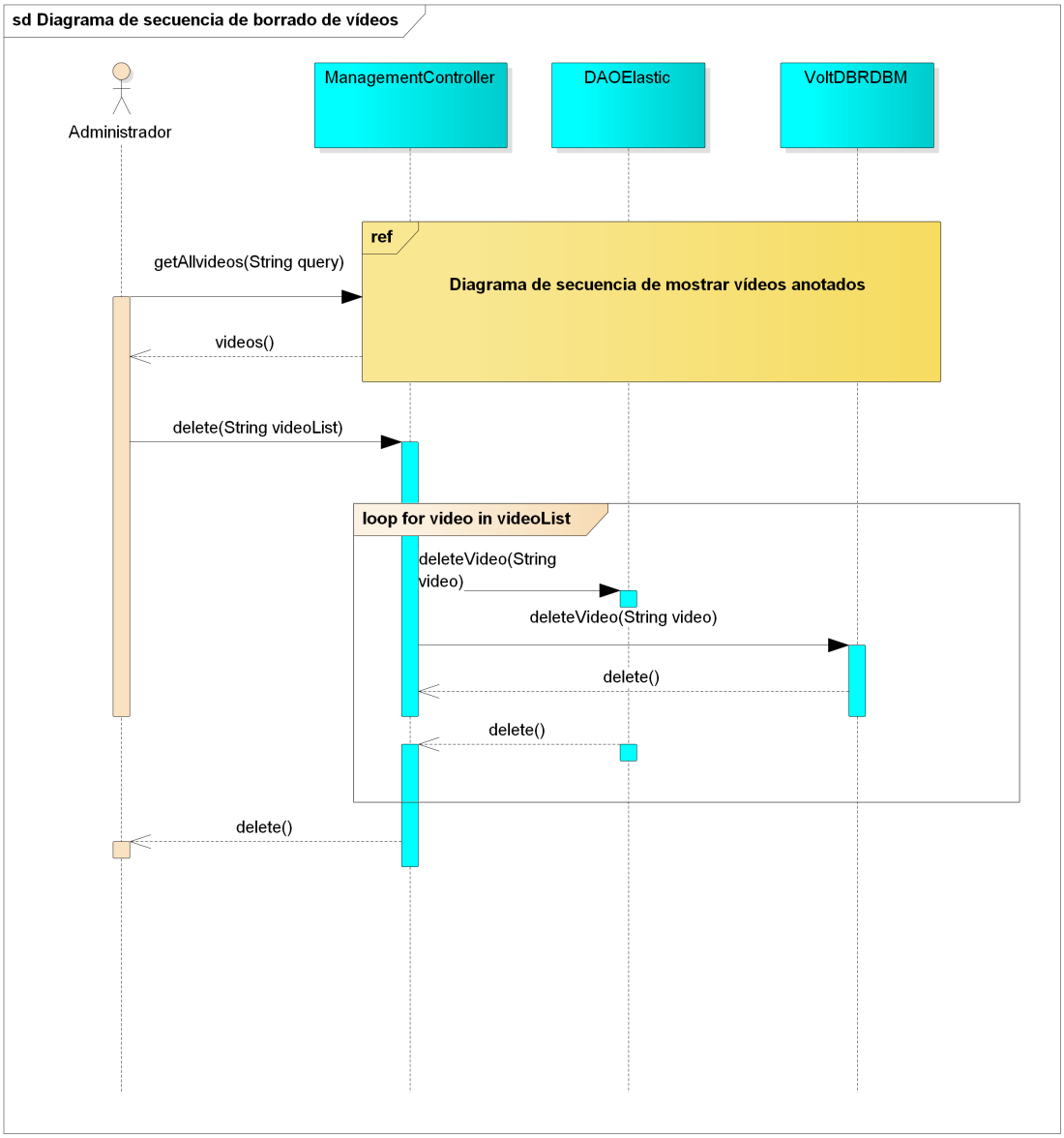

Figura 6.15: Diagrama de secuencia: "Eliminación de vídeos anotados"

En primer lugar, se llama al diagrama de secuencia "Mostrar vídeos anotados" puesto que, para eliminar un conjunto de vídeos, el usuario, en primer lugar, debe poder visualizar y seleccionar que vídeos quiere eliminar. Una vez mostrados y seleccionados los vídeos, el usuario llama al servicio de eliminación de un conjunto de vídeos. Así, para cada vídeo seleccionado, se eliminan de ambas bases de datos los vídeos asociados. Nótese que, en este caso, no hay ningún mecanismo de sincronización entre las dos bases de datos para la eliminación de los vídeos.

6.5.5. Actualizar vídeos anotados

El diagrama de secuencia de la figura 6.16 permite ver las interacciones del administrador con el sistema cuando se desea actualizar un conjunto de vídeos del sistema. Este diagrama de secuencia se corresponde con el caso de uso **CU.11** ("Actualizar vídeos anotados").

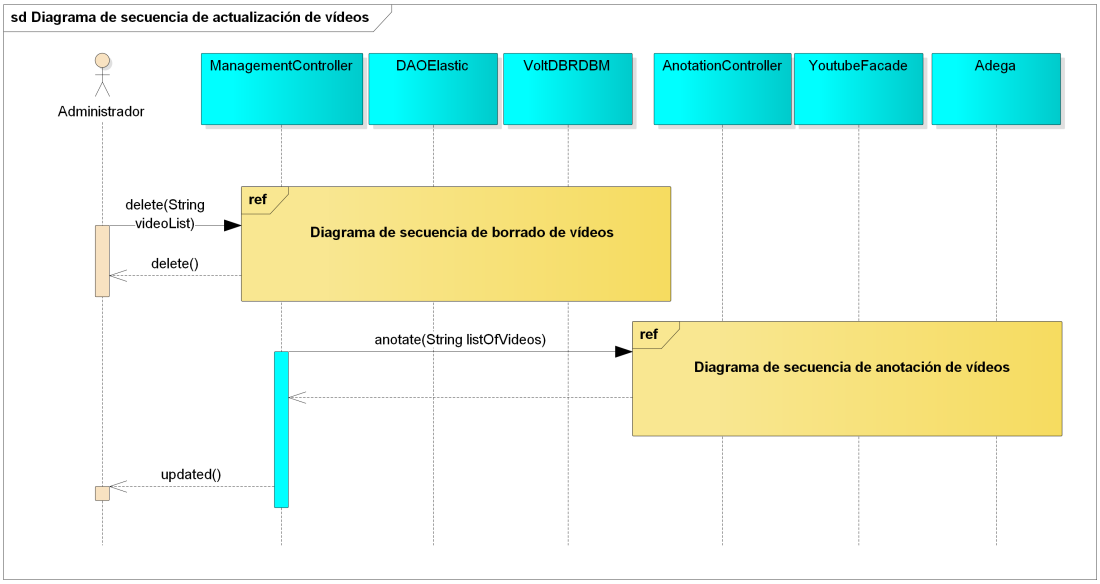

Figura 6.16: Diagrama de secuencia: "Actualización de vídeos anotados"

La interacción se inicia mediante la eliminación de los vídeos seleccionados por el usuario. Cada vídeo eliminado se vuelve a anotar siguiendo el mismo proceso que el descrito en el "diagrama de secuencia de anotación de vídeos". El principal motivo para realizarlo de esta manera reside en evitar que queden nodos huérfanos

si se realiza una actualización por modificación de tablas directamente (puesto que los grafos pueden no ser iguales al actualizar).

6.5.6. Procesamiento de consulta de usuario

El diagrama de secuencia de la figura 6.17 pretende mostrar las interacciones entre los distintos objetos del sistema cuando un usuario utiliza el sistema pregunta-respuesta. Se corresponde con los casos de uso **CU. 14** ("Introducir consulta") y **CU. 15** ("Visualizar resultado").

La interacción se inicia con una pregunta al sistema, es decir, con la llamada al método de resolución de preguntas. [3] Primero, se realiza el procesamiento de la consulta mediante el algoritmo ADEGA. Para ello, se obtienen el contexto, los nodos raíz y el grafo. Una vez obtenido el grafo de la consulta, se llama a la clase de comparación de grafos conceptuales.

Inicialmente, esta clase realiza un filtrado de los vídeos mediante una búsqueda por palabras llamando a la clase "*Keyword Search*",. Esta clase consulta al DAO de ElasticSearch por un conjunto de vídeos en función de una lista de términos separada por comas. Como resultado se obtiene una tabla hash en la que la clave es el identificador del vídeo y el valor es la similaridad obtenida mediante ElasticSearch.

Una vez obtenida esta tabla hash (`Map`), se calculan las relevancias añadiendo el peso de cada uno de los nodos del grafo para cada vídeo (el peso de los nodos se obtienen colateralmente de la consulta a ElasticSearch). Ya con el cálculo realizado, se ordenan los resultados en función de la relevancia y se realiza el filtrado de los 100 mejores. Este filtrado se realiza independientemente de que se utilice la comparación de grafos conceptuales o no, puesto que resulta incongruente devolver una lista de vídeos muy larga en la que los vídeos menos relevantes tienen, muy probablemente, una relevancia muy próxima a 0.

[3] Se está obviando la interacción inicial con el controlador que redirige a esta llamada por falta de espacio.

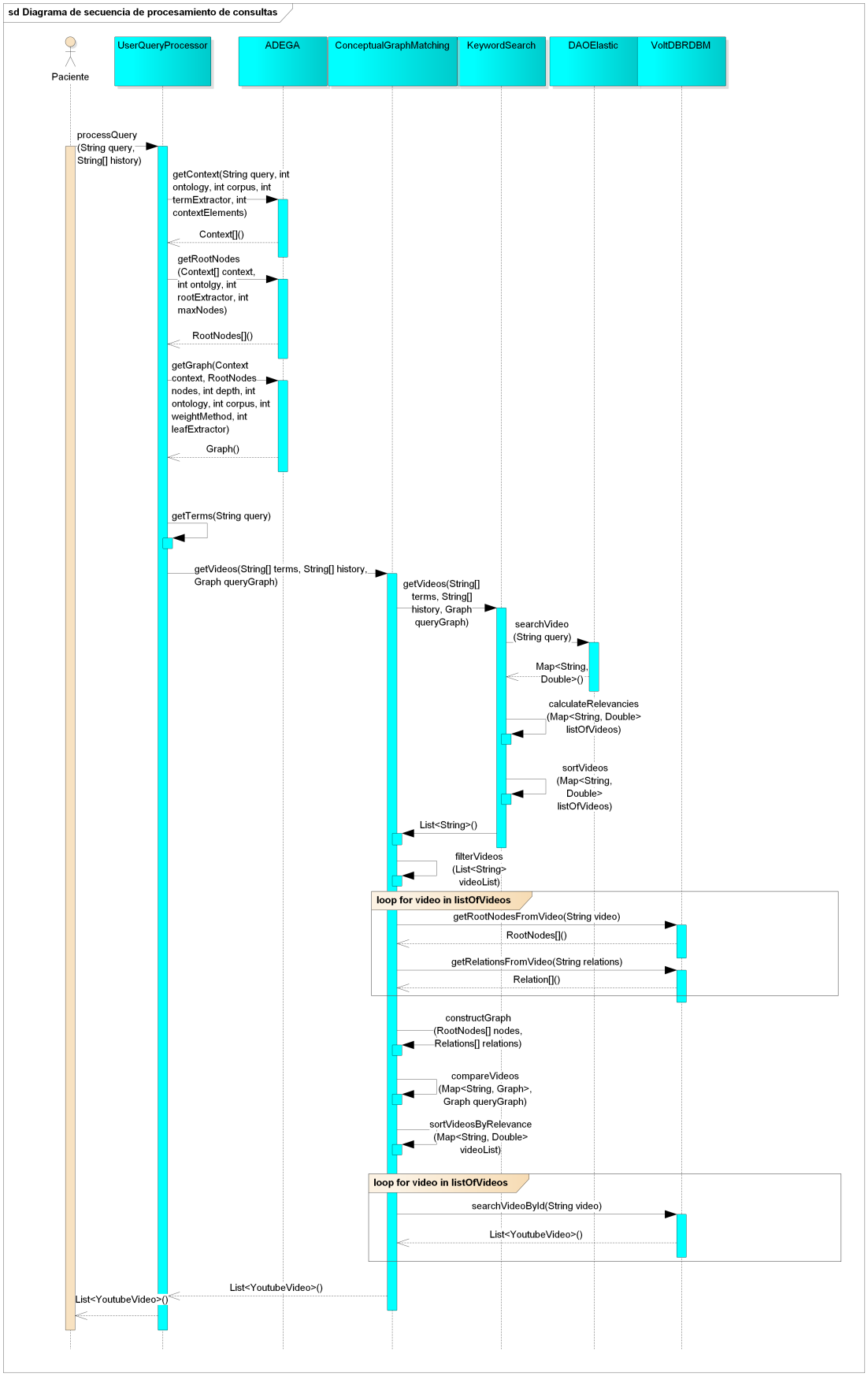

Figura 6.17: Diagrama de secuencia: procesamiento de consulta de usuario

Esta lista ordenada se devuelve al comparador de grafos que, primero, recupera los grafos completos para cada uno de los vídeos[4]. Una vez recuperados los grafos, se construye el grafo a partir de los datos recuperados y se comparan los vídeos utilizando el algoritmo descrito en el sección "Comparativa de grafos". Por último, se ordenan los vídeos en función de su relevancia y se devuelven al llamador, finalizando así la interacción.

6.6. Interfaces de usuario (*frontend*)

6.6.1. Diseño de la interfaz de usuario

Para el diseño de la interfaz de usuario se utiliza la herramienta **Lumzy** que permite crear prototipos (*wireframes*) de cualquier aplicación. En este apartado solo se mostrarán los prototipos más relevantes de la aplicación.

Así, la página principal de la aplicación se muestra en la figura 6.18. Como se puede observar, existe una barra de navegación lateral que permite acceder a las funcionalidades principales de la aplicación. La barra superior muestra el nombre de la aplicación y permite navegar hasta la ventana prinicipal. En la parte derecha se muestran algunas estadísticas de la aplicación.

Por otra parte, otra de las páginas con más carga visual es la pantalla que permite mostrar los vídeos de un canal. Dicha pantalla aparece reflejada en el prototipo de la figura 6.19. Como se puede ver, se mantiene tanto la barra de navegación lateral como la barra superior. Los vídeos aparecen mostrados en una lista y se permite tanto visualizarlos directamente desde la página como ver la descripción y título de los mismos. Los vídeos pueden ser marcados para anotación utilizando el "*checkbox*" de la parte superior derecha de cada vídeo. Se incluye

[4]Se podría argumentar que sería más eficiente pasar los nodos raíz de los vídeos directamente al comparador de grafos conceptual. Sin embargo, esto no sería correcto desde el punto de vista del diseño del patrón Strategy puesto que los algoritmos implementan la misma interfaz, la cual no devuelve ningún tipo de grafo ni de nodo raíz; únicamente devuelve vídeos y su correspondiente relevancia (puesto que es lo único que la clase que realiza la petición necesita, al fin y al cabo). Además, desde ElasticSearch no se pueden recuperar las relaciones entre los nodos.

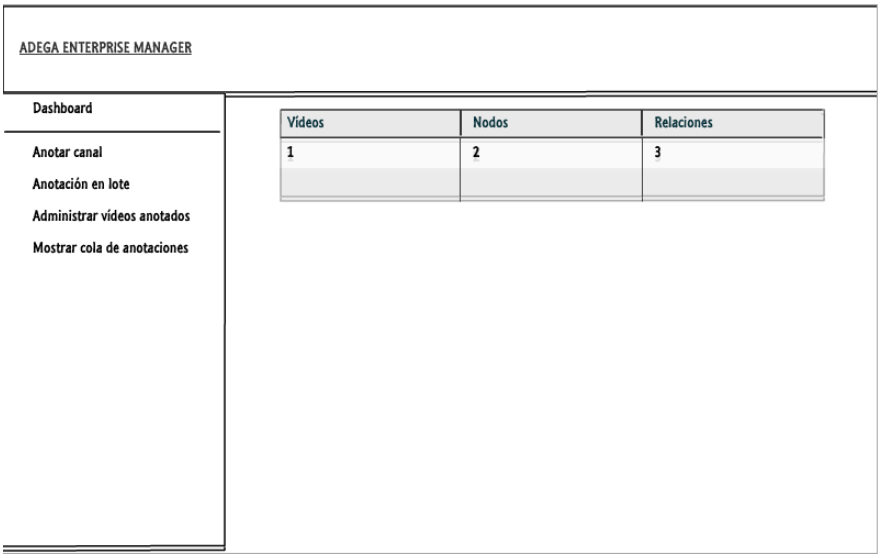

Figura 6.18: Prototipo de la página principal de la aplicación

un paginador que permite ver los vídeos poco a poco y un botón que permite mandar los vídeos seleccionados para su anotación.

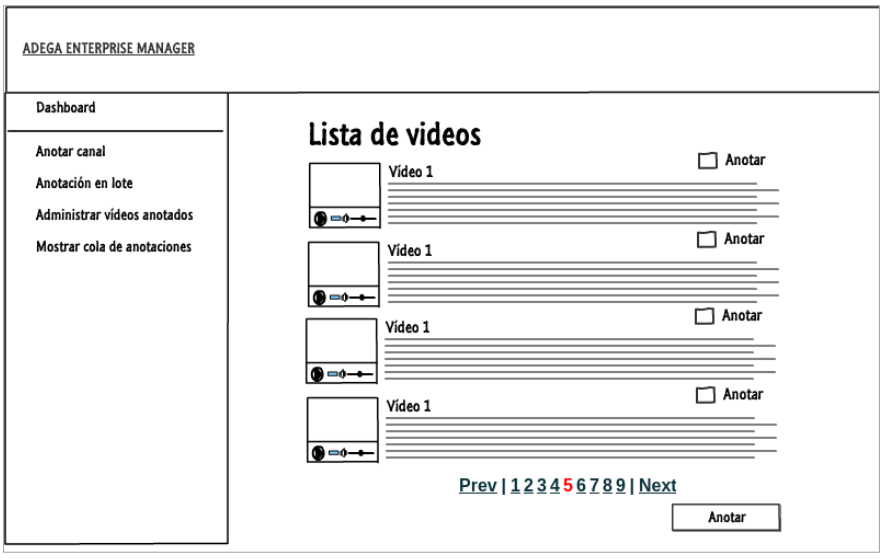

Figura 6.19: Prototipo de la página de mostrar vídeos de un canal

Por último, otra de las páginas importantes de la página de búsquedas del sistema pregunta-respuesta. Dicha página aparece reflejada en la figura 6.20. Como se puede observar, simplemente se incluye un campo que permite realizar las

búsquedas. Los resultados se devuelven justo debajo por orden de relevancia de la misma manera que en el prototipo de la figura 6.19.

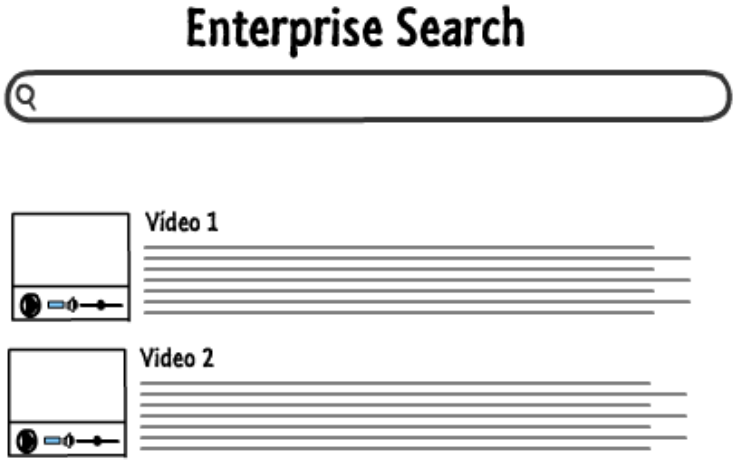

Figura 6.20: Prototipo de la página de pregunta respuesta.

6.6.2. Implementación de la interfaz de usuario

Para la implementación de la interfaz de usuario se ha decidido por implementar una **Single Page Application** (SPA) utilizando *AngularJS* y *Bulma*. Así, el esqueleto de la aplicación consiste tanto en la barra de navegación izquierda como en la barra de título y en la parte derecha se intercambian las distintas páginas de la aplicación: mostrar vídeos, buscar canales, mostrar resultados de las anotaciones... Cada una de estas páginas se corresponde con un *controlador* en AngularJS.

Así, la página principal de la aplicación tiene el aspecto mostrado en la figura 6.21. Como se puede observar, en la parte izquierda de la página existe una barra de navegación que permite acceder a las principales funcionalidades de la aplicación. La parte central alberga el contenido en cuestión de cada funcionalidad que, en este caso, muestra estadísticas del sistema ("**dashboard**").

Por otra parte, la página que muestra los vídeos de un canal tiene el aspecto mostrado en la figura 6.22. Como se puede ver, se permite visualizar el vídeo

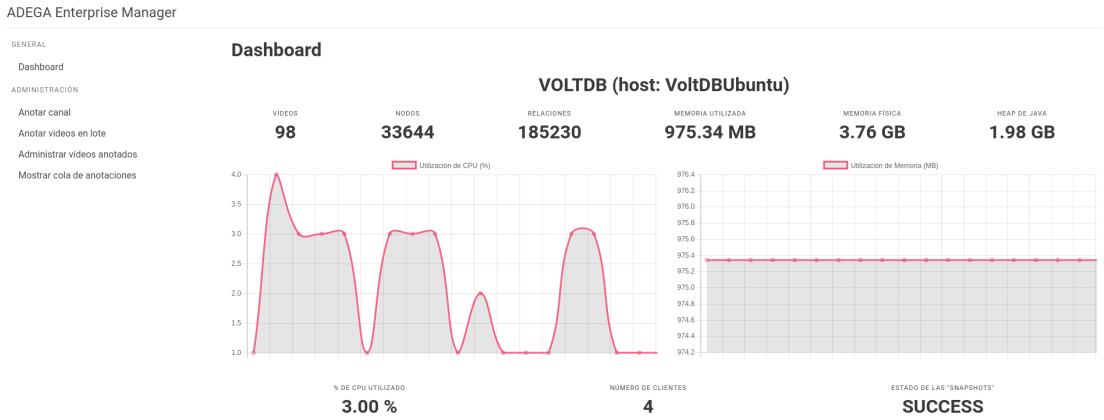

Figura 6.21: Implementación de la pantalla principal de la aplicación.

directamente (pulsando sobre la miniatura), se muestran algunos datos básicos y se permite añadir para selección en el "checkbox" adjunto a cada vídeo. Además, se permite ver los metadatos específicos de cada vídeo pulsando sobre el botón "Ver metadatos". Los vídeos seleccionados pueden ser enviados para su anotación pulsando sobre el botón inferior derecho "Anotar seleccionados". Por último, la interfaz pregunta respuesta tiene el aspecto mostrado en la figura 6.23. Esta interfaz simplemente consiste en un buscador que muestra la respuesta al usuario inmediatamente debajo.

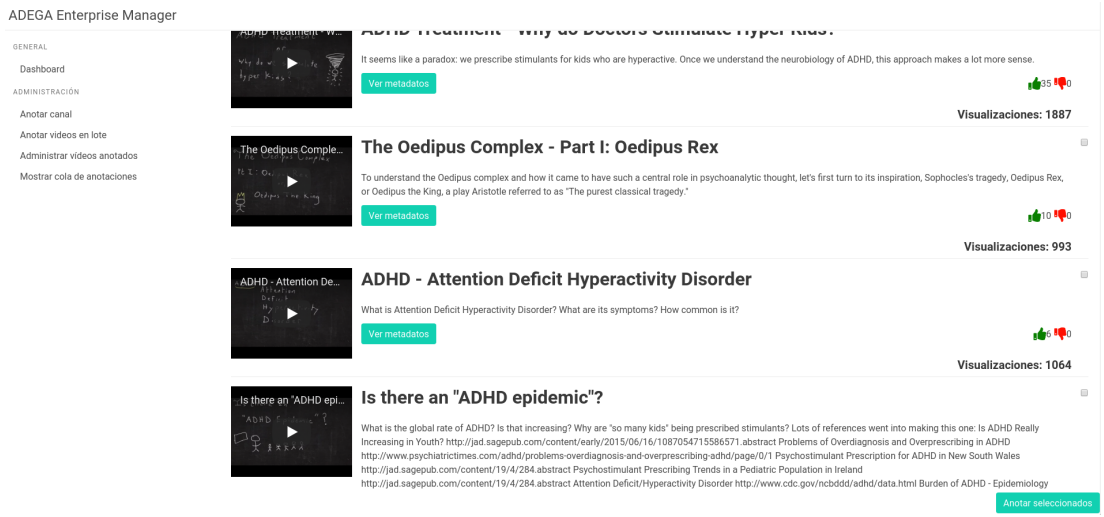

Figura 6.22: Implementación de la visualización de vídeos de un canal.

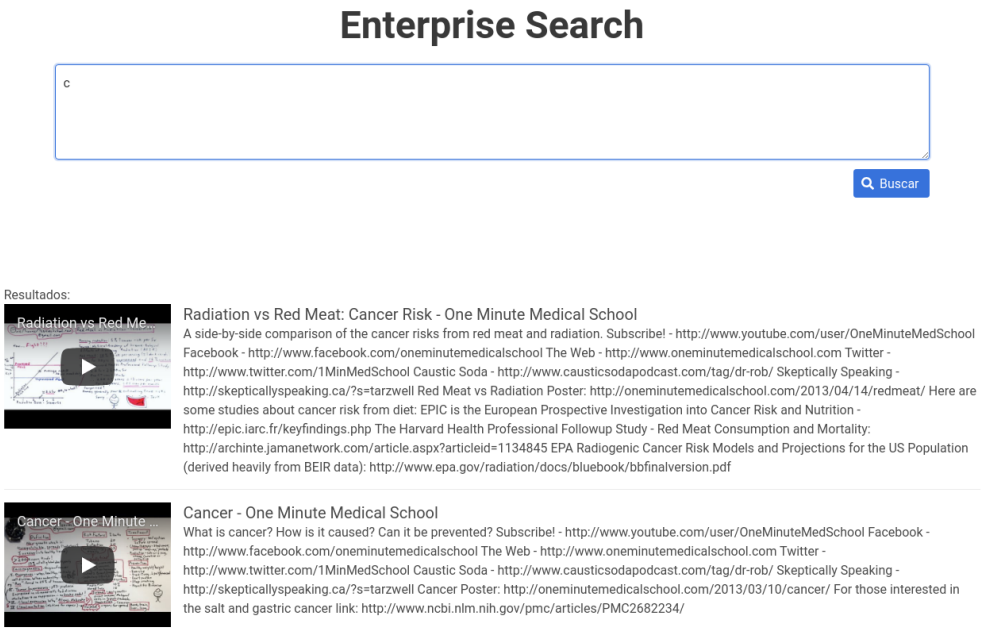

Figura 6.23: Implementación de la interfaz del sistema pregunta-respuesta.

Capítulo 7

Pruebas y validación

7.1. Pruebas unitarias

Las pruebas unitarias sirven para asegurar el correcto funcionamiento de un fragmento o módulo de código. Para la implementación de estas pruebas unitarias se ha utilizado JUnit, un framework Java para la creación de pruebas unitarias. Las pruebas que se implementen serán pruebas de caja negra, por simplicidad.

Para la especificación de las pruebas unitarias, se definirá el método a probar así como los parámetros que se le suministran, tal y como especifica la siguiente plantilla:

Identificador:	PU-X
Método a probar:	Método a probar
Parámetros:	
Parámetros que se le suministran al método	
Resultado esperado:	Resultado esperado de la prueba unitaria
Precondiciones:	Precondiciones de la prueba
Ejemplo de llamada	
Ejemplo de una llamada	

Así, el catálogo de pruebas realizado es el siguiente:

Identificador:	PU-1
Método a probar:	YoutubeFacade.getVideoFromId
Parámetros:	
"lIqT3WX SUc"	
Resultado esperado:	Ejecución correcta. Se obtiene un objeto con los datos del vídeo. El título del vídeo es "Turning the Page to a New Decade"
Precondiciones:	No hay
Ejemplo de llamada	
new YoutubeFacade().getVideoFromId("lIqT3WX_SUc")	

Identificador:	PU-2
Método a probar:	SubtitleDownloader.getvideoSubtitle
Parámetros:	
"lIqT3WX SUc"	
Resultado esperado:	Se obtiene un objeto con los subtitulos del vídeo. Los únicos subtítulos disponibles son de tipo ASR y EN, es decir, autogenerados y en inglés.
Precondiciones:	No hay
Ejemplo de llamada	
new SubtitleDownloader().getVideoSubtitles("lIqT3WX_SUc")	

Identificador:	PU-3
Método a probar:	SubtitleDownloader.findDownloadUrl
Parámetros:	
s.getvideoSubtitles("lIqT3WX_SUc").get(0) "lIqT3WX_SUc"	
Resultado esperado:	Se obtiene una URL que corresponde con los subtítulos autogenerados del vídeo (en formato XML)
Precondiciones:	No hay
Ejemplo de llamada	
SubtitleDownloader s = new SubtitleDownloader(); s.findDownloadUrl(s.getVideoSubtitles("lIqT3WX_SUc").get(0), "lIqT3WX_SUc")	

Identificador:	PU-4
Método a probar:	SubtitleDownloader.downloadSubtitle
Parámetros:	
s.findDownloadUrl(s.getVideoSubtitles("lIqT3WX_SUc").get(0)), "subtitulos.txt", DownloadMode.NOT_CLEAN	
Resultado esperado:	Se obtiene un fichero denominado "subtítulos.txt" que contiene los subtítulos descargados de Youtube. Los subtítulos contienen etiquetas XML.
Precondiciones:	No hay
Ejemplo de llamada	
s.downloadSubtitle(s.findDownloadUrl(s.getVideoSubtitles("lIqT3WX_SUc").get(0)), "subtitulos.txt", DownloadMode.NOT_CLEAN)	

Identificador:	PU-5
Método a probar:	SubtitleDownloader.downloadSubtitle
Parámetros:	
s.findDownloadUrl(s.getVideoSubtitles("lIqT3WX_SUc").get(0)), "subtitulos.txt", DownloadMode.CLEAN	
Resultado esperado:	Se obtiene un fichero denominado "subtítulos.txt" que contiene los subtítulos descargados de Youtube. Los subtítulos **no** contienen etiquetas XML.
Precondiciones:	No hay
Ejemplo de llamada	
s.downloadSubtitle(s.findDownloadUrl(s.getVideoSubtitles("lIqT3WX_SUc").get(0)), "subtitulos.txt", DownloadMode.CLEAN)	

Identificador:	PU-5
Método a probar:	SubtitleDownloader.downloadSubtitle
Parámetros:	
s.findDownloadUrl(s.getVideoSubtitles("aaaa").get(0)), "subtitulos.txt", DownloadMode.CLEAN	
Resultado esperado:	Se devuelve un error asociado a que el vídeo no existe en la base de datos de Youtube.
Precondiciones:	No hay
Ejemplo de llamada	
s.downloadSubtitle(s.findDownloadUrl(s.getVideoSubtitles("aaaa").get(0)), "subtitulos.txt", DownloadMode.CLEAN)	

Identificador:	PU-6
Método a probar:	SubtitleDownloader.dowloadSubtitle
Parámetros:	
s.findDownloadUrl(s.getVideoSubtitles("GRxofEmo3HA").get(0)), "subtitulos.txt", DownloadMode.CLEAN	
Resultado esperado:	Se devuelve un error asociado a que el vídeo no posee subtítulos.
Precondiciones:	No hay
Ejemplo de llamada	
s.downloadSubtitle(s.findDownloadUrl(s.getVideoSubtitles("GRxofEmo3HA").get(0)), "subtitulos.txt", DownloadMode.CLEAN)	

Identificador:	PU-7
Método a probar:	Mangement.update
Parámetros:	
"lIqT3WX_SUc"	
Resultado esperado:	Se actualiza el contenido tanto de la anotación del vídeo como de los metadatos de Youtube
Precondiciones:	El vídeo debe estar anotado previamente en la base de datos. Preferentemente, su contenido en la base de datos debe estar a "null" (para que sea más fácil de comprobar)
Ejemplo de llamada	
new Management().update("lIqT3WX_SUc")	

Identificador:	PU-8
Método a probar:	Management.update
Parámetros:	
"aaaaaa"	
Resultado esperado:	El sistema devuelve un error asociado a que no se puede actualizar el vídeo.
Precondiciones:	El vídeo no debe estar anotado en las bases de datos
Ejemplo de llamada	
new Management().update("aaaaaaa")	

Identificador:	PU-9
Método a probar:	Management.delete
Parámetros:	
"lIqT3WX_SUc"	
Resultado esperado:	El vídeo es eliminado de la base de datos.
Precondiciones:	El vídeo debe estar anotado previamente en la base de datos.
Ejemplo de llamada	
new Management().delete("lIqT3WX_SUc")	

Identificador:	PU-10
Método a probar:	Management.delete
Parámetros:	
"aaaaaa"	
Resultado esperado:	El sistema devuelve un error asociado a que no se puede eliminar el vídeo.
Precondiciones:	El vídeo no debe estar anotado en las bases de datos
Ejemplo de llamada	
new Management().update("aaaaaaa")	

Identificador:	PU-11
Método a probar:	Management.delete
Parámetros:	
null	
Resultado esperado:	El sistema devuelve un error asociado a que no se puede eliminar el vídeo.
Precondiciones:	El vídeo no debe estar anotado en las bases de datos
Ejemplo de llamada	
new Management().update(null)	

Identificador:	PU-12
Método a probar:	Management.getGraph
Parámetros:	
"lIqT3WX_SUc"	
Resultado esperado:	Se obtiene el grafo asociado al vídeo.
Precondiciones:	El vídeo debe estar anotado previamente en la base de datos.
Ejemplo de llamada	
new Management().getGraph("lIqT3WX_SUc")	

Identificador:	PU-13
Método a probar:	YoutubeFunctionalities.getAllVideosFromChannel
Parámetros:	
"lIqT3WX_SUc"	
Resultado esperado:	Se obtiene una lista de "PlayListItems" con los vídeos asociados al canal
Precondiciones:	No hay
Ejemplo de llamada	
new YoutubeFunctionalities.getAllVideosFromChannel("lIqT3WX_SUc")	

7.2. Pruebas de integración

Las pruebas de integración se realizan una vez se han superado las pruebas unitarias sobre el software. El propósito principal de las mismas es probar que los distintos módulos que conforman el software funcionen bien en su conjunto.

De nuevo, para la codificación de las pruebas se utiliza JUnit. Para la realización de estas pruebas, lo más sencillo es llamar a los distintos servicios expuestos en la aplicación. Para ello, se utiliza una librería denominada "MockMVC" que permite realizar llamadas a los distintos servicios, configurando distintos parámetros sobre dichas llamadas y comprobar las respuestas a las llamadas a los servicios. La plantilla para este tipo de pruebas es muy similar que la utilizada para las pruebas unitarias. El catálogo de pruebas de integración es el siguiente:

Identificador:	PI-1
Servicio a probar:	/anotate
Parámetros:	
"lIqT3WX_SUc"	
Resultado esperado:	El sistema añade el trabajo a la cola de anotaciones. Se anota el vídeo en las bases de datos.
Precondiciones:	Las bases de datos están desplegadas y el vídeo no está anotado en la base de datos.
Ejemplo de llamada	
this.mock.perform(post(/anotate") .param(."notation","lIqT3WX_SUc")) .andExpect(status().isOk()) .andExpect(content() .contentType(MediaType.APPLICATION_JSON_UTF8));	

Identificador:	PI-2
Servicio a probar:	/anotate
Parámetros:	
"qwertyuiop"	
Resultado esperado:	El sistema devuelve un error indicando que el vídeo no existe
Precondiciones:	Las bases de datos están desplegadas.
Ejemplo de llamada	
this.mock.perform(post(/processQuery") .param(.ªnotation", "qwertyuiop")) .andExpect(status().isInternalServerError()) .andExpect(content().contentType(MediaType.APPLICATION_JSON_UTF8));	

Identificador:	PI-3
Servicio a probar:	/anotate
Parámetros:	
"lIqT3WX_SUc", "FukBZp9gyQQ"	
Resultado esperado:	El sistema añade un trabajo por cada vídeo a la cola de anotaciones. Se anota cada vídeo en las bases de datos.
Precondiciones:	Las bases de datos están desplegadas y los vídeos no están anotados en las base de datos.
Ejemplo de llamada	
this.mock.perform(post(/anotate") .param(.ªnotation","lIqT3WX_SUc")) .param("anotation", "FukBZp9gyQQ") .andExpect(status().isOk()).andExpect(content() .contentType(MediaType.APPLICATION_JSON_UTF8));	

Identificador:	PI-4
Servicio a probar:	/processQuery
Parámetros:	
"I have diabetes and hypertension."	
Resultado esperado:	El sistema devuelve un conjunto de vídeos relevantes en función de la consulta del usuario.
Precondiciones:	Las bases de datos están desplegadas y contienen vídeos que traten diabetes e hipertensión.
Ejemplo de llamada	
this.mock.perform(post("anotate") .param("query", "I have diabetes and hypertension")) .andExpect(status().isOk())	

Identificador:	PI-5
Servicio a probar:	/processQuery
Parámetros:	
"I have many stomach issues from last 6 months. I have gas, burp which tastes like vomit, abdomen pain in the lower side like sharp pain, headaches, confusion.." [1]	
Resultado esperado:	El sistema devuelve un conjunto de vídeos relevantes en función de la consulta del usuario.
Precondiciones:	Las bases de datos están desplegadas y contienen vídeos que traten dolencias de estómago.
Ejemplo de llamada	
this.mock.perform(post("anotate") .param("query", "`Insertar consulta del usuario aquí`")) .andExpect(status().isOk())	

7.3. Pruebas de validación de requisitos

7.3.1. Pruebas de validación de requisitos funcionales.

Este tipo de pruebas permite verificar que los requisitos de un proyecto están satisfechos. Para la definición de las pruebas se utilizará la siguiente plantilla:

Identificador	P-00X
Casos de uso involucrados	Casos de uso que se comprueban con la prueba
Descripción	Descripción de la prueba a realizar
Resultado esperado	Resultados esperados de la ejecución de la prueba
Estado	Superada o fallida
Observaciones	Este campo indica los motivos de fallo de la prueba (en caso de que los hubiere)

Así, el conjunto de pruebas a realizar se muestra a continuación:

Identificador	P-001
Casos de uso involucrados	CU1, CU2, CU3, CU4
Descripción	Intentar anotar un único vídeo mediante el anotador por lotes.
Resultado esperado	Se anota el vídeo en la base de datos y se muestra el vídeo en la lista de vídeos anotados.
Estado	Superada

Identificador	P-002
Casos de uso involucrados	CU1, CU2, CU3, CU4
Descripción	Intentar anotar varios vídeos mediante el anotador por lotes
Resultado esperado	Se anota el vídeo en la base de datos y se muestra el estado de cada vídeo en la lista de vídeos anotados.
Estado	Superada

Identificador	P-003
Casos de uso involucrados	CU5
Descripción	Intentar buscar el canal denominado como “American Diabetes Association”.
Resultado esperado	Se encuentra el canal en cuestión, junto a su descripción.
Estado	Superada

Identificador	P-004
Casos de uso involucrados	CU5, CU6
Descripción	Intentar buscar los vídeos asociados al canal que tiene por nombre “American Diabetes Association”
Resultado esperado	Se muestra tanto el canal en la lista de canales a ver y se muestran los vídeos asociados a dicho canal.
Estado	Superada

Identificador	P-005
Casos de uso involucrados	CU1, CU3, CU4, CU5, CU6
Descripción	Se intenta anotar un vídeo que no tenga subtítulos mediante el método de búsqueda de canales.
Resultado esperado	El vídeo aparece en la lista de vídeos anotados como “Sin subtítulos”
Estado	Superada

Identificador	P-006
Casos de uso involucrados	CU1, CU3, CU4, CU5, CU6
Descripción	Intentar anotar un conjunto de vídeos con subtítulos mediante el método de búsqueda de canales.
Resultado esperado	Se anota los vídeos en la base de datos y se muestra el estado de cada vídeo en la lista de vídeos anotados.
Estado	Superada

Identificador	P-007
Casos de uso involucrados	CU1, CU2, CU3, CU4
Descripción	Intentar anotar un conjunto de vídeos tanto con subtítulos como sin ellos mediante el método de anotación por lotes.
Resultado esperado	Se anotan los vídeos que tengan subtítulos en la base de datos. Se muestra el estado correcto de cada vídeo (finalizado correctamente o sin subtítulos) al finalizar las anotaciones.
Estado	Superada

Identificador	P-008
Casos de uso involucrados	CU1, CU2, CU3, CU9
Descripción	Anotar un vídeo mediante el método de anotación por lotes y comprobar que aparece en la lista de vídeos anotados en la base de datos.
Resultado esperado	Se anota el vídeo en la base de datos y aparece en la lista de vídeos anotados.
Estado	Superada

Identificador	P-009
Casos de uso involucrados	CU1, CU2, CU3, CU9, CU10
Descripción	Anotar un vídeo mediante el método de anotación por lotes, comprobar que aparece en la lista de vídeos anotados en la base de datos, eliminarlo y comprobar que no aparece en la lista de vídeos anotados en la base de datos.
Resultado esperado	Al eliminar el vídeo de la base de datos este desaparece de la lista de vídeos anotados.
Estado	Superada

Identificador	P-010
Casos de uso involucrados	CU1, CU2, CU3, CU9, CU10, CU11
Descripción	Anotar un vídeo mediante el método de anotación por lotes, modificar el título manualmente de las bases de datos, actualizar el vídeo y ver que el título del vídeo cambia al título correcto.
Resultado esperado	El título del vídeo al final del proceso se cambia al título correcto real.
Estado	Superada

Identificador	P-011
Casos de uso involucrados	CU1, CU3, CU5, CU6, CU9, CU12
Descripción	Anotar un vídeo mediante el método de búsqueda de vídeos y comprobar que se puede visualizar el grafo desde el sistema de administración de vídeos anotados.
Resultado esperado	Se anota el vídeo en la base de datos y se muestra el grafo del vídeo correctamente.
Estado	Superada

Identificador	P-012
Casos de uso involucrados	CU1, CU2, CU14, CU15
Descripción	Anotar 5 vídeos similares en temática en la base de datos de tal forma que únicamente aparezcan esos vídeos, realizar una pregunta al sistema pregunta respuesta y verificar que se devuelve el vídeo más relevante.
Resultado esperado	El sistema pregunta-respuesta devuelve el vídeo más relevante de los 5 existentes.
Estado	Superada

Identificador	P-013
Casos de uso involucrados	CU14, CU15
Descripción	Se le proporciona al sistema pregunta-respuesta una consulta nula (es decir, sin contenido).
Resultado esperado	El sistema pregunta-respuesta muestra un error para esa petición y continúa atendiendo peticiones.
Estado	Superada

Identificador	P-014
Casos de uso involucrados	CU1, CU2, CU3, CU4
Descripción	Intentar anotar mediante el anotador en lote un conjunto de vídeos nulo (es decir, ningún vídeo.
Resultado esperado	El sistema continúa atendiendo peticiones y no se muestra ningún vídeo nuevo en la cola de anotación de vídeos.
Estado	Superada

Identificador	P-015
Casos de uso involucrados	CU1, CU3, CU5, CU6
Descripción	Intentar anotar mediante el sistema de búsqueda de vídeos un conjunto de vídeos nulo (es decir, ningún vídeo).
Resultado esperado	El sistema continúa atendiendo peticiones y no se muestra ningún vídeo nuevo en la cola de anotación de vídeos.
Estado	Superada

Identificador	P-016
Casos de uso involucrados	CU13
Descripción	Ver que las estadísticas de los vídeos de la base de datos se actualizan con la anotación de vídeos. Comparar la diferencia de vídeos entre la BD original y la BD ampliada con los nuevos vídeos anotados.
Resultado esperado	El sistema actualiza correctamente las estadísticas de la base de datos.
Estado	Superada

Identificador	P-017
Casos de uso involucrados	CU13
Descripción	Ver que las estadísticas de los vídeos de la base de datos concuerdan con los valores proporcionados por las bases de datos VoltDB y ElasticSearch. Para la comparación, se utiliza el VoltDB Management Center y la API de estadísticas de ElasticSearch.
Resultado esperado	El sistema muestra correctamente los datos.
Estado	Superada

Identificador	P-018
Casos de uso involucrados	CU5, CU6, CU7
Descripción	Visualizar los metadatos de un vídeo de la plataforma Youtube.
Resultado esperado	El sistema muestra correctamente los metadatos asociados al vídeo.
Estado	Superada

Identificador	P-019
Casos de uso involucrados	CU5, CU6, CU8
Descripción	Visualizar los subtítulos de un vídeo de la plataforma Youtube.
Resultado esperado	El sistema muestra correctamente los subtítulos asociados al vídeo.
Estado	Superada

7.3.2. Pruebas de validación de requisitos no funcionales.

A continuación se muestran las pruebas sobre los distintos requisitos no funcionales de la aplicación.

- **RQNF-01:** rendimiento del sistema de consultas. Cumplido: el sistema tarda menos del tiempo especificado.
- **RQNF-02:** idioma utilizado por el usuario. Cumplido: el sistema responde a preguntas en el idioma inglés.
- **RQNF-03:** interfaz usable. Cumplido: el sistema cumple los heurísticos de Nielsen y supera las pruebas de usabilidad.
- **RQNF-04:** interfaz gráfica independiente del motor. Cumplido: el sistema es una aplicación web, por lo que se puede utilizar en casi cualquier sistema (excepto los más antiguos).
- **RQNF-05:** extensibilidad Cumplido: el sistema se ha implementado utilizando distintos patrones de diseño y arquitectura permitiendo así la extensibilidad del mismo.

7.3.3. Matrices de trazabilidad

En la figura 7.1 se muestra la matriz de trazabilidad entre los casos de uso y las pruebas de validación del software. Así, se puede comprobar que los requisitos del proyecto han sido todos validados correctamente. Así, se puede comprobar que los requisitos del proyecto han sido todos validados correctamente.

Cuadro 7.1: Matriz: casos de uso - pruebas de validación.

	P. 001	P. 002	P. 003	P. 004	P. 005	P. 006	P. 007	P. 008	P. 009	P. 010	P. 011	P. 012	P. 013	P. 014	P. 015	P. 016	P. 017	P. 018	P. 019
CU. 01	↰	↰			↰	↰	↰	↰	↰	↰	↰	↰	↰	↰	↰				
CU. 02	↰	↰				↰	↰	↰	↰	↰		↰	↰						
CU. 03	↰	↰	↰	↰	↰	↰	↰	↰	↰	↰	↰		↰	↰	↰				
CU. 04	↰	↰			↰	↰	↰						↰	↰					
CU. 05			↰	↰	↰	↰					↰				↰			↰	↰
CU. 06				↰	↰	↰					↰				↰			↰	
CU. 07																		↰	
CU. 08																			↰
CU. 09								↰	↰	↰	↰								
CU. 10									↰	↰									
CU. 11										↰									
CU. 12											↰								
CU. 13																	↰		
CU. 14												↰	↰			↰			
CU. 15												↰	↰						

7.4. Validación de la interfaz de usuario.

Tras la realización de cada incremento, se ha evaluado de forma progresiva la interfaz gráfica de la aplicación.

La evaluación se realiza mediante dos formas: mediante los heurísticos creados por Jacob Nielsen [23] y mediante pruebas del sistema con usuarios reales.

7.4.1. Heurísticos de Nielsen

Los resultados de la evaluación para cada heurístico son los siguientes:

- **Visibilidad del estado del sistema:** se debe mantener a los usuarios informados del estado del sistema, dando una retroalimentación adecuada en un tiempo razonable.
    - La interfaz muestra tanto el estado del sistema como las operaciones que se están realizando en el mismo en tiempo razonable. En concreto, se muestran los distintos estados de anotación de un vídeo sabiendo cuando comienza y cuando termina.
- **Utilizar el lenguaje de los usuarios:** el sistema debe utilizar el lenguaje natural de los usuarios, con palabras o frases que le sean conocidas, evitando vocabulario técnico propio del sistema y desconocido por el usuario.
    - Los términos que se utilizan en la aplicación son adecuados para los usuarios que van a utilizar el sistema.
- **Control y libertad para el usuario:** los usuarios deben tener una forma fácil de salir de funciones del sistema en las que han entrado por error.
    - Es posible salir de cualquier función utilizando la barra de navegación izquierda de la interfaz.
- **Consistencia y estándares:** el lenguaje utilizado por el sistema debe ser el acorde al de la plataforma y ámbito en que está implementado de modo que el usuario no tenga que preguntarse el significado de las palabras y acciones del sistema.

  - Se utiliza un lenguaje adecuado que permite al usuario reconocer los mensajes del sistema.

- **Prevención de errores:** se deben eliminar acciones que puedan llevar al usuario a cometer errores, o advertir sobre el peligro de una acción y preguntar si desea ejecutarla.

  - La única peligrosidad del sistema reside en las acciones de actualización y borrado, ambas remarcadas adecuadamente.

- **Minimizar la carga de memoria del usuario:** el sistema debe evitar que el usuario deba memorizar información, mostrando lo que necesite para realizar las acciones.

  - Toda la información se muestra en la interfaz mediante un lenguaje sencillo y permitiendo consultar toda la información necesaria a la vez.

- **Flexibilidad y eficiencia de uso:** el uso de atajos permiten mejorar la eficiencia de usuarios experimentados sin necesidad de dificultar el uso de usuarios inexpertos.

  - No existen atajos. Sin embargo, las funcionalidades exigen poco esfuerzo en ser ejecutadas, por lo que se realizan rápidamente.

- **Diálogos estéticos y diseño minimalista:** la interfaz no debe contener información irrelevante, ya que cada unidad de información disminuye la visibilidad relativa de la información importante.

  - La interfaz es lo más minimalista posible, por lo que no hay información superflua.

- **Ayudar a los usuarios a conocer, diagnosticar y recuperarse de los errores:** los mensajes de error deben ser claros y sin códigos extraños, permitiendo al usuario identificar perfectamente el error.

  - Siempre que se produce un error se notifica al usuario de forma clara y sencilla, sin utilizar códigos de error.

- **Ayuda y documentación:** lo ideal es que un sistema sea usable sin documentación, no obstante, es posible necesitarla. En ese caso, la documentación debe ser fácil de encontrar, clara y concreta.

    - Existen los manuales de usuarios anexos a este documento. Además, las funcionalidades del sistema son autodescriptivas.

7.4.2. Pruebas de usabilidad

Las pruebas de usabilidad tienen por objetivo comprobar que el software desarrollado resulta fácil de utilizar para los tipos de usuarios objetivo, es decir, para aquellos usuarios para los cuales se ha desarrollado el software.

En el caso de este proyecto, se ha realizado una evaluación[2] con tres usuarios a los cuales se les ha pedido que cubran el **cuestionario SUS** [1]. El principal propósito de este cuestionario es proporcionar un "test fácil de completar, fácil de puntuar y que permitiera establecer comparaciones cruzadas entre productos"[24]. La plantilla de la tabla 7.2 indica las preguntas que conforman el cuestionario y el método de cálculo del mismo. Sin embargo, la puntuación obtenida se multiplica por 0.25 para obtener así una puntuación sobre 10 (más fácil de ponderar). Por otra parte, se ha preguntado a los usuarios posibles mejoras de la interfaz gráfica. Teniendo en cuenta dicha plantilla, las puntuaciones obtenidas por los usuarios para cada pregunta en orden de aparición, son las siguientes:

- Primer usuario: 4, 2, 5, 1, 3, 3, 4, 5, 1, 5. Puntuación total: 8.25
- Segundo usuario: 5, 1, 4, 1, 5, 1, 5, 1, 5, 1. Puntuación total: 9.75
- Tercer usuario: 4, 2, 5, 1, 4, 3, 3, 2, 5, 1. Puntuación total: 8
- Cuarto usuario: 5, 1, 5, 1, 5, 1, 5, 1, 4, 1. Puntuación total: 9.75
- Quinto usuario: 3, 1, 5, 1, 4, 2, 5, 1, 4, 1. Puntuación total: 8.25

[2]Los cuestionarios se refieren únicamente a la interfaz del sistema anotador

	Escala de usabilidad (SUS)				
	1	2	3	4	5
Creo que me gustará visitar con frecuencia este sitio web.					
Encontré el sitio innecesariamente complejo.					
Pienso que el sitio web es fácil de usar.					
Creo que necesitaría apoyo de un experto para utilizar el sitio web.					
Encontré las diversas posibilidades del sitio web bastante bien integradas.					
Pienso que hay demasiada inconsistencia en el sitio web.					
Creo que la mayoría de la gente podría hacer uso del sitio web rápidamente.					
He encontrado el sitio web bastante incómodo de utilizar.					
Me he sentido muy seguro haciendo uso del sitio web.					
Necesitaría aprender muchas cosas antes de poder manejarme con el sitio web.					

Evaluación										
1	2	3	4	5	6	7	8	9	10	**TOTAL**
__ - 1	5 - __	__ - 1	5 - __	__ - 1	5 - __	__ - 1	5 - __	__ - 1	5 - __	

Cuadro 7.2: Plantilla de evaluación de usabilidad.

Así, a partir de los cuestionarios de usabilidad, se mejora la interfaz gráfica desde una interfaz como de la figura 7.1 a una interfaz como la mostrada en la figura 7.2 que es la interfaz que se utiliza actualmente. Los principales cambios residen, principalmente, en mejorar el menú lateral permitiendo acceder a las funcionalidades principales directamente. También se ha producido una mejora de la visualización, modificando los colores oscuros por unos más claros (dando un aspecto minimalista). Por último, se mejora la nomenclatura de las funcionalidades, para que sean más claras.

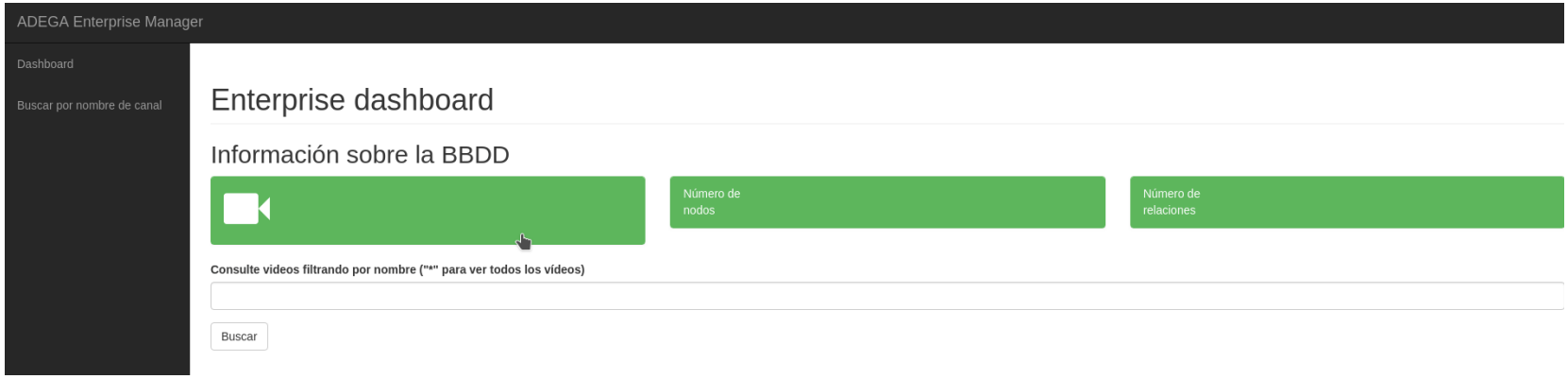

Figura 7.1: Interfaz antigua, antes de las pruebas de usabilidad.

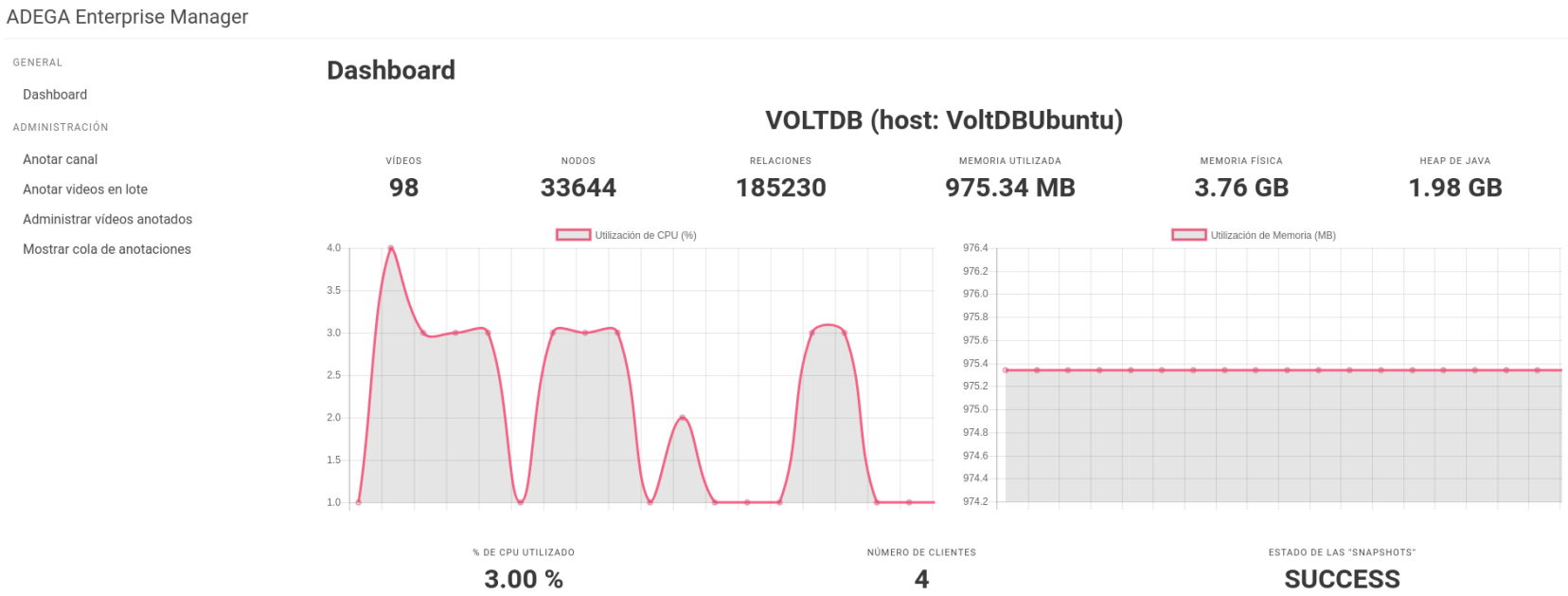

Figura 7.2: Interfaz nueva, después de las pruebas de usabilidad.

Capítulo 8

Conclusiones y trabajo futuro

8.1. Conclusiones

En el presente proyecto se ha desarrollado un sistema pregunta-respuesta (**Enterprise Search**) que permite responder a preguntas del usuario relacionadas con temática médica. Además, se ha desarrollado una aplicación que da soporte al sistema pregunta-respuesta permitiendo la anotación de vídeos y la administración de los mismos (**Adega Enterprise Manager**). Ambos sistemas utilizan el anotador semántico ADEGA que, en su versión actual, utiliza únicamente la ontología MeSH.

Por un lado, se ha implementado un sistema anotador que permite descargar los subtítulos de cualquier vídeo de Youtube siempre que existan, pese a las restricciones impuestas por la propia API de Youtube. Además, se ha implementado un sistema de almacenamiento eficiente y escalable que permite ampliar las capacidades de almacenamiento del sistema simplemente añadiendo nuevos nodos.

Por otra parte, se ha implementado un sistema pregunta-respuesta que, utilizando los grafos devueltos por el anotador semántico ADEGA, permite recomendar vídeos a preguntas abiertas del paciente en cuestión. Para la recomendación se ha utilizado un algoritmo compuesto de dos pasos: un primer filtrado mediante una búsqueda de palabras clave y un refinamiento de los vídeos a recomendar mediante una comparativa de grafos.

Dicha comparativa otorga un índice de similaridad entre cada uno de los vídeos que se procesan en el sistema y el grafo de la consulta del usuario. Dicho índice tiene en cuenta tanto el número de conceptos comunes en ambos grafos como el número de relaciones comunes entre conceptos iguales de los grafos y está basado en un estadístico bien conocido denominado *coeficiente de Soresen-Dice*. La principal ventaja de este método de comparación es que resulta especialmente preciso al tener en cuenta la presencia de todos los conceptos y de las relaciones semánticas en los mismos.

Como el tamaño de los grafos del sistema no es despreciable y el número de vídeos anotados en el sistema también puede ser muy elevado, se ha decidido acelerar la búsqueda realizando un filtrado inicial mediante una búsqueda por palabras. Para ello, se ha utilizado un motor de búsqueda denominado como *ElasticSearch*. Así, se realiza una búsqueda de cada uno de los términos de la consulta del usuario en los grafos asociados a los vídeos. En función del peso de los nodos y de la similaridad proporcionada por el motor de búsqueda se ordenan los vídeos obtenidos de las consultas. Nótese que esto es una puntuación acumulada para cada uno de los términos obtenidos del grafo de la consulta del usuario.

8.2. Trabajo futuro

Aún cuando los resultados del proyecto son satisfactorios, existen múltiples mejoras aplicables al mismo.

En primer lugar, se debería realizar un despliegue distribuido tanto de la base de dato VoltDB como de la base de datos ElasticSearch. Esta mejora permitiría mejores tiempos de recuperación de información así como el escalamiento del sistema en función de los vídeos anotados de forma horizontal y lineal (cuantos más vídeos, más máquinas serán precisas añadir para dar soporte al almacenamiento).

En segundo lugar, sería preciso extender el sistema pregunta-respuesta para utilizar un lenguaje distinto al inglés. Esto requiere únicamente modificar ADEGA para que de soporte a la ontología en el lenguaje que se desea añadir puesto

que el sistema desarrollado no tiene una dependencia fuerte con la ontología que se use. Aún así, habría que cambiar el identificador de la ontología que se utiliza en las llamadas a ADEGA.

En tercer lugar, sería preciso reducir aún más los tiempos de respuesta del sistema pregunta-respuesta. Una de las formas de realizarlo sería mediante una intersección eficiente de los grafos en en algoritmo de similaridad presentado en el proyecto. Para ello, se pueden utilizar librerías como "**Google Guava**" que dan soporte al manejo de grafos de una manera más eficiente.

Por otra parte, la interfaz gráfica para gestionar la anotación de nuevos vídeos debería ser extendida para permitir elegir qué nodos del grafo se almacenan en el sistema y cuales no. También se debería ampliar el sistema de anotación para permitir, desde la propia interfaz gráfica, modificar los parámetros que se suministran al anotador semántico para realizar el proceso de anotación (por ejemplo: permitir modificar la profundidad de los grafos que genera el anotador).

Por último, sería interesante que el sistema pregunta-respuesta tuviese en cuenta el histórico médico del paciente, es decir, si la consulta, por ejemplo, trata sobre hipertensión y el paciente también tiene diabetes el sistema debería tener en cuenta esos dos factores. Esto incluye expandir la consulta con los términos asociados al histórico del paciente. También sería preciso gestionar de alguna manera los pacientes registrados en el sistema (para poder distinguirlos).

Apéndice A

Manual técnico de despliegue

Para realizar el despliegue de las dos bases de datos, no se utilizará ningún sistema de contenedores software como Docker. Esto tiene varios motivos:

- Hay que editar ficheros de configuración y características del kernel para que las bases de datos funcionen. Esto es más sencillo si se hace en el sistema directamente.
- Los binarios de la aplicación son autocontenidos, es decir, no se necesita ninguna instalación para ninguna de las dos bases de datos.
- La comunicación entre bases de datos es más sencilla entre las bases de datos puesto que es más fácil asignar una dirección IP para su uso a las bases de datos (en concreto, la dirección siempre será localhost y lo que varía en realidad son los puertos).

En primer lugar, es preciso realizar un clonado del repositorio del proyecto con el siguiente comando:

```
git clone https://gitlab.citius.usc.es/efren.rama/adega-youtube.git
```

Como dependencia tenemos "python" y "java" si no están instalados:

```
1 sudo apt install python-dev
  sudo apt install openjdk-8-jdk
```

A.1. Instalación de VoltDB

Descargamos la última versión de VoltDB y la descomprimimos con los siguientes comandos:

```
wget https://downloads.voltdb.com/technologies/server/voltdb-latest.
    tar.gz
tar zxvf voltdb*.tar.gz
```

Deshabilitamos las THP (Transparent Huge Tables):

```
echo never >/sys/kernel/mm/transparent_hugepage/enabled
echo never >/sys/kernel/mm/transparent_hugepage/defrag
```

Creamos un directorio para la base de datos y lo inicializamos con el siguiente comando:

```
mkdir database
./voltdb*/bin/voltdb init -D database --config=./deploy/
    despliegueVoltDB.xml
```

El anterior comando utiliza el fichero de configuración que permite inicializar la exportación. Puede ser necesario cambiar la dirección IP del mismo en función de las necesidades. Ahora, arrancamos la base de datos:

```
./voltdb*/bin/voltdb start -D database/
```

Ahora, inicializamos la base de datos ejecutando el script de inicialización (.SQL). Para ello, ejecutamos el siguiente comando:

```
./voltdb-community-8.1.1/bin/sqlcmd
1> file 'init.sql'
```

A.2. Instalación de ElasticSearch

Descargamos la versión 6.2.0 de ElasticSearch y la descomprimimos:

```
wget https://artifacts.elastic.co/downloads/elasticsearch/
    elasticsearch-6.2.0.tar.gz
tar zxvf elasticsearch-6.2.0.tar.gz
```

Aumentamos el tamaño de memoria virtual con el siguiente comando:

```
sysctl -w vm.max_map_count=262144
```

Arrancamos ElasticSearch con el siguiente comando:

```
./bin/elasticsearch
```

A.3. Despliegue de la aplicación

Para la compilación del proyecto y ejecución del .jar, se tiene que ejecutar el siguiente comando:

```
  mvn package
  java -jar target/adegayoutube-0.3.0.jar
```

Apéndice B

Manual de usuario

Si ingresamos a la pantalla principal de la aplicación se tiene la vista mostrada en la figura B.1.

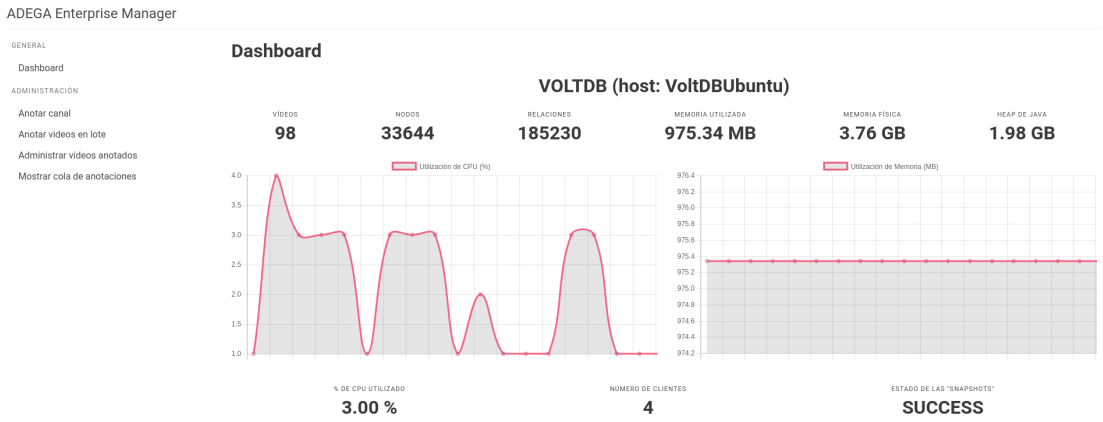

Figura B.1: Pantalla principal de la aplicación.

Para anotar un conjunto de vídeos mediante el método de búsqueda de canales y vídeos, se pulsa en el botón "Anotar canal" del menú lateral izquierdo. Esto muestra una pantalla en la que se pueden consultar canales, tal y como se muestra en la figura B.2.

Si pulsamos en el botón "Ver vídeos" asociado a cada canal es posible ver los vídeos de dicho canal. Así, se mostraría la pantalla de la figura B.3. En dicha pantalla es posible seleccionar distintos vídeos para su anotación utilizando el "check" de la parte superior derecha de cada vídeo. En cada página se muestran

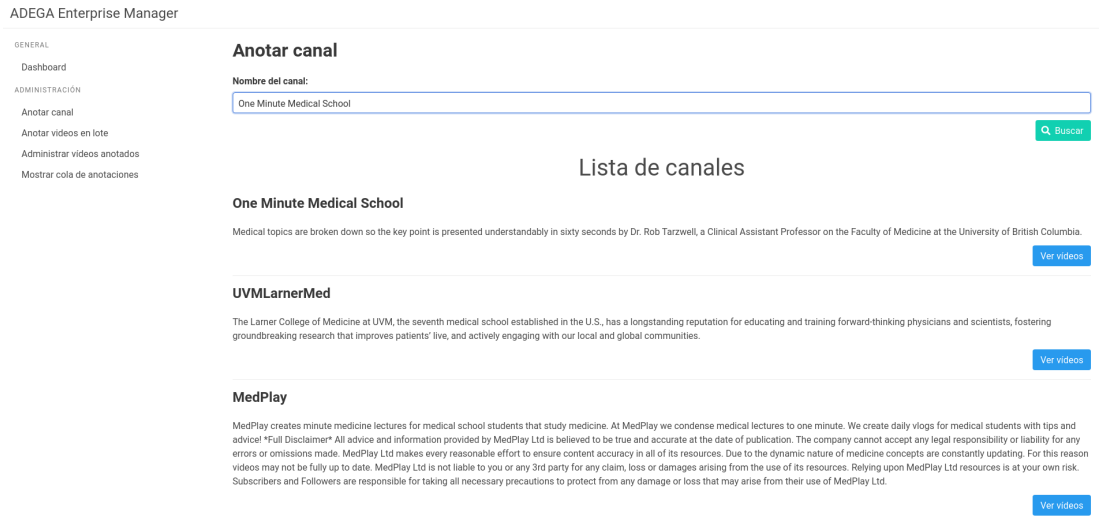

Figura B.2: Pantalla de búsqueda de canales.

10 vídeos pudiendo pasar de página utilizando los botones de paginación de la parte inferior (no mostrados). Para mandar los vídeos a anotación, se utiliza el botón "Anotar seleccionados". Si se pulsa en la miniatura del vídeo es posible visualizarlo.

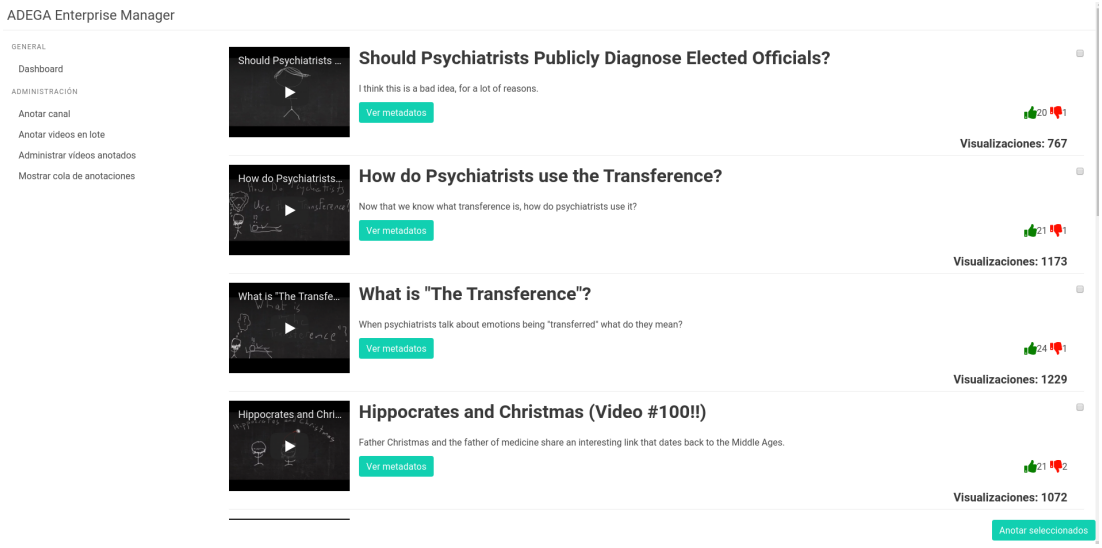

Figura B.3: Pantalla de visualización de vídeos.

Una vez mandados los vídeos a anotación se muestra la siguiente pantalla de la figura B.4.

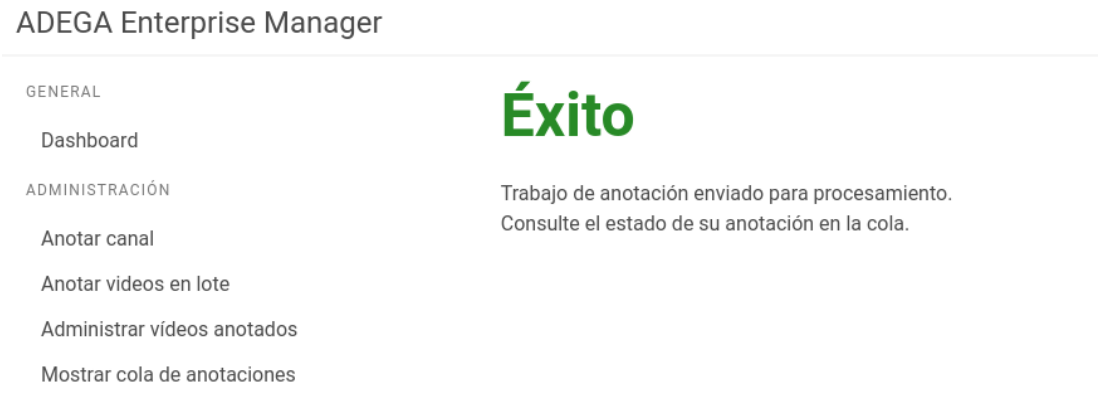

Figura B.4: Pantalla de éxito.

Es posible visualizar los trabajos en ejecución si se pulsa en el botón "Mostrar cola de anotaciones" del menú lateral izquierdo. Así, se muestra la pantalla de la figura B.5.

Figura B.5: Pantalla de visualización de trabajos de anotación.

De cada vídeo, es posible ver el grafo que genera Adega para el mismo pulsando en el botón "Ver grafo". El grafo mostrado tiene el aspecto de la figura B.6. Con la rueda del ratón se puede añadir o quitar zoom y con un movimiento de arrastre del ratón puede moverse la cámara del grafo. Para salir se pulsa en la "X" de la parte superior derecha.

Para acceder a la pantalla de anotación de vídeos en lote, se pulsa sobre el botón "Anotar vídeos en lote" del menú lateral izquierdo. Eso muestra la pantalla de la figura B.7. Simplemente se añaden las URLs separadas por un retorno de carro ("enter"). Para comenzar la anotación se pulsa en "Anotar" y se redirige a la pantalla de la figura B.4.

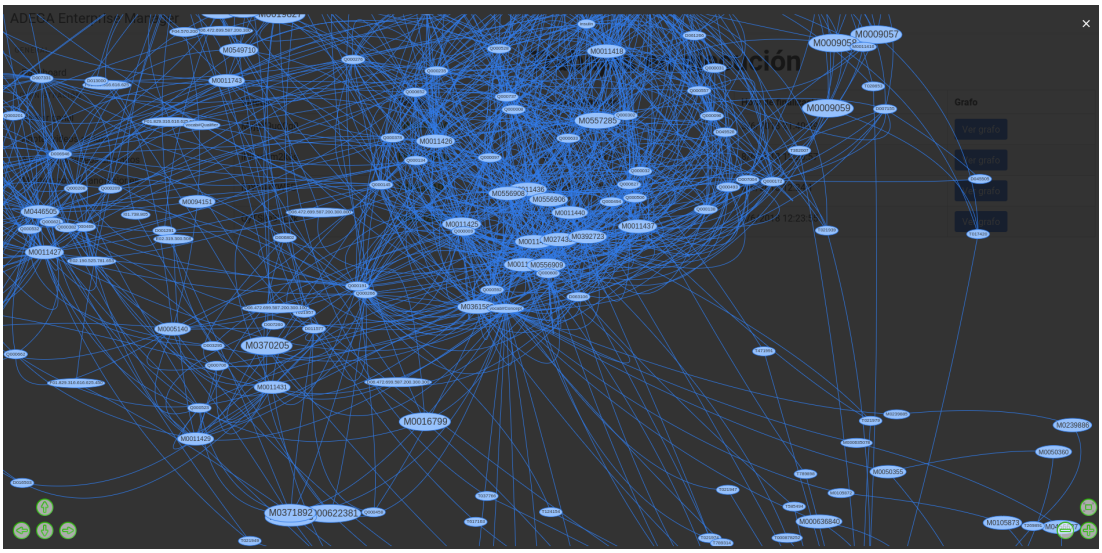

Figura B.6: Pantalla de visualización de grafos.

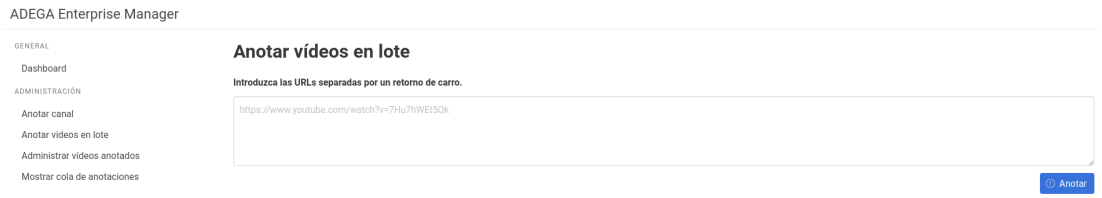

Figura B.7: Pantalla de anotación de vídeos en lote.

Por último, para administrar los vídeos anotados se accede a través del botón "Administrar vídeos anotados" del menú lateral izquierdo. Eso muestra un cuadro de búsqueda que permite dos posibilidades:

- Buscar por "*": muestra todos los vídeos de la base de datos.
- Buscar por una consulta normal: muestra los vídeos relacionados con esa temática.

Una vez realizada la búsqueda, la pantalla es similiar a la de la figura B.8. Si se pulsa sobre el checkbox asociado al vídeo se permiten dos operaciones (para el conjunto de vídeos seleccionados): eliminarlos o actualizarlos. Existe un paginador en la parte inferior que muestra más páginas de vídeos.

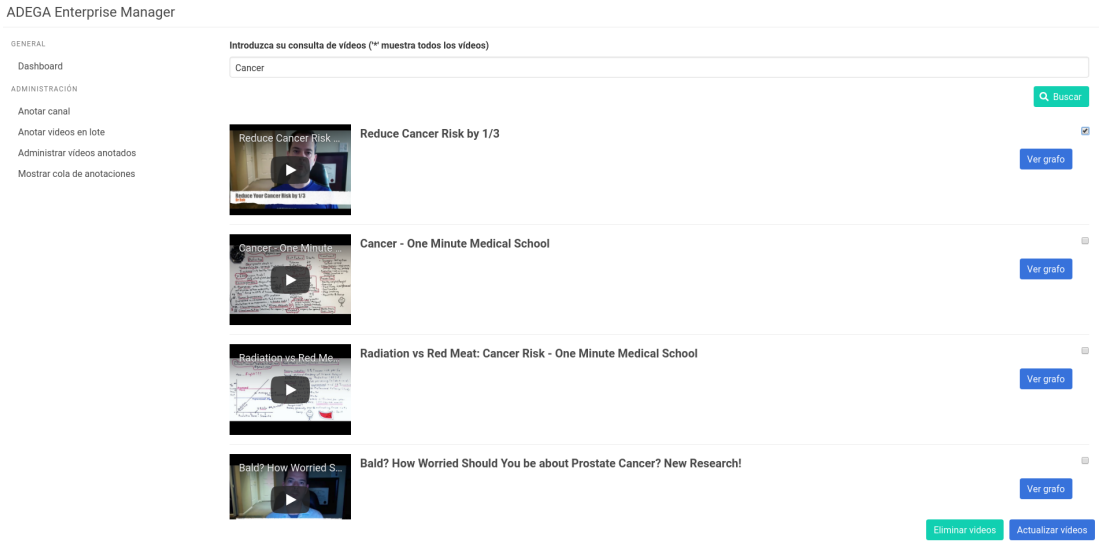

Figura B.8: Pantalla de administración de vídeos anotados.

Al sistema pregunta-respuesta se accede desde la url "(IP y puerto)/adegaQuery". Simplemente se muestra un cuadro de búsqueda que permite mostrar los vídeos relacionados con la consulta en lenguaje natural del usuario, tal y como muestra la figura B.9.

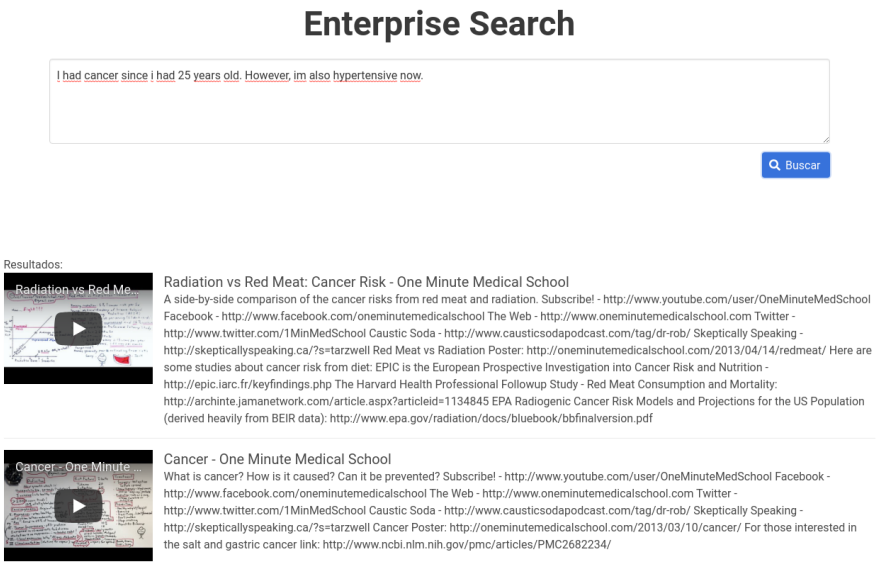

Figura B.9: Pantalla del sistema pregunta respuseta.

Bibliografía

[1] John Brooke. *SUS: A quick and dirty usability scale*. 1996.

[2] Vitae Consultores. *Guía Salarial Sector TI Galicia 2015-2016*. https://es.scribd.com/document/288511179/Guia-Salarial-Sector-TI-Galicia-2015-2016. Consultado el 19-4-18. 2016.

[3] Gaudenz Alder y David Benson. *Draw.io*. http://draw.io. Consultado el 24-6-18.

[4] ElasticSearch. *ElasticSearch as NoSQL Database*. https://www.elastic.co/blog/found-elasticsearch-as-nosql. Consultado el 26-6-18.

[5] ElasticSearch. *Match Query*. https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-multi-match-query.html. Consultado el 20-6-18.

[6] ElasticSearch. *Standard Analyzer*. https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-standard-analyzer.html. Consultado el 20-6-18.

[7] Apache Foundation. *Tomcat 7*. https://tomcat.apache.org/. Consultado el 7-6-18.

[8] Erich Gamma y col. *Design Patterns: Elements of Reusable Object-oriented Software*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995. ISBN: 0-201-63361-2.

[9] Estefania Natalia Otero García. «Descubrimiento de grafos enlazados para la anotación semántica de documentos». Tesis de mtría. Universidad de Santiago de Compostela, abr. de 2017.

[10] Isabel García-Rodeja Gayoso. *USC Investigación*. http://imaisd.usc.es/ftp/oit/documentos/591_gl.pdf. Consultado el 19-4-18. 2008.

[11] Proyecto GNU. *AUCTex*. https://www.gnu.org/s/auctex/. Consultado el 24-6-18.

[12] Manuel Montes-y Gómez, Aurelio López-López y Alexander Gelbukh. «Information Retrieval with Conceptual Graph Matching». En: *Database and Expert Systems Applications*. Ed. por Mohamed Ibrahim, Josef Küng y Norman Revell. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, págs. 312-321.

[13] Google. *AngularJS*. https://angularjs.org/. Consultado el 6-6-18.

[14] Google. *GSON*. https://github.com/google/gson. Consultado el 6-6-18.

[15] Google. *Youtube API V3*. https://developers.google.com/youtube/. Consultado el 7-6-18.

[16] Vicerrectorado de investigación e innovación. *Calculadora de Contratos*. http://imaisd.usc.es/ferramentas/calculadora/calculadoracontratos.asp. Consultado el 19-4-18. 2018.

[17] Project Management Institute. «Guía de los Fundamentos Para La Dirección de Proyectos (Guía del PMBOK). 5ed.» En: 2012. Cap. 5, pág. 125.

[18] Jetbrains. *Intellij IDEA*. https://www.jetbrains.com/idea/. Consultado el 7-6-18.

[19] Erich Gamma y Kent Beck. *JUnit 5*. https://junit.org/junit5/. Consultado e 24-6-18.

[20] Crunch Frog LLC. *Lumzy*. http://lumzy.com. Consultado el 24-6-18.

[21] Microsoft. *Microsoft Project*. https://products.office.com/es-es/project/project-and-portfolio-management-software. Consultado el 7-6-18.

[22] Craig S. Mullins. *What is a NewSQL Database System?* http://www.dbta.com/Columns/DBA-Corner/What-Is-a-NewSQL-Database-System-104489.aspx. Consultado el 12-4-18. 2016.

[23] Jacob Nielsen. *10 Usability Heuristics for Interface Design*. https://www.nngroup.com/articles/ten-usability-heuristics/. Consultado el 24-6-18.

[24] SIDAR. *Algunos cuestionarios conocidos*. https://www.sidar.org/recur/desdi/traduc/es/visitable/nuevos/CuestCon.htm. Consultado el 25-6-18.

[25] Pivotal Software. *Spring Boot*. https://spring.io/projects/spring-boot. Consultado el 6-6-18.

[26] SourceForge. *Google2SRT*. https://google2srt.sourceforge.io/es/. Consultado el 6-6-18.

[27] Richard Stallman. *Emacs*. https://www.gnu.org/software/emacs/. Consultado el 7-6-18.

[28] Sparx Systems. *Enterprise Architect*. http://sparxsystems.com/products/ea/. Consultado el 7-6-18.

[29] WBSTool Team. *WBSTool*. www.wbstool.com. Consultado el 24-6-18.

[30] Jeremy Thomas. *Bulma*. https://bulma.io/. Consultado el 7-6-18.

[31] Linus Torvalds. *Git*. https://git-scm.com/. Consultado el 7-6-18.

[32] Almende B. V. *VisJS*. http://visjs.org/. Consultado el 6-6-18.

[33] VoltDB. *Chapter 10. Availability*. https://docs.voltdb.com/UsingVoltDB/ChapKSafety.php. Consultado el 24-6-18.

[34] VoltDB. *CREATE TABLE*. https://docs.voltdb.com/UsingVoltDB/ddlref_createtable.php. Consultado el 20-6-18.

[35] VoltDB. *Database Benefits & Features*. https://www.voltdb.com/product/features-benefits/. Consultado el 24-6-18.