



Escola Técnica Superior de Enxeñaría
Universidade de Santiago de Compostela

Análisis de sentimiento de términos mediante la recuperación y procesamiento de publicaciones digitales en tiempo real con tecnologías Big Data

Autor:

Rodrigo Martínez Castaño

Directores:

Juan Carlos Pichel Campos

Pablo Gamallo Otero

Grado en Ingeniería Informática

Julio de 2016

Trabajo de Fin de Grado presentado en la *Escola Técnica Superior de Enxeñaría* de la *Universidade de Santiago de Compostela* para la obtención del Grado en Ingeniería Informática.

© 2016 Rodrigo Martínez Castaño, Algunos derechos reservados.

Esta memoria se publica bajo la licencia *Creative Commons Attribution-ShareAlike 4.0 International*¹ (CC BY-SA 4.0), bajo la que se ceden los derechos de distribución y elaboración de obras derivadas para cualquier finalidad —incluso comercial— siempre y cuando se reconozca adecuadamente la autoría, se proporcione un enlace a la licencia y se indique si se han realizado cambios. Además, toda obra derivada deberá difundirse bajo la licencia de la obra original. La figura 3.2 fue cedida para su inclusión en la presente memoria y no se verá afectada por la licencia descrita. Todos los derechos relativos a la misma pertenecen a Aerospike, Inc.

Todo el código fuente que acompaña a la presente memoria es software libre: puede ser redistribuido y/o modificado bajo los términos de la *GNU General Public License* como es publicada por la Free Software Foundation en su versión 3².

¹<https://creativecommons.org/licenses/by-sa/4.0/legalcode>

²<https://www.gnu.org/licenses/gpl-3.0.html>

Agradecimientos

Me gustaría agradecer a mis tutores —Juan Carlos Pichel Campos y Pablo Gamallo Otero— su grandísimo apoyo y confianza, al igual que a mi familia y a mi círculo más cercano, quienes me habéis dado tanta fuerza y ánimos en estas últimas semanas.



D. Juan Carlos Pichel Campos, Profesor del Departamento de Electrónica y Computación de la Universidade de Santiago de Compostela, y **D. Pablo Gamallo Otero**, Profesor del Departamento de Lengua Española de la Universidade de Santiago de Compostela,

INFORMAN:

Que la presente memoria, titulada *Análisis de sentimiento de términos mediante la recuperación y procesamiento de publicaciones digitales en tiempo real con tecnologías Big Data*, presentada por **D. Rodrigo Martínez Castaño** para superar los créditos correspondientes al Trabajo de Fin de Grado de la titulación de Grado en Ingeniería Informática, ha sido realizada bajo nuestra dirección en el Centro Singular de Investigación en Tecnoloxías da Información de la Universidad de Santiago de Compostela.

Y para que así conste a efectos oportunos, expiden el presente informe en Santiago de Compostela a 8 de julio de 2016:

El director,

El codirector,

El alumno,

Juan Carlos Pichel Campos Pablo Gamallo Otero Rodrigo Martínez Castaño

Índice general

1. Introducción	1
1.1. Big Data	2
1.2. NoSQL	2
1.3. Aplicación del PLN sobre publicaciones digitales	2
1.4. Motivación del proyecto	3
1.5. Fuentes de publicaciones y limitaciones	3
1.6. Objetivos	4
2. Gestión del proyecto	7
2.1. Alcance del proyecto	9
2.1.1. Descripción del alcance del producto	9
2.1.2. Criterios de aceptación del producto	9
2.1.3. Entregables del proyecto	10
2.1.4. Exclusiones del proyecto	10
2.1.5. Restricciones del proyecto	10
2.1.6. Supuestos del proyecto	10
2.2. Catálogo de requisitos del sistema	10
2.2.1. Casos de uso del sistema	10
2.2.2. Requisitos	19
2.3. Metodología de desarrollo	37
2.4. Planificación del proyecto	38
2.4.1. Estructura de Descomposición del Trabajo (EDT)	38
2.4.2. Definición de las actividades	40
2.4.3. Cronograma del proyecto: Diagrama de Gantt	43
2.5. Gestión de riesgos	46
2.5.1. Análisis cualitativo	46
2.5.2. Estrategias de acción	47
2.5.3. Catálogo de riesgos	47
2.6. Gestión de la configuración	54

2.6.1.	Identificación de los elementos de la configuración	54
2.6.2.	Control de cambios	55
2.6.3.	Control de versiones	55
2.7.	Análisis de costes	56
2.7.1.	Costes directos	56
2.7.2.	Costes indirectos	57
2.7.3.	Coste total del proyecto	57
3.	Análisis de tecnologías y herramientas	59
3.1.	Tecnologías empleadas en el desarrollo	61
3.1.1.	Librerías	61
3.1.2.	Otras librerías	62
3.1.3.	Frameworks	62
3.1.4.	Otros frameworks	64
3.1.5.	Servidores web	65
3.1.6.	Bases de datos	65
3.1.7.	IDEs de desarrollo	69
3.1.8.	Lenguajes de programación	69
3.2.	Tecnologías empleadas en la documentación	69
3.2.1.	Overleaf	69
3.2.2.	WBSTool	70
3.2.3.	ProjectLibre	70
3.2.4.	Draw.io	70
3.2.5.	StarUML	70
3.2.6.	Enterprise Architect	70
3.2.7.	OwnCloud	70
4.	Diseño de software	71
4.1.	Arquitectura del sistema	72
4.1.1.	Módulo de obtención de publicaciones digitales	73
4.1.2.	Módulo de análisis de sentimiento en tiempo real	75
4.1.3.	Módulo de procesamiento periódico de la LPA	78
4.1.4.	Módulo de cálculo de sentimiento asociado a términos arbitrarios	81
4.1.5.	API para realizar consultas personalizadas y recuperar resultados	81
4.1.6.	Interfaz gráfica	84
4.2.	Patrones aplicados	87
4.2.1.	Entorno de desarrollo	88
4.3.	Diagramas de clases	90

4.3.1. Módulo de obtención de publicaciones digitales	90
4.3.2. Módulo de análisis de sentimiento en tiempo real	92
4.3.3. Módulo de procesamiento periódico de la LPA	94
4.3.4. Módulo de cálculo de sentimiento asociado a términos arbitrarios	94
4.3.5. API para realizar consultas personalizadas y recuperar resultados	95
4.4. Diagramas de interacción	97
4.5. Tablas en los distintos SGBD	99
5. Pruebas y validación	101
6. Conclusiones y trabajo futuro	109
6.1. Conclusiones	110
6.2. Trabajo futuro	111
A. Manual técnico de despliegue	113
A.1. Generación de ejecutables	113
A.2. Arranque de contenedores	113
B. Manual de usuario	117
B.1. Consultar el ratio de positividad de un término arbitrario	117
B.2. Consultar el ratio de positividad de los términos de la LPA	119
B.3. Gestionar LPA	119
B.4. Consultar la evolución en directo de los términos más activos de la LPA	120
C. Glosario	121
Bibliografía	125

Índice de figuras

2.1. Diagrama de casos de uso del sistema.	11
2.2. Ejemplo de tablero Kanban. Autor: Oliver Tacke (Flickr).	38
2.3. Diagrama de Descomposición del Trabajo del proyecto.	39
2.4. Cronograma del proyecto: Diagrama de Gantt (1/2).	44
2.5. Cronograma del proyecto: Diagrama de Gantt (2/2).	45
3.1. Interfaces utilizadas por Docker. (Dominio Público).	65
3.2. Almacenamiento de los datos en espacios de nombres basados en disco y RAM en Aerospike. (Ilustración cedida por Aerospike, ©2016 Ae- rospike, Inc.).	67
4.1. Diagrama de arquitectura del sistema.	72
4.2. Vista de visualización en tiempo real de términos con gran actividad. .	84
4.3. Vista de consulta sobre los términos de la LPA.	85
4.4. Búsqueda personalizada en la interfaz (1/2).	85
4.5. Búsqueda personalizada en la interfaz (2/2).	86
4.6. Gestión de la LPA en la interfaz.	86
4.7. Patrón DAO Factory.	87
4.8. Diagrama de clases con atributos y funciones del Módulo de obtención de publicaciones digitales (1/2)	90
4.9. Diagrama de clases con atributos y funciones del Módulo de obtención de publicaciones digitales (2/2)	91
4.10. Diagrama de clases con atributos y funciones del módulo de análisis de sentimiento en tiempo real (1/2)	92
4.11. Diagrama de clases con atributos y funciones del módulo de análisis de sentimiento en tiempo real (2/2)	93
4.12. Diagrama de clases con atributos y funciones del módulo de procesa- miento periódico de la LPA	94
4.13. Diagrama de clases con atributos y funciones del módulo de cálculo de sentimiento asociado a términos arbitrario	94

4.14. Diagrama de clases con atributos y funciones de la API para realizar consultas personalizadas y recuperar resultados (1/2)	95
4.15. Diagrama de clases con atributos y funciones de la API para realizar consultas personalizadas y recuperar resultados (1/2)	96
4.16. Diagrama de interacción de alto nivel para la consulta de un término arbitrario.	97
4.17. Diagrama de interacción de alto nivel para la consulta de un término de la LPA.	97
4.18. Diagrama de interacción de alto nivel para la visualización de datos en tiempo real.	98
4.19. Diagrama de interacción de alto nivel para añadir un término a la LPA.	98
4.20. Tablas en los SGBD MariaDB, HBase y Aerospike.	99

Índice de cuadros

2.1. Escala de estabilidad de los casos de uso.	11
2.2. Escala de importancia de los casos de uso.	12
2.3. CU-01	12
2.4. CU-02	13
2.5. CU-03	13
2.6. CU-04	14
2.7. CU-05	14
2.8. CU-06	15
2.9. CU-07	16
2.10. CU-08	17
2.11. CU-09	17
2.12. CU-10	18
2.13. CU-11	18
2.14. Códigos asignados a los distintos módulos del sistema.	19
2.15. Escala de importancia de los requisitos.	19
2.16. RF-01-01	23
2.17. RF-01-02	23
2.18. RF-01-03	23
2.19. RF-01-04	24
2.20. RF-01-05	24
2.21. RF-01-06	24
2.22. RF-01-07	24
2.23. RF-02-01	25
2.24. RF-02-02	25
2.25. RF-02-03	25
2.26. RF-02-04	25
2.27. RF-02-05	25
2.28. RF-02-06	26
2.29. RF-03-01	27

2.30. RF-03-02	27
2.31. RF-03-03	27
2.32. RF-03-04	27
2.33. RF-04-01	28
2.34. RF-04-02	28
2.35. RF-04-03	28
2.36. RF-05-01	29
2.37. RF-05-02	29
2.38. RF-05-03	29
2.39. RF-05-04	29
2.40. RF-05-05	30
2.41. RF-05-06	30
2.42. RF-05-07	30
2.43. RF-05-08	30
2.44. RF-05-09	31
2.45. RF-05-10	31
2.46. RF-06-01	32
2.47. RF-06-02	32
2.48. RF-06-03	32
2.49. RF-06-04	32
2.50. RF-06-05	33
2.51. RNF-01	34
2.52. RNF-02	34
2.53. RNF-03	34
2.54. RNF-04	35
2.55. RNF-05	35
2.56. RNF-06	35
2.57. RNF-07	35
2.58. RNF-08	36
2.59. RNF-09	36
2.60. Escala de impacto de los riesgos.	46
2.61. Escala de probabilidad de ocurrencia de los riesgos.	46
2.62. Escala de exposición de los riesgos.	46
2.63. RS-01	48
2.64. RS-02	48
2.65. RS-03	49
2.66. RS-04	49

2.67. RS-05	49
2.68. RS-06	50
2.69. RS-07	50
2.70. RS-08	50
2.71. RS-09	50
2.72. RS-10	51
2.73. RS-11	51
2.74. RS-12	51
2.75. RS-13	51
2.76. RS-14	52
2.77. RS-15	52
2.78. RS-16	52
2.79. RS-17	52
2.80. RS-18	53
2.81. Elementos de la configuración del proyecto.	54
2.82. Elementos de la configuración del proyecto (código).	54
5.1. PR-01	101
5.2. PR-02	101
5.3. PR-03	102
5.4. PR-04	102
5.5. PR-05	102
5.6. PR-06	103
5.7. PR-07	103
5.8. PR-08	103
5.9. PR-09	104
5.10. PR-10	104
5.11. PR-11	104
5.12. PR-12	105
5.13. PR-13	105
5.14. PR-14	105
5.15. PR-15	106
5.16. PR-16	106
5.17. PR-17	106
5.18. PR-18	107
5.19. PR-19	107
5.20. PR-20	107
5.21. PR-22	108

5.22. PR-23	108
-----------------------	-----

Índice de fragmentos de código

4.1. Definición básica de la topología Storm sin multi-instanciación.	76
4.2. Definición de la topología con una pista de paralelización 2 y un máximo de 8 hilos para cada <i>spout/bolt</i>	77
4.3. Fragmento de código en el que se muestra la conexión con HBase desde una aplicación Spark escrita en Scala.	78
4.4. Procesado de los resultados recuperados (1/2).	79
4.5. Procesado de los resultados recuperados (2/2).	80
4.6. Ejemplo de fichero <i>crontab</i> para la ejecución cada minuto del módulo de procesamiento de la LPA.	80
4.7. Cálculo de los separadores de las ventanas y de la ventana correspondiente a una marca temporal	82
4.8. Script de reinicio de los distintos contenedores de desarrollo.	89

Capítulo 1

Introducción

1.1. Big Data	2
1.2. NoSQL	2
1.3. Aplicación del PLN sobre publicaciones digitales	2
1.4. Motivación del proyecto	3
1.5. Fuentes de publicaciones y limitaciones	3
1.6. Objetivos	4

1.1. Big Data

En la sociedad digital moderna, se estima que cada día se crean al menos 2,5 trillones de bytes de datos (2,5 Exabytes), tanto que el 90 % de los datos en todo el mundo fueron creados tan sólo en los últimos dos años. Estos datos provienen de todo tipo de fuentes: sensores utilizados para obtener información sobre el clima, publicaciones en redes sociales, blogs, imágenes digitales y vídeos, etc. Así, por ejemplo, Twitter genera unos 8 Terabytes de datos al día, mientras que Facebook captura unos 100 Terabytes.

Se conoce como Big Data al almacenamiento y procesamiento de estas masivas cantidades de datos en el momento que superan la capacidad del software tradicional para ser manejados y gestionados. Una de las principales características de esta cantidad de información es que, en muchos casos, se encuentra sin estructurar.

1.2. NoSQL

NoSQL, también denominado “No solo SQL”, es un término que hace referencia a una amplia clase de sistemas de gestión de bases de datos con grandes diferencias respecto a los sistemas relacionales (RDBMS). Los datos almacenados no requieren estructuras fijas y son sistemas que suelen escalar bien horizontalmente. Algunas desventajas con respecto a los RDBMS son que no garantizan completamente ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad) y no suelen soportar operaciones de tipo JOIN. Entre los sistemas NoSQL existen tres grandes categorías: columna ancha (HBase, Cassandra, Accumulo, etc.), clave-valor (Aerospike, Redis, Memcached, etc.) y documentales (MongoDB, CouchDB, Solr, etc.). Estos gestores son idóneos para almacenar grandes cantidades de datos por su excelente escalabilidad y rendimiento. Además desaparece el concepto de “esquema”, pudiendo almacenar registros con distintos atributos en una misma tabla.

1.3. Aplicación del PLN sobre publicaciones digitales

El Procesamiento del Lenguaje Natural (PLN) está considerada como una de las metodologías más apropiadas para poder estructurar y organizar la información textual accesible a través de Internet. El procesamiento lingüístico de grandes cantidades de texto es una tarea compleja que requiere el uso de varias subtarefas organizadas en módulos interrelacionados.

Las publicaciones digitales, presentes en redes sociales como Facebook o Twitter, fuentes mediáticas, blogs, etc., son datos que pueden procesarse a gran escala para

extraer información como el sentimiento expresado sobre marcas, entidades, organizaciones, productos, etc.

1.4. Motivación del proyecto

En la actualidad, la reputación en las redes es crucial para las empresas y organizaciones. Conocer la opinión generada sobre sus productos o la propia marca supone la toma de decisiones estratégicas cruciales. Un buen ejemplo es el de la compañía Netflix [1] que produce y selecciona contenidos que sabe que con gran probabilidad de acierto gustarán a la mayor parte de sus clientes o a grandes grupos de perfiles. Para ello procesa las opiniones sobre los contenidos ya existentes, la tasa de finalización de las series y películas, etc. Además dispone de un algoritmo de recomendación basado en los hábitos y gustos de sus usuarios.

En los últimos años han surgido numerosas plataformas enfocadas al manejo, procesamiento y almacenamiento de estas cantidades masivas de datos con una filosofía de crecimiento horizontal. Es decir, se pretende aumentar el rendimiento añadiendo nuevas máquinas similares sin la necesidad de incrementar exclusivamente la capacidad de cómputo de forma vertical. Se trata de plataformas que permiten realizar computación distribuida de forma bastante generalista y siguiendo innovadores modelos de programación. En la actualidad existen decenas de frameworks Big Data. Los más populares son *opensource* y están mantenidos por la fundación Apache con la colaboración de grandes compañías.

Con este proyecto se busca desarrollar una plataforma modular y escalable en *commodity hardware* que permita analizar la opinión de términos arbitrarios a lo largo del tiempo y en directo. Que sea una plataforma modular permitirá la inclusión futura de nuevos componentes para la obtención de nuevas publicaciones provenientes de otras fuentes, el procesamiento de otros idiomas, etc. Se utilizarán métodos alternativos y gratuitos para obtener todas las publicaciones digitales posibles, sin tener que recurrir a opciones de pago no accesibles al gran público, como por ejemplo el acceso al *Firehose* de Twitter. Además debe permitir visualizar de forma gráfica los resultados procesados.

1.5. Fuentes de publicaciones y limitaciones

En esta primera versión del sistema —que constituye el Trabajo de Fin de Grado— se obtendrán y procesarán publicaciones de la red social Twitter en español. Las redes sociales suelen contar con una o varias APIs con notorias restricciones. Twitter permite a través de su “Streaming API” [3] el acceso a un flujo global de tweets. En

este caso la limitación en el flujo global de tweets consiste en el porcentaje de publicaciones recibidas: es una muestra muy poco significativa de los tweets producidos en cada instante, entorno al 1 % del total. El acceso a los flujos de usuarios concretos de forma simultánea está limitado a un pequeño número (*User streams*). Existe un acceso especial a esta API mediante un *endpoint* diferente: el Firehose. Este acceso es de pago, ofrecido directamente por Twitter a través de Gnip, sin intermediarios y bajo demanda. Su elevado precio lo hace exclusivo a grandes empresas. Otra API útil de Twitter es la “Search API” [4]. Permite realizar búsquedas a través de diversos filtros: término de búsqueda, fecha, emisor, tweets recientes o populares, etc. De nuevo surge el problema en las restricciones que impone la empresa: 180 peticiones cada 15 minutos. Debido a las restricciones a través de las APIs, estas tomarán un papel menos relevante en favor de otras técnicas de obtención de datos: el *web scraping*.

A través del portal web de Twitter es posible acceder de forma similar a la “Search API”, simulando la interacción de un usuario. Es por tanto factible realizar búsquedas con numerosos filtros. De esta forma es posible automatizar la obtención de tweets en tiempo real mediante técnicas como un “ataque de frecuencias”, utilizando como filtro para un número elevado de búsquedas simultáneas los términos más frecuentes en un determinado idioma. Este método está limitado por las restricciones del cliente web. Por ejemplo, al realizar una búsqueda, Twitter devuelve una muestra de 20 tweets aleatorios (en caso de haberse producido al menos dicha cantidad) cada 20 segundos (limitación existente también a través de la “Search API”).

1.6. Objetivos

El objetivo principal del proyecto consiste en el desarrollo de un sistema software que permita la obtención automática de grandes cantidades de publicaciones digitales y el procesamiento de los sentimientos asociados a términos variables mediante el uso de módulos PLN integrados en tecnologías Big Data en tiempo real. Se podrán consultar resultados sobre términos arbitrarios. Además, una serie de términos serán procesados de forma periódica y sin intervención para su consulta rápida. Esta lista se denomina Lista de Procesamiento Automático (LPA). De forma más concreta, los objetivos del TFG son los que siguen:

- O1. Desarrollar un módulo de **obtención de publicaciones digitales** que gestione la persistencia de las mismas.
- O2. Desarrollar un módulo que permita el **análisis de sentimiento en tiempo real** de las publicaciones obtenidas.

- O3. Desarrollar un módulo que **procese los términos de LPA periódicamente** sobre las nuevas publicaciones digitales obtenidas desde la última ejecución y que almacene los resultados asociados.
- O4. Desarrollar un módulo para **calcular el sentimiento de términos arbitrarios** en publicaciones ya procesadas mediante el uso de filtros básicos.
- O5. **API para gestionar la ejecución de trabajos personalizados y la consulta de resultados.**
- O6. Desarrollar una **interfaz web que haga uso de la API** permitiendo la ejecución de trabajos personalizados y la consulta de resultados.

Capítulo 2

Gestión del proyecto

2.1. Alcance del proyecto	9
2.1.1. Descripción del alcance del producto	9
2.1.2. Criterios de aceptación del producto	9
2.1.3. Entregables del proyecto	10
2.1.4. Exclusiones del proyecto	10
2.1.5. Restricciones del proyecto	10
2.1.6. Supuestos del proyecto	10
2.2. Catálogo de requisitos del sistema	10
2.2.1. Casos de uso del sistema	10
2.2.2. Requisitos	19
Requisitos funcionales	23
Requisitos no funcionales	34
2.3. Metodología de desarrollo	37
2.4. Planificación del proyecto	38
2.4.1. Estructura de Descomposición del Trabajo (EDT)	38
2.4.2. Definición de las actividades	40
2.4.3. Cronograma del proyecto: Diagrama de Gantt	43
2.5. Gestión de riesgos	46
2.5.1. Análisis cualitativo	46
2.5.2. Estrategias de acción	47
2.5.3. Catálogo de riesgos	47
2.6. Gestión de la configuración	54
2.6.1. Identificación de los elementos de la configuración	54
2.6.2. Control de cambios	55

2.6.3. Control de versiones	55
2.7. Análisis de costes	56
2.7.1. Costes directos	56
2.7.2. Costes indirectos	57
2.7.3. Coste total del proyecto	57

2.1. Alcance del proyecto

En esta sección se incluye una descripción de los procesos necesarios para asegurar que el proyecto incluye el trabajo requerido para completar el proyecto satisfactoriamente.

2.1.1. Descripción del alcance del producto

El producto a construir será un sistema software que se encargará de la obtención de *tweets* en tiempo real, el análisis del sentimiento de cada uno de ellos con la ayuda de una serie de módulos PLN a medida que son recuperados y del almacenamiento de los resultados en una base de datos de forma permanente. Además, el sistema permitirá consultar los resultados obtenidos una vez procesados mediante una interfaz gráfica que haga uso de una API. La consulta de los resultados implica la realización de operaciones sobre las entradas almacenadas a medida que se van recuperando y debido a la gran cantidad de datos generables por el sistema, se emplearán tecnologías Big Data tanto para el procesamiento de las publicaciones digitales como para su almacenamiento y posterior consulta.

2.1.2. Criterios de aceptación del producto

El producto se considerará aceptado si se cumplen los siguientes criterios mínimos:

- Permite la obtención de *tweets* escritos en castellano en tiempo real y su almacenamiento.
- Permite el análisis de sentimiento en castellano de los *tweets* recuperados en tiempo real y la persistencia de los resultados asociados a los *tweets* ya almacenados.
- Permite realizar consultas básicas sobre los datos procesados: operación de agregación de resultados de los *tweets* filtrados por un término arbitrario en un intervalo de tiempo, obteniendo un ratio de positividad.
- Realiza operaciones de agregación de resultados para una lista de términos mutable sobre los *tweets* todavía no procesados de forma periódica y autónoma.
- Es posible ejecutar consultas a través de una API.
- Es posible recuperar los resultados de las consultas realizadas a través de una API.

2.1.3. Entregables del proyecto

Los elementos a entregar tras la finalización del proyecto son:

1. Software funcional que cumple los criterios de aceptación.
2. Código fuente íntegro del producto y de otros elementos secundarios para su puesta en marcha.
3. Ejecutables de los distintos módulos del sistema.
4. La presente memoria junto con los manuales técnicos y de usuario anexados.

2.1.4. Exclusiones del proyecto

Se excluyen del proyecto todos los requisitos que pongan en riesgo la finalización a tiempo del proyecto, ya sea por su complejidad o por su impracticidad en el tiempo establecido. No se pondrá y configurará en marcha un clúster con todas las tecnologías necesarias puestas a punto. Por tanto, el sistema será desarrollado para su uso en un clúster pero no puesto en marcha en producción. No se gestionarán límites de acceso o un control de acceso para el uso del frontal de la aplicación.

2.1.5. Restricciones del proyecto

El sistema podrá ser ejecutado sobre máquinas de arquitectura x86-64 con distribuciones modernas de GNU/Linux. Se utilizarán versiones comunitarias (gratuitas) de tecnologías *opensource* y técnicas de recuperación de *tweets* que no supongan un gasto adicional. El gasto tecnológico sólo se contempla para los RRHH y para el *hardware* necesario.

2.1.6. Supuestos del proyecto

El proyecto se elabora bajo el supuesto del correcto funcionamiento de los módulos de Programación neurolingüística (PLN), ofreciendo un gran nivel de acierto en la estimación del sentimiento asociado a los tweets.

2.2. Catálogo de requisitos del sistema

2.2.1. Casos de uso del sistema

Los casos de uso describen de manera informal la interacción entre los usuarios del sistema, denominados Actores, y el sistema a desarrollar. Los casos de uso no profundizan en detalles técnicos de la implementación o en el comportamiento del

sistema. La interacción de los usuarios con el sistema es sencilla, ya que está limitada a realizar diversos tipos de consultas como se detalla a continuación.

Antes de comentar detalladamente los distintos casos de uso del sistema, se definirán dos escalas mediante las que serán categorizados. Se trata de la escala de estabilidad, que indica la probabilidad de que un caso de uso no tenga que ser modificado durante el desarrollo, y la escala de importancia, que indica la relevancia de su implementación.

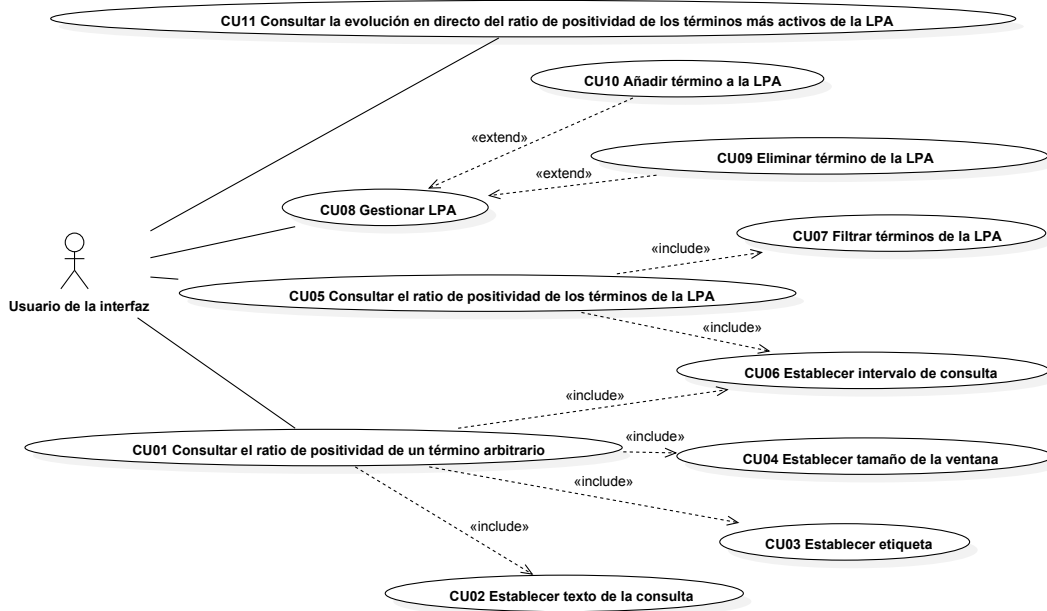


Figura 2.1: Diagrama de casos de uso del sistema.

Alta	El caso de uso está bien definido, al igual que su alcance. Es improbable que sufra modificaciones durante el desarrollo.
Media	El caso de uso en cuestión está bien definido pero su alcance no está correctamente delimitado, podría modificarse a lo largo del proyecto.
Baja	El caso de uso en cuestión puede sufrir importantes cambios debido que quizás no se comprenden en profundidad las implicaciones que tiene.

Tabla 2.1: Escala de estabilidad de los casos de uso.

A continuación se describen las especificaciones de los casos de uso.

Alta	El caso de uso es imprescindible para que los objetivos del producto se satisfagan.
Media	El caso de uso no es considerado crucial, es menos importante pero necesario para que los objetivos se cumplan.
Baja	El caso de uso permite satisfacer los objetivos de forma más robusta, pero no es imprescindible.

Tabla 2.2: Escala de importancia de los casos de uso.

Identificador	CU-01
Nombre	Consultar el ratio de positividad de un término arbitrario
Descripción	El sistema deberá mostrar al usuario el resultado del ratio de positividad de una consulta arbitraria que atiende a unos filtros preestablecidos.
Precondición	Ninguna
Secuencia normal	1. El usuario accede a la sección 'Búsqueda' 2. El usuario introduce el texto de la consulta y el intervalo de búsqueda de forma obligatoria. Opcionalmente también completa la etiqueta y el tamaño de la ventana del intervalo de búsqueda. 3. El usuario ejecuta la consulta presionando el botón 'Obtener resultados'. 4. El sistema procesa la consulta y almacena los resultados. 5. Los resultados son mostrados por pantalla representandos en una gráfica.
Postcondición	Ninguna
Excepciones	- En el paso 2, el usuario establece parámetros incorrectos de filtrado o no completa alguno de los campos obligatorios en el momento de ejecutar la consulta en el punto 3. - En el paso 3, el sistema muestra una advertencia sugiriendo la modificación de los campos del filtro cuando el usuario presiona el botón 'Obtener resultados'. La consulta no se ejecuta. - En el paso 4, si se produce un error interno durante el procesamiento de la consulta, se informa al usuario de que la misma no pudo completarse.
Estabilidad	Alta
Importancia	Alta

Tabla 2.3: CU-01

Identificador	CU-02
Nombre	Establecer texto de la consulta
Descripción	El sistema permitirá introducir el texto de la consulta por pantalla
Precondición	Ninguna
Secuencia normal	1. El usuario introduce el texto de la consulta. 2. El sistema almacena el texto de forma temporal hasta el momento de la ejecución de la consulta.
Postcondición	El sistema almacena de forma temporal el texto de la consulta hasta el momento de su ejecución.
Excepciones	- El usuario introduce una consulta no válida o vacía. El sistema muestra un error sugiriendo la modificación del campo.
Estabilidad	Alta
Importancia	Alta

Tabla 2.4: CU-02

Identificador	CU-03
Nombre	Establecer etiqueta
Descripción	El sistema permitirá introducir la etiqueta de la consulta por pantalla
Precondición	Ninguna
Secuencia normal	1. El usuario introduce el texto de la etiqueta. 2. El sistema almacena el texto de forma temporal hasta el momento de la ejecución de la consulta.
Postcondición	El sistema almacena de forma temporal el texto de la etiqueta hasta el momento de la ejecución de la consulta.
Excepciones	- El usuario introduce una etiqueta no válida. El sistema muestra un error sugiriendo la modificación del campo.
Estabilidad	Alta
Importancia	Baja

Tabla 2.5: CU-03

Identificador	CU-04
Nombre	Establecer tamaño de la ventana
Descripción	El sistema permitirá establecer el tamaño de la ventana de la consulta por pantalla (tiempo entre resultados parciales).
Precondición	Ninguna
Secuencia normal	1. El usuario introduce el tamaño de la ventana. 2. El sistema almacena el tamaño de la ventana de forma temporal hasta el momento de la ejecución de la consulta.
Postcondición	El sistema almacena de forma temporal el tamaño de la ventana hasta el momento de la ejecución de la consulta.
Excepciones	- El usuario introduce un tamaño de ventana no válido. El sistema muestra un error sugiriendo la modificación del campo.
Estabilidad	Alta
Importancia	Media

Tabla 2.6: CU-04

Identificador	CU-05
Nombre	Establecer intervalo de consulta
Descripción	El sistema permitirá introducir un intervalo temporal para filtrar los resultados.
Precondición	Ninguna
Secuencia normal	1. El usuario introduce fecha y hora para los dos extremos del intervalo. 2. El sistema almacena el intervalo de forma temporal hasta el momento de la ejecución de la consulta.
Postcondición	El sistema almacena de forma temporal el intervalo hasta el momento de la ejecución de la consulta.
Excepciones	- El usuario introduce fechas y horas no válidas o vacías o un intervalo no válido. El sistema muestra un error sugiriendo la modificación del intervalo.
Estabilidad	Alta
Importancia	Baja

Tabla 2.7: CU-05

Identificador	CU-06
Nombre	Consultar el ratio de positividad de los términos de la LPA
Descripción	El sistema deberá mostrar al usuario el ratio de positividad de los términos seleccionados de la LPA en el intervalo indicado.
Precondición	Ninguna
Secuencia normal	1. El usuario accede a la sección 'Explorar'. 2. El usuario completa los campos de filtrado: términos pertenecientes a la lista que desea consultar e intervalo de tiempo sobre el que quiere obtener resultados. 3. El usuario ejecuta la consulta presionando el botón 'Obtener resultados'. 4. El sistema procesa la consulta. 5. Los resultados son mostrados por pantalla representados en una gráfica.
Postcondición	Se muestra por pantalla el resultado de la consulta.
Excepciones	- En el paso 2, el usuario establece parámetros incorrectos de filtrado. - En el paso 3, el sistema muestra un error sugiriendo la modificación de los campos del filtro cuando el usuario presiona el botón 'Obtener resultados' y la consulta no se ejecuta. - En el paso 4, si se produce un error interno durante el procesamiento de la consulta, se informa al usuario de que la misma no pudo completarse.
Estabilidad	Alta
Importancia	Media

Tabla 2.8: CU-06

Identificador	CU-07
Nombre	Filtrar términos de la LPA
Descripción	El sistema permitirá al usuario filtrar los términos de la LPA de forma dinámica que quiera consultar.
Precondición	Ninguna
Secuencia normal	1. El usuario selecciona los términos presentes en la LPA sobre los que quiere comparar los resultados. 2. El sistema almacena la lista de términos de forma temporal hasta el momento de la ejecución de la consulta.
Postcondición	El sistema almacena de forma temporal la lista de términos hasta el momento de la ejecución de la consulta.
Excepciones	- El usuario no selecciona ningún término. Por consiguiente el sistema muestra una advertencia al usuario.
Estabilidad	Alta
Importancia	Baja

Tabla 2.9: CU-07

Identificador	CU-08
Nombre	Gestionar LPA
Descripción	El sistema permitirá al usuario añadir y eliminar términos de la Lista de Procesamiento Automático.
Precondición	Para poder eliminar términos deben existir previamente en la LPA.
Secuencia normal	1. El usuario accede a la sección 'Gestionar LPA' 2. El sistema muestra por pantalla la lista de términos actuales y un campo para añadir nuevos términos. 3. El usuario elimina términos presionando el botón 'Borrar' que acompaña a cada elemento de la lista o añade términos introduciendo el nuevo término y presionando el botón 'Añadir'.
Postcondición	Los términos son retirados de la LPA en caso de ser borrados. Si son añadidos pasan a formar parte de la LPA.
Excepciones	- Un término introducido en el paso 2 no es válido o está vacío, motivo por el que el sistema muestra una advertencia informando del suceso. La acción no se completa.
Estabilidad	Media
Importancia	Baja

Tabla 2.10: CU-08

Identificador	CU-09
Nombre	Eliminar término de la LPA
Descripción	El usuario presiona el botón 'Borrar' que acompaña a un término de la LPA.
Precondición	Para poder eliminar términos deben existir previamente en la LPA.
Secuencia normal	1. El usuario presiona el botón 'Borrar' 2. El sistema procesa la petición y elimina dicho término de la LPA
Postcondición	El término se elimina de la LPA.
Excepciones	- Si se produce un error durante la eliminación, se informa del problema al usuario.
Estabilidad	Media
Importancia	Baja

Tabla 2.11: CU-09

Identificador	CU-10
Nombre	Añadir término a la LPA
Descripción	El usuario introduce un nuevo término y presiona el botón 'Añadir' para añadirlo a la LPA.
Precondición	Ninguna
Secuencia normal	1. El usuario introduce el texto del término que desea añadir 2. A continuación presiona el botón 'Añadir' 3. El sistema procesa la petición e incorpora el nuevo término a la LPA
Postcondición	El nuevo término es incorporado a la LPA.
Excepciones	- El usuario introduce un término no válido, vacío o ya presente en la LPA. Al ejecutar el paso 2, se informa al usuario con una advertencia de lo ocurrido.
Estabilidad	Media
Importancia	Baja

Tabla 2.12: CU-10

Identificador	CU-11
Nombre	Consultar la evolución en directo del ratio de positividad de los términos más activos de la LPA
Descripción	El sistema permite al usuario consultar en tiempo real la evolución de los términos de la LPA más relevantes con un criterio propio.
Precondición	Deben existir resultados sobre los términos de la LPA en el sistema.
Secuencia normal	1. El usuario accede a la sección 'Tiempo real' 2. El usuario visualiza una selección de términos cuyos valores se van actualizando en tiempo real.
Postcondición	Ninguna
Excepciones	- Existe un error en el sistema al tratar de recuperar la información, por lo que se muestra al usuario una advertencia.
Estabilidad	Media
Importancia	Baja

Tabla 2.13: CU-11

2.2.2. Requisitos

A continuación se especifican una serie de requisitos del producto a desarrollar. Son fruto de los objetivos mencionados y fueron perfilados en las reuniones previas realizadas con los tutores del proyecto. Se muestran divididos en dos categorías: requisitos funcionales y requisitos no funcionales. Los requisitos funcionales incorporan el prefijo “RF” en su identificador, mientras que los no funcionales hacen lo propio con el prefijo “RNF”. Después del prefijo correspondiente se incorpora un código que referencia al módulo del sistema al que afecta. Estos códigos se muestran en la siguiente tabla:

01	Módulo de obtención de publicaciones digitales.
02	Módulo de análisis de sentimiento en tiempo real.
03	Módulo de procesamiento periódico de la LPA.
04	Módulo de cálculo de sentimiento asociado a términos arbitrarios.
05	API para realizar consultas personalizadas y recuperar resultados.
06	Interfaz gráfica.

Tabla 2.14: Códigos asignados a los distintos módulos del sistema.

Los distintos niveles de importancia de los requisitos se detallan a continuación.

Esencial	El requisito es imprescindible para cubrir los criterios mínimos de aceptación.
Importante	El requisito no es imprescindible para cubrir los criterios mínimos de aceptación pero supone una carencia de calidad en el producto final.
Opcional	El requisito es deseable, pero su no compleción no supone la no aceptación.

Tabla 2.15: Escala de importancia de los requisitos.

Antes de profundizar en la especificación, los requisitos se listan agrupados a continuación:

Enumeración de los requisitos funcionales

Módulo de obtención de publicaciones digitales (01)

RF-01-01 - Recuperar *tweets* a partir de la *Streaming API* de Twitter.

RF-01-02 - Recuperar *tweets* por medio de técnicas de *web scraping* dirigidas hacia el portal web de Twitter.

RF-01-03 - Generar identificadores internos para cada *tweet* generado.

- RF-01-04 - Almacenar el contenido, idioma e identificador interno de cada *tweet* en un *buffer* de salida.
- RF-01-05 - Almacenar todos los datos del *tweet* recuperados en la base de datos de persistencia.
- RF-01-06 - Permitir la configuración de los parámetros de ejecución sin necesidad de recompilación.
- RF-01-07 - Almacenar los identificadores originales de los *tweets* en un *buffer* exclusivo por fuente de extracción.

Módulo de análisis de sentimiento en tiempo real (02)

- RF-02-01 - Adaptación de los módulos PLN a una topología Storm.
- RF-02-02 - Lectura destructiva de publicaciones digitales del *buffer* de salida del primer módulo.
- RF-02-03 - Almacenar en un *buffer* de salida el identificador y el resultado de cada publicación procesada.
- RF-02-04 - Traslado asíncrono de los resultados del *buffer* de salida a la base de datos de persistencia.
- RF-02-05 - Procesamiento en tiempo real de las publicaciones a medida que están disponibles en el *buffer* de entrada.
- RF-02-06 - En caso de no existir un módulo para procesar las publicaciones en un determinado idioma, omitirlas.

Módulo de procesamiento periódico de la LPA (03)

- RF-03-01 - Procesamiento periódico de las publicaciones que contienen al menos un término de la LPA y fechadas *a posteriori* del último procesado.
- RF-03-02 - Almacenamiento de los nuevos resultados agrupados por término en cada ejecución.
- RF-03-03 - Permitir la selección de los parámetros de ejecución sin necesidad de recompilación.

RF-03-04 - Normalizar el contenido de las publicaciones durante la ejecución para evitar falsos negativos a la hora de contrastarlo con los términos de la LPA.

Módulo de cálculo de sentimiento asociado a términos arbitrarios (04)

RF-04-01 - Permitir la selección de los parámetros de ejecución sin necesidad de compilación.

RF-04-02 - Normalizar el contenido de las publicaciones durante la ejecución para evitar falsos negativos a la hora de contrastarlo con la consulta.

RF-04-03 - Almacenar el resultado de cada consulta con los resultados totales y agrupados por ventanas.

API para realizar consultas personalizadas y recuperar resultados (05)

RF-05-01 - Lanzar trabajo de cálculo de sentimiento asociado a términos arbitrarios.

RF-05-02 - Obtener el resultado de una consulta procesada a partir de su identificador.

RF-05-03 - Buscar resultados de consultas realizadas con anterioridad.

RF-05-04 - Añadir un término a la LPA.

RF-05-05 - Eliminar un término de la LPA.

RF-05-06 - Obtener el siguiente resultado disponible a un *timestamp* de un término de la LPA.

RF-05-07 - Obtener el resultado de un término de la LPA en un intervalo temporal.

RF-05-08 - Localizar un determinado número de términos que son los más populares de la LPA en el último espacio de tiempo indicado.

RF-05-09 - Localizar términos de la LPA a partir de su nombre.

RF-05-010 - Obtener todos los términos de la LPA.

Interfaz gráfica (06)

RF-06-01 - Consultar en tiempo real la evolución de los términos más relevantes de la LPA.

RF-06-02 - Mostrar la comparación entre los términos de la LPA seleccionados en un intervalo de tiempo.

RF-06-03 - Añadir y eliminar términos de la LPA.

RF-06-04 - Consultar el ratio de positividad de un término arbitrario.

RF-06-05 - Consultar el ratio de positividad de un término de la LPA.

Enumeración de los requisitos no funcionales

RNF-01 - Almacenamiento de *tweets* en un formato genérico en HBase.

RNF-02 - Identificadores únicos que permitan una distribución entre nodos equitativa y un acceso secuencial eficiente en HBase.

RNF-03 - Posibilidad de acople de nuevos módulos PLN para soportar otros idiomas o características.

RNF-04 - Utilización de HBase como sistema de base de datos para la persistencia de las publicaciones.

RNF-05 - Utilización de MariaDB como sistema de base de datos para la persistencia de los resultados.

RNF-06 - Utilización de tablas Aerospike a modo de buffer.

RNF-07 - Utilización de marcas temporales con precisión de milisegundos para indicar fecha y hora en todos los módulos.

RNF-08 - Utilización de Spark como sistema de agregación de resultados para las consultas sobre las publicaciones almacenadas en HBase.

RNF-09 - Permitir la ejecución paralela de varias instancias de los distintos módulos PLN durante el análisis en tiempo real de las publicaciones.

Requisitos funcionales**Módulo de obtención de publicaciones digitales (01)**

Identificador	RF-01-01
Nombre	Recuperar <i>tweets</i> a partir de la <i>Streaming API</i> de Twitter.
Descripción	Los <i>tweets</i> se recuperarán consumiendo el <i>stream</i> gratuito ofrecido por Twitter en su API pública en directo.
Importancia	Importante

Tabla 2.16: RF-01-01

Identificador	RF-01-02
Nombre	Recuperar <i>tweets</i> por medio de técnicas de <i>web scraping</i> dirigidas hacia el portal web de Twitter.
Descripción	Los <i>tweets</i> se recuperarán mediante <i>web scraping</i> del portal de búsqueda de Twitter haciendo búsquedas paralelas con un conjunto de términos que maximicen el número de resultados: una lista con las palabras más frecuentes del castellano de al menos 100 términos.
Importancia	Esencial

Tabla 2.17: RF-01-02

Identificador	RF-01-03
Nombre	Generar identificadores internos para cada <i>tweet</i> generado.
Descripción	Los identificadores deberán ser independientes de la publicación original y se debe garantizar casi con total seguridad la no generación de dos identificadores repetidos.
Importancia	Esencial

Tabla 2.18: RF-01-03

Identificador	RF-01-04
Nombre	Almacenar el contenido, idioma e identificador interno de cada <i>tweet</i> en un <i>buffer</i> de salida.
Descripción	Se debe mantener un <i>buffer</i> interno para evitar volcados constantes que penalicen el rendimiento.
Importancia	Opcional

Tabla 2.19: RF-01-04

Identificador	RF-01-05
Nombre	Almacenar todos los datos del <i>tweet</i> recuperados en la base de datos de persistencia.
Descripción	Se debe mantener un <i>buffer</i> interno para evitar volcados constantes que penalicen el rendimiento.
Importancia	Esencial

Tabla 2.20: RF-01-05

Identificador	RF-01-06
Nombre	Permitir la configuración de los parámetros de ejecución sin necesidad de recompilación.
Descripción	Listas de términos para <i>scraping</i> , idiomas, tiempo de ejecución, credenciales de uso de la API pública de Twitter, tiempo de espera de cada hilo cada vez que procesa los términos asignados, número de hilos dedicados a <i>scraping</i> , tamaño del <i>buffer</i> y datos de acceso a las bases de datos.
Importancia	Opcional

Tabla 2.21: RF-01-06

Identificador	RF-01-07
Nombre	Almacenar los identificadores originales de los <i>tweets</i> en un <i>buffer</i> exclusivo por fuente de extracción.
Descripción	Los datos deben expirar como mínimo a los 2 días para evitar duplicidades provocadas por las técnicas de <i>scraping</i> y la generación de identificadores internos únicos.
Importancia	Opcional

Tabla 2.22: RF-01-07

Módulo de análisis de sentimiento en tiempo real (02)

Identificador	RF-02-01
Nombre	Adaptación de los módulos PLN a una topología Storm.
Descripción	Adecuación de los módulos PLN existentes a las interfaces utilizadas en Storm para su correcta ejecución y paralelización.
Importancia	Esencial

Tabla 2.23: RF-02-01

Identificador	RF-02-02
Nombre	Lectura destructiva de publicaciones digitales del <i>buffer</i> de salida del primer módulo.
Descripción	Cada vez que los datos son leídos del <i>buffer</i> , estos deben retirarse para evitar el procesamiento múltiple de los mismos contenidos.
Importancia	Opcional. Esencial en caso de implementarse RF-01-04.

Tabla 2.24: RF-02-02

Identificador	RF-02-03
Nombre	Almacenar en un <i>buffer</i> de salida el identificador y el resultado de cada publicación procesada.
Descripción	-
Importancia	Opcional

Tabla 2.25: RF-02-03

Identificador	RF-02-04
Nombre	Traslado asíncrono de los resultados del <i>buffer</i> de salida a la base de datos de persistencia.
Descripción	-
Importancia	Opcional

Tabla 2.26: RF-02-04

Identificador	RF-02-05
Nombre	Procesamiento en tiempo real de las publicaciones a medida que están disponibles en el <i>buffer</i> de entrada.
Descripción	-
Importancia	Esencial

Tabla 2.27: RF-02-05

Identificador	RF-02-06
Nombre	En caso de no existir un módulo para procesar las publicaciones en un determinado idioma, omitirlas.
Descripción	Las instancias de los módulos PLN incluidas deben escuchar el “stream” de publicaciones en castellano, no deben aceptar las publicaciones en otro idioma.
Importancia	Importante

Tabla 2.28: RF-02-06

Módulo de procesamiento periódico de la LPA (03)

Identificador	RF-03-01
Nombre	Procesamiento periódico de las publicaciones que contienen al menos un término de la LPA y fechadas <i>a posteriori</i> del último procesado.
Descripción	La ejecución periódica puede realizarse con Crontab o cualquier sistema similar cada minuto.
Importancia	Esencial

Tabla 2.29: RF-03-01

Identificador	RF-03-02
Nombre	Almacenamiento de los nuevos resultados agrupados por término en cada ejecución.
Descripción	Los resultados referentes a cada términos de la LPA procesados deben almacenarse en la tabla de resultados de la LPA.
Importancia	Esencial

Tabla 2.30: RF-03-02

Identificador	RF-03-03
Nombre	Permitir la selección de los parámetros de ejecución sin necesidad de recompilación.
Descripción	Estos parámetros son el idioma, intervalo temporal y atributos de la base de datos de persistencia para la lectura de publicaciones, de la utilizada para almacenar los resultados y del <i>framework</i> de procesamiento.
Importancia	Opcional

Tabla 2.31: RF-03-03

Identificador	RF-03-04
Nombre	Normalizar el contenido de las publicaciones durante la ejecución para evitar falsos negativos a la hora de contrastarlo con los términos de la LPA.
Descripción	Todos los caracteres no ingleses permutarlos por el caracter inglés más parecido, elimina caracteres no alfanuméricos y convertir todas las letras a minúsculas.
Importancia	Opcional

Tabla 2.32: RF-03-04

Módulo de cálculo de sentimiento asociado a términos arbitrarios (04)

Identificador	RF-04-01
Nombre	Permitir la selección de los parámetros de ejecución sin necesidad de compilación.
Descripción	Se trata del intervalo temporal, idioma, identificador de la consulta, etiqueta, cadena de búsqueda, tamaño de las ventanas en el intervalo y atributos de la base de datos de persistencia para la lectura de publicaciones, de la utilizada para almacenar los resultados y del <i>framework</i> de procesamiento.
Importancia	Opcional

Tabla 2.33: RF-04-01

Identificador	RF-04-02
Nombre	Normalizar el contenido de las publicaciones durante la ejecución para evitar falsos negativos a la hora de contrastarlo con la consulta.
Descripción	Todos los caracteres no ingleses permutarlos por el caracter inglés más parecido, elimina caracteres no alfanuméricos y convertir todas las letras a minúsculas.
Importancia	Opcional

Tabla 2.34: RF-04-02

Identificador	RF-04-03
Nombre	Almacenar el resultado de cada consulta con los resultados totales y agrupados por ventanas.
Descripción	Tanto los resultados parciales como los totales, acompañados del número de publicaciones procesadas para cada caso, deben almacenarse en la tabla de resultados. Estos han de guardarse en formato JSON ligados al identificador del trabajo, la cadena de búsqueda, la marca temporal de la consulta, la etiqueta y el intervalo de inicio y fin a consultar en la tabla de resultados de consultas personalizadas.
Importancia	Importante

Tabla 2.35: RF-04-03

API para realizar consultas personalizadas y recuperar resultados (05)

Identificador	RF-05-01
Nombre	Lanzar trabajo de cálculo de sentimiento asociado a términos arbitrarios.
Descripción	Se invocará al módulo correspondiente con los datos proporcionados en la interfaz y se asignará un identificador único a la tareas, devolviendo la URL de consulta del resultado.
Importancia	Esencial

Tabla 2.36: RF-05-01

Identificador	RF-05-02
Nombre	Obtener el resultado de una consulta procesada a partir de su identificador.
Descripción	Dado el identificador, se devolverá la cadena de consulta asociada al trabajo, el número total de publicaciones y procesadas y el resultado y los resultados parciales y número de publicaciones procesadas parciales por ventana. Todo empaquetado en formato JSON.
Importancia	Esencial

Tabla 2.37: RF-05-02

Identificador	RF-05-03
Nombre	Buscar resultados de consultas realizadas con anterioridad.
Descripción	Permitirá localizar búsquedas realizadas con anterioridad a partir del texto de consulta utilizado.
Importancia	Opcional

Tabla 2.38: RF-05-03

Identificador	RF-05-04
Nombre	Añadir un término a la LPA.
Descripción	El término se añadirá a la tabla que representa la LPA.
Importancia	Opcional

Tabla 2.39: RF-05-04

Identificador	RF-05-05
Nombre	Eliminar un término de la LPA.
Descripción	El término será retirado de la tabla que representa la LPA en caso de existir.
Importancia	Opcional

Tabla 2.40: RF-05-05

Identificador	RF-05-06
Nombre	Obtener el siguiente resultado disponible a un <i>timestamp</i> de un término de la LPA.
Descripción	Se devolverá el siguiente resultado procesado de forma automática para un término dado presente en la LPA.
Importancia	Opcional

Tabla 2.41: RF-05-06

Identificador	RF-05-07
Nombre	Obtener el resultado de un término de la LPA en un intervalo temporal.
Descripción	Proporcionando marcas de agua de inicio y fin y un término de la LPA, se proporcionará el conjunto de resultados parciales existentes junto con el número de publicaciones procesadas al igual que el total para ambos casos.
Importancia	Esencial

Tabla 2.42: RF-05-07

Identificador	RF-05-08
Nombre	Localizar un determinado número de términos que son los más populares de la LPA en el último espacio de tiempo indicado.
Descripción	Proporcionando un número de términos y un espacio temporal en minutos se proporcionará dicho número de términos que fueron los más presentes en publicaciones digitales en el espacio de tiempo indicado inmediatamente anterior al momento actual.
Importancia	Opcional

Tabla 2.43: RF-05-08

Identificador	RF-05-09
Nombre	Localizar términos de la LPA a partir de su nombre.
Descripción	Proporcionando una secuencia de caracteres se devolverán los términos de la LPA que coinciden con el patrón.
Importancia	Opcional

Tabla 2.44: RF-05-09

Identificador	RF-05-10
Nombre	Obtener todos los términos de la LPA.
Descripción	Se devolverán todos los términos existentes en la tabla que representa la LPA.
Importancia	Opcional

Tabla 2.45: RF-05-10

Interfaz gráfica (06)

Identificador	RF-06-01
Nombre	Consultar en tiempo real la evolución de los términos más relevantes de la LPA.
Descripción	Se mostrará en tiempo real la evolución de los tres términos más relevantes en el momento de la primera consulta.
Importancia	Opcional

Tabla 2.46: RF-06-01

Identificador	RF-06-02
Nombre	Mostrar la comparación entre los términos de la LPA seleccionados en un intervalo de tiempo.
Descripción	Se mostrarán en una gráfica los términos seleccionados y su evolución en el intervalo seleccionado de sentimiento y número de publicaciones obtenidas.
Importancia	Opcional

Tabla 2.47: RF-06-02

Identificador	RF-06-03
Nombre	Añadir y eliminar términos de la LPA.
Descripción	Se permitirá añadir y eliminar términos de la LPA de forma visual.
Importancia	Opcional

Tabla 2.48: RF-06-03

Identificador	RF-06-04
Nombre	Consultar el ratio de positividad de un término arbitrario.
Descripción	Indicando los mismos parámetros que consume la API en el requisito RF-05-01, se podrá consultar el resultado del trabajo de cómputo una vez haya terminado.
Importancia	Esencial

Tabla 2.49: RF-06-04

Identificador	RF-06-05
Nombre	Consultar el ratio de positividad de un término de la LPA.
Descripción	Proporcionando los mismos parámetros que consume la API e indicados en el requisito RF-05-07 se podrán visualizar los resultados de forma gráfica en el intervalo proporcionado.
Importancia	Esencial

Tabla 2.50: RF-06-05

Requisitos no funcionales

Identificador	RNF-01
Nombre	Almacenamiento de <i>tweets</i> en un formato genérico en HBase.
Descripción	El objetivo es la adaptabilidad del sistema a nuevas fuentes de publicaciones, evitando almacenar los <i>tweets</i> con sus campos propios y estableciendo unos campos genéricos y relevantes: identificador interno, contenido, idioma, marca de agua original de la publicación, identificación externo de la publicación, nombre del autor, apodo del autor, identificador externo del autor y sentimiento asociado. La fuente de la publicación se incluirá en el identificador.
Importancia	Opcional

Tabla 2.51: RNF-01

Identificador	RNF-02
Nombre	Identificadores únicos que permitan una distribución entre nodos equitativa y un acceso secuencial eficiente en HBase.
Descripción	Los identificadores deberán permitir una lectura secuencial eficiente, por lo que deberán ser de alguna forma secuenciales entre sí y además deberán tener un formato que permita su distribución lo más equitativa posible en un hipotético clúster, evitando la sobre carga de nodos concretos.
Importancia	Opcional

Tabla 2.52: RNF-02

Identificador	RNF-03
Nombre	Posibilidad de acople de nuevos módulos PLN para soportar otros idiomas o características.
Descripción	La inclusión de nuevos módulos para soportar nuevos idiomas debe ser muy sencilla, sin necesidad de realizar casi ninguna modificación.
Importancia	Opcional

Tabla 2.53: RNF-03

Identificador	RNF-04
Nombre	Utilización de HBase como sistema de base de datos para la persistencia de las publicaciones.
Descripción	-
Importancia	Esencial

Tabla 2.54: RNF-04

Identificador	RNF-05
Nombre	Utilización de MariaDB como sistema de base de datos para la persistencia de los resultados.
Descripción	-
Importancia	Esencial

Tabla 2.55: RNF-05

Identificador	RNF-06
Nombre	Utilización de tablas Aerospike a modo de <i>buffer</i> .
Descripción	-
Importancia	Opcional

Tabla 2.56: RNF-06

Identificador	RNF-07
Nombre	Utilización de marcas temporales con precisión de milisegundos para indicar fecha y hora en todos los módulos.
Descripción	En los distintos módulos, cuando se trabaje con unidades de tiempo se utilizarán marcas temporales con milisegundos.
Importancia	Importante

Tabla 2.57: RNF-07

Identificador	RNF-08
Nombre	Utilización de Spark como sistema de agregación de resultados para las consultas sobre las publicaciones almacenadas en HBase.
Descripción	-
Importancia	Esencial

Tabla 2.58: RNF-08

Identificador	RNF-09
Nombre	Permitir la ejecución paralela de varias instancias de los distintos módulos PLN durante el análisis en tiempo real de las publicaciones.
Descripción	Deberá estar soportada la ejecución paralela de varias instancias de los distintos módulos PLN.
Importancia	Opcional

Tabla 2.59: RNF-09

2.3. Metodología de desarrollo

Las metodologías ágiles permiten una puesta en marcha del proyecto rápida, dedicando un tiempo suficiente pero ajustado para la planificación y análisis. Debido a la incertidumbre que generan las diversas tecnologías utilizadas y desconocidas de forma previa al comienzo del proyecto, se estima que una metodología ágil es la más adecuada. Existe mayor flexibilidad y tolerancia a cambios en los requisitos, en las prioridades, mejor gestión del riesgo, etc.

Para el desarrollo del proyecto se utiliza una metodología denominada Kanban. Se trata de una metodología ágil y está basada en el desarrollo incremental con un flujo de trabajo continuo, dividiendo el trabajo en distintas partes. Es una metodología que permite cambiar prioridades rápidamente y una muy buena visión del flujo de trabajo.

En un tablero —tablero Kanban— se establecen una serie de columnas personalizables como podrían ser: “pendiente”, “en curso” y “terminado” en un tablero sencillo. En contraposición con Scrum, el tablero es continuo. A medida que se avanza, las nuevas tareas (mejoras, incidencias o funcionalidades) se acumulan en la sección de tareas pendientes, de forma que se van priorizando periódicamente. Kanban no limita el tiempo para realizar una determinada cantidad de trabajo como sí hace Scrum con los *sprints*. Es un progreso continuo en el desarrollo de las tareas. Esto convierte a Kanban en una metodología más adecuada cuando las iteraciones no son estables o muy predecibles, como es el caso. Sin embargo sí limita la cantidad de trabajo permitido por columna con tareas pendientes de pasar a la siguiente o de completarse —*Work In Progress (WIP) limits*—. Estos límites se establecen de acuerdo a las capacidades de desarrollo en cada una de las columnas normalmente, aunque también pueden establecerse límites por persona o equipo. Establecer estos límites favorece la calidad en el software producido, la toma correcta de decisiones y la compleción de las tareas.

El proyecto en cuestión supone el uso de varias tecnologías nuevas en las que el desarrollador no dispone de experiencia previa, como son Apache Spark, Aerospike y Apache Storm entre otras. Algunos requisitos pueden no ser cubiertos completamente debido al límite temporal y a la incertidumbre que ocasiona el desconocimiento de las tecnologías que supondrán el núcleo del proyecto. Kanban se basa en hacer lo justo y necesario pero con calidad. Es muy flexible y permite priorizar las tareas pendientes según las necesidades del momento, permitiendo dar respuesta a tareas imprevistas. Debido a la incertidumbre comentada es muy probable que surjan numerosos proble-

mas y necesidades de cambio o adaptación para llevar a cabo los requisitos.

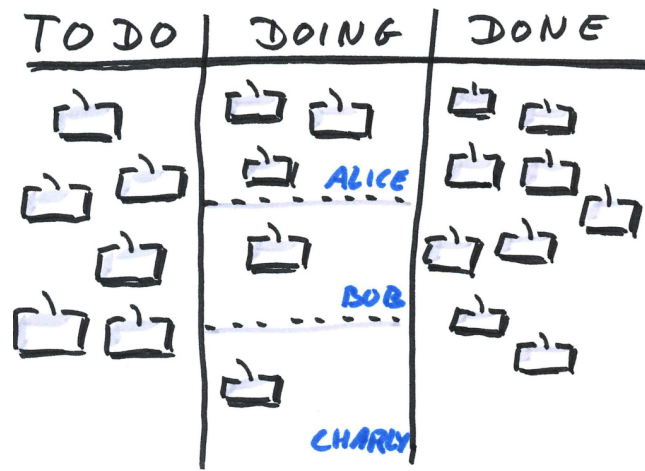


Figura 2.2: Ejemplo de tablero Kanban. Autor: Oliver Tacke (Flickr).

2.4. Planificación del proyecto

La estimación total de horas es de 412,5, total de horas de trabajo autónomo y trabajo presencial mediatne tutorías y evaluación. Por tanto se toma este tiempo como base de la planificación. En total se estima unos 80 días de trabajo y 30 horas semanales. A pesar de utilizar una metodología ágil como Kanban, es importante realizar una estimación aproximada y no involucrarse en el proyecto totalmente a ciegas.

2.4.1. Estructura de Descomposición del Trabajo (EDT)

La Estructura de Descomposición del Trabajo (EDT) es una representación jerárquica de los principales paquetes de actividades que conforman el trabajo a ser ejecutado por el equipo de proyecto. Su propósito es organizar y definir el alcance total aprobado, sirviendo como base para la planificación del proyecto.

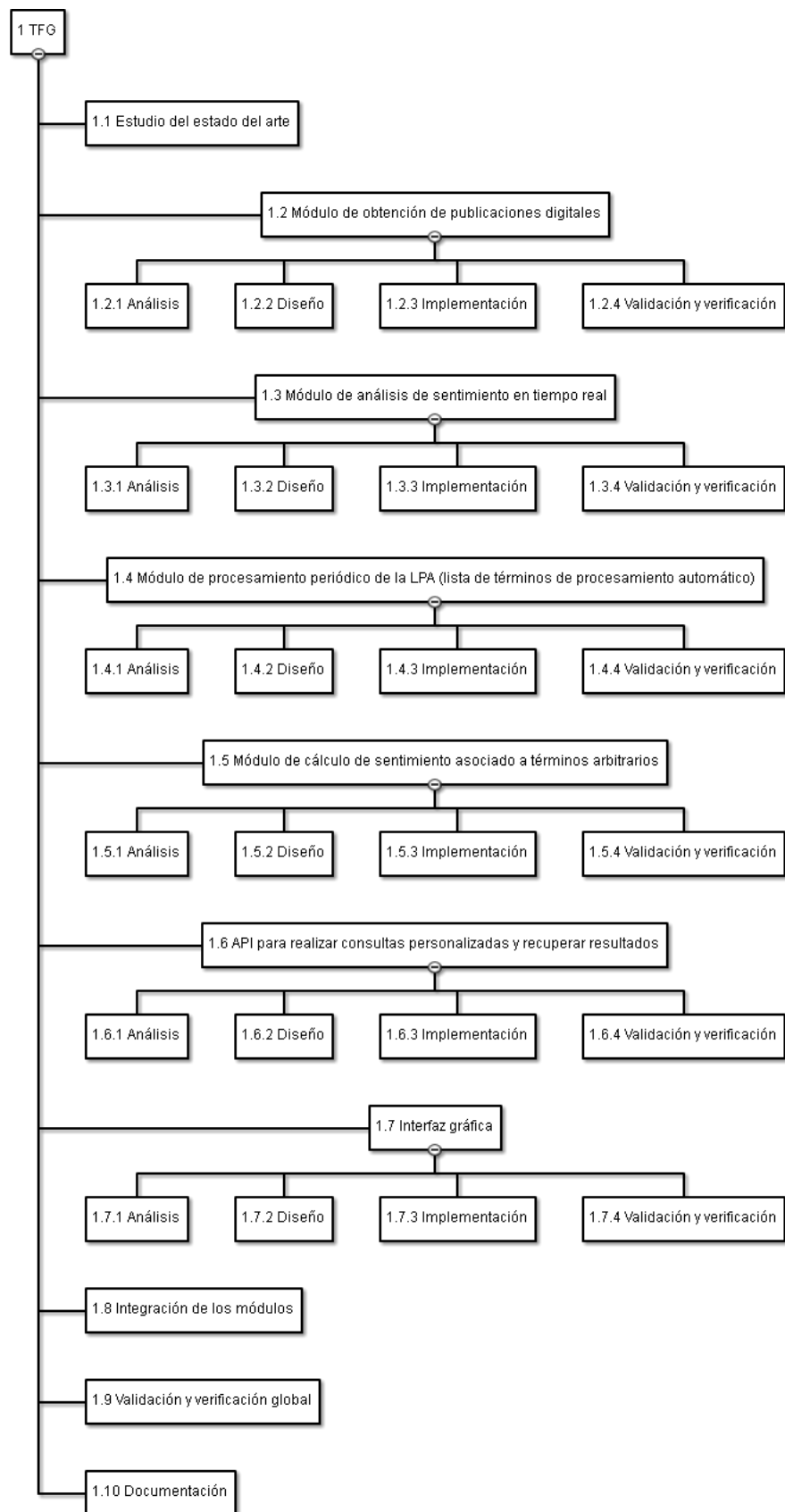


Figura 2.3: Diagrama de Descomposición del Trabajo del proyecto.

2.4.2. Definición de las actividades

- 1.1 Estudio del estado del arte. Estudio y selección de las distintas alternativas tecnológicas disponibles en la actualidad para desarrollar el proyecto. Esto afecta principalmente a los frameworks Big Data que serán utilizados.

Estimación temporal: 60 horas

- 1.2 Módulo de obtención de publicaciones digitales. Este módulo se corresponde con el primer objetivo del proyecto.

- 1.2.1 Análisis. Recopilar, examinar y formular los distintos requisitos del módulo y estudiar las restricciones aplicables.

- 1.2.2 Diseño. Definición precisa de la arquitectura del módulo y definición de cada uno de sus componentes.

- 1.2.3 Implementación. Codificación del módulo.

- 1.2.4 Validación y verificación. Tareas que permiten asegurar que este módulo esté acorde a su especificación y que satisfaga sus objetivos iniciales.

Estimación temporal: 61 horas

- 1.3 Módulo de análisis de sentimiento en tiempo real. Este módulo se corresponde con el segundo objetivo del proyecto.

- 1.3.1 Análisis. Recopilar, examinar y formular los distintos requisitos del módulo y estudiar las restricciones aplicables.

- 1.3.2 Diseño. Definición precisa de la arquitectura del módulo y definición de cada uno de sus componentes.

- 1.3.3 Implementación. Codificación del módulo.

- 1.3.4 Validación y verificación. Tareas que permiten asegurar que este módulo esté acorde a su especificación y que satisfaga sus objetivos iniciales.

Estimación temporal: 61 horas

- 1.4 Módulo de procesamiento periódico de la LPA (lista de términos de procesamiento automático). Este módulo se corresponde con el tercer objetivo del proyecto.

- 1.4.1 Análisis. Recopilar, examinar y formular los distintos requisitos del módulo y estudiar las restricciones aplicables.

- 1.4.2 Diseño. Definición precisa de la arquitectura del módulo y definición de cada uno de sus componentes.

1.4.3 Implementación. Codificación del módulo.

1.4.4 Validación y verificación. Tareas que permiten asegurar que este módulo esté acorde a su especificación y que satisfaga sus objetivos iniciales.

Estimación temporal: 38 horas

1.5 Módulo de cálculo de sentimiento asociado a términos arbitrarios. Este módulo se corresponde con el cuarto objetivo del proyecto.

1.5.1 Análisis. Recopilar, examinar y formular los distintos requisitos del módulo y estudiar las restricciones aplicables.

1.5.2 Diseño. Definición precisa de la arquitectura del módulo y definición de cada uno de sus componentes.

1.5.3 Implementación. Codificación del módulo.

1.5.4 Validación y verificación. Tareas que permiten asegurar que este módulo esté acorde a su especificación y que satisfaga sus objetivos iniciales.

Estimación temporal: 42 horas

1.6 API para realizar consultas personalizadas y recuperar resultados. Este módulo se corresponde con el quinto objetivo del proyecto.

1.6.1 Análisis. Recopilar, examinar y formular los distintos requisitos del módulo y estudiar las restricciones aplicables.

1.6.2 Diseño. Definición precisa de la arquitectura del módulo y definición de cada uno de sus componentes.

1.6.3 Implementación. Codificación del módulo.

1.6.4 Validación y verificación. Tareas que permiten asegurar que este módulo esté acorde a su especificación y que satisfaga sus objetivos iniciales.

Estimación temporal: 32 horas

1.7 Interfaz gráfica. Este módulo se corresponde con el sexto objetivo del proyecto.

1.7.1 Análisis. Recopilar, examinar y formular los distintos requisitos del módulo y estudiar las restricciones aplicables.

1.7.2 Diseño. Definición precisa de la arquitectura del módulo y definición de cada uno de sus componentes.

1.7.3 Implementación. Codificación del módulo.

1.7.4 Validación y verificación. Tareas que permiten asegurar que este módulo esté acorde a su especificación y que satisfaga sus objetivos iniciales.

Estimación temporal: 36 horas

1.8 Integración de los módulos. Todas las tareas que permiten garantizar que los diferentes módulos se integran correctamente en la aplicación.

Estimación temporal: 10 horas

1.9 Validación y verificación global. Todas las tareas de comprobación y análisis que aseguran que el proyecto es acorde a su especificación y cumple las necesidades previstas. Se comprueba que el software se esté construyendo correctamente según los requisitos funcionales y no funcionales (verificación) y que el producto construido sea el correcto (validación), cumpliendo las expectativas iniciales.

Estimación temporal: 10 horas

1.10 Documentación. Todas las tareas relacionadas con la documentación del proyecto; la documentación de los distintos módulos y la memoria global.

Estimación temporal: 62 horas

2.4.3. Cronograma del proyecto: Diagrama de Gantt

El diagrama de Gantt es un cronograma cuyo objetivo es representar el tiempo de dedicación previsto para un conjunto de tareas o actividades a lo largo de un tiempo total determinado. Este conjunto de tareas, agrupadas en paquetes de trabajo, forman en su conjunto el trabajo a realizar para la compleción del proyecto. En el diagrama se muestra también la asignación de recursos para cada tarea.

Las tareas son secuenciales y están agrupadas en paquetes de trabajo asociados a los distintos módulos del sistema. La tarea de documentación, que incluye la creación de esta memoria supone una dedicación parcial durante todo el marco temporal del proyecto. Por esta razón se le asigna un porcentaje constante (15%) y al resto de actividades el restante (85%).

Las reuniones realizadas con los tutores coinciden con las etapas de análisis de cada módulo. Por ello se les asigna un 10% en dichas tareas. A continuación se muestran dos figuras con el Diagrama de Gantt del proyecto.

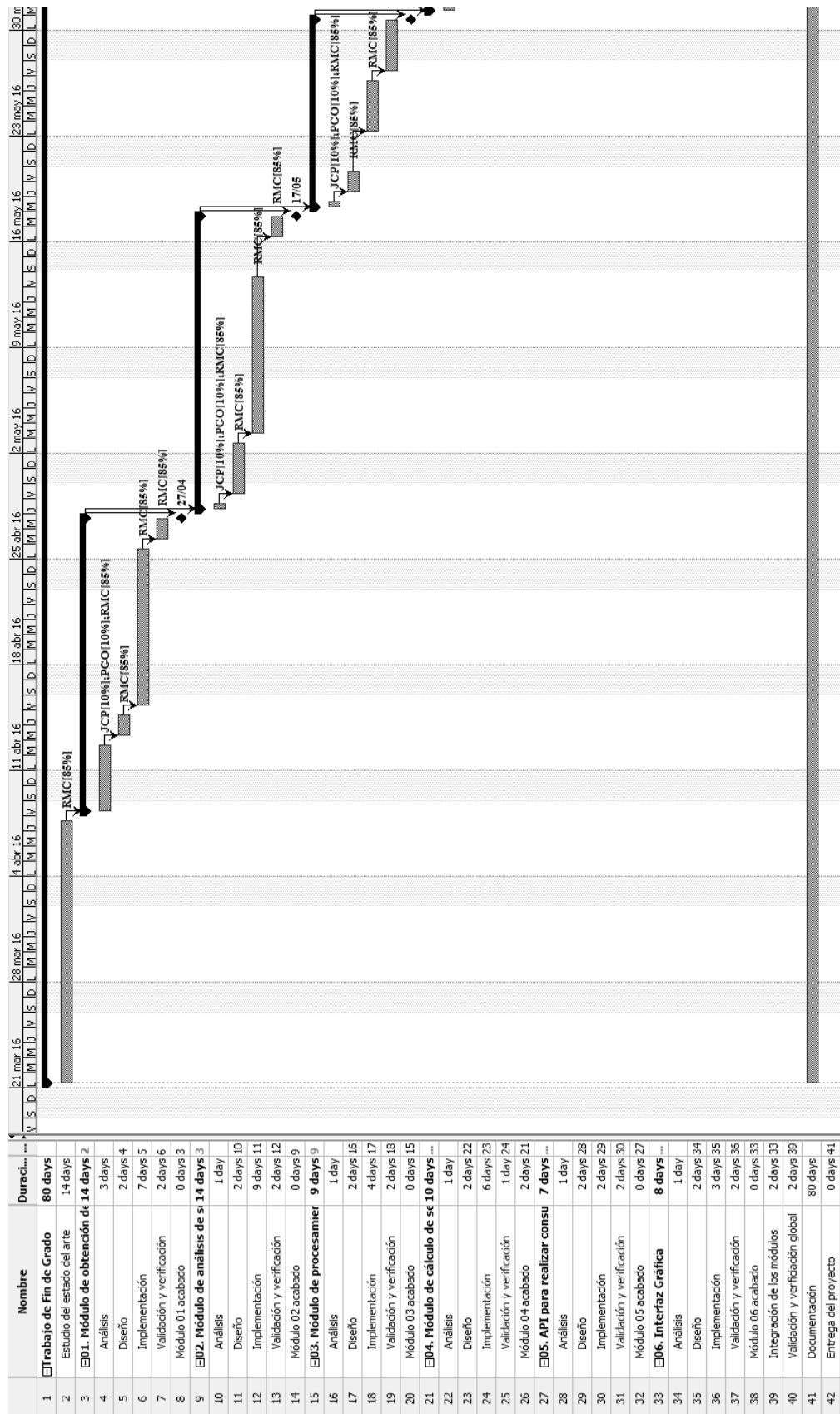


Figura 2.4: Cronograma del proyecto: Diagrama de Gantt (1/2).

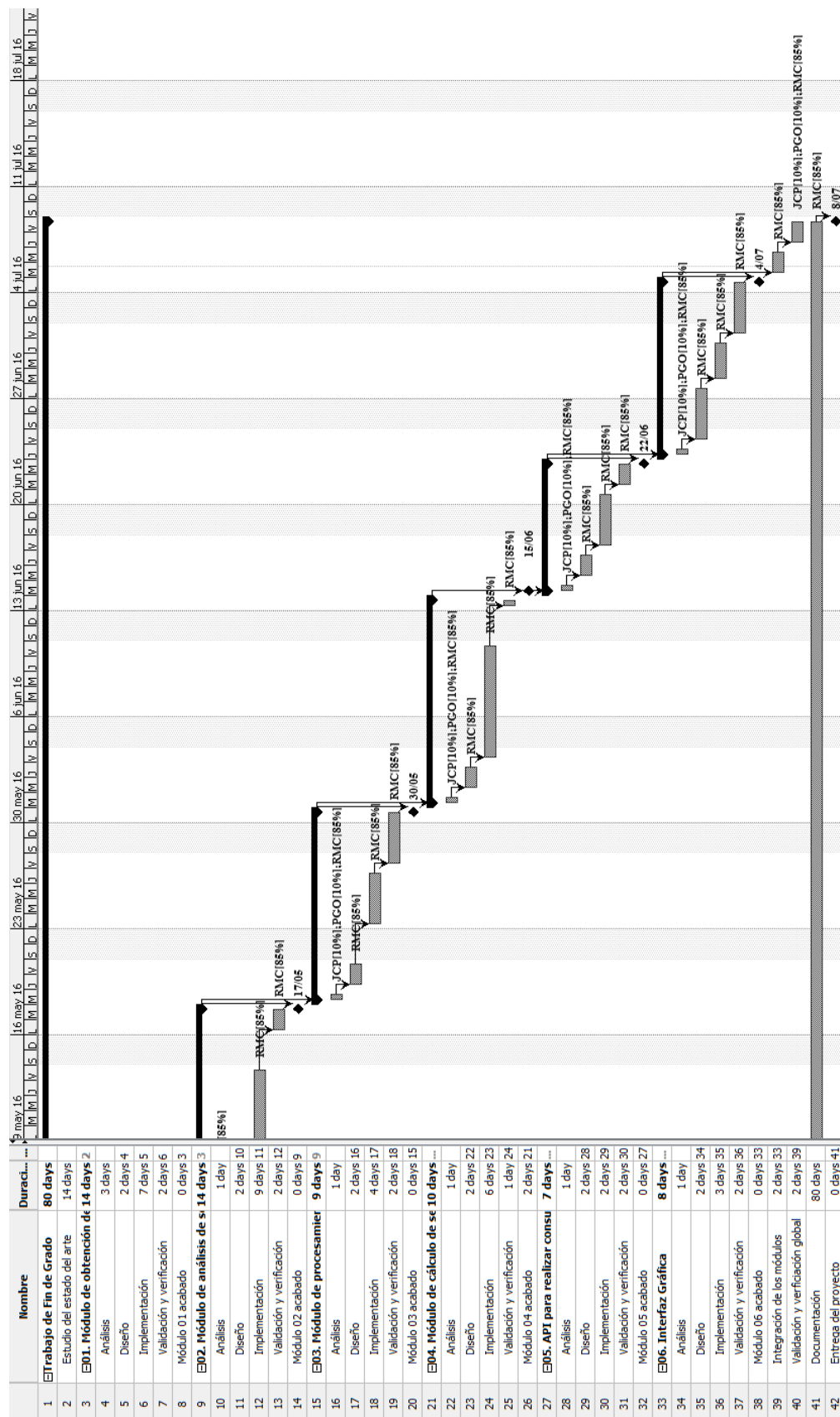


Figura 2.5: Cronograma del proyecto: Diagrama de Gantt (2/2).

2.5. Gestión de riesgos

En este apartado se identificarán y categorizarán posibles riesgos relacionados con el proyecto que puedan poner en peligro el desarrollo del mismo.

2.5.1. Análisis cualitativo

Se emplearán dos escalas principales: nivel de impacto en el proyecto y probabilidad de ocurrencia. El nivel de impacto se refiere al grado de afectación del proyecto en cuanto a sus posibilidades de compleción en caso de concreción del riesgo. La probabilidad es la estimación de probabilidad del riesgo de concretarse. Además, también se indicará la exposición al riesgo. Este es un parámetro que combina la probabilidad y el impacto. La exposición al riesgo se define a continuación, igual que el resto de escalas utilizadas.

Desastroso	En caso de ocurrencia, supondría la cancelación directa del proyecto.
Grave	La ocurrencia podría provocar alteraciones moderadas en la planificación o en el alcance.
Moderado	La ocurrencia podría provocar alteraciones leves en la planificación o en el alcance sin afectar a la consecución de los objetivos del proyecto.
Leve	La ocurrencia podría provocar pequeños retrasos sin llegar a alterar las fechas establecidas de la planificación.

Tabla 2.60: Escala de impacto de los riesgos.

Muy alta	Probabilidad estimada igual o superior al 75 %.
Alta	Probabilidad estimada entre el 50 y el 75 %.
Media	Probabilidad estimada entre el 25 y el 50 %.
Baja	Probabilidad estimada igual o inferior al 25 %.

Tabla 2.61: Escala de probabilidad de ocurrencia de los riesgos.

		Probabilidad			
		Muy alta	Alta	Media	Baja
Impacto	Desastroso	<i>Muy alta</i>	<i>Muy alta</i>	<i>Muy alta</i>	<i>Media</i>
	Grave	<i>Muy alta</i>	<i>Alta</i>	<i>Alta</i>	<i>Media</i>
	Moderado	<i>Alta</i>	<i>Media</i>	<i>Media</i>	<i>Baja</i>
	Leve	<i>Media</i>	<i>Media</i>	<i>Baja</i>	<i>Baja</i>

Tabla 2.62: Escala de exposición de los riesgos.

2.5.2. Estrategias de acción

- **Minimización.** Las estrategias de minimización buscan reducir el impacto del riesgo cuando aparecen.
- **Prevención.** Las estrategias de prevención buscan evitar que el riesgo aparezca.
- **Plan de contingencia.** Los planes de contingencia definen acciones a realizar cuando el riesgo aparece.
- **Aceptación del riesgo.** Se asume el impacto del riesgo y sus consecuencias.

2.5.3. Catálogo de riesgos

Antes de profundizar en la especificación, los riesgos se listan agrupados a continuación:

Índice de riesgos

RS-1 - Planificación demasiado optimista.

RS-2 - Codificación apurada retrasa el proyecto debido a errores de programación.

RS-3 - Imposibilidad de avanzar a buen ritmo por la aparición de numerosos problemas.

RS-4 - El mecanismo de obtención de tweets no es suficientemente potente para incorporarse a un sistema en tiempo real.

RS-5 - Análisis de sentimiento erróneo debido a una mala adaptación de los módulos PLN.

RS-6 - Pérdida de publicaciones digitales.

RS-7 - Duplicación de publicaciones.

RS-8 - La velocidad de procesamiento de publicaciones y resultados no es escalable y por tanto el sistema no sería eficaz en un entorno real.

RS-9 - Robo del material de trabajo.

RS-10 - Inclemencias meteorológicas.

RS-11 - Enfermedades.

RS-12 - Fallos eléctricos.

RS-13 - Retraso del proyecto.

RS-14 - Falta de precisión en la planificación.

RS-15 - Errores en pruebas.

RS-16 - Problemas de integración.

RS-17 - Inexperiencia.

RS-18 - Incumplimiento de plazos.

Especificación de riesgos

Identificador	RS-01
Nombre	Planificación demasiado optimista.
Descripción	La planificación asigna tiempo menor del necesario a las tareas y por tanto se podría producir un retraso en cadena. Aparece cuando uno de los paquetes de trabajo se retrasa, haciendo que se retrase el inicio del siguiente.
Probabilidad	Alta
Impacto	Grave
Exposición	Alta
Tipo de estrategia	Prevención

Tabla 2.63: RS-01

Identificador	RS-02
Nombre	Codificación apurada retrasa el proyecto debido a errores de programación.
Descripción	El desarrollo demasiado veloz puede propiciar la generación de errores que acaben por ralentizar el flujo. Aparece cuando el desarrollo de un módulo acaba con un margen moderado respecto a la planificación y se detectan numerosos problemas a continuación.
Probabilidad	Alta
Impacto	Moderado
Exposición	Media
Tipo de estrategia	Minimización.

Tabla 2.64: RS-02

Identificador	RS-03
Nombre	Imposibilidad de avanzar a buen ritmo por la aparición de numerosos problemas.
Descripción	La aparición de problemas relacionados con la configuración de los sistemas utilizados puede ralentizar el proyecto. El riesgo se manifiesta cuando el desarrollo no avanza durante dos días seguidos.
Probabilidad	Media
Impacto	Grave
Exposición	Alta
Tipo de estrategia	Plan de contingencia.

Tabla 2.65: RS-03

Identificador	RS-04
Nombre	El mecanismo de obtención de tweets no es suficientemente potente para incorporarse a un sistema en tiempo real.
Descripción	No es posible alimentar al sistema con suficientes publicaciones. Se manifiesta cuando el sistema procesa las publicaciones un 50 % más rápido que la velocidad de obtención.
Probabilidad	Baja
Impacto	Desastroso
Exposición	Media
Tipo de estrategia	Aceptación

Tabla 2.66: RS-04

Identificador	RS-05
Nombre	Análisis de sentimiento erróneo debido a una mala adaptación de los módulos PLN.
Descripción	No se obtiene el mismo resultado para la misma entrada al procesarla con los módulos originales que con los módulos adaptados a la nueva infraestructura.
Probabilidad	Baja
Impacto	Grave
Exposición	Media
Tipo de estrategia	Prevención

Tabla 2.67: RS-05

Identificador	RS-06
Nombre	Pérdida de publicaciones digitales.
Descripción	El riesgo se manifiesta cuando más de un 5 % de las publicaciones obtenidas no llegan a ser procesadas.
Probabilidad	Media
Impacto	Leve
Exposición	Baja
Tipo de estrategia	Minimización

Tabla 2.68: RS-06

Identificador	RS-07
Nombre	Duplicación de publicaciones.
Descripción	Se manifiesta cuando más de un 1 % de las publicaciones obtenidas se encuentra duplicada.
Probabilidad	Media
Impacto	Moderado
Exposición	Media
Tipo de estrategia	Prevención

Tabla 2.69: RS-07

Identificador	RS-08
Nombre	La velocidad de procesamiento de publicaciones y resultados no es escalable y por tanto el sistema no sería eficaz en un entorno real.
Descripción	Aparece cuando no es posible escalar horizontalmente la capacidad de procesamiento en tiempo real y de procesamiento de resultados añadiendo una o más máquinas a un hipotético clúster.
Probabilidad	Baja
Impacto	Desastroso
Exposición	Media
Tipo de estrategia	Prevención

Tabla 2.70: RS-08

Identificador	RS-09
Nombre	Robo del material de trabajo.
Descripción	El riesgo se manifiesta cuando la estación de trabajo es extraída sin permiso, evitando la continuación temporal del proyecto y la pérdida de tiempo durante la reconfiguración de un nuevo equipo.
Probabilidad	Baja
Impacto	Grave
Exposición	Media
Tipo de estrategia	Plan de contingencia, minimización.

Tabla 2.71: RS-09

Identificador	RS-10
Nombre	Inclemencias meteorológicas.
Descripción	Una tormenta eléctrica tendría un leve impacto debido a las medidas de protección ya instaladas. En el caso de una inundación el resultado sería la posible inutilización del equipo de trabajo y en cualquier caso, la imposibilidad de continuar el trabajo en el lugar habitual de manera temporal.
Probabilidad	Baja
Impacto	Grave
Exposición	Media
Tipo de estrategia	Plan de contingencia, minimización.

Tabla 2.72: RS-10

Identificador	RS-11
Nombre	Enfermedades.
Descripción	Una enfermedad paralizaría el desarrollo del proyecto debido a la existencia de un único ejecutor.
Probabilidad	Media
Impacto	Grave
Exposición	Alta
Tipo de estrategia	Prevención

Tabla 2.73: RS-11

Identificador	RS-12
Nombre	Fallos eléctricos.
Descripción	Un fallo eléctrico provocaría la incapacidad temporal de seguir trabajando y posibles daños en los equipos utilizados.
Probabilidad	Baja
Impacto	Grave
Exposición	Media
Tipo de estrategia	Prevención, minimización.

Tabla 2.74: RS-12

Identificador	RS-13
Nombre	Retraso del proyecto.
Descripción	La fecha de entrega del producto final puede sufrir variaciones frente a la establecida inicialmente.
Probabilidad	Alta
Impacto	Grave
Exposición	Alta
Tipo de estrategia	Prevención

Tabla 2.75: RS-13

Identificador	RS-14
Nombre	Falta de precisión en la planificación.
Descripción	La falta de precisión puede dar lugar a la aparición de tareas no contempladas que alarguen la ejecución de sus tareas padre.
Probabilidad	Alta
Impacto	Moderado
Exposición	Media
Tipo de estrategia	Prevención, minimización.

Tabla 2.76: RS-14

Identificador	RS-15
Nombre	Errores en pruebas.
Descripción	Durante las pruebas de la aplicación es probable no detectar algunos errores que afecten al funcionamiento del sistema debido a no ser lo suficiente exhaustivas.
Probabilidad	Baja
Impacto	Moderado
Exposición	Baja
Tipo de estrategia	Prevención, minimización

Tabla 2.77: RS-15

Identificador	RS-16
Nombre	Problemas de integración.
Descripción	Aunque los módulos funcionen de forma individual es posible que aparezcan problemas a la hora de orquestarlos conjuntamente y que el comportamiento no sea el esperado.
Probabilidad	Media
Impacto	Desastroso
Exposición	Muy alta
Tipo de estrategia	Prevención

Tabla 2.78: RS-16

Identificador	RS-17
Nombre	Inexperiencia.
Descripción	Debido al uso de nuevas tecnologías que no se había utilizado con anterioridad por el ejecutor del proyecto, es posible que se produzca una ralentización que afecte a las fechas previstas para los paquetes de trabajo.
Probabilidad	Alta
Impacto	Moderado
Exposición	Media
Tipo de estrategia	Minimización, plan de contingencia

Tabla 2.79: RS-17

Identificador	RS-18
Nombre	Incumplimiento de plazos.
Descripción	Los plazos establecidos para los hitos podrían no ser respetados.
Probabilidad	Media
Impacto	Desastroso
Exposición	Muy alta
Tipo de estrategia	Minimización

Tabla 2.80: RS-18

2.6. Gestión de la configuración

2.6.1. Identificación de los elementos de la configuración

Un Elemento de Configuración del Software (ECS) es cada uno de los elementos del proyecto que se verán afectados por la Gestión de la Configuración. Los elementos de la configuración del proyecto se identifican a continuación:

1. Planificación del proyecto
2. Diseño y Arquitectura del software (diagramas)
3. Memoria del proyecto
4. Código fuente de los distintos módulos

	Descripción	Código
1	Documento con la planificación temporal del proyecto creado con Project Libre.	PLT
2	Documento elaborado con Draw.io con los distintos diagramas que representan el diseño y la arquitectura del proyecto.	DAS
3	La presente memoria del proyecto.	MEM
4	Código fuente de los distintos módulos del proyecto	Ver cuadro 2.82

Tabla 2.81: Elementos de la configuración del proyecto.

Los códigos asignados a los paquetes de código fuente se corresponden con el nombre del repositorio utilizado en GitLab. Varios módulos pueden compartir repositorios.

ID	Módulo del sistema	Código
02	Módulo de obtención de publicaciones digitales	dpsa-polypus
03	Módulo de análisis de sentimiento en tiempo real. El segundo repositorio contiene el código del demonio que traslada los resultados del <i>buffer</i> a la base de datos de persistencia.	dpsa-storm, dpsa-btop
04	Módulo de procesamiento periódico de la LPA	dpsa-spark
05	Módulo de cálculo de sentimiento asociado a términos arbitrarios	dpsa-spark
06	API para realizar consultas personalizadas y recuperar resultados	dpsa-api
07	Interfaz gráfica	dpsa-gui

Tabla 2.82: Elementos de la configuración del proyecto (código).

2.6.2. Control de cambios

El desarrollo de las tareas que componen el alcance del proyecto son de carácter individual, por lo que la gestión de la configuración se limita al control de versiones de los distintos elementos de la configuración. Solo una persona modificará los elementos de la configuración.

2.6.3. Control de versiones

Se ha decidido crear un repositorio por módulo asignados al mismo grupo de trabajo en la plataforma GitLab¹, cuyo principal servicio es la utilización de Git de forma remota. Como excepción, los dos módulos que realizan cálculos de agregación sobre las publicaciones digitales almacenadas en la base de datos de persistencia se encuentran en el mismo repositorio. El único desarrollador trabaja sobre la rama “master” de los repositorios. Cada cambio en los repositorios se acompaña de un mensaje explicativo. Las modificaciones insuficientes para definir una tarea completada también se guardan diariamente para evitar riesgos relacionados con la pérdida de información en el equipo local.

Para controlar el versionado del documento de diagramas se utilizó Draw.io² utilizando el almacenamiento de Google Drive³. Existe un historial de versiones que se genera a medida que se edita el documento, pero también es posible crear puntos de revisión, similares a los *commits* de Git.

Para el control de versiones de la memoria se utiliza el sistema integrado de Overleaf⁴, el editor que fue empleado. Al igual que Draw.io, va construyendo progresivamente el historial de modificaciones. También permite crear versiones asignando una descripción. Internamente funciona con un repositorio Git.

Para el documento de planificación se generan copias del documento actual, añadiendo al nombre del archivo la versión que le corresponda. Tanto para este documento como para otros documentos como diagramas elaborados con aplicaciones locales, se utiliza la plataforma OwnCloud⁵ en un servidor propio para sincronizar la información en una ubicación remota.

¹<https://gitlab.com>

²<https://draw.io>

³<https://drive.google.com>

⁴<https://overleaf.com>

⁵<https://owncloud.org>

2.7. Análisis de costes

Este apartado pretende analizar el coste económico que conlleva el desarrollo del proyecto. Para los costes en material informático no aplicables en su integridad al proyecto, se realiza una proporción de 80 días de uso sobre el tiempo de amortización establecido —4 años, 1460 días— si todavía se encuentra en su *vida útil*. Esto supondría un 0,055 % aproximado del coste total.

2.7.1. Costes directos

1. **Estación de trabajo.** Se trata de un Lenovo M92P con un microprocesador Intel(R) Core(TM) i5-3470T de doble núcleo con una velocidad de reloj de hasta 3,60 GHz con 4GB de RAM y propiedad del desarrollador. Adquirido de segunda mano previamente al inicio del proyecto por 200 euros y ampliando sus características: nuevo disco duro SSD de 250GB y un segundo módulo de memoria RAM de 8GB. Por lo que el coste total asciende a 334 euros. Suponiendo una vida útil de cuatro años para el equipo, esto supone un coste proporcional de unos **18,4 euros**.
2. **Dos monitores de 24 y 22 pulgadas.** El primero es propiedad del Centro Singular de Investigación en Tecnoloxías da Información. No es de reciente adquisición y el coste proporcional se considera despreciable. En cuanto al segundo monitor, es propiedad del desarrollador y fue adquirido en 2014 por 99 euros. Considerando de nuevo una vida útil estándar de 4 años, el coste atribuible al proyecto es de alrededor de **5,5 euros**.
3. **Software y tecnologías utilizadas.** Todo el software y tecnologías empleadas son de uso gratuito: Apache Spark, Apache HBase, Apache Storm, Aerospike, NetBeans, IntelliJ IDEA, Docker, AngularJS, Oracle JDK, etc. Se utilizaron las versiones de prueba de StarUML 2 y Enterprise Architect. Para eliminar los límites de la versión gratuita del servicio Overleaf, empleado para la redacción de la memoria, se pagó una mensualidad de la modalidad Premium cuyo coste para estudiantes asciende a **3,9 euros**.
4. **Sistema operativo.** Para la mayor parte del desarrollo se utilizó Ubuntu, distribución GNU/Linux sin licencia de uso. También se utilizó Windows 10 y el coste de la licencia está incluido en el coste de la estación de trabajo.
5. **Mano de obra.** Se asignará un sueldo al desarrollador en base a los salarios actuales. Para ello se ha consultado una estimación en el portal Experteer.es⁶

⁶http://www.experteer.es/salary_calculator/

en el área de Santiago de Compostela para recién titulados en el área de Tecnologías de la información. La estimación es de 18.000 euros anuales brutos a jornada completa. Ya que el producto no va a ser comercializado, se ignoran costes relativos a la Seguridad Social que tendría que pagar el empleador a mayores o la cuota de autónomos correspondiente. Suponiendo 52 fines de semana, hay alrededor de 261 días laborables. En consecuencia, el coste del desarrollador asciende a unos 8,6 euros la hora. Estimando unas 412 horas de trabajo aproximadas, el coste por mano de obra asciende a **3.543,2 euros**.

Los costes directos suman un total aproximado de **3.571 euros**.

2.7.2. Costes indirectos

Además de los costes mencionados, es necesario tener en cuenta costes indirectos como el consumo eléctrico o la conexión a Internet. Como certifica la Secretaria General de la Universidade de Santiago de Compostela [6], se aplicará el 20 % de los costes directos para calcular los costes indirectos. Esto es alrededor de **714,2 euros**.

2.7.3. Coste total del proyecto

En total, el coste de la ejecución del proyecto asciende a **4285,2 euros**.

Capítulo 3

Análisis de tecnologías y herramientas

3.1. Tecnologías empleadas en el desarrollo	61
3.1.1. Librerías	61
Twitter4j	61
HTMLUnit	61
LinguaKit / CitiusSentiment [8]	61
3.1.2. Otras librerías	62
3.1.3. Frameworks	62
Hadoop	62
Spark	63
Storm	63
Docker	64
3.1.4. Otros frameworks	64
3.1.5. Servidores web	65
3.1.6. Bases de datos	65
Aerospike	65
HBase	68
MariaDB	68
3.1.7. IDEs de desarrollo	69
3.1.8. Lenguajes de programación	69
3.2. Tecnologías empleadas en la documentación	69
3.2.1. Overleaf	69
3.2.2. WBSTool	70

3.2.3. ProjectLibre	70
3.2.4. Draw.io	70
3.2.5. StarUML	70
3.2.6. Enterprise Architect	70
3.2.7. OwnCloud	70

La mayor parte del desarrollo relativo al procesado de información se hizo sobre Ubuntu 16.04, debido en gran parte al soporte nativo de las distribuciones nativas GNU/Linux de Docker, sistema de virtualización utilizado para la realización de un desarrollo ágil sobre las distintas tecnologías que se comentarán a continuación. El desarrollo de la parte web se hizo en Windows 10.

3.1. Tecnologías empleadas en el desarrollo

A continuación se describen las tecnologías empleadas para el desarrollo del proyecto y su implicación en el mismo.

3.1.1. Librerías

Twitter4j

Twitter4j¹ es una librería no oficial Java para la API de Twitter. Se utilizó en el módulo de obtención de publicaciones digitales para consumir la *Streaming API* como canal secundario de obtención de *tweets*. Se utilizó la versión 4.0.4.

HTMLUnit

HTMLUnit² es una librería para Java que modela documentos HTML y proporciona una API que permite invocar páginas, rellenar formularios, hacer clic en enlaces, etc. Soporta JavaScript y permite simular navegadores. Es la herramienta principal utilizada para realizar *Scraping web*. Se utiliza para la obtención de tweets a partir de la invocación del servicio de búsqueda de la aplicación web de Twitter, como se detallará más adelante. Se utilizó la versión 2.20.

LinguaKit / CitiusSentiment [8]

LinguaKit³ es una herramienta para el procesamiento del lenguaje natural que contiene varios módulos. Se utilizó una versión de 2014 que disponía de la mayoría de los módulos utilizados portados a Java gracias al software Perldoop⁴ [9]. Los módulos *Normalizer* y *Sentiment* fueron traducidos manualmente en su momento con la ayuda del software ya mencionado, disponiendo así, de todos los módulos necesarios

¹<http://twitter4j.org>

²<http://htmlunit.sourceforge.net/>

³<https://github.com/citiususc/Linguakit>

⁴<https://github.com/citiususc/perldoop>

para hacer el análisis de sentimiento en Java. Se utiliza la librería JRegex⁵ de forma indirecta.

Los módulos utilizados son *Sentence segmentation*, *Normalizer*, *Tokenizer*, *Splitter*, *NER*, *PoS —Part-of-Speech— Tagger* y *Sentiment*. Conectando las salidas de cada módulo al siguiente, en el orden de mención, se obtiene un resultado para el texto introducido: positivo, negativo o neutro.

1. Sentence segmentation: divide el texto en frases.
2. Normalizer: normaliza, emitiendo etiquetas semánticas, elementos del contenido de la publicación como abreviaturas, emoticonos, etc. que se suelen utilizar en las redes sociales.
3. Tokenizer: cada frase se divide en una secuencia de tokens (o palabras).
4. Splitter: separa las palabras compuestas por contracciones. Por ejemplo: *comerse = comer + se*, *del = de + el*, etc.
5. NER: reconocimiento de entidades nombradas que pueden contener varias palabras. Por ejemplo *Santiago de Compostela*.
6. PoS Tagger: realiza el etiquetado morfo-sintáctico. Por ejemplo: *nombre masculino singular*, *adjetivo femenino plural*, etc.
7. Sentiment: utiliza un léxico polarizado y un clasificador Naïve Bayes entrenado con tweets para proporcionar la salida final ya comentada.

3.1.2. Otras librerías

Angular-Chart⁶ es una librería para AngularJS que permite la utilización de la librería JavaScript Chart.js⁷ para la representación de datos en gráficas de forma nativa. Spark-HBase Connector⁸ permite realizar escaneos en HBase desde Spark de forma paralela, dando soporte al uso de prefijos en las *rowkeys*.

3.1.3. Frameworks

Hadoop

Apache Hadoop⁹ [12] es la base de la mayoría de los proyectos de Big Data. Es un *framework* para el almacenamiento, procesamiento y análisis de grandes cantidades

⁵<http://jregex.sourceforge.net/>

⁶<https://jtblin.github.io/angular-chart.js/>

⁷<http://www.chartjs.org>

⁸<https://github.com/nerdammer/spark-hbase-connector>

⁹<http://hadoop.apache.org>

de datos. Es escalable, distribuido y tolerante a fallos. Está basado en los conceptos desarrollados por Google a finales de la década de los 90 y los primeros años del presente siglo. GFS (*Google File System*) es un sistema de ficheros distribuidos que sirvió como inspiración para HDFS (*Hadoop Distributed Filesystem*). MapReduce [10] se presentó como una metodología de procesamiento distribuido que también sirvió como inspiración para una implementación propia en Hadoop que corre sobre YARN, el planificador y gestor de recursos del clúster Hadoop.

Spark

Apache Spark¹⁰ [11] permite describir flujos de trabajo arbitrarios con una, dos o más etapas de procesamiento. Es un modelo mucho más flexible que Hadoop MapReduce. Procesa los datos en memoria, mientras que Hadoop MapReduce escribe en disco tras superar el tamaño de un pequeño buffer en etapas intermedias del procesamiento. Spark se carga y permanece en memoria, mientras que Hadoop MapReduce mata los procesos creados cuando un trabajo termina. Spark debería ejecutar los programas de forma notablemente más rápida mientras la memoria RAM permita cargar los datos necesarios para ejecutar un determinado trabajo. Es necesario en el proyecto para realizar consultas sobre las publicaciones obtenidas realizando operaciones de filtrado y agregación.

Storm

Apache Storm¹¹ [2] es un *framework* de cómputo en tiempo real y distribuido que permite realizar el procesamiento de flujos de datos no delimitados. Es decir, una vez en ejecución puede consumir información nueva al instante según la va recibiendo. Existen cuatro conceptos básicos que ayudan a describir su funcionamiento.

Topología. Una topología es un grafo de computación donde los nodos representan cálculos individuales y las aristas representan los datos fluyendo entre nodos.

Tupla. Los nodos de una topología envían datos entre ellos en forma de tuplas. Una tupla es una lista ordenada de valores, donde cada valor tiene asignado un nombre. Un nodo puede emitir tuplas de forma opcional.

Stream. Un “stream” es una secuencia no delimitada de tuplas entre dos nodos de una topología. Los nodos pueden aceptar uno o más “streams” como entrada.

Spout. Un “spout” es una fuente de un “stream” en una topología. Normalmente leen datos de una fuente externa y emiten tuplas dentro de la topología.

¹⁰<http://spark.apache.org>

¹¹<http://storm.apache.org>

Bolt. Los “bolts” consumen “streams” y realizan algún cálculo o transformación sobre los datos recibidos. Finalmente emiten de forma opcional una o varias nuevas tuplas a sus “streams” de salida.

Docker

El uso de contenedores Docker¹² [13] permite obtener las ventajas de la virtualización (aislamiento, seguridad, flexibilidad, agilidad, portabilidad, etc.) sin penalizar al rendimiento de E/S característico en otros métodos reales de virtualización. Docker utiliza características de aislamiento de recursos del kernel de Linux, tales como *cgroups* y *namespaces* para permitir que contenedores independientes se ejecuten en una misma máquina Linux. Los contenedores se pueden definir como espacios que proveen un entorno virtual con su propio espacio de procesos y redes.

Los contenedores Docker son imágenes Docker desplegadas que se encuentran paradas o en ejecución. Una vez arrancado un contenedor a partir de una imagen, es posible realizar cambios directamente sobre el contenedor en ejecución y guardar el estado, generando una nueva imagen. Cada imagen está formada por un conjunto de imágenes apiladas. Esto se debe a que normalmente, para crear una imagen, utilizaremos como base otra ya existente. La otra posibilidad para genera imágenes es a partir de un Dockerfile. En este caso, cada instrucción del Dockerfile también genera una imagen. Los Dockerfiles son scripts que detallan una serie de modificaciones que se ejecutarán sobre una imagen base principalmente. Además de realizar dichas modificaciones, es posible realizar otras acciones como exponer puertos, montar volúmenes del anfitrión, etc. Siempre es necesario indicar una imagen base, incluso para partir de cero (imagen “scratch”). Normalmente escogeremos una imagen base de alguna distribución como pueden ser CentOS, Debian o Ubuntu. Existen imágenes oficiales de Docker basadas en las principales distribuciones GNU/Linux actuales.

3.1.4. Otros frameworks

Además de los *frameworks* mencionados —los más importantes del proyecto—, se utilizaron otros para distintas partes del proyecto. Para el desarrollo de la interfaz gráfica con el patrón MVC (Modelo Vista Controlador), se utilizó AngularJS¹³. Para las hojas de estilos se utilizó MaterializeCSS¹⁴ con leves modificaciones. Todos los módulos, a excepción de la interfaz gráfica, se desarrollaron con el gestor de proyectos

¹²<https://www.docker.com>

¹³<https://angularjs.org>

¹⁴<http://materializecss.com>

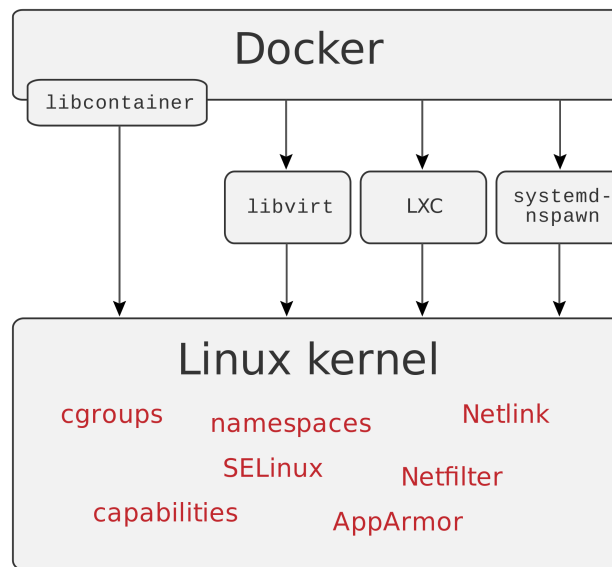


Figura 3.1: Interfaces utilizadas por Docker. (Dominio Público).

Maven¹⁵. Jersey¹⁶ fue empleado para la creación de la API RESTful.

3.1.5. Servidores web

Jetty¹⁷ fue el contenedor de aplicaciones utilizado para el despliegue de la API. Para la interfaz web se utilizó Apache HTTP Server¹⁸.

3.1.6. Bases de datos

Aerospike

Aerospike¹⁹ [14] se define como “Una base de datos NoSQL que ofrece velocidad y escalabilidad”. La función principal es la de almacenar información en formato clave-valor y está escrita en C. Se utilizó la versión 3.8.2.3.

Clientes inteligentes. Los clientes están al corriente de la configuración del clúster de la base de datos. Detectan transacciones fallidas y reencaminan peticiones a nodos con réplicas de los datos de forma automática. Gestionan *pools* de conexiones de forma transparente. También permiten almacenar tipos de datos básicos, objetos

¹⁵<https://maven.apache.org>

¹⁶<https://jersey.java.net>

¹⁷<http://www.eclipse.org/jetty/>

¹⁸<https://httpd.apache.org>

¹⁹<http://www.aerospike.com/>

binarios (*blobs*), listas, mapas e incluso tipos nativos de los distintos lenguajes soportados a través de la serialización. Disponen de soporte para operaciones atómicas.

Almacenamiento. Aerospike puede almacenar datos en discos de estado sólido (SSDs), RAM o discos mecánicos tradicionales. Utiliza un modelo de datos sin esquemas. Las escrituras se realizan en bloques grandes y las lecturas en bloques pequeños para reducir la latencia. Aerospike gestiona la desfragmentación y la liberación de RAM mediante la eliminación de referencias a registros expirados o con expiración establecida en caso de ser necesario y sin intervención externa. La información se almacena en *namespaces* (espacios de nombres), configurables de forma independiente. Se trata de contenedores de alto nivel y están formados por registros, índices y políticas. Según las políticas establecidas, pueden configurarse para ser usados en RAM o disco (SSD). En el caso de configurarse para su uso en memoria volátil, es posible habilitar la persistencia de los datos en disco. Las políticas permiten establecer el número de réplicas que existirán para un registro y su fecha de expiración. Dentro de cada espacio de nombres se definen los conjuntos (*sets*), un concepto similar a tabla en las bases de datos relacionales. Cada registro en un conjunto puede tener varias “columnas”, denominadas *bins*. Al no existir la necesidad de definir un esquema, cada registro puede tener columnas distintas. La escritura en SSDs está optimizada, utilizando patrones de bajo nivel para la lectura y la escritura, además, independientemente de la configuración de los espacios de nombres, todos los índices de la base de datos se encuentran almacenados en RAM.

Distribución. Los nodos se monitorizan entre sí para comprobar la conectividad. No existe un nodo *master*, si no que todos los nodos son iguales (*peers*). Se utiliza un algoritmo de consenso para determinar qué nodos son considerados parte del clúster cuando un nodo se añade o retira. Se utilizan algoritmos de distribución *hash* para dividir el índice principal en secciones y asignarlas a determinados nodos. La distribución aleatoria de los datos garantiza una carga similar en todas las máquinas del clúster, reduciendo previsiblemente la latencia. Existe soporte para consistencia eventual e inmediata. Los datos se redistribuyen cuando los clústeres son reconfigurados. Además está soportada la replicación entre datacenters con consistencia eventual.

Acceso a datos. Aerospike ejecuta las consultas en todos los nodos en paralelo. Soporta filtros de equidad y rango y permite la definición de funciones (*UDFs* en Lua) para extender la funcionalidad de la base de datos y aumentar el rendimiento. Con esta técnica es posible crear estructuras de datos complejas y funciones.

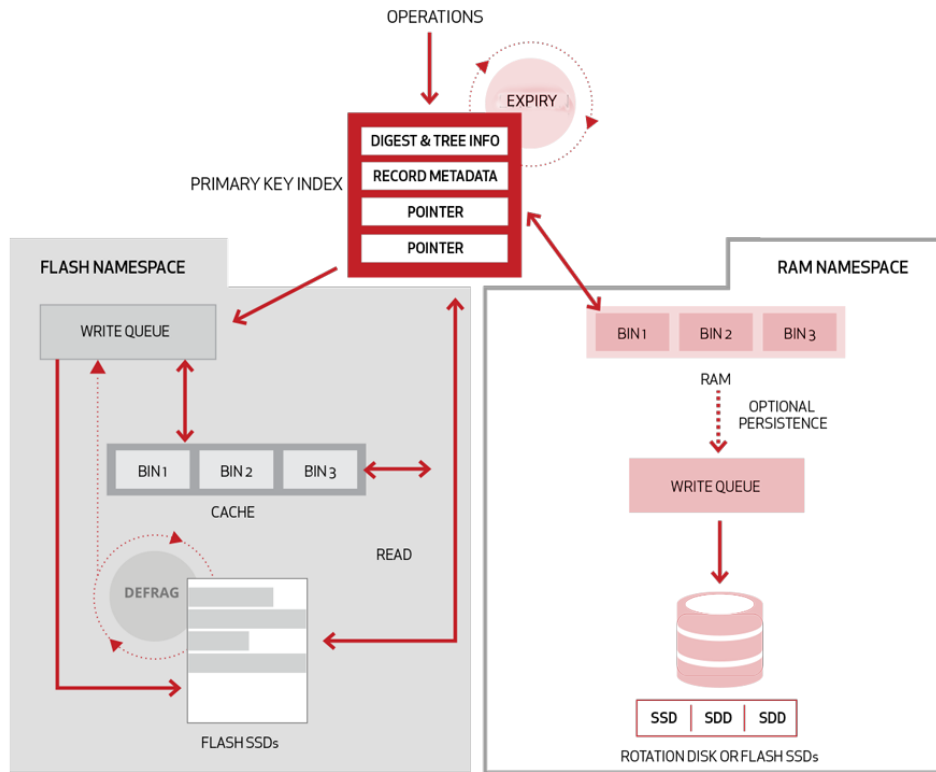


Figura 3.2: Almacenamiento de los datos en espacios de nombres basados en disco y RAM en Aerospike. (Ilustración cedida por Aerospike, © 2016 Aerospike, Inc.).

En el proyecto se utiliza como una base de datos temporal para la información, ya que almacenar todos los datos directamente en Aerospike supondría un uso de recursos muy elevado. Los tweets obtenidos se almacenan en HBase. Al mismo tiempo, se almacena en un set de Aerospike el identificador y contenido de cada publicación. Esta información es la expresamente necesaria para realizar el análisis de sentimientos y actualizar dicho cálculo en la base de datos destinada a la persistencia de los datos (HBase en este caso). Debido a las técnicas aplicadas para la captación de tweets, es posible que entre las iteraciones realizadas para un mismo o distinto término, se obtengan tweets ya repetidos. Como se explicará más adelante, se genera un identificador único para cada publicación que no coincide con el identificador único del tweet, entre otros aspectos, para soportar diversas fuentes. Para evitar el almacenamiento y procesamiento de tweets repetidos, se dispone de otro conjunto dedicado a tweets exclusivamente, en el que se almacenan los identificadores únicos de los mismos. Cuando se vacían los *buffers* del módulo de obtención de tweets se comprueba que no hayan sido recuperados con anterioridad. En caso negativo, se almacena y se añade una entrada al conjunto de tweets recuperados con su identificador. Estos

registros se añaden con una expiración de dos días.

HBase

HBase²⁰ [7] es una base de datos no relacional distribuida y escalable para grandes cantidades de información. Los *RegionServers* son los nodos que sirven y manejan regiones. Las regiones son el elemento básico de disponibilidad y distribución para las tablas y a su vez están formadas por *Stores* para cada *Column Family*. Las *Column Families* son agrupaciones de columnas. No existe esquema, característica tradicional de los RDBMS por lo que las columnas pueden variar entre distintos registros de una misma tabla. Cada columna es una tupla consistente en el nombre de la misma, su valor y un *timestamp*. Fue modelada a partir de Google BigTable [5] y está escrita en Java. Se ejecuta sobre HDFS (Hadoop Distributed File System) en modo distribuido y está por tanto, fuertemente integrada con el ecosistema Hadoop. Utiliza Zookeeper²¹ —Servicio centralizado de configuración y registro de nombres— de forma autogestionada o externalizada. HBase ordena las filas lexicográficamente según la clave de fila (*rowkey*). Debido a este orden, si se utilizan claves secuenciales se generan áreas calientes (*hotspotting*): un *RegionServer* será el objetivo de inserción y lectura de las nuevas filas hasta que se divida la región en dos, generando el *hotspotting* en el mismo *RegionServer* o en otro distinto. Esto se debe a que cada región sólo puede estar en un sólo *RegionServer*. Una de las grandes ventajas de HBase es su integración con el sistema de archivos distribuido de Hadoop. Ya que el sistema está destinado al almacenamiento permanente de las publicaciones, el uso de bases de datos NoSQL basadas en el uso intensivo de memoria o discos de estado sólido como Aerospike no parecen la solución más viable económicamente. Además, el uso de Spark para la agrupación de resultados también supone un uso intensivo de memoria. Otra de las ventajas de HBase es el almacenamiento de las filas en orden lexicográfico, de modo que los escaneos son ágiles en rangos secuenciales. Se utilizó la versión 1.1.5.

MariaDB

MariaDB²² es un sistema de gestión de bases de datos derivado de MySQL²³ con soporte para bases de datos relacionales. En el proyecto se utiliza este gestor para el almacenamiento de resultados de los distintos procesamiento llevados a cabo por los módulos Spark y para el almacenamiento de la LPA (Lista de términos de Procesamiento Automático). Se utilizó la versión 10.1.

²⁰<https://hbase.apache.org/>

²¹<https://zookeeper.apache.org/>

²²<https://mariadb.org/>

²³<https://www.mysql.com/>

3.1.7. IDEs de desarrollo

Se utilizaron principalmente dos IDEs para el desarrollo del proyecto. Se utilizó IntelliJ IDEA²⁴ 2016 para el desarrollo en el lenguaje Scala (aplicaciones Spark). Por otro lado se utilizó Notepad++²⁵ para el desarrollo web (HTML, CSS y Javascript). Para el desarrollo en Java se utilizó NetBeans²⁶ 8.1. Para la creación de *dockerfiles* se utilizó Gedit²⁷ 3.

3.1.8. Lenguajes de programación

A continuación se enumeran los distintos lenguajes de programación —en orden descendente de uso— utilizados durante el desarrollo del producto. Cabe destacar la necesidad expresa de aprender Scala²⁸ en el marco del proyecto para poder realizar los dos módulos Spark con conexión a HBase.

1. Java
2. Scala
3. JavaScript
4. HTML
5. Dockerfile
6. Bash
7. CSS

3.2. Tecnologías empleadas en la documentación

3.2.1. Overleaf

Overleaf²⁹ es un editor colaborativo del Latex, en línea y con renderizado casi instantáneo. Tienen integrado un potente control de versiones y permite exportar a un PDF el documento una vez terminado. Es utilizado, como ya ha sido comentado, para la redacción de la presente memoria.

²⁴<https://www.jetbrains.com/idea/>

²⁵<https://notepad-plus-plus.org>

²⁶<https://netbeans.org>

²⁷<https://wiki.gnome.org/Apps/Gedit>

²⁸<http://www.scala-lang.org>

²⁹<https://www.overleaf.com>

3.2.2. WBSTool

WBSTool³⁰ es un editor en línea de Estructuras de Descomposición de Tareas. Permite de forma sencilla generar EDTs y exportarlos una vez terminados. Ha sido utilizado para la generación del EDT mostrado en la sección de Gestión del Proyecto.

3.2.3. ProjectLibre

ProjectLibre³¹ es un software de administración de proyectos. Se utilizó para la creación del Diagrama de Gantt con el que se muestra la planificación del proyecto en la sección de Gestión del Proyecto. Se utilizó la versión 1.6.2.

3.2.4. Draw.io

Draw.io³² es un editor de gráficos compatible con Google Drive³³. Permite crear diagramas de todo tipo y fue utilizado para la generación de los diagramas mostrados en la sección de Diseño de Software.

3.2.5. StarUML

StarUML³⁴ es una herramienta de modelado UML. Permite la creación de diagramas de casos de uso, funcionalidad que fue aprovechada para la elaboración del diagrama de casos de uso del sistema, incluido en la sección de Gestión del Proyecto. Se utilizó la versión de evaluación. Se utilizó la versión 2.7.0.

3.2.6. Enterprise Architect

Enterprise Architect³⁵ es una herramienta de modelado y diseño visual basado en el OMG UML³⁶. También fue utilizado para la generación de los diagramas de la sección de Diseño de Software. Se utilizó la versión 12.

3.2.7. OwnCloud

OwnCloud³⁷ es una herramienta que puede instalarse en un servidor propio y que permite principalmente proveer de un servicio privado de almacenamiento de datos en línea y sincronización entre equipos.

³⁰<http://wbstool.com>

³¹<http://projectlibre.com>

³²<https://www.draw.io>

³³<https://drive.google.com/drive/>

³⁴<http://staruml.io/>

³⁵<http://www.sparxsystems.com.au/products/ea/>

³⁶<http://www.omg.org/spec/UML/>

³⁷<https://owncloud.org/>

Capítulo 4

Diseño de software

4.1. Arquitectura del sistema	72
4.1.1. Módulo de obtención de publicaciones digitales	73
4.1.2. Módulo de análisis de sentimiento en tiempo real	75
4.1.3. Módulo de procesamiento periódico de la LPA	78
4.1.4. Módulo de cálculo de sentimiento asociado a términos arbitrarios	81
4.1.5. API para realizar consultas personalizadas y recuperar resultados	81
4.1.6. Interfaz gráfica	84
4.2. Patrones aplicados	87
4.2.1. Entorno de desarrollo	88
4.3. Diagramas de clases	90
4.3.1. Módulo de obtención de publicaciones digitales	90
4.3.2. Módulo de análisis de sentimiento en tiempo real	92
4.3.3. Módulo de procesamiento periódico de la LPA	94
4.3.4. Módulo de cálculo de sentimiento asociado a términos arbitrarios	94
4.3.5. API para realizar consultas personalizadas y recuperar resultados	95
4.4. Diagramas de interacción	97
4.5. Tablas en los distintos SGBD	99

4.1. Arquitectura del sistema

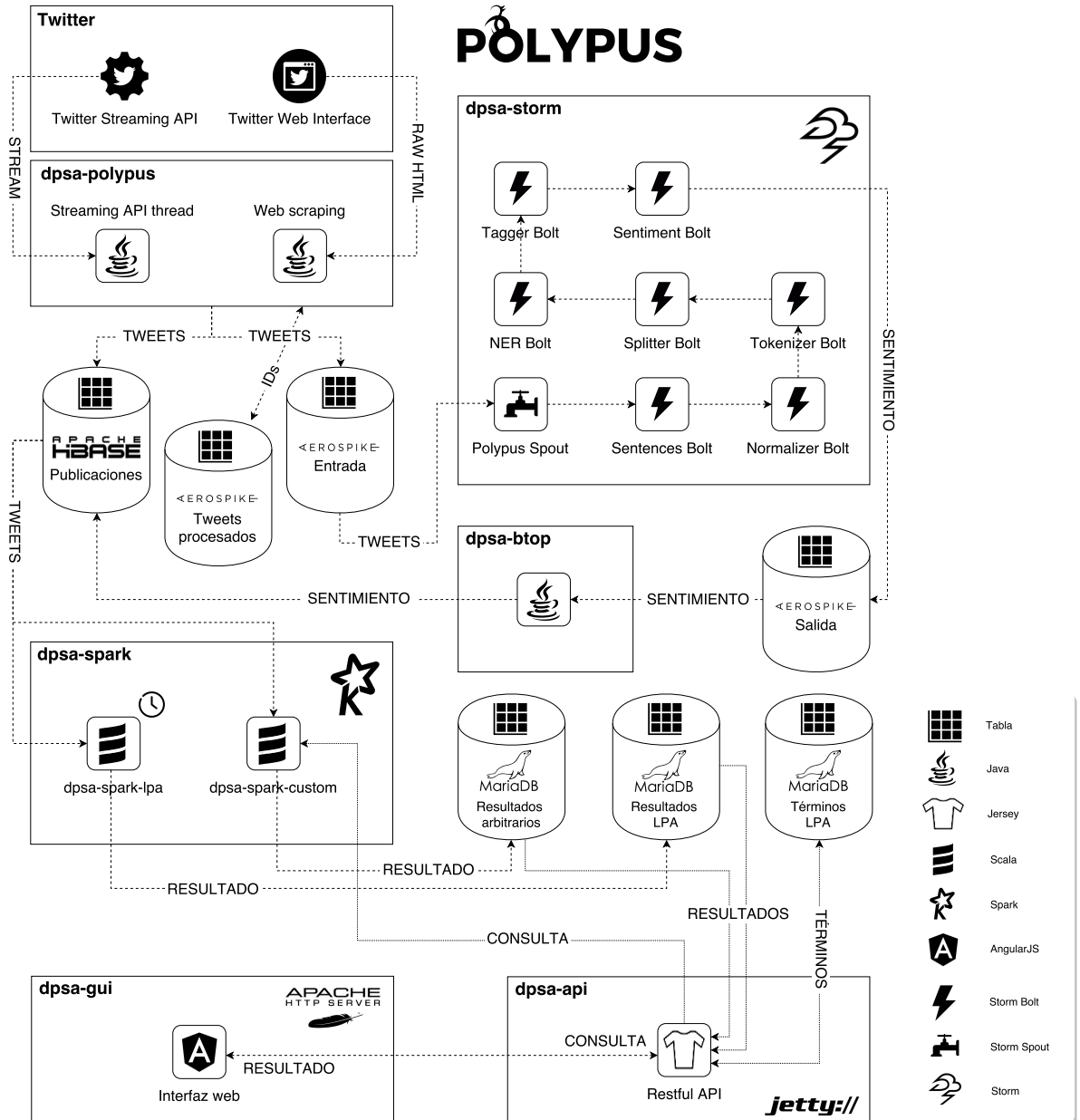


Figura 4.1: Diagrama de arquitectura del sistema.

A continuación se describe y detalla el funcionamiento de cada módulo del sistema y las conexiones entre ellos. También se describe el entorno de ejecución para el desarrollo.

4.1.1. Módulo de obtención de publicaciones digitales

El primer módulo es el encargado de obtener las publicaciones digitales que más tarde serán procesadas. Esta implementación trabaja exclusivamente con *tweets*. Se trata una aplicación escrita en Java y compuesta por varios hilos. Por un lado existe un hilo que lee el *streaming* de *tweets* emitido por Twitter a través de la librería Twitter4j. Por otro lado, existe un número configurable de hilos para realizar *web scraping*. A estos hilos se les asigna equitativamente una porción del total de términos de una lista configurable. Históricamente se utiliza una lista con las palabras más frecuentes de cada idioma a procesar. En este caso el castellano. Las características del procesador (frecuencia, número de núcleos, etc.) y de la conexión a Internet comprometen el tamaño de la lista de términos para conseguir un comportamiento eficiente. Para cada término de la lista, se invoca la URL `https://twitter.com/search?f=tweets&vertical=default&q=<TERMINO>lang:<IDIOMA>&src=typd` y se extraen los *tweets* presentes con sus atributos. Esta URL se corresponde con las búsquedas en directo del portal web.

Es posible ejecutar varias instancias de la aplicación en distintos nodos, repartiendo la lista de términos manualmente. En este caso, e incluso cuando se ejecuta en un solo nodo, es necesario evitar el almacenamiento de *tweets* repetidos. Ya que los *tweets* se almacenan como publicaciones digitales genéricas —para fomentar la abstracción de las distintas partes del sistema—, se genera un identificador único para cada publicación, por lo que no se repetirán nunca —los identificadores— aún cuando se recupere el mismo *tweet*. Esto supone la no detección de elementos repetidos.

Para reducir al mínimo las posibilidades de repetición de *tweets*, ya sea debido al uso de técnicas de *scraping* o a la multi-instanciación en varios nodos —también afectaría al consumo duplicado de la “Streaming API” si no se deshabilita—, se gestiona un buffer exclusivo para *tweets* —en caso de incluir otro tipo de publicaciones habría que incluir nuevos *buffers*—. Dicho *buffer* se implementa como una tabla —*set*— de Aerospike y sólo se almacenan los identificadores originales utilizados por Twitter. Los elementos escritos en dicha tabla expiran a los dos días, por lo que los identificadores se autoliberan tras un tiempo más que prudente de forma automática. Al incluir el *set* en un *namespace* en RAM, el acceso es muy rápido y no debería ralentizar demasiado al módulo de obtención de *tweets*. De esta forma, cada vez que va a realizar un volcado de los *tweets* obtenidos, se comprueba uno por uno que no estén en el *buffer*. Los volcados se realizan cuando hay suficientes publicaciones acumuladas en un *buffer* interno de un hilo. De cara a cada hilo, este *buffer* interno permite pre-filtrar las publicaciones repetidas que se obtienen consultando los mismo términos mientras no se vacía. El volcado de cada *buffer* interno supone la realización de dos

acciones: la escritura en la base de datos de persistencia en HBase y la escritura en el *buffer* externo implementado en Aerospike denominado “Entrada”. En este *buffer* se escriben exclusivamente el identificador, idioma y contenido de la publicación. Constituirá la entrada para el módulo de análisis de sentimiento. En la base de datos de persistencia se almacenan todos los atributos recuperados. También se escribe el valor del sentimiento a -2 para detectar publicaciones no analizadas en los módulos de procesamiento.

Generación de identificadores únicos. Los identificadores de las publicaciones digitales están diseñados para su uso en HBase. El diseño consta de dos partes fundamentales. En primer lugar, los identificadores están formados por una marca de tiempo en milisegundos —*timestamp*—, un identificador de la fuente origen de la publicación y una secuencia aleatoria de cuatro dígitos. HBase lee siempre las primeras entradas de las tablas durante las localizaciones y escaneos. Utiliza un orden lexicográfico para almacenar las filas, lo que implica que se leerán primero las filas más antiguas —*timestamp* inferior—. La mayor parte del tiempo se estima que se accederá a las últimas filas añadidas —las últimas publicaciones disponibles—, por lo que la solución es utilizar la diferencia entre el máximo número de tipo *Long* representable —9223372036854775807L— y el *timestamp* actual. Así, las publicaciones más recientes siempre estarán antes en la ordenación que lleva a cabo HBase, siendo leídas en primer lugar. Para respetar el orden lexicográfico, también es necesario que coincida el tamaño de todos los identificadores. Por tanto, aunque el programa tendría que sobrevivir cientos de generaciones para fallar en este aspecto, se comprueba el tamaño del *timestamp* y en caso de ser necesario se hace *padding* de ceros por la izquierda a 19 caracteres. El identificador de la fuente se ha decidido que tenga 5 dígitos —número fijo para mantener el tamaño constante de todos los identificadores—. Por último, el número aleatorio generado siempre tiene cuatro cifras. La razón por la que se añade es la siguiente. El sistema operativo no actualiza a cada milisegundo el reloj, por lo que una llamada para comprobar el *timestamp* actual podría ser inexacta y por tanto generar dos identificadores idénticos para dos publicaciones distintas —si se diese la casualidad de que el prefijo del identificador, explicado a continuación, también coincidiese—. No se puede garantizar la generación de identificadores únicos en momentos próximos pero es sumamente improbable —mismo milisegundo y dos números aleatorios de dos y cuatro cifras generados en distintos momentos y por separado con el mismo valor—. El diseño, en caso de utilizarse el capturador de publicaciones en más de un nodo, podría mejorarse, alterando el número aleatorio en función del *hostname*, por ejemplo. Por tanto, hasta ahora, la generación del identificador se realiza

con la siguiente estructura: `(Long.MAX_VALUE - Timestamp)_twtr_ALEATORIO[4]`.

En segundo lugar, se añade un prefijo a todos los identificadores generados que va desde “00” hasta “31”. Esto es, 32 prefijos posibles que se asignan en *Round-robin* en el módulo de captación de publicaciones. Esta característica permite pre-particionar la tabla de HBase para que exista una región por prefijo desde el principio. Cada región sólo puede estar en un servidor de regiones (nodo). Esta distribución de las claves de forma secuencial en distintas regiones evita el fenómeno del *hotspotting* —acceso repetido a una misma región y por consiguiente, a un mismo nodo— [15]. Al realizar esta distribución se pierde el acceso secuencial a las claves, por lo que será necesario instanciar varios escáneres durante las lecturas (acción muy beneficiosa en algunos casos, proporcionando velocidades mayores que los escaneos secuenciales en una misma región). Por tanto, la estructura final sería: `ROUND-ROBIN(01-32)_(Long.MAX_VALUE - Timestamp)_twtr_ALEATORIO[4]`.

4.1.2. Módulo de análisis de sentimiento en tiempo real

El módulo de análisis en tiempo real es una aplicación Java en la que se define una topología Storm. Existe una fuente de información —*spout*— que se encarga de recuperar las publicaciones digitales almacenadas por el primer módulo en el *buffer* Aerospike de entrada. El *spout* lee las publicaciones almacenadas en el buffer de forma destructiva —una vez leídas las elimina del *buffer*—. Para cada publicación emite una tupla compuesta por el identificador y contenido de la publicación. Según el idioma de la publicación, se emite la tupla en un *stream* determinado. Sólo existen *bolts* que implementen la solución en castellano, por lo que las publicaciones en otros idiomas no son recogidas por ningún *bolt*. Los *bolts* son clases especiales en las que se adapta a la interfaz necesaria el código de los distintos módulos PLN que ya habían sido traducidos a Java. Los distintos módulos operan de forma secuencial, consumiendo cada uno en su entrada la salida del anterior. Es posible indicar la instanciación de varios *bolts* o *spouts* del mismo tipo mediante el denominado *parallelism hint*. Este parámetro se refiere al número inicial de hilos —*executors*— y, por defecto, este valor también es el máximo. Sin embargo, es posible incrementar el número de tareas de forma dinámica mediante la propiedad *numTasks*. Esto supone que la topología realizará un reescalado automático según las necesidades de cada momento hasta un número máximo de hilos correspondiente al número de tareas indicado. Los *bolts* se comportan como elementos independientes sin restricciones ya que es irrelevante el *bolt/spout* de origen para asignar la instancia del *bolt* destino —de tipo consecutivo en la secuencia de procesamiento—. Debido a esta característica se utiliza el *shuffle*

grouping como método de agrupación de los *streams*, es decir, asignación aleatoria.

```
TopologyBuilder builder = new TopologyBuilder();

builder.setSpout("polypus", new PolypusSpout());

builder.setBolt("sentences", new es_SentencesBolt())
    .shuffleGrouping("polypus", "lang_es");

builder.setBolt("normalizer", new es_NormalizerBolt())
    .shuffleGrouping("sentences", "lang_es");

builder.setBolt("tokens", new es_TokensBolt())
    .shuffleGrouping("normalizer", "lang_es");

builder.setBolt("splits", new es_SplitterBolt())
    .shuffleGrouping("tokens", "lang_es");

builder.setBolt("ner", new es_NERBolt())
    .shuffleGrouping("splits", "lang_es");

builder.setBolt("tagger", new es_TaggerBolt())
    .shuffleGrouping("ner", "lang_es");

builder.setBolt("nbayes", new es_NbayesBolt())
    .shuffleGrouping("tagger", "lang_es");
```

Fragmento de código 4.1: Definición básica de la topología Storm sin multi-instanciación.

La entrada y salida de cada *bolt* se corresponde con la entrada y salida tradicionales de las clases Java originales de los módulos PLN. En cada uno se recupera la tupla emitida por otro *bolt* de tipo previo en la secuencia de procesamiento y en un *stream* del mismo idioma. Cuando las publicaciones llegan a una instancia del último módulo —*bolt*—, los resultados (sentimiento positivo, negativo o neutro) se guardan en un *buffer* de salida en Aerospike. Un pequeño módulo escrito en Java es el encargado de realizar lecturas destructivas y las consiguientes actualizaciones del sentimiento en la tabla de HBase con los valores -1, 0 o 1 —negativo, neutro o positivo—. El uso de un demonio externo para la tarea en conjunto con otro *buffer* en RAM se debe a la intención de liberar las instancias de los *bolts* que escriben los resultados lo más rápido posible para que puedan seguir procesando información, evitando la creación de un cuello de botella o la necesidad de instanciar un número mayor de *bolts* del último paso desperdiciando recursos innecesariamente.

```
TopologyBuilder builder = new TopologyBuilder();

builder.setSpout("polypus", new PolypusSpout(), 2);

builder.setBolt("sentences", new es_SentencesBolt(), 2)
    .setNumTasks(8)
    .shuffleGrouping("polypus", "lang_es");

builder.setBolt("normalizer", new es_NormalizerBolt(), 2)
    .setNumTasks(8)
    .shuffleGrouping("sentences", "lang_es");

builder.setBolt("tokens", new es_TokensBolt(), 2)
    .setNumTasks(8)
    .shuffleGrouping("normalizer", "lang_es");

builder.setBolt("splits", new es_SplitterBolt(), 2)
    .setNumTasks(8)
    .shuffleGrouping("tokens", "lang_es");

builder.setBolt("ner", new es_NERBolt(), 2)
    .setNumTasks(8)
    .shuffleGrouping("splits", "lang_es");

builder.setBolt("tagger", new es_TaggerBolt(), 2)
    .setNumTasks(8)
    .shuffleGrouping("ner", "lang_es");

builder.setBolt("nbayes", new es_NbayesBolt(), 2)
    .setNumTasks(8)
    .shuffleGrouping("tagger", "lang_es");
```

Fragmento de código 4.2: Definición de la topología con una pista de paralelización 2 y un máximo de 8 hilos para cada *spout/bolt*.

4.1.3. Módulo de procesamiento periódico de la LPA

El módulo de procesamiento periódico de la LPA es una aplicación Spark escrita en Scala. En primer lugar, la aplicación realiza la lectura de los términos de la LPA desde una tabla en una base de datos en MariaDB. Dado el intervalo de tiempo de búsqueda con dos *timestamps*, se generan las claves de comienzo y de parada como se explica más adelante. Gracias a la ayuda del conector de HBase *Spark-HBase Connector*, es posible realizar una conexión de forma sencilla e incluso indicar el prefijo con el que fueron distribuidas las claves:

```
val result = sc.hbaseTable[(Option[String], Option[Long], Int, Option[String], Option[String])](hbase_table) // No se recoge la rowkey
    .select("content", "post_timestamp", "sentiment", "language", "author_nick")
    .inColumnFamily(hbase_cf)
    .withStartRow(getStartRowkey(stop_timestamp)) // ts2 porque las filas se guardan en orden inverso
    .withStopRow(getStopRowkey(start_timestamp)) // lo mismo
    .withSalting(BUCKETS).filter { post =>
[...]
```

Fragmento de código 4.3: Fragmento de código en el que se muestra la conexión con HBase desde una aplicación Spark escrita en Scala.

El contenido de las publicaciones se normaliza antes de ser comparado con la lista de términos de la LPA recuperada de MariaDB. Todos los caracteres se convierten a minúsculas, se eliminan caracteres no ingleses y se sustituyen por los más parecidos y se eliminan todos los caracteres no alfanuméricos y espacios sobrantes. Para cada publicación que contenga algún término de la lista se emite una tupla —tantas veces como términos coincidentes—, con el sentimiento recuperado como valor y como clave “TERMINO_result” y otra tupla con la clave “TERMINO_posts” y valor 1 para contar las publicaciones procesadas. Esto se logra respectivamente con los métodos de filtrado —*filter*— y de mapa plano (*flatMap*), para poder emitir varias tuplas. Por último se aplica la función *reduce*(*_+_*) al resultado para obtener la agregación de las tuplas con las mismas claves. Los resultados se ordenan y se recuperan de los distintos nodos con los métodos *sortByKey(true)* y *collect*. Por último se escriben los resultados obtenidos para cada término en la tabla correspondiente de MariaDB. Cabe destacar que en cada ejecución, se consulta la marca temporal de la última actualización de los términos en la tabla de resultados de la LPA. Por ello sólo se procesan las publicaciones que no habían sido procesadas con anterioridad y posteriores a la última fecha de barrido.

Como se ha comentado, este módulo se ejecuta de forma periódica cada minuto,

```
[...]
var normalizedContent: String = null
if (post._1.nonEmpty) {
    // Contenido de la publicación normalizado
    normalizedContent = SerializedMethods.normalizeString(post._1.get)
}
// Si se escribe algún valor en el buffer Aerospike sin estar en HBase (só
// lo disponible la columna sentiment, se omite)
post._1.nonEmpty && post._2.nonEmpty && post._4.nonEmpty && post._4.
    nonEmpty && post._5.nonEmpty &&
    // Se comprueba que el post haya sido procesado
    post._3 != -2 &&
    // Coincide el idioma con el solicitado
    post._4.get.equals(language) &&
    // El timestamp de la publicación está en rango
    post._2.get >= (start_timestamp) && post._2.get <= (stop_timestamp)
    && (terms.map(term => " " + term + " ")
        .exists(normalizedContent.contains) || // La publicación contiene alguno
        de los términos dentro de la publicación
        terms.map(term => term + " ")
        .exists(normalizedContent.startsWith) || // La publicación contiene
        alguno de los términos al comienzo de la publicación
        terms.map(term => " " + term)
        .exists(normalizedContent.endsWith)) // La publicación contiene alguno
        de los términos al final de la publicación
}
[...]
```

Fragmento de código 4.4: Procesado de los resultados recuperados (1/2).

por lo que para automatizar la ejecución se utiliza Cron. Para ello es necesario definir un fichero *crontab* en el sistema similar al siguiente:

```
[...]
.flatMap(
  post => {
    // Contenido de la publicación normalizado
    val normalizedContent = SerializedMethods.normalizeString(post._1.get)
    // Términos contenidos en el medio de la publicación
    val termsInside = terms.filter(term => normalizedContent.contains(" " +
      term + " "))
    // Términos contenidos al principio de la publicación
    val termsBegin = terms.filter(term => normalizedContent.startsWith(term
      + " "))
    // Términos contenidos al final de la publicación
    val termsEnd = terms.filter(term => normalizedContent.endsWith(" " +
      term))
    // Se concatenan las listas
    var termsToEmit = termsInside ::: termsBegin ::: termsEnd
    // Se eliminan términos repetidos
    termsToEmit = termsToEmit.distinct

    // Se añaden las secuencias parciales a
    var buffer = new ListBuffer[(String, Double)]()
    termsToEmit.foreach { term =>
      buffer += ((term + "_result", post._3))
      buffer += ((term + "_posts", 1))
    }

    // Emisión de la secuencia completa
    buffer.toIndexedSeq
  }
).reduceByKey(_ + _).sortByKey(true).collect()
```

Fragmento de código 4.5: Procesado de los resultados recuperados (2/2).

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/default/java:/opt/spark/default
HOME=/root

#####
# minuto (0-59),
# | hora (0-23),
# | | día del mes (1-31),
# | | | mes (1-12),
# | | | | día de la semana (0-6 donde 0=Domingo)
# | | | | | comandos
#####
* * * * * spark-submit /dpsa/dpsa-spark-lpa.jar --language=es
```

Fragmento de código 4.6: Ejemplo de fichero *crontab* para la ejecución cada minuto del módulo de procesamiento de la LPA.

Generación de las claves de comienzo y de parada dado un intervalo temporal. Dado que la aplicación recibe como parámetros dos marcas de tiempo, es necesario generar la clave de inicio y de parada para poder escanear la tabla de HBase. Para ello se procede de la siguiente manera. En primer lugar se calcula la diferencia entre el *timestamp* y el *Long* más grande con *padding* de ceros por la izquierda a 19 caracteres. Quedan dos incógnitas por resolver para generar el par de claves. En primer lugar la fuente, para hacer la aplicación más genérica se utiliza el siguiente código “AAAAA”. Ya que la fuente es un código que se genera con caracteres alfabéticos, las mayúsculas se encuentran antes que las minúsculas en el orden lexicográfico. Como la “a” es la primera letra del abecedario, con cinco aes garantizamos que la clave sea la mínima para la partida y máxima para la clave de parada con “zzzzz” por las mismas razones. En cuanto a los cuatro números aleatorios que finalizan la clave se utilizan, por las razones ya expuestas, las cadenas “00000” y “99999” respectivamente. Cabe destacar que la clave de partida se genera con la marca temporal de cierre del intervalo y que la de parada con la de inicio. Esto se debe a la alteración que se realiza en las claves, almacenando en sentido inverso todas las filas.

4.1.4. Módulo de cálculo de sentimiento asociado a términos arbitrarios

Esta aplicación de Spark escrita en Scala comparte muchas similitudes con la anterior, de modo que se comentarán los aspectos diferenciales. En vez de procesar los términos de la LPA, se procesa exclusivamente la cadena de búsqueda proporcionada. Es posible definir, además del intervalo temporal, el tamaño de las ventanas que contendrá. En el módulo anterior, se procesan las nuevas publicaciones, mientras que en este módulo, todas las coincidentes con la cadena de búsqueda. Por eso cobra sentido el concepto de ventana, permitiendo definir subintervalos de tiempo en los que producir resultados parciales para poder visualizar progresivamente la evolución del sentimiento asociada a una consulta. Entre los intervalos dados, se calcula el número de ventanas al proporcionarse su tamaño. Para cada publicación, se calcula la ventana a la que pertenece:

4.1.5. API para realizar consultas personalizadas y recuperar resultados

La API, implementada en Java con el *framework* RESTful Jersey, permite consultar resultados y ejecutar consultas personalizadas en Spark. Todos los datos que maneja están presentes en MariaDB. Este módulo tiene integradas las dependencias

```

// Número de ventanas
val windows: Long =
Math.ceil((stop_timestamp - start_timestamp) / window_size).toLong

// Número de milisegundos de diferencia en el intervalo
val milliseconds: Long = stop_timestamp - start_timestamp

// Tamaño de la ventana
val windowSize: Long = Math.ceil(milliseconds / windows.toDouble).toLong

// Último separador entre ventanas
val lastSeparator: Long = (windows * windowSize + start_timestamp) -
    windowSize

// Lista de separadores
val separators = List.range(start_timestamp, lastSeparator, windowSize)

def getWindow(post_timestamp: Long, separators: List[Long]): Int = {
    // La ventana correspondiente es la posición del separador más
    // grande de aquellos con un valor menor que el del timestamp dado
    return separators.indexOf(separators.filter(_ <= post_timestamp).max)
}

```

Fragmento de código 4.7: Cálculo de los separadores de las ventanas y de la ventana correspondiente a una marca temporal

de Jetty. Es posible ejecutar directamente la aplicación en un servidor Jetty embebido mediante el comando `mvn jetty:run`. A continuación se indican los métodos disponibles:

- M1. GET /api/1.0/lpa
Obtener términos de la LPA.
- M2. GET /api/1.0/lpa/search?q=
Obtener términos de la LPA filtrados por una cadena “q”.
- M3. GET /api/1.0/lpa/trending?no= &min=
Obtener un número “no” de términos más relevantes en los últimos “min” minutos.
- M4. GET /api/1.0/lpa/{termino}/results?b= &e=
Obtener los resultados de un término desde la marca temporal “b” hasta la “e”.
- M5. GET /api/1.0/lpa/{termino}/next?since=
Obtener el siguiente resultado de un término de la LPA a una marca temporal “since”.

- M6. DELETE /api/1.0/lpa/{termino} Eliminar un término de la LPA.
- M7. PUT /api/1.0/lpa/{termino} Añadir un término a la LPA.
- M8. GET /api/1.0/results/{id} Obtener los resultados de una consulta personalizada a partir de su ID.
- M9. GET /api/1.0/results?q= Localizar consultas personalizadas a través de una cadena “q”.
- M10. POST /api/1.0/submit Lanzar un trabajo de agregación de resultados personalizado enviando un JSON con la siguiente estructura:

```
{
  "search_query": "",
  "language": "es",
  "start_timestamp": "",
  "stop_timestamp": "",
  "window_size": "",
  "job_tag": ""
}
```

4.1.6. Interfaz gráfica

La interfaz gráfica permite una interacción casi directa con la API. Las funcionalidades se corresponden con los casos de uso del sistema. Está implementada en AngularJS y realiza peticiones RESTful al servidor donde se encuentra desplegada la API.

Existen cuatro vistas: “Tiempo real”, “Explorar”, “Búsqueda” y “Gestionar LPA” que permiten respectivamente visualizar los términos de la LPA más activos en tiempo real, comparar el sentimiento atribuido a los distintos términos de la LPA, realizar búsquedas personalizadas sobre el conjunto total de publicaciones procesadas y eliminar o añadir términos a la LPA. Cada vista es gestionada por su controlador que a su vez se encarga de la comunicación con la API a través de los servicios correspondientes. La interfaz simplemente gestiona de manera visual la interacción con la API y muestra los resultados representados en gráficas para su mejor visualización.

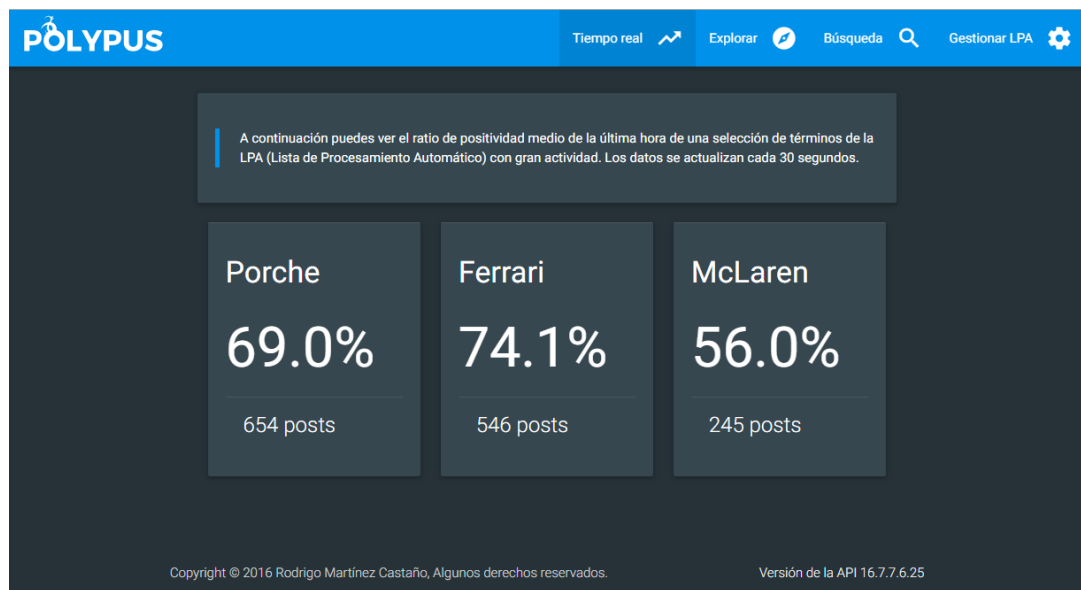


Figura 4.2: Vista de visualización en tiempo real de términos con gran actividad.



Figura 4.3: Vista de consulta sobre los términos de la LPA.

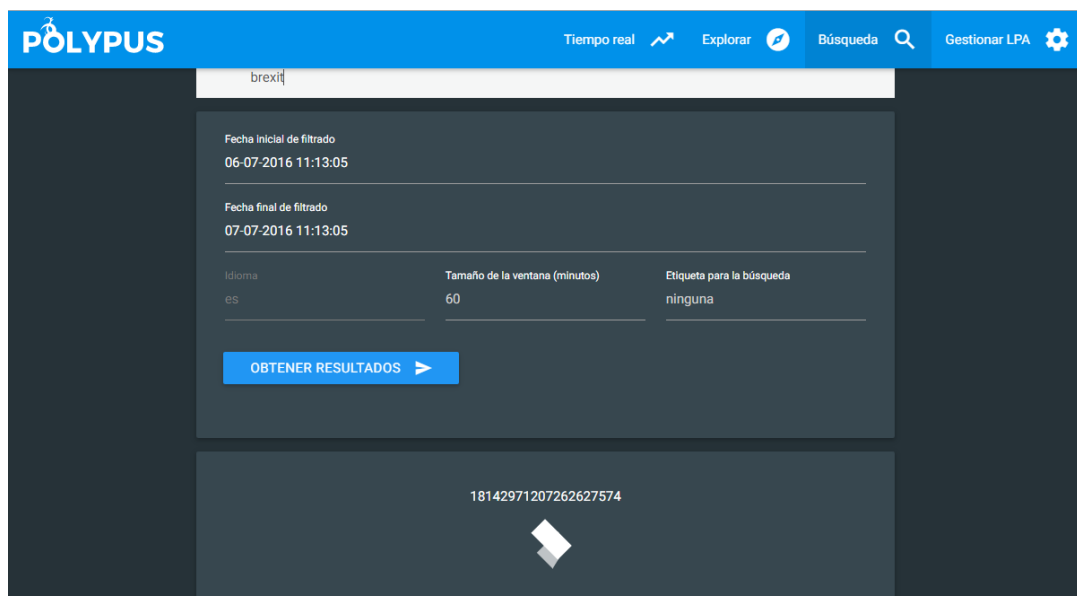


Figura 4.4: Búsqueda personalizada en la interfaz (1/2).

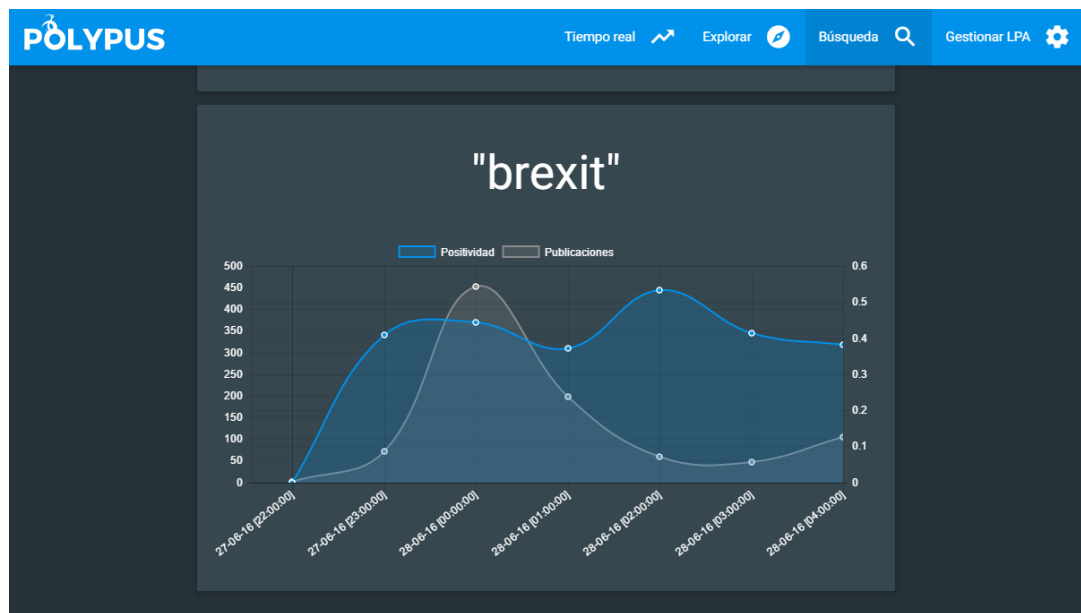


Figura 4.5: Búsqueda personalizada en la interfaz (2/2).

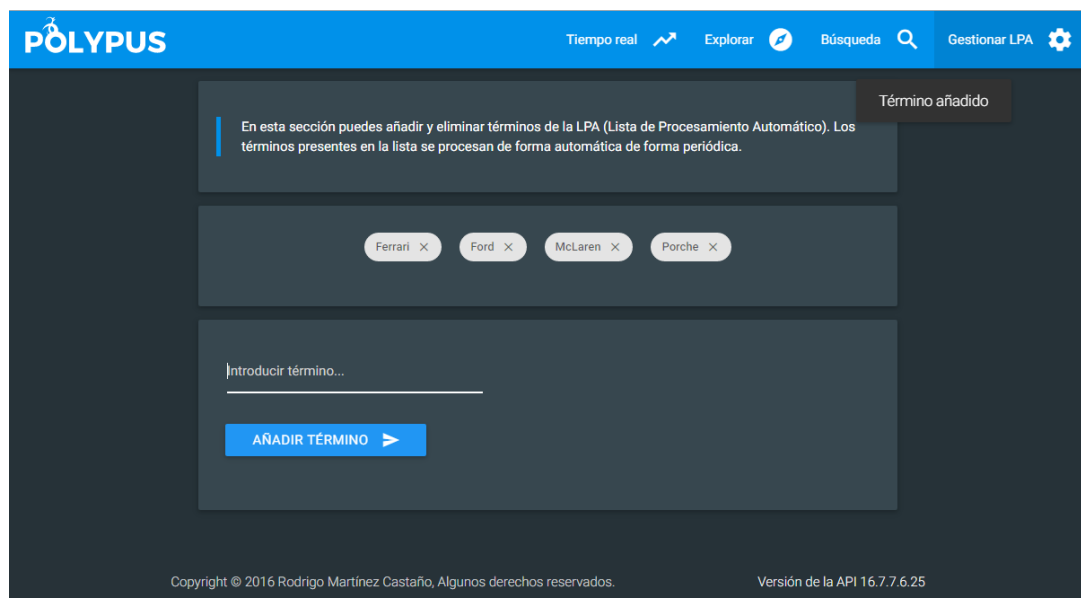


Figura 4.6: Gestión de la LPA en la interfaz.

4.2. Patrones aplicados

El patrón *singleton* es un patrón de diseño que restringe la creación de objetos pertenecientes a una clase. Garantiza que una sola clase tenga una instancia y proporciona un punto de acceso global a ella. Este patrón se aplica en varios de los módulos en las implementaciones de los DAOs —*Data Access Object*—. DAO es un patrón de diseño que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos: bases de datos, archivos, etc.

El patrón *Factory Method* consiste en la utilización de una clase constructora dedicada a la instanciación de objetos de un subtipo de un tipo determinado. Los tipos serían las distintas interfaces de acceso a datos —Interfaces DAO—. Los subtipos serían las distintas implementaciones para los distintos SGBD utilizados de esas interfaces. En el caso del módulo de obtención de *tweets* se escribe en dos SGBD distintos. En otros casos, como el de la API, en el que sólo se accede a uno, contribuye a la modularidad y fácil adaptación futura a nuevas implementaciones o cambios de SGBD.

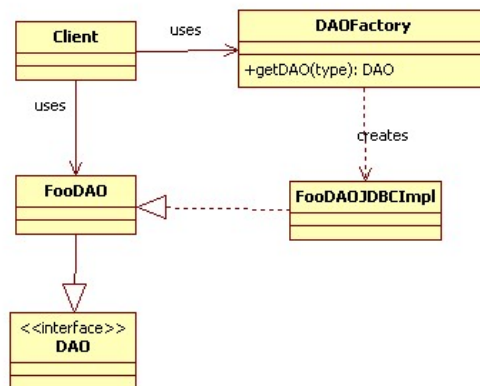


Figura 4.7: Patrón DAO Factory.

Para la interfaz gráfica, al utilizar AngularJS se aplicó el patrón Modelo Vista Controlador (MVC). El controlador recibe todas las peticiones y trabaja con el modelo —implementado con *Services*— para preparar los datos necesarios para la Vista. Esta utiliza los datos preparados por el controlador para generar una respuesta final de cara al usuario. El modelo se encarga del mantenimiento de los datos, la vista de mostrar los datos preparados al usuario y el controlador hace de intermediario entre Vista y Modelo.

4.2.1. Entorno de desarrollo

Todas estas aplicaciones necesitan un entorno para poder ser ejecutadas. El entorno de desarrollo no es un clúster y los distintos sistemas se despliegan y se conectan mediante la tecnología de virtualización Docker. El uso de Docker también facilitará en el futuro la ejecución clusterizada del sistema. Una vez realizada la orquestación de los distintos contenedores se garantiza que el entorno de ejecución va a ser idéntico al de desarrollo (mismas versiones y dependencias de las distintas aplicaciones empleadas).

La estrategia seguida fue la de la creación de una imagen por cada servicio necesario y se busca la utilización de la misma imagen base en cada una: CentOS 7. Se realizaron más imágenes de las enumeradas a continuación —teniendo en mente la ejecución clusterizada— pero las utilizadas directamente en el desarrollo fueron las de Aerospike, HBase, MariaDB, Spark y Storm y Apache HTTP Server. Esto facilitó el desarrollo y pruebas. Por ejemplo, la imagen de MariaDB creaba de forma automática la base de datos necesaria y las respectivas tablas. Aerospike hacía lo propio con los *namespaces*. Para ejecutar una instalación limpia (ante un problema con la información almacenada, por ejemplo), simplemente había que volver a lanzar los contenedores.

Como se puede observar, el contenedor de Storm está ligado al de Aerospike (lectura del *buffer*, pero también al de HBase. Esto se debe a dos motivos, por un lado porque desde este mismo contenedor también se ejecutaba la aplicación *dpsa-btop*, encargada de trasladar los resultados del *buffer* de salida a HBase. En segundo lugar, porque HBase auto gestiona Zookeeper cuando se ejecuta en modo *standalone*. Por tanto, para el desarrollo, se utiliza directamente la instancia de Zookeeper autogestionada por HBase. El contenedor de Spark está ligado al de MariaDB y al de HBase para la lectura de términos de la LPA y escritura de resultados y para la lectura de publicaciones digitales respectivamente. La API se despliega en el mismo contenedor que Spark. El contenedor Httpd fue el utilizado para el despliegue de la aplicación web (Interfaz gráfica) que hace uso de la API.

```
#!/bin/bash

docker kill hbase ; docker rm hbase
docker run -tid --name hbase -h hbase -p 60000:60000 -p 60010:60010 -p
    60020:60020 -p 60030:60030 -p 2888:2888 -p 3888:3888 -p 2181:2181 -p
    3181:3181 hbase

docker kill aerospike ; docker rm aerospike
docker run -tid --name aerospike -h aerospike -p 3000:3000 -p 3001:3001 -p
    3002:3002 -p 3003:3003 aerospike

docker kill storm ; docker rm storm
docker run -tid --name storm -h storm --link hbase --link aerospike -p
    3772:3772 -p 3773:3773 -p 6627:6627 -p 6700:6700 -p 6701:6701 -p
    6702:6702 -p 6703:6703 -p 8000:8000 -p 8080:8080 -v $(pwd)/dpsa-storm:/
    dpsa:rw storm

docker kill mariadb ; docker rm mariadb
docker run -d --name mariadb -h mariadb -p 3306:3306 mariadb

docker kill spark ; docker rm spark
docker run -tid --name spark -h spark --link hbase --link mariadb -v $(pwd)/
    dpsa-spark:/dpsa:rw spark

docker kill httpd ; docker rm httpd
docker run -tid --name httpd -h httpd --link spark -v $(pwd)/dpsa-gui:/dpsa:
    rw httpd
```

Fragmento de código 4.8: Script de reinicio de los distintos contenedores de desarrollo.

4.3. Diagramas de clases

4.3.1. Módulo de obtención de publicaciones digitales

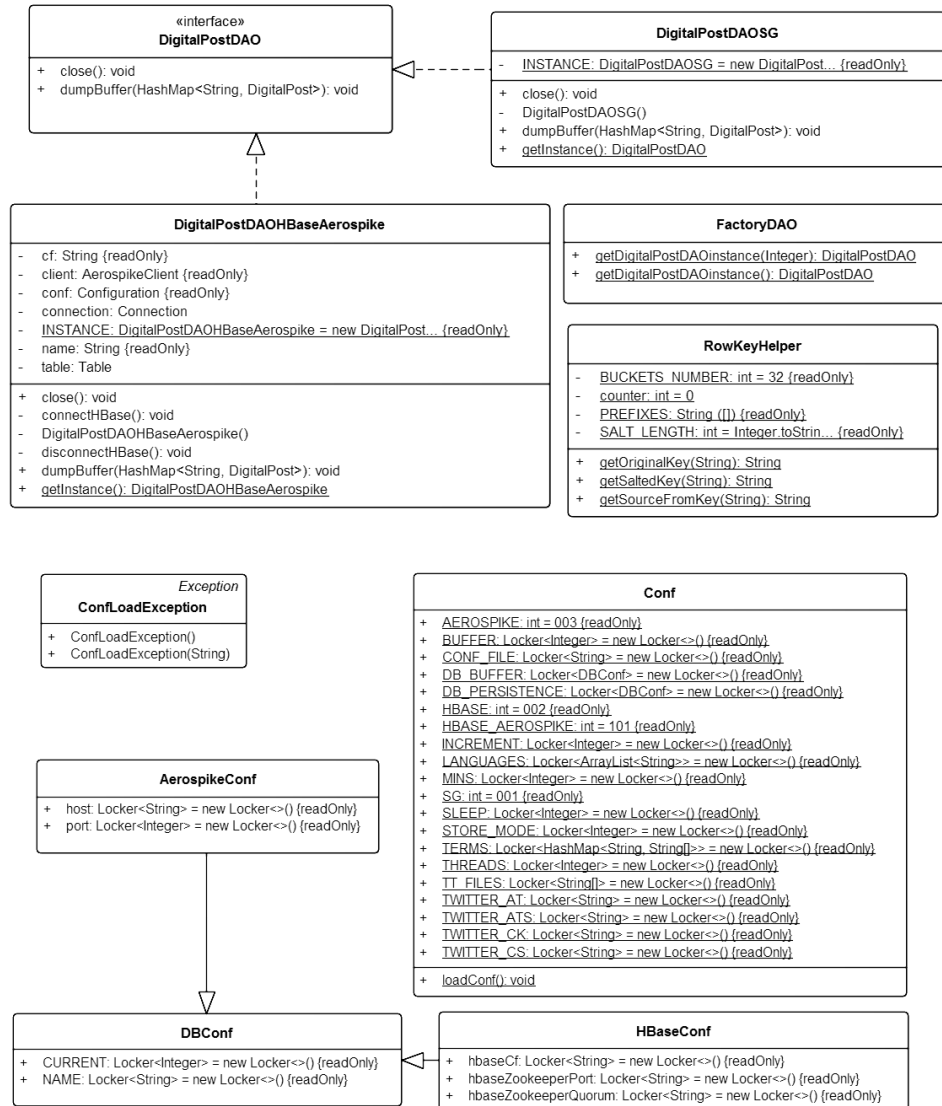


Figura 4.8: Diagrama de clases con atributos y funciones del Módulo de obtención de publicaciones digitales (1/2)

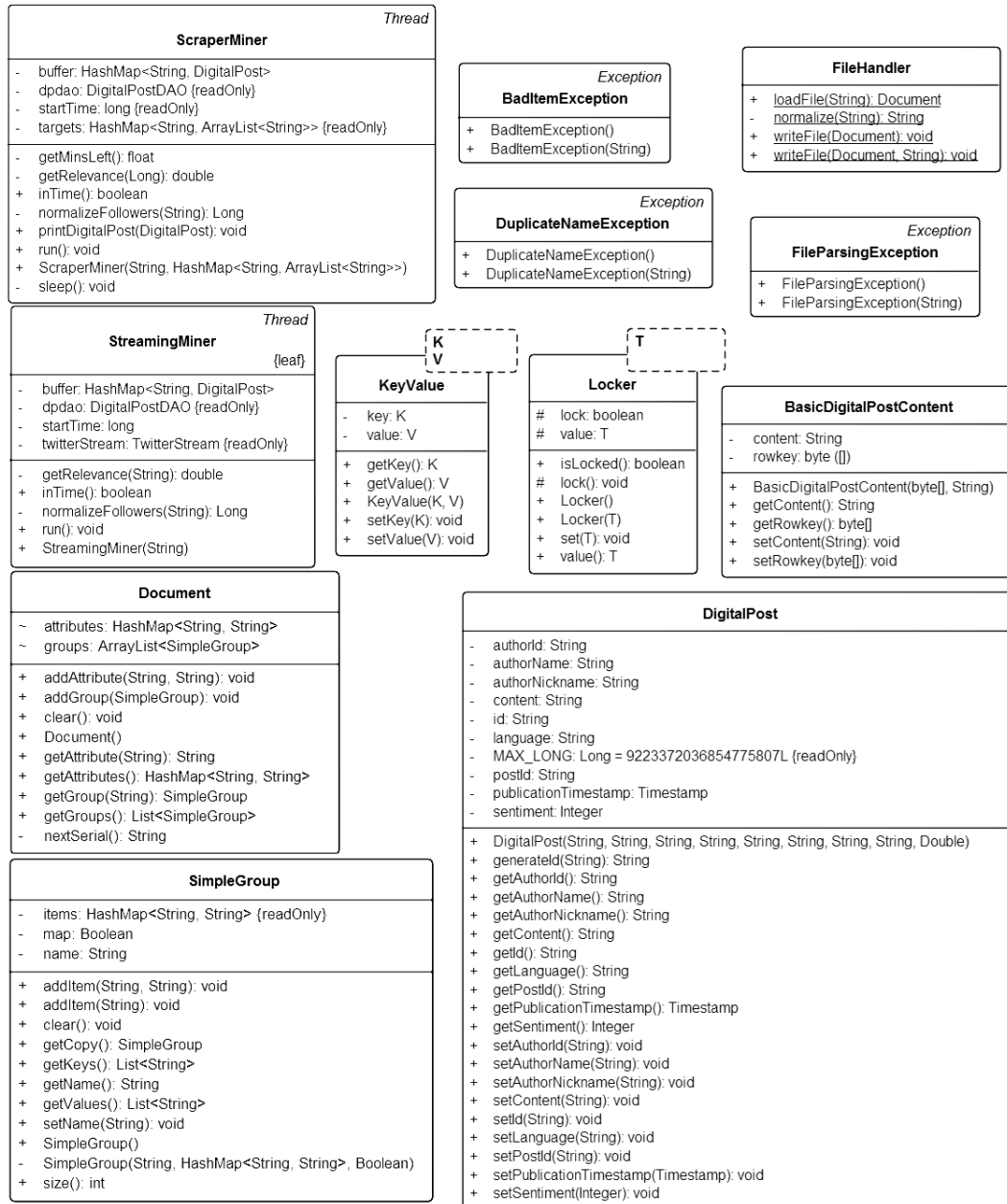


Figura 4.9: Diagrama de clases con atributos y funciones del Módulo de obtención de publicaciones digitales (2/2)

4.3.2. Módulo de análisis de sentimiento en tiempo real

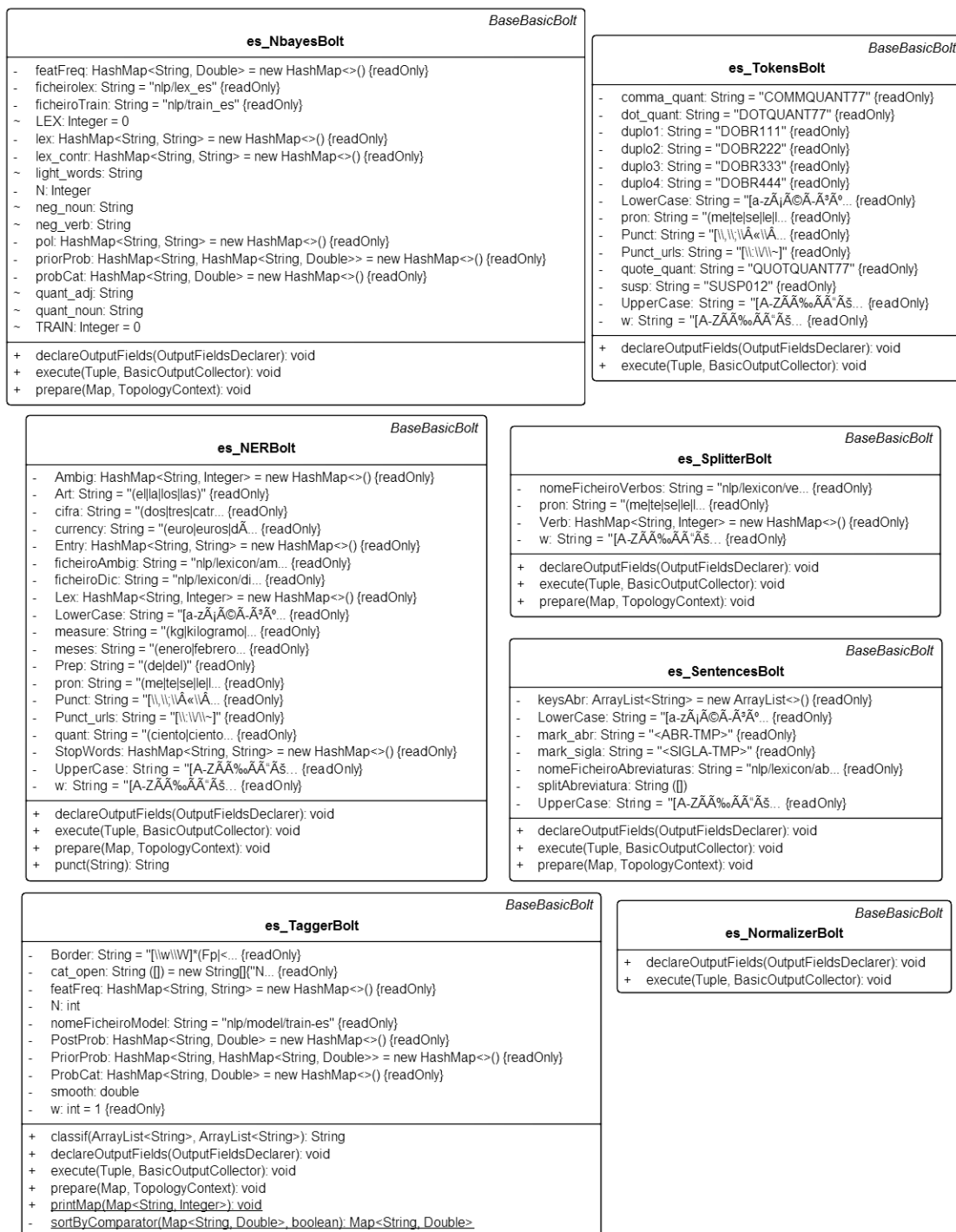


Figura 4.10: Diagrama de clases con atributos y funciones del módulo de análisis de sentimiento en tiempo real (1/2)

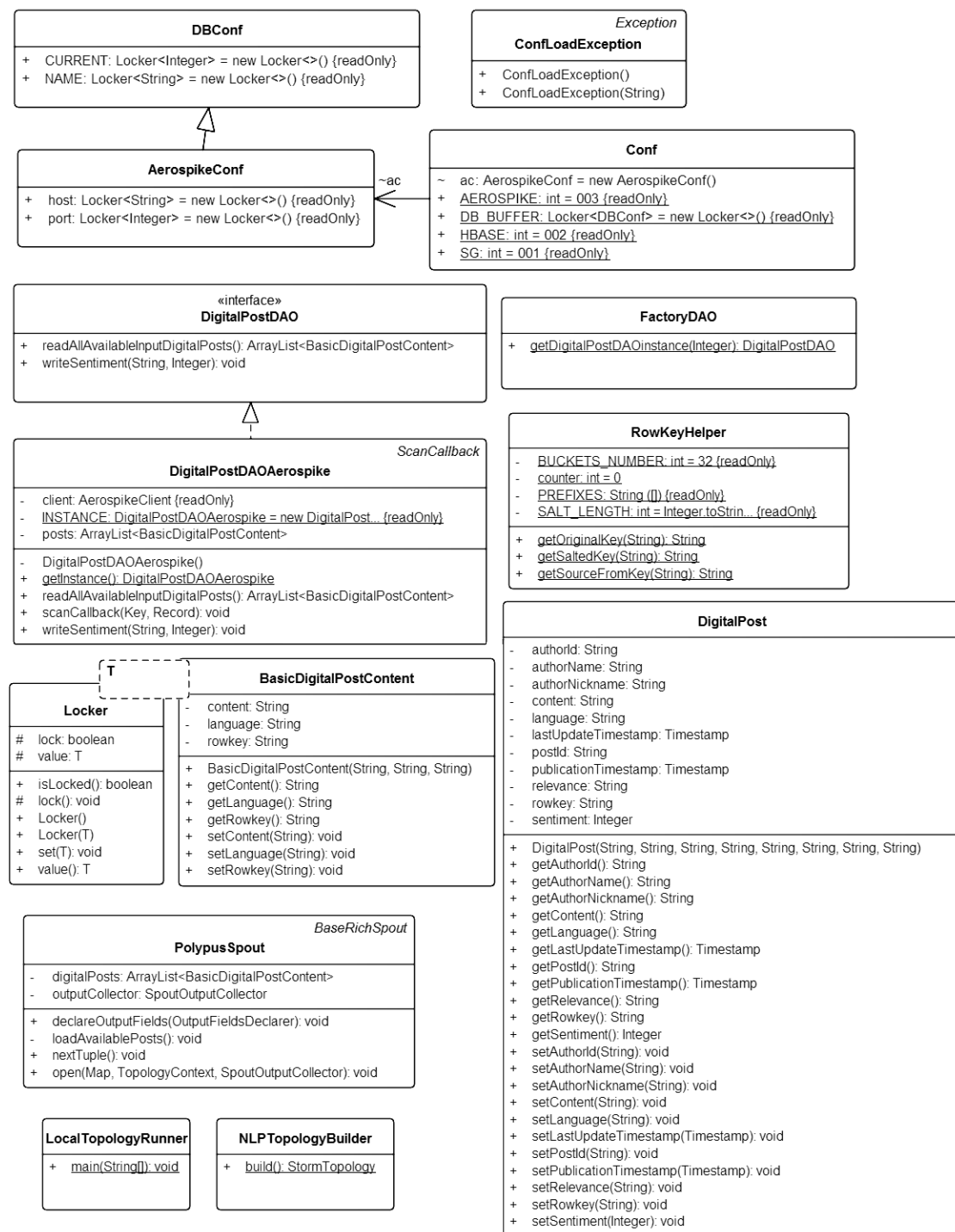


Figura 4.11: Diagrama de clases con atributos y funciones del módulo de análisis de sentimiento en tiempo real (2/2)

4.3.3. Módulo de procesamiento periódico de la LPA

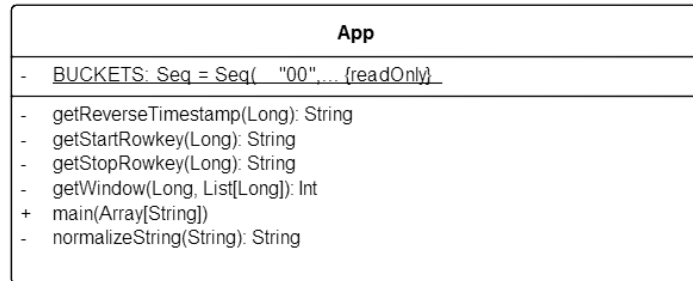


Figura 4.12: Diagrama de clases con atributos y funciones del módulo de procesamiento periódico de la LPA

4.3.4. Módulo de cálculo de sentimiento asociado a términos arbitrarios

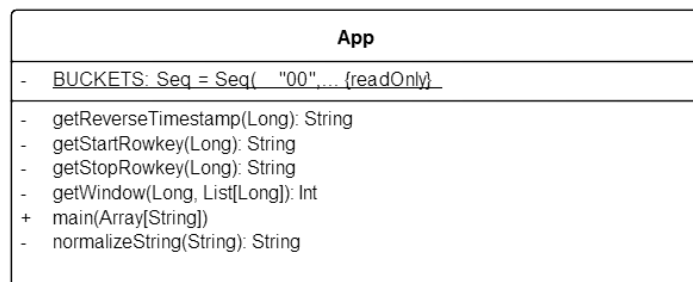


Figura 4.13: Diagrama de clases con atributos y funciones del módulo de cálculo de sentimiento asociado a términos arbitrario

4.3.5. API para realizar consultas personalizadas y recuperar resultados

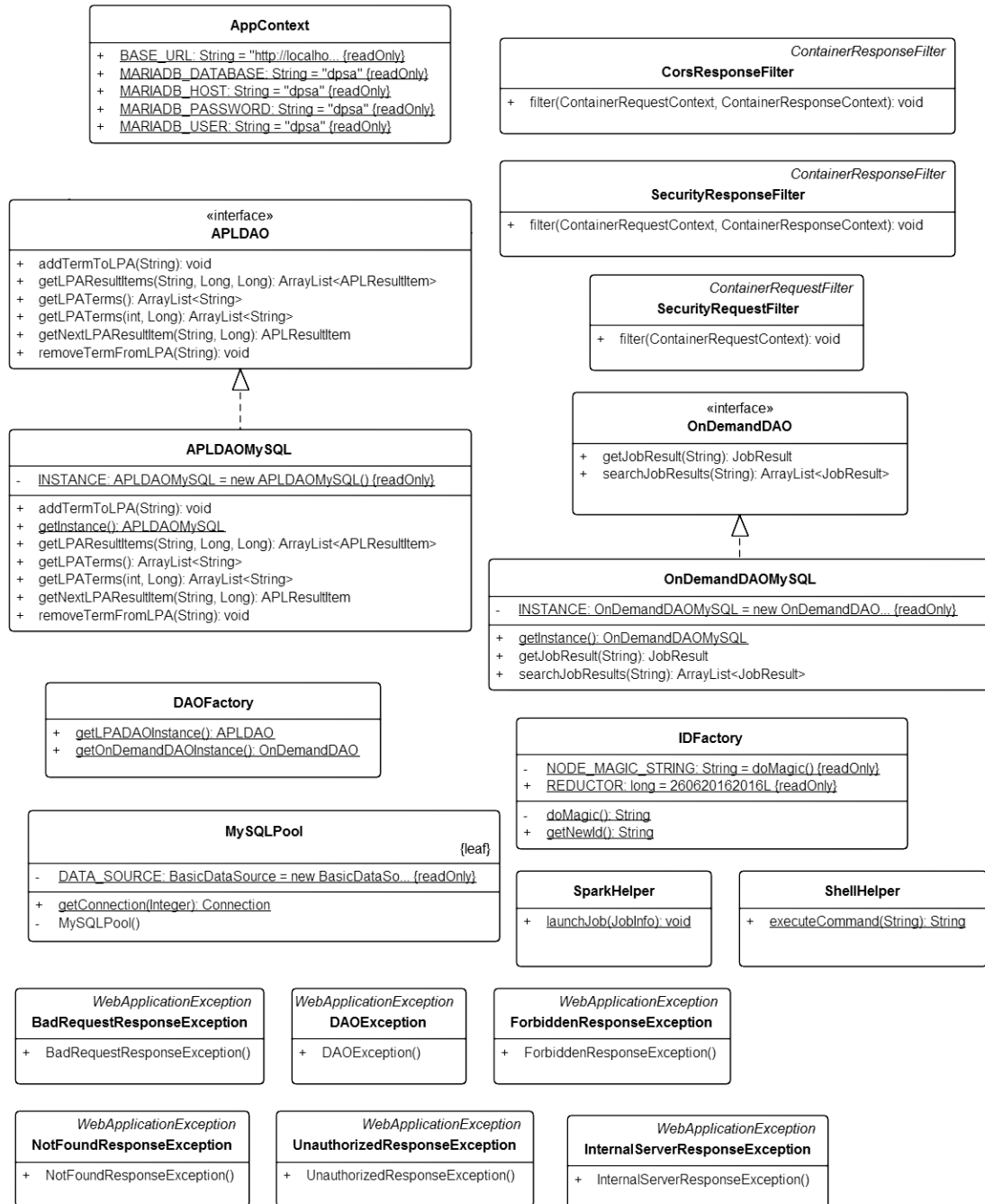


Figura 4.14: Diagrama de clases con atributos y funciones de la API para realizar consultas personalizadas y recuperar resultados (1/2)

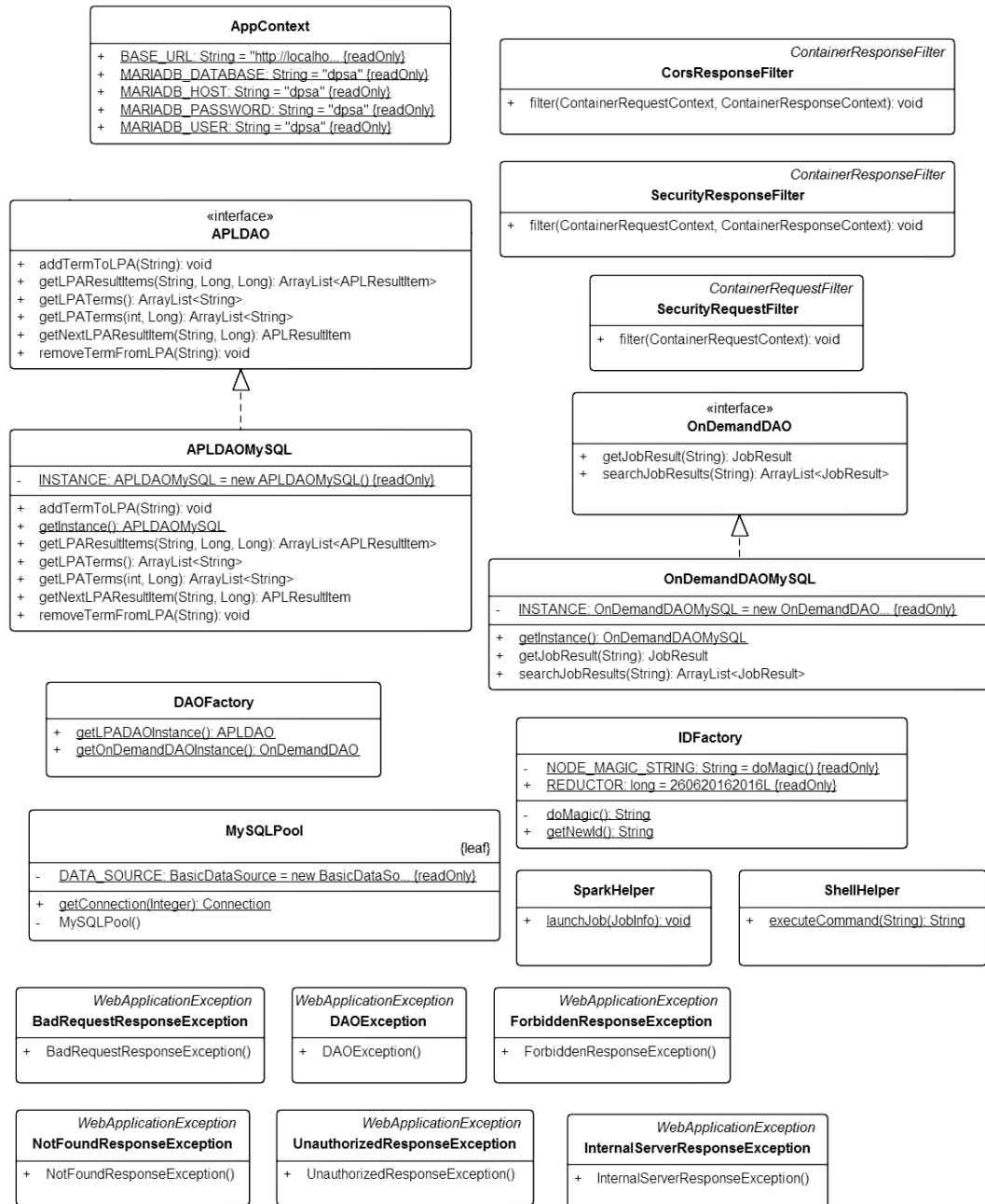


Figura 4.15: Diagrama de clases con atributos y funciones de la API para realizar consultas personalizadas y recuperar resultados (1/2)

4.4. Diagramas de interacción

Los diagramas expuestos a continuación tratan de representar la interacción del usuario con el sistema y cómo se desencadena la comunicación entre los distintos módulos en un nivel de abstracción elevado, debido a la complejidad que supondría la elaboración de un diagrama de secuencia detallado.

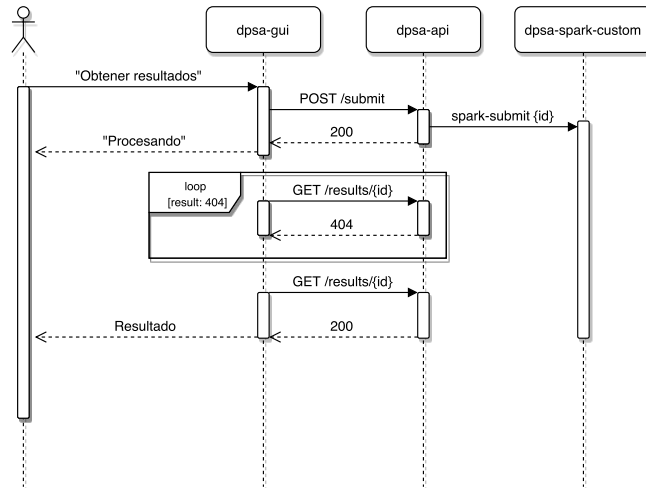


Figura 4.16: Diagrama de interacción de alto nivel para la consulta de un término arbitrario.

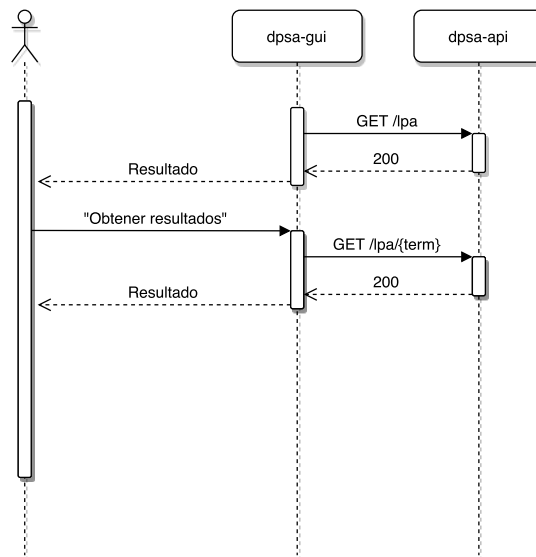


Figura 4.17: Diagrama de interacción de alto nivel para la consulta de un término de la LPA.

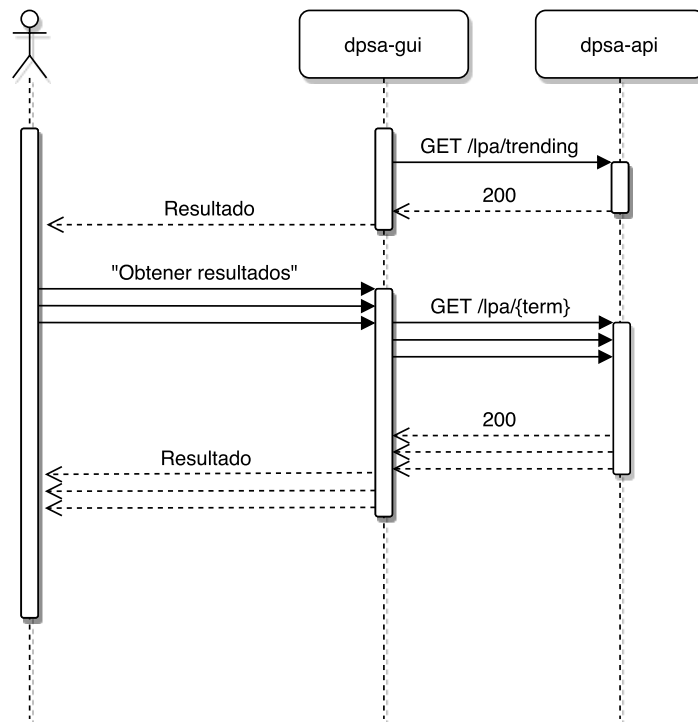


Figura 4.18: Diagrama de interacción de alto nivel para la visualización de datos en tiempo real.

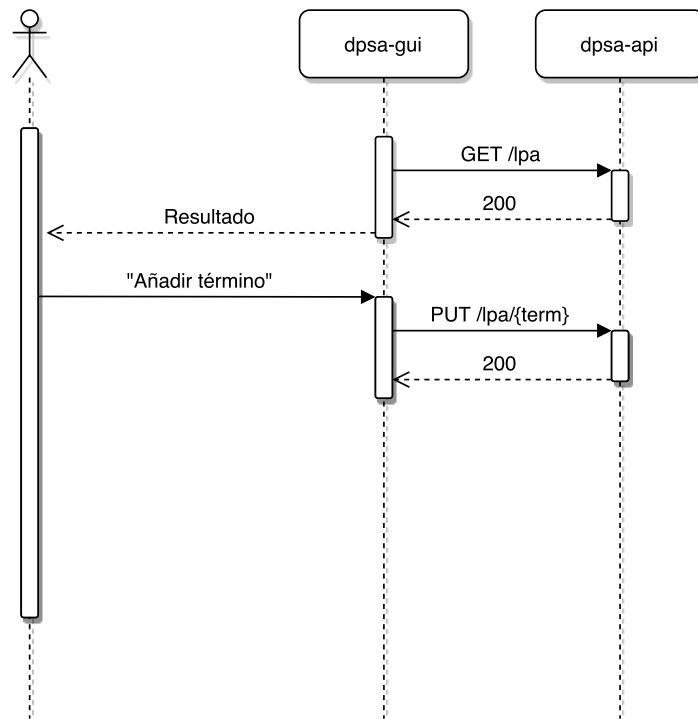


Figura 4.19: Diagrama de interacción de alto nivel para añadir un término a la LPA.

4.5. Tablas en los distintos SGBD

A continuación se muestra una figura con las distintas tablas utilizadas por el sistema en los distintos SGBD con sus atributos.

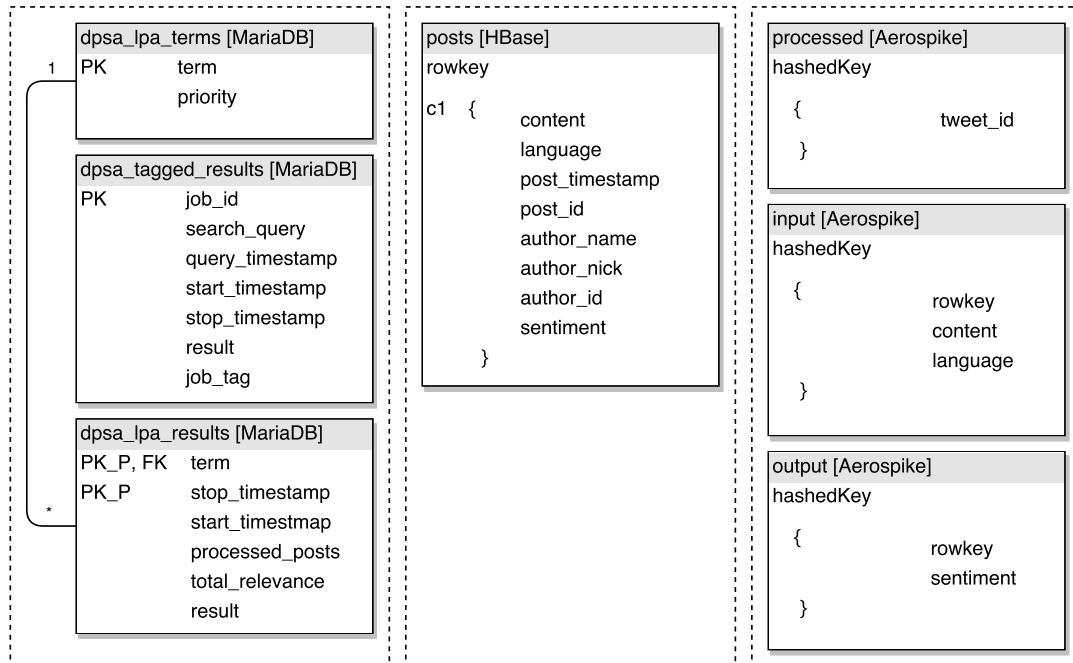


Figura 4.20: Tablas en los SGBD MariaDB, HBase y Aerospike.

Capítulo 5

Pruebas y validación

Se han diseñado una serie de pruebas para verificar el correcto funcionamiento de los módulos y garantizar el cumplimiento de los requisitos necesarios para alcanzar el criterio mínimo de aceptación.

Identificador de la prueba	PR-01
Requisitos involucrados	RF-01-01
Descripción	Se ejecutará el módulo de obtención de <i>tweets</i> para comprobar a continuación que las publicaciones se almacenan tanto en HBase como en Aerospike cada vez que se llena el <i>buffer</i> interno.
Resultado esperado	Tanto en HBase como en Aerospike deben existir las mismas publicaciones tras un volcado.
Estado	Superada

Tabla 5.1: PR-01

Identificador de la prueba	PR-02
Requisitos involucrados	RF-01-02, RF-01-03
Descripción	Generación de una batería de 100,000 identificadores, comprobando que no se repita ninguno.
Resultado esperado	100,000 identificadores únicos.
Estado	Superada

Tabla 5.2: PR-02

Identificador de la prueba	PR-03
Requisitos involucrados	RF-01-04, RF-01-05
Descripción	Comprobar que se almacena el contenido, idioma e identificador en el <i>buffer</i> de Aerospike y que se almacena además la marca temporal de la publicación, el identificador externo de la publicación, nombre, apodo e identificador del autor y sentimiento con valor -2.
Resultado esperado	Las mismas publicaciones en ambas bases de datos con los atributos comentados.
Estado	Superada

Tabla 5.3: PR-03

Identificador de la prueba	PR-04
Requisitos involucrados	RF-01-06
Descripción	Comprobar que los parámetros del fichero de configuración son utilizados en lugar de los por defecto.
Resultado esperado	Se imprimen por pantalla los parámetros leídos y coinciden con los proporcionados.
Estado	Superada

Tabla 5.4: PR-04

Identificador de la prueba	PR-05
Requisitos involucrados	RF-01-07
Descripción	Comprobar que no haya identificadores duplicados en el <i>buffer</i> de identificadores de los <i>tweets</i> . Para ello introducir 3 <i>tweets</i> de forma manual en el <i>buffer</i> y realizar un <i>bypass</i> del módulo de recuperación de publicaciones para que sólo pueda leer esos mismos tres <i>tweets</i> .
Resultado esperado	Los <i>tweets</i> no se escriben ni en el <i>buffer</i> externo ni en la tabla de persistencia.
Estado	Superada

Tabla 5.5: PR-05

Identificador de la prueba	PR-06
Requisitos involucrados	RF-02-01, RF-02-05
Descripción	Comprobar que la topología de Storm devuelve el sentimiento en tiempo real y que coincide con el proporcionado por una ejecución de los módulos originales. Para ello realizar el procesamiento de tres publicaciones con los módulos originales e insertar los mismos módulos en el <i>buffer</i> de Aerospike para que la aplicación lo compute.
Resultado esperado	Se obtiene el mismo resultado en el <i>buffer</i> de salida y en la ejecución de los módulos originales.
Estado	Superada

Tabla 5.6: PR-06

Identificador de la prueba	PR-07
Requisitos involucrados	RF-02-02
Descripción	Comprobar que la lectura por parte del módulo de análisis de sentimiento en tiempo real es destructiva en el <i>buffer</i> de entrada. Insertar tres publicaciones para su procesado.
Resultado esperado	El <i>buffer</i> de entrada queda vacío.
Estado	Superada

Tabla 5.7: PR-07

Identificador de la prueba	PR-08
Requisitos involucrados	RF-02-03, RF-02-04
Descripción	Comprobar que se almacenan los resultados en el <i>buffer</i> de salida y que se produce el traslado de la información a la base de datos de persistencia, eliminando los registros del <i>buffer</i> . Se puede realizar en el contexto existente tras la finalización de la prueba anterior.
Resultado esperado	Al ejecutar la aplicación de traslado <i>dpsa-btop</i> , se vacía el <i>buffer</i> de salida y se escriben los resultados en HBase.
Estado	Superada

Tabla 5.8: PR-08

Identificador de la prueba	PR-09
Requisitos involucrados	RF-02-06
Descripción	Comprobar que se omiten las publicaciones etiquetadas en un idioma distinto del castellano. Para ello añadir tres <i>tweets</i> en el <i>buffer</i> de entrada con el idioma etiquetado con “en” (inglés) y ejecutar el analizador de sentimiento en tiempo real.
Resultado esperado	Los <i>tweets</i> se eliminan del <i>buffer</i> pero no se reflejan en el <i>buffer</i> de salida.
Estado	Superada

Tabla 5.9: PR-09

Identificador de la prueba	PR-10
Requisitos involucrados	RF-03-01, RF-03-02
Descripción	Comprobar que el módulo de procesamiento periódico de la LPA se ejecuta cada minuto y que almacena los resultados en MariaDB.
Resultado esperado	Cada minuto es posible ver la creación de un nuevo trabajo Spark y la actualización de los resultados parciales en caso de haberlos de los distintos términos de la LPA.
Estado	Superada

Tabla 5.10: PR-10

Identificador de la prueba	PR-11
Requisitos involucrados	RF-03-03, RF-04-01
Descripción	Comprobar que se sobrescriben los parámetros pasados por argumento del módulo de procesamiento periódico de la LPA.
Resultado esperado	Se muestran por pantalla los parámetros recibidos correctos.
Estado	Superada

Tabla 5.11: PR-11

Identificador de la prueba	PR-12
Requisitos involucrados	RF-03-04, RF-04-02
Descripción	Comprobar que la función de normalización del contenido de las publicaciones digitales funciona correctamente probándola con una publicación aleatoria con símbolos y letras mayúsculas y minúsculas con caracteres latinos.
Resultado esperado	La misma cadena es transformada a minúsculas sin caracteres no anglosajones (se utiliza el más parecido) y sin los símbolos presentes en el contenido original.
Estado	Superada

Tabla 5.12: PR-12

Identificador de la prueba	PR-13
Requisitos involucrados	RF-04-03
Descripción	Comprobar que los resultados se almacenan correctamente al realizar una consulta personalizada en la base de datos con los atributos especificados en el requisito asociado a la prueba.
Resultado esperado	Aparece en la tabla de resultados una nueva entrada con la información obtenida.
Estado	Superada

Tabla 5.13: PR-13

Identificador de la prueba	PR-14
Requisitos involucrados	RF-04-03
Descripción	Comprobar que los resultados almacenados tras la realización de una consulta personalizada son correctos. Para ello insertar manualmente tres <i>tweets</i> con su correspondiente sentimiento en la tabla de HBase y comparar los resultados obtenidos a mano con los almacenados en la base de datos al procesar la consulta que contenga un término presente en las tres publicaciones.
Resultado esperado	Los resultados almacenados en la tabla de resultados se corresponden con los calculados manualmente.
Estado	Superada

Tabla 5.14: PR-14

Identificador de la prueba	PR-15
Requisitos involucrados	RF-05-01, RF-05-02
Descripción	Comprobar que tras lanzar un trabajo personalizado a través de la API, se ejecute en el sistema y devuelva como respuesta el identificador del trabajo. Acto seguido comprobar que se pueden obtener los resultados del mismo a través de una consulta al ID del trabajo recibido.
Resultado esperado	La operación se completa sin problemas y el resultado obtenido es el mismo que el presente en la base de datos.
Estado	Superada

Tabla 5.15: PR-15

Identificador de la prueba	PR-16
Requisitos involucrados	RF-05-03
Descripción	Comprobar que es posible localizar consultas realizadas con anterioridad a partir de la cadena de búsqueda utilizada en su momento.
Resultado esperado	Proporcionando una cadena de comparación, comprobar que coincide el resultado devuelto de búsquedas filtradas con el existen en la base de datos.
Estado	Superada

Tabla 5.16: PR-16

Identificador de la prueba	PR-17
Requisitos involucrados	RF-05-03, RF-05-04
Descripción	Comprobar que es posible añadir un término a la LPA y retirarlo a continuación. Comprobar también que no sucede nada si ya existe el término a añadir y que se devuelve un error en caso de tratar eliminar un término no existente.
Resultado esperado	En cada paso, la base de datos muestra datos coherentes con la acción realizada.
Estado	Superada

Tabla 5.17: PR-17

Identificador de la prueba	PR-18
Requisitos involucrados	RF-05-06
Descripción	Comprobar que dada una marca temporal y un término de la LPA, se obtiene el siguiente resultado disponible almacenado.
Resultado esperado	El resultado obtenido coincide con el inmediatamente posterior a la marca temporal indicada en la tabla de resultados.
Estado	Superada

Tabla 5.18: PR-18

Identificador de la prueba	PR-19
Requisitos involucrados	RF-05-07
Descripción	Comprobar que se obtienen los datos indicados en el requisito asociado y que corresponden con los almacenados en la base de datos para un término de la LPA y un intervalo con datos.
Resultado esperado	El resultado recibido debe coincidir con el presente en la BD.
Estado	Superada

Tabla 5.19: PR-19

Identificador de la prueba	PR-20
Requisitos involucrados	RF-05-08
Descripción	Comprobar que se obtienen los tres términos (indicando este número como parámetro) con más publicaciones en los últimos 5 minutos (dejando el sistema en funcionamiento 10 minutos).
Resultado esperado	Los tres términos devueltos coinciden con los que tienen más publicaciones en los últimos 5 minutos.
Estado	Superada

Tabla 5.20: PR-20

Identificador de la prueba	PR-22
Requisitos involucrados	RF-05-09, RF-05-10
Descripción	Comprobar que se pueden obtener términos de la LPA filtrados por nombre y que se puede obtener la lista de todos los términos utilizando una lista de términos personalizada.
Resultado esperado	Para las búsquedas filtradas y para la solicitud de la totalidad de términos, los resultados coinciden con el contenido presente en la base de datos.
Estado	Superada

Tabla 5.21: PR-22

Identificador de la prueba	PR-23
Requisitos involucrados	RF-06-01, RF-06-02, RF-06-03, RF-06-04, RF-06-05
Descripción	Comprobar que se realizan las llamadas a la API de forma correcta a partir de los datos de los formularios en el inspector de un navegador web. Ejecutar todos los casos de uso.
Resultado esperado	Las llamadas a la API se realizan de forma correcta y los parámetros son traducidos correctamente del formulario a la petición enviada.
Estado	Superada

Tabla 5.22: PR-23

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones	110
6.2. Trabajo futuro	111

6.1. Conclusiones

El proyecto llevado a cabo supone un primer contacto con las grandes posibilidades del procesamiento masivo de datos. El arte de recuperar cantidades ingentes de datos y de obtener información interesante donde sólo hay datos inconexos que no revelan nada sin ser procesados. La implementación del sistema en arquitecturas distribuidas permite un escalado horizontal.

Se describe e implementa un sistema de captación de *tweets* que supera los medios gratuitos oficiales. Se utilizan *buffers* de interconexión entre los distintos módulos que permiten a cada módulo continuar su labor rápidamente gracias a las posibilidades de almacenamiento en RAM de Aerospike.

Con Storm se convierte el sistema de tuberías tradicional de los módulos PLN en una topología paralelizable. Cada instancia de cada módulo es independiente y espera trabajo de cualquier instancia inmediatamente anterior en la secuencia original de ejecución. La gran ventaja es que aparece la paralelización a nivel de etapas de procesamiento —bolts— por lo que si una determinada etapa es más costosa computacionalmente, puede paralelizarse más que el resto. Así es posible reducir cuellos de botella que aparecerían en otros paradigmas como MapReduce optimizando mejor los recursos disponibles.

Cada *tweet* recuperado es categorizado en función del sentimiento calculado. El procesamiento en tiempo real con Spark permite realizar consultas sobre el conjunto total de datos ya procesados. Así, es posible averiguar si los usuarios de la red social —en el contexto concreto de este Trabajo de Fin de Grado— comentan positiva o negativamente en mayor o menor medida determinadas marcas, productos, servicios, lugares, etc. Es posible tanto realizar consultas arbitrarias como explorar una serie de términos (LPA) que el sistema consulta periódicamente para permitir una interacción mucho más veloz al usuario y generando a su vez históricos de los términos paulatinamente. Estas consultas son incrementales y realizan tareas de agregación, por lo que la carga que soporta el sistema es relativamente ligera.

La consulta de sentimiento tanto de términos arbitrarios como de términos presentes en la LPA se puede realizar a través de una API RESTful, así como la gestión de la lista de términos. Esto permite una desconexión con la interfaz que deja abierto un gran abanico de posibilidades para la adaptación del sistema a distintos ámbitos, tanto para un usuario final —en los diversos tipos de dispositivos— como para sis-

temas que aprovechen la información de forma automatizada. Para acompañar a la API, se creó una interfaz que cubre los casos de uso del sistema y mediante la que es posible interactuar con el mismo sin conocimientos técnicos.

Es importante crear software modular, permitiendo una fácil evolución futura. En este caso concreto, un aspecto fundamental a tener en cuenta es la escalabilidad ya que el sistema podría llegar a procesar cantidades de datos que en un principio podrían parecer exageradas. Con un mayor número de máquinas de similares características es posible aumentar la potencia del sistema de forma lineal. La utilización de técnicas alternativas para la captación de *tweets* permite obtener una muestra mucho mayor que con las herramientas disponibles pero que sigue representando un pequeño porcentaje del total de publicaciones. Existen algunos sistemas de análisis de sentimiento de *tweets* gratuitos, pero suelen utilizar un catálogo de publicaciones obtenido de la ya comentada “Streaming API” (insuficiente para el procesamiento de términos arbitrarios en ventanas temporales próximas al momento actual) o realizar consultas sobre un término específico sobre la “Search API”, devolviendo tan solo un puñado de resultados procesados.

6.2. Trabajo futuro

El primer paso a partir de este momento es realizar la configuración de un clúster con los distintos *frameworks* para poner en marcha el sistema de forma distribuida, ya que está preparado y fue concebido pensando para ello.

La técnica utilizada para la captación de un número mayor de *tweets* funciona, y es fácil alcanzar más de 250 *tweets* por segundo en un solo idioma con las condiciones adecuadas. Sin embargo, esto sigue siendo una cifra poco significativa con respecto al total de publicaciones generadas en cada instante. Debido a las limitaciones establecidas por Twitter en las consultas en directo, no hay muchas esperanzas de mejora por esta vía. Sin embargo existe una solución bastante eficaz que podría potenciar el sistema. Twitter permite localizar *tweets* publicados en el pasado indicando intervalos de tiempo y una cadena de búsqueda, además de otros filtros como el idioma. Solo es posible buscar del día anterior al actual hacia atrás mediante este mecanismo, por lo que esta técnica no mejoraría la captación de *tweets* en tiempo real. Sin embargo, se comprobó en la práctica que mediante este sistema, Twitter no ofrece una muestra aleatoria de las coincidencias, si no que se listan todas las publicaciones que coinciden con el filtro. Es decir, sería posible recuperar casi la totalidad de *tweets* con un retraso de un día aplicando la misma técnica que se aplicó a las búsquedas en directo.

Cada publicación se valora por igual independientemente de quién lo haya escrito. Introduciendo el concepto de relevancia, se pueden producir resultados más realistas. Si se calcula un parámetro para cada publicación en función del autor, que funcione como multiplicador del resultado, los resultados obtenidos serían más realistas. Algunos expertos estiman que alrededor de un 4% —de media, ya que también influyen factores como la hora a la que se publique— de los seguidores de un usuario en Twitter leen una determinado mensaje que publique. Un posible valor para la relevancia de un *tweet* podría ser considerar que dicha publicación es compartida en opinión por la mitad de los seguidores del autor de ese 4% que lo han visto. Se podrían tener en cuenta otros factores como las acciones realizadas directamente sobre el *tweet*, como los *retweets* o los *likes*. En este último caso, no tendría mucho sentido tenerlos en cuenta en tiempo real, ya que no serían representativos al estar al principio del ciclo de vida útil del *tweet*, sin embargo, sí que tendría más sentido en la recuperación de publicaciones del día anterior.

Buscando de nuevo una mayor representación de la población, es interesante incluir diversas fuentes de opinión, ya que el conjunto de los usuarios de Twitter no es representativo del total de la población. Se podrían incluir noticias de periódicos con su respectiva relevancia en función del número de lectores, carácter regional o estatal, etc. También se podrían incluir publicaciones de Facebook, entradas en blogs, etc.

El sistema descrito procesa publicaciones en castellano, pero está diseñado para añadir soporte a más idiomas. Sería interesante adaptar los módulos de *LinguaKit* ya existentes para otros idiomas a Java para poder incorporarlos al sistema.

Una de las grandes características del sistema es la inclusión de una lista de términos de procesamiento automático que evita consultas pesadas sobre el sistema al almacenar progresivamente los datos asociados a los términos presentes en dicha lista. Sería interesante que la lista evolucionase automáticamente, localizando temas o términos emergentes entre las publicaciones recuperadas o que consultase a fuentes externas para la obtención de los mismos como Google Trends.

El sistema podría evolucionar realizando otras clasificaciones. Por ejemplo, detectando las publicaciones que tratan sobre películas para acabar mostrando al usuario las películas mejor valoradas de la semana o permitirle simplemente poder filtrar consultas por categorías o contextos.

Apéndice A

Manual técnico de despliegue

Antes de comenzar es necesario instalar Docker¹.

A.1. Generación de ejecutables

Para generar los ejecutables de la API (dpsa-api), el módulo de traslado de resultados (dpsa-btop), el módulo de obtención de *tweets* (dpsa-polypus), los módulos Spark (dpsa-spark-custom y dpsa-spark-lpa) y el módulo de análisis de sentimiento en tiempo real (dpsa-storm), tan solo es necesario ejecutar en cada directorio el siguiente comando:

```
mvn package
```

Esto generará un “.jar” por módulo en el subdirectorio “target” de cada uno.

A.2. Arranque de contenedores

Una vez en el directorio “dockerfiles”, en primer lugar es necesario descargar las versiones binarias (*tar.gz*) de HBase, Spark y Storm ya que en los directorios originales de cada *framework* se incluían en el subdirectorio “archives” y han sido eliminados para ahorrar espacio. Además también es necesario incluir el *tar.gz* de Oracle JDK 8 en el mismo subdirectorio para los mismos tres casos. El resto de *dockerfiles* obtienen de la red los recursos necesarios.

Se introducen los ejecutables del paso anterior en la carpeta “shared”. También se incluye el contenido del directorio “dpsa-gui”. Del directorio “dpsa-polypus” copiamos los siguientes ficheros: “dpc.conf” y “rae_freq_es_250.sg”. Del directorio “dpsa-api” hacemos lo mismo con el fichero “spark-launcher.sh”. Por último copiamos al completo

¹<https://docs.docker.com/engine/installation/>

el directorio “dpsa-api”.

Ahora ya es posible arrancar los contenedores. Con el siguiente comando construimos las imágenes y ejecutamos los contenedores necesarios. Los contenedores de MariaDB y Aerospike se autoconfiguran y no necesitan interacción manual.

```
bash aio.sh
```

Entramos en el contenedor de Spark:

```
docker exec -ti spark bash
```

Para lanzar la API con Jetty embebido es necesario instalar *wget* y Maven:

```
yum -y install wget
```

A continuación ejecutamos los siguientes comandos para instalar Maven:

```
MAVEN_VERSION=3.3.9
MAVEN_BIN_ARCHIVE=apache-maven-$MAVEN_VERSION-bin.tar.gz
MAVEN_DOWNLOAD_LINK=http://apache.rediris.es/maven/maven-3/$MAVEN_VERSION/
    binaries/$MAVEN_BIN_ARCHIVE
MAVEN_INSTALL_DIR=/opt/apache/maven
mkdir -p $MAVEN_INSTALL_DIR
cd $MAVEN_INSTALL_DIR
wget $MAVEN_DOWNLOAD_LINK
tar xzf $MAVEN_BIN_ARCHIVE
rm -f $MAVEN_BIN_ARCHIVE
ln -sf apache-maven-$MAVEN_VERSION current
echo 'export M2_HOME=$MAVEN_INSTALL_DIR/current' > /etc/profile.d/apache-
    maven.sh
echo 'if ! echo $PATH | grep -q '$MAVEN_INSTALL_DIR'/current/bin ; then
export PATH=$M2_HOME/bin:$PATH; fi' >> /etc/profile.d/apache-maven.sh
. /etc/profile.d/apache-maven.sh
```

Establecemos `JAVA_HOME` con el comando `export JAVA_HOME=/usr/java/default` y acto seguido `mvn jetty:run` dentro del directorio `/shared/dpsa-api`.

Tras unos segundos la API entraría en funcionamiento y podríamos acceder al portal web a través de la siguiente dirección desde un navegador en el anfitrión: `http:127.0.0.1/` y poder realizar algunas acciones como gestionar la LPA.

Ahora nos conectamos al contenedor de HBase:

```
docker exec -ti hbase bash
```

Ejecutamos `hbase shell` y creamos la tabla necesaria introduciendo en la consola `create 'posts', 'c1'`.

Nos conectamos al contenedor de Storm para lanzar el recolector de *tweets*:

```
docker exec -ti storm bash
```

Lanzamos el módulo:

```
cd /shared; java -jar /shared/polypus.jar
```

Lanzamos ahora la topología de Storm en otra terminal:

```
java -jar dpsa-storm.jar
```

Por último lanzamos *btop* para trasladar los resultados a HBase:

```
java -jar dpsa-btop.jar
```

Para habilitar el procesamiento periódico y poder empezar a visualizar información en las secciones “Explorar” y “Tiempo real” es necesario en el contenedor de Spark configurar Cron como se mencionó en el cuarto capítulo de esta memoria. Sin necesidad de realizar más configuraciones, a través de la interfaz ahora ya es posible realizar búsquedas arbitrarias:

```
docker exec -ti spark bash
```

```
crontab -e
```

```
SHELL=/bin/bash
```

```
PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/default/java:/opt/spark/default
```

```
HOME=/root
```

```
* * * * * spark-submit /shared/dpsa-spark-lpa.jar --language=es
```

Apéndice B

Manual de usuario

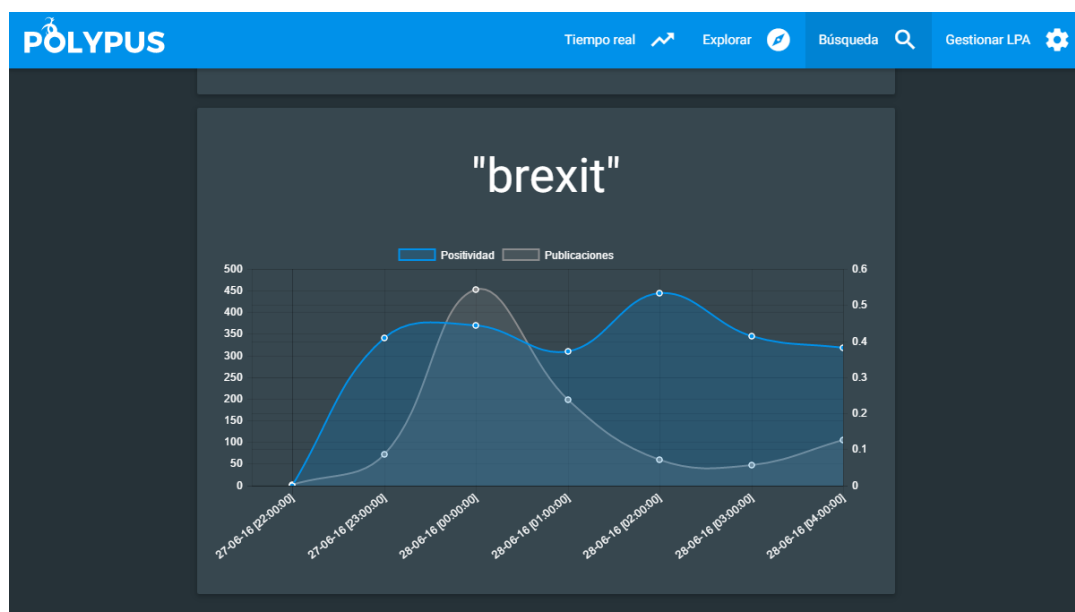
A través de la URL `http://127.0.0.1/`, si estamos ejecutando el sistema en local, podemos acceder a la interfaz web.

B.1. Consultar el ratio de positividad de un término arbitrario

Pulsando en “Búsqueda” en la barra de navegación podemos acceder a la vista adecuada. En primer lugar existe un campo para introducir el término de búsqueda. A continuación hay una serie de filtros: dos campos relativos al intervalo temporal de búsqueda que queremos acotar, el filtro de idioma (forzado a castellano), el tamaño de la ventana en minutos y un campo para definir la etiqueta de la consulta con la que queremos que se almacene en el sistema. Por defecto, el intervalo se establece a un día de duración hacia atrás desde el momento actual. Una vez completados los filtros a nuestro gusto el botón “Obtener resultados” nos permite lanzar el trabajo.

The screenshot shows the POLYPUS web application interface. At the top, there is a blue header with the POLYPUS logo and navigation links: "Tiempo real", "Explorar", "Búsqueda", and "Gestionar LPA". Below the header, a search bar contains the text "brexit". Underneath the search bar, there are filter settings: "Fecha inicial de filtrado" (06-07-2016 11:13:05), "Fecha final de filtrado" (07-07-2016 11:13:05), "Idioma" (es), "Tamaño de la ventana (minutos)" (60), and "Etiqueta para la búsqueda" (ninguna). A blue button labeled "OBTENER RESULTADOS" is positioned below the filters. At the bottom of the form, a unique identifier "18142971207262627574" is displayed above a loading spinner.

Una vez procesado, el resultado se mostrará por pantalla instantáneamente en una gráfica con dos elementos: ratio de positividad y número de publicaciones procesadas para cada ventana:



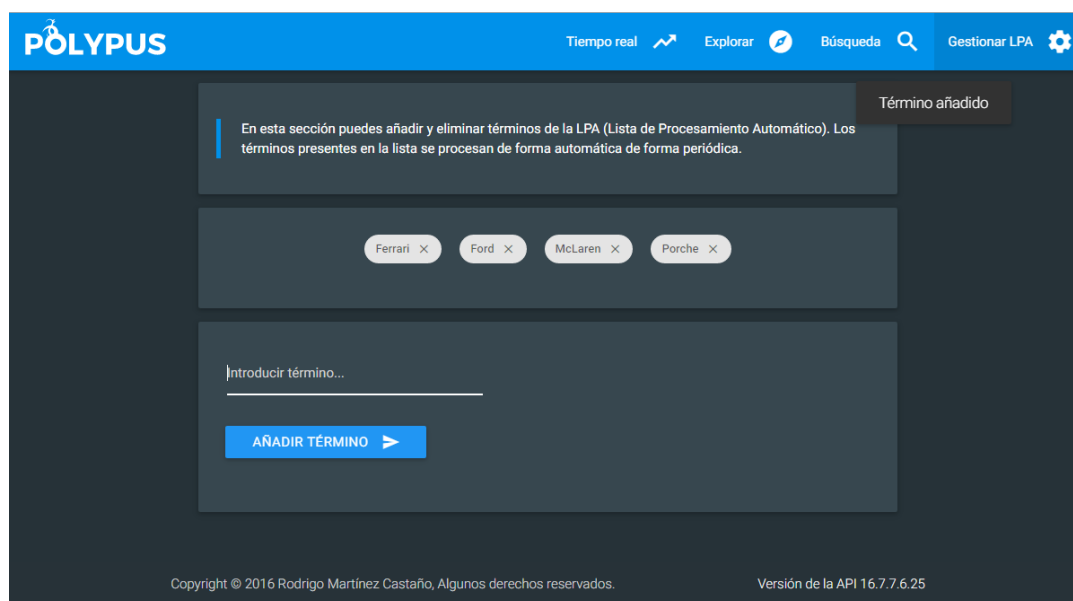
B.2. Consultar el ratio de positividad de los términos de la LPA

En la sección “Explorar” podemos visualizar, nada más cargar la página, una serie de botones azules. Son los términos disponibles en la LPA. Para seleccionar uno basta con pulsar su botón y a continuación cubrimos los datos de filtrado y ejecutamos la consulta de forma análoga a las búsquedas de términos arbitrarios:



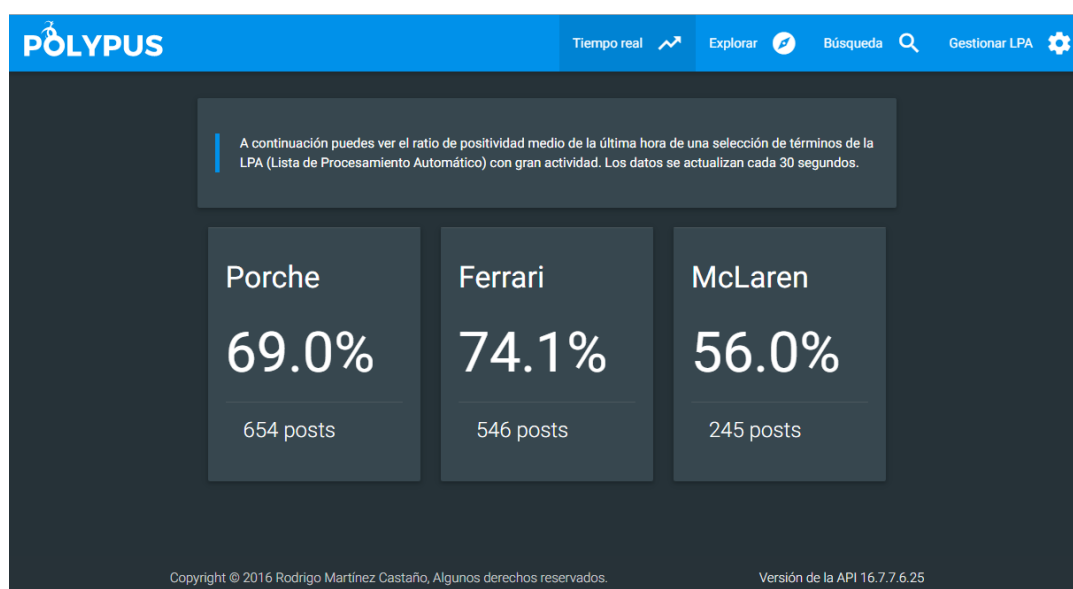
B.3. Gestionar LPA

En la sección “Gestionar LPA” es posible añadir y eliminar términos de la LPA. Al cargar la página se muestran varias etiquetas blancas con un término escrito en su interior y una cruz en el lado derecho. Son los términos presentes en la LPA. Para eliminarlos basta con pulsar la equis. Para añadir nuevos términos es suficiente con introducirlos en el cajetín inferior y pulsar en “Añadir término”. Los cambios se visualizan instantáneamente:



B.4. Consultar la evolución en directo de los términos más activos de la LPA

En la sección “Tiempo Real” podemos visualizar la evolución de una selección de términos de la LPA en tiempo real. Se escogen los términos con más actividad en la última hora (más publicaciones procesadas). Cada minuto se realiza una actualización de los datos de forma automática.



Apéndice C

Glosario

ACID Acrónimo de Atomicity, Consistency, Isolation and Durability: Atomicidad, Consistencia, Aislamiento y Durabilidad.

Algoritmo de consenso Conjunto prescrito de instrucciones o reglas bien definidas, ordenadas y finitas que define cómo un conjunto de procesos han de ponerse de acuerdo en un valor una vez uno o varios de ellos hagan una sugerencia.

Alta disponibilidad Es un protocolo de diseño del sistema y su implementación asociada que asegura un cierto grado absoluto de continuidad operacional durante un período de medición dado.

API La interfaz de programación de aplicaciones es el conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Buffer Es un espacio de memoria, en el que se almacenan datos de manera temporal, normalmente para un único uso.

Centro de datos Ubicación donde se concentran los recursos necesarios para el procesamiento de la información de una organización.

Commit Consignar un conjunto de cambios "tentativos" de forma permanente.

Crontab Cron es un administrador regular de procesos en segundo plano que ejecuta procesos o guiones a intervalos regulares. Los procesos que deben ejecutarse y la hora en la que deben hacerlo se especifican en el fichero crontab.

Desfragmentación Proceso mediante el cual se acomodan los archivos en un disco para que no se aprecien fragmentos de cada uno de ellos, de tal manera que el archivo quede contiguo y sin espacios dentro del mismo.

Función hash Una función hash tiene como entrada un conjunto de elementos, que suelen ser cadenas, y los convierte en un rango de salida finito, normalmente cadenas de longitud fija.

Endpoint URL donde un servicio puede ser accedido por una aplicación cliente.

Escalado horizontal Un sistema escala horizontalmente si al agregar más nodos al mismo, el rendimiento de éste mejora.

Framework Estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software.

Git Software de control de versiones.

Hardware Partes físicas de un sistema informático.

Hostname Nombre único y relativamente informal que se le da a un dispositivo conectado a una red informática.

IDE Aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software.

Latex Sistema de composición de textos, orientado a la creación de documentos escritos que presenten una alta calidad tipográfica.

Master La arquitectura maestro/esclavo es una arquitectura de sistemas en los que un componente realiza labores de control, supervisión y coordinación (maestro), y los restantes realizan tareas específicas (esclavos), devolviendo los resultados al maestro.

NoSQL Clase de sistemas de gestión de bases de datos que difieren del modelo clásico del sistema de gestión de bases de datos relacionales (RDBMS) en aspectos importantes: no usan SQL como el principal lenguaje de consultas, los datos almacenados no requieren estructuras fijas y no suelen soportar operaciones JOIN ni garantizar completamente ACID. Suelen escalar bien horizontalmente.

Opensource El código abierto es el software distribuido y desarrollado libremente. Se focaliza más en los beneficios prácticos (acceso al código fuente) que en cuestiones éticas o de libertad que tanto se destacan en el software libre.

Join La sentencia join en SQL permite combinar registros de dos o más tablas en una base de datos relacional.

Padding Relleno.

Peer Participantes en un sistema con los mismos privilegios y capacidades.

PLN Campo de las ciencias de la computación, inteligencia artificial y lingüística que estudia las interacciones entre las computadoras y el lenguaje humano.

Pool de conexiones Colección de conexiones abiertas a una base de datos de manera que puedan ser reutilizadas al realizar múltiples consultas o actualizaciones.

Publicación digital Cualquier contenido difundido en medios digitales por un usuario y principalmente compuesto por texto, para poder realizar el análisis de sentimiento.

RDBMS Sistema de gestión de bases de datos que sigue el modelo relacional.

Repositorio Git Lugar centralizado donde se almacena y mantiene toda la información relativa al control de versiones realizado con Git.

RESTful Estilo de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web.

Scraping web Técnica utilizada mediante programas de software para extraer información de sitios web.

Sincronización Acción mediante la que dos o más dispositivos se actualizan de forma que contengan los mismos datos.

Timestamp Secuencia de caracteres que denotan la fecha y hora.

Tweet Mensajes de texto plano de corta longitud enviados a través de la red social Twitter.

Twitter Firehose *Endpoint* de la *Streaming API* de Twitter que devuelve todos los *tweets* públicos.

Twitter Search API API formada por un conjunto de *endpoints* que permiten realizar búsquedas con distintos filtros en el servicio Twitter.

Twitter Streaming API API formada por un conjunto de *endpoints* que permiten recuperar distintos tipos de flujos de *tweets*.

UML Lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.

Bibliografía

- [1] Zach Bulygo. “How Netflix uses analytics to select movies, create content, and make multimillion dollar decisions”. [En línea]. Septiembre de 2013, [Marzo de 2016]. Disponible en: <https://blog.kissmetrics.com/how-netflix-uses-analytics/>
- [2] Sean T. Allen, Matthew Jankowski y Peter Pathirana, *Storm Applied, Strategies for real-time event processing*, Manning Publications, New York, 2015.
- [3] Twitter, Inc. “The Streaming APIs documentation”. [En línea]. [Marzo de 2016]. Disponible en: <https://dev.twitter.com/streaming/overview>
- [4] Twitter, Inc. “The Search API documentation”. [En línea]. [Marzo de 2016]. Disponible en: <https://dev.twitter.com/rest/public/search>
- [5] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes y Robert E. Gruber (2006). “Bigtable: A Distributed Storage System for Structured Data”. OSDI ’06. [En línea]. Disponible en: <http://static.googleusercontent.com/media/research.google.com/gl//archive/bigtable-osdi06.pdf>
- [6] Universidade de Santiago de Compostela, “USC Investigación”, [En línea]. Mayo de 2008, [Julio de 2016] Disponible en: http://imaisd.usc.es/ftp/oit/documentos/591_gl.pdf
- [7] The Apache Software Foundation. “Apache HBase™ Reference Guide”. [En línea]. Marzo de 2016, [Marzo de 2016]. Disponible en: <https://hbase.apache.org/book.html>
- [8] P. Gamallo, M. García y S. Fernández-Lanza, “Tass: A naive-bayes strategy for sentiment analysis on spanish tweets”, en Workshop on Sentiment Analysis en SEPLN (TASS2013), Madrid, España, 2013, pp. 126-132.

- [9] José M. Abuín, Juan C. Pichel, Tomás F. Pena, Pablo Gamallo, y Marcos García, “Perldoop: Efficient Execution of Perl Scripts on Hadoop Clusters”, en IEEE International Conference on Big Data, Washington DC, USA, 2014, pp. 1-6.
- [10] Jeffrey Dean y Sanjay Ghemawat (2004). “MapReduce: Simplified Data Processing on Large Clusters”. OSDI '04. [En línea]. Disponible en: <http://static.googleusercontent.com/media/research.google.com/gl/archive/mapreduce-osdi04.pdf>
- [11] The Apache Software Foundation. “Apache Spark™ documentation”. [En línea]. Marzo de 2016, [Marzo de 2016]. Disponible en: <http://spark.apache.org/docs/1.6.1/>
- [12] The Apache Software Foundation. “Apache Hadoop™ documentation”. [En línea]. Enero de 2016, [Marzo de 2016]. Disponible en: <http://hadoop.apache.org/docs/r2.7.2/>
- [13] Docker.com. “The Docker user guide”. [En línea]. [Junio de 2016]. Disponible en <https://docs.docker.com/v1.8/userguide/>
- [14] Aerospike, Inc. “Aerospike Architecture”. [En línea]. [Junio de 2016]. Disponible en <http://www.aerospike.com/technologies/>
- [15] Alex Baranau. “HBaseWD: Avoid RegionServer Hotspotting Despite Sequential Keys”. [En línea]. Abril de 2012, [Junio de 2016]. Disponible en: <https://sematext.com/blog/2012/04/09/hbasewd-avoid-regionserver-hotspotting-despite-writing-records-with-sequential-keys/>

