Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

"HATE CRIME CLASSIFICATION": A LA CAZA DEL DELITO DE ODIO EN TWITTER

Alumno/Alumna: Hernandez Barandika, Fernando

Director/Directora (1): Azanza Sese, Maider

Director/Directora (2): Ceberio Uribe, Josu

Curso: 2017-2018

Fecha: martes, 24 de julio de 2018

Contenido

Índice de Tablas

Índice de Tablas

1. Introducción

En los últimos años se advierte una proliferación de conductas de distinto signo que suele agruparse bajo la denominación de "delitos de odio". Todas tienen en común el hecho de que las victimas lo son por formar parte de colectivos que tradicionalmente han sufrido la discriminación, el hostigamiento y en ocasiones también la violencia por profesar una religión, por su género, etnia, orientación sexual, etc. La concienciación sobre lo intolerable de estos ataques ha conllevado, desde una perspectiva penal, a un aumento de los comportamientos considerados delictivos, llegándose a castigar, en la actualidad, no sólo aquellos supuestos en los que personas concretas son objeto de hechos violentos por pertenecer a alguno de esos colectivos, sino también aquellos casos en los que se emiten mensajes que "se limitan" a incitar al odio y a la discriminación, así como aquellos que entrañan humillación o menosprecio de personas que pertenezcan a alguno de los grupos considerados vulnerables.

Twitter[1] es un servicio de microblogging con gran repercusión social y una de las redes sociales más utilizadas actualmente. Se estima que tiene más de 500 millones de usuarios, generando 65 millones de tuits al día y maneja más de 800 000 peticiones de búsqueda diarias (Kalil, 2018).

Por ello, es una plataforma en la que se constatan muchas de las conductas penalmente relevantes que, según Art.22 del Código Orgánico Integral Penal (Ley Orgánica 10/1995, de 23 de noviembre, De las circunstancias que agravan la responsabilidad criminal, 1995), son las acciones u omisiones que ponen en peligro o producen resultados lesivos, descriptibles y demostrables. Alentados por sucesos concretos, no son pocos los usuarios que recurren a Twitter para lanzar mensajes que, en apariencia, coinciden con la descripción de un delito de odio. Teniendo en cuenta que la pena de los autores del delito se agrava en caso de recurrir a internet y a las redes sociales para su comisión, resulta un objetivo de primer nivel recabar información sobre estos mensajes para poder dilucidar: a) cuales estarían amparados por la libertad de expresión; b) cuales son y qué características tienen aquellos mensajes que pueden ser considerados delictivos.

Recientemente, se ha visto cómo algunos magistrados de diferentes comunidades autónomas han condenado a usuarios de Twitter por

[1] https://twitter.com/

comentarios vejatorios sobre el accidente de avión de Germanwindg (Vidales, 2017) o por difundir mensajes de odio hacia las mujeres asesinadas por violencia machista (Pérez, 2018).

En vista de la realidad actual a la que se enfrenta la anteriormente citada red social, y en colaboración con la Cátedra de Derechos Humanos y Poderes Públicos de la UPV/EHU, este proyecto pretende, por tanto, recabar información sobre el espectro de conductas que conforman la base de los delitos de odio. Para ello se desarrollará una aplicación web que contará con una herramienta de *text mining* que sea capaz de identificar de manera autónoma dichos tweets y de clasificarlos dentro del espectro de conductas, basándose en las directrices establecidas por los juristas que colaboran en este proyecto.

1.1. Motivación

La elección de este proyecto ha venido motivada por varios factores, el primero de ellos es que se va a desarrollar una herramienta de la que no se tiene constancia que existan antecedentes. Esto supondrá un reto en la búsqueda de herramientas que se adecuen tanto al presupuesto como a las necesidades del cliente. Teniendo en cuenta el elevado número de tweets que se obtienen como resultado de una búsqueda, resulta extremadamente laborioso para las personas el etiquetado de los mensajes; es por ello que se ha detectado la necesidad de desarrollar un sistema autónomo que sea capaz de realizar esta tarea teniendo como eje principal la minería de datos.

Otro factor que ha determinado la preferencia por el desarrollo de esta idea es la existencia de un cliente "real" que añade un nivel de exigencia superior, dado que se tratará de satisfacer sus expectativas lo más fielmente posible, teniendo en cuenta las limitaciones propias del proyecto. Asimismo, se tendrán en cuenta las posibles modificaciones y los cambios de última hora, debido tanto a imprevistos surgidos durante el desarrollo de la aplicación, como a las propias exigencias del cliente.

Por último, también se tomará en consideración el estado de la minería de texto en la actualidad, la forma en que ésta evoluciona y cómo esos avances pueden ayudar a este proyecto.

1.2. Definiciones, acrónimos y abreviaturas

- **Tweet:** Cada uno de los mensajes que se publican en Twitter.
- **Retweet:** Republicación de un tweet lanzado por otro usuario.
- **Algoritmo:** Conjunto ordenado de operaciones sistemáticas que permite hacer un cálculo y hallar la solución de un tipo de problemas.
- **Weka:** Es una plataforma de software para el aprendizaje automático y la minería de datosescrito en Java y desarrollado en la Universidad de Waikato.
- **Twitter4J:** Es una librería no oficial escrita en java para utilizar el API de Twitter.
- **Apache Tomcat:** Es un servidor web usado para proyectos java por su implementación de servlets o páginas JSP
- **Servlet:** Es una clase en el lenguaje de programación Java, utilizada para ampliar las capacidades de un servidor.
- **JSP:** Es una tecnología que ayuda a los desarrolladores de software a crear páginas web dinámicas basadas en HTML y XML, entre otros tipos de documentos. JSP es similar a PHP, pero usa el lenguaje de programación Java.
- **XAMPP:** es un paquete de software libre, que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script PHP y Perl.
- **Front-end:** Es la parte del software que interactúa con el usuario
- **Back-end:** Es la parte del software que procesa los datos recibidos por el front-end

2. Planteamiento inicial

2.1. Objetivos

El objetivo del proyecto es crear una aplicación web que permita realizar una búsqueda en Twitter y clasificar automáticamente los resultados que se obtengan, determinando si su contenido puede ser penalmente relevante o no.

Para ello, los usuarios registrados en la aplicación, que en primera instancia serán los juristas que colaboran en el desarrollo de esta herramienta, podrán partir de una búsqueda previamente realizada o crear un nuevo proyecto en el que elegirán un término a rastrear en Twitter. Consideraremos proyecto el entorno de trabajo en el que el usuario habrá introducido un único término a buscar en Twitter y habrá puesto un nombre identificativo a esa búsqueda para que así, cada vez que acceda al sistema, pueda recuperar todo el trabajo hecho hasta ese momento sobre ese término. El usuario podrá buscar tantos términos como desee, pero cada uno de ellos deberá estar en un proyecto diferente. Una vez acabada la búsqueda sobre un término, los juristas etiquetarán los tweets de forma manual, determinando si pueden ser relevantes o no, y en caso de serlo a que subconjunto de conductas penales pertenecen.

Cuando se hayan etiquetado un cierto número de tweets de un proyecto concreto, el sistema, con la ayuda de una herramienta de *text mining* generará varios modelos de aprendizaje para dicho proyecto, que el usuario podrá seleccionar para que clasifique todos los tweets de manera autónoma que hasta ahora han quedado sin clasificar del mismo. Para seleccionar el modelo que más le pueda convenir para cada proyecto el usuario dispondrá de unas estadísticas para ver el porcentaje de precisión y exhaustividad que se obtienen aplicando esos modelos a los tweets ya etiquetados. Más adelante veremos porqué se ha decidido utilizar estos dos estadísticos y de dónde se obtienen.

Tras ser clasificados por la aplicación, el usuario podrá revisar los resultados obtenidos y supervisar su tasa de acierto, brindándosele la oportunidad de hacer las correcciones oportunas. De este modo, la herramienta recibiría la retroalimentación apropiada y podría reducir su margen de error en futuros proyectos.

Con esta aplicación se pretende recabar información sobre las diferentes conductas que conforman la base de los delitos de odio en una de las redes sociales con mayor difusión a nivel mundial

La aplicación web constará de tres módulos:

- El primero de ellos se encargará de la recopilación de tweets basándose en los requisitos establecidos por el equipo de juristas.
- El segundo realizará un procesamiento previo, analizará y clasificará los tweets en base al criterio de los juristas.
- El tercer y último módulo permitirá visualizar los resultados obtenidos y efectuar correcciones, si estas fuesen necesarias.

2.2. Alcance

Se ha decidido utilizar la metodología Scrum, que es una metodología de gestión y desarrollo del software que se basa en un proceso iterativo e incremental. Dicha metodología consiste en dividir el desarrollo de la aplicación en diferentes subprocesos llamados sprint, los cuales tendrán una duración aproximada entre 2-3 semanas.

Con el scrum lo que se hace es obtener una lista de funcionalidades, llamadas Scrum backlog, que se ordenarán de forma prioritaria. Mientras haya tareas que realizar en el scrum backlog, por cada una de las tareas se creará un sprint backlog (se crearan las tareas a realizar en ese sprint), se programará cada tarea, se hará una revisión por parte del alumno para ver que ha salido mal e intentar mejorarlo, se realizará una entrega al cliente y se le consultará para ver si hay nuevos requisitos.

Es vital que entre cada uno de los sprint el valor del producto haya aumentado, ya que lo importante es el producto, no el avance del proyecto.

Un punto muy favorable al realizar scrum es la facilidad de subsanar los errores cometidos, ya que, al ser periodos cortos entre sprint y sprint apenas se pierde tiempo.

En nuestro caso concreto habrá una reunión principal con el cliente, la Cátedra de Derechos Humanos y Poderes Públicos de la UPV/EHU, en la que se fijarán los objetivos y se intentarán aclarar dudas. Una vez definido los objetivos, se hará una división de las tareas a realizar y durante el desarrollo de esas tareas se harán reuniones con los directores del proyecto para tratar posibles dudas o problemas. Cuando

se disponga de una aplicación en fase de pruebas, pero con alguna funcionalidad ya implementada, se volverá a hacer una reunión con el cliente para que vea su desarrollo y que indique posibles cambios o dudas que le surjan. Según avance el proyecto y se tengan más funcionalidades desarrolladas se seguirán haciendo más reuniones con el mismo fin. Una vez finalizada la aplicación, se concertará una última cita con el cliente para que valide y muestre su opinión al respecto.

A día de hoy el scrum es una herramienta innovadora y muy utilizada por las numerosas compañías de desarrollo de software informático siendo la que mejor alternativa ofrece a las especificaciones que el cliente, en este caso la Cátedra de Derechos Humanos y Poderes Públicos de la UPV/EHU, quiera introducir a lo largo del proceso de creación y permitiendo al desarrollador hacer frente a posibles errores de la aplicación y variables jurídicas.

Las tareas que forman parte del proyecto, se representan en la Estructura de Descomposición del Trabajo (EDT), representado en Figura 1 – EDT.

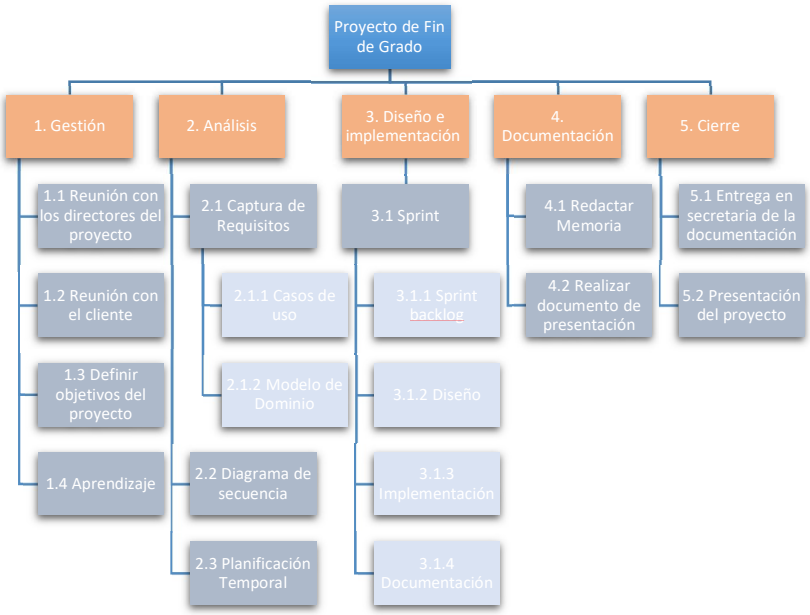


FIGURA 1 – EDT

2.3. Descripción de tareas

Se procederá a describir cada una de las tareas reflejadas en el EDT:

1. Gestión

1.1. Reunión con los directores del proyecto

Duración: 1 hora

Descripción: Realizar una reunión con los directores del proyecto exponiendo las ideas a realizar, funcionalidades que se ha de desarrollar en el trabajo, siendo las descripciones muy concisas y claras.

Salidas/Entregables: Una descripción informal del proyecto

1.2. Reunión con el cliente

Duración: 2 horas

Descripción: Realizar una reunión con los clientes del proyecto, consultándoles las necesidades que tiene, exponiéndole dudas surgidas en la reunión con los directores del proyecto y fijar una línea para el desarrollo del proyecto

Salidas/Entregables: Una descripción informal, pero más detallada del proyecto

Precedentes: Paquete de trabajo 1.1

1.3. Definir objetivos del proyecto

Duración:20 horas

Descripción: Buscar información sobre las herramientas necesarias para la realización del proyecto. Creación de los diagramas necesarios para incluir en el DOP y la redacción del mismo

Entrada: Información recabada en los apartados 1.1 y 1.2

Salidas/Entregables: El documento de objetivos del proyecto

Recursos necesarios: Microsoft Word, Microsoft PowerPoint, Microsoft Proyect

Precedentes: Paquetes de trabajo 1.1 y 1.2

1.4. Aprendizaje

Duración: 25 Horas

Descripción: Tras saber las necesidades del cliente, se realizará una valoración de las herramientas presentes en el mercado, seleccionando las que mejor se ajusten a las necesidades del proyecto, y una vez seleccionadas se realizará el estudio de su utilización

Recursos necesarios: PC, Internet

2. Análisis y captura de requisitos

2.1.1. Casos de uso

Duración: 10 horas

Descripción: A partir de la información recogida en la reunión con el cliente, se formalizarán los requisitos en forma de diagramas de casos de uso con sus correspondientes explicaciones, así como las acciones que harán cada uno de los actores que formarán parte de ellas.

Entrada: DOP

Salidas/Entregables: Diagrama de casos de uso

Recursos necesarios: Visual Paradigm

Precedentes: Paquetes de trabajo 1.3

2.1.2. Modelo de Dominio

Duración:12 horas

Descripción: Se realizará el diagrama con la representación de las entidades que tomarán parte en el sistema, indicando sus atributos y la relación existente entre ellas.

Entrada: Diagrama de casos de uso

Salidas/Entregables: Modelo de dominio

Recursos necesarios: Visual Paradigm

Precedentes: Paquetes de trabajo 2.1.1

2.2. Planificación Temporal

Duración:6 horas

Descripción: Se establecerá un plan para el desarrollo de las tareas y se realizará una estimación aproximada del tiempo que llevará la realización de cada una de ellas

Entrada: DOP, casos de uso

Salidas/Entregables: Una planificación de las tareas con los tiempos aproximados para su realización

Precedentes: Paquetes de trabajo 1.3 y 2.1

2.3. Sprint

2.3.1. Sprint backlog

Duración: 2 Horas

Descripción: Se seleccionará las tareas que forme parte del Sprint

Entrada: Lista de tareas de todo el proyecto

Salidas/Entregables: Lista de tareas para ese sprint

Precedentes: Scrum backlog

2.3.2. Diseño

Duración: 12 Horas

Descripción: Se realizará el diseño de la tarea seleccionada. Se diseñará tanto la interfaz web como la estructura, bases de datos y las pruebas a realizar.

Para el diseño de la interfaz del usuario, habrá que intentar que este sea lo más user-friendly posible ya que la interacción con el usuario es muy importante porque es él el que alimentara la base de datos en un primer momento y revisará los datos obtenidos por el programa.

Para el diseño de la base de datos habrá que tener en cuenta el gran número de datos almacenados.

En el diseño de la estructura de la aplicación se tendrá en cuenta cuales son los algoritmos más eficientes para realización del *text-mining* y el conjunto de datos disponibles para su realización.

También se realizará el diagrama que muestre el funcionamiento de los métodos que se hayan para el funcionamiento de la aplicación. Este diagrama mostrará la comunicación entre los diferentes módulos del programa para llevar a cabo las funcionalidades requeridas.

Por último, se definirán el conjunto de pruebas a realizar para probar el buen funcionamiento del software.

Entrada: Tarea a realizar, modelo de domino

Salidas/Entregables: Diseño del software perteneciente a ese sprint.

Recursos necesarios: ordenador, visual paradigm, MySQL Workbench y documentación realizada hasta el momento

Precedentes: Paquete de trabajo 3.1.1 y Paquetes de trabajo 2.1

2.3.3. Implementación

Duración: 30 horas

Descripción: Se implementará lo previamente diseñado para cada tarea seleccionada.

Entrada: Implementación realizada en el paquete de trabajo 3.1.2

Salidas/Entregables: Parte de software con las funcionalidades implementadas

Recursos necesarios: Servidor en la nube, apache, Tomcat, eclipse, programa de gestión XAMPP, Mysql, WEKA

Precedentes: Paquete de Trabajo 3.1.2

2.3.4. Documentación

Duración: 6 Horas

Descripción: Se documentará todo lo realizado para ese sprint

Entrada: Información obtenida en el paquete de trabajo 3.1.2 y 3.1.3

Salidas/Entregables: Documentación actualizada

Recursos necesarios: ordenador y Microsoft Word.

Precedentes: Paquete de Trabajo 3.1.2 y 3.1.3

3. Documentación

3.1. Redactar Memoria

Duración: 30 horas

Descripción: Realizar los documentos restantes y poner toda la documentación realizada hasta el momento en común

Entrada: Toda la documentación realizada hasta el momento y el programa finalizado

Salidas/Entregables: Memoria

Recursos necesarios: ordenador y Microsoft Word

Precedentes Paquetes de trabajo 1, 2, 3 y 4

3.2. Realizar documento de presentación

Duración: 6 horas

Descripción: Realizar el PowerPoint que se utilizará el día de la presentación

Entrada: Memoria y aplicación finalizada

Salidas/Entregables: Documento de presentación

Recursos necesarios: Memoria y ordenador con PowerPoint

Precedentes: Paquete de trabajo 5.1

4. *Cierre*

4.1. Entrega en secretaria de la documentación

Duración: 30 minutos

Descripción: Se entregará en secretaría la documentación requerida

Entrada: Toda la documentación realizada.

Recursos necesarios: Trabajo impreso

Precedentes: Paquete de trabajo 5.2

4.2. Presentación del proyecto

Duración: 30 minutos

Descripción: Presentación y defensa del proyecto ante el tribunal.

Entrada: Documento de presentación

Recursos necesarios: Ordenador con PowerPoint

Precedentes: Tener finalizado todos los paquetes de trabajos anteriores.

2.4. Planificación Temporal

Tras haber realizado un desglose de las tareas a realizar se calculará el tiempo estimado de la realización del proyecto.

Tareas	Tiempo dedicado
1. Gestión	**48 horas**
1.1. Reunión con los directores del proyecto	1 hora
1.2. Reunión con el cliente	2 horas
1.3. Definir objetivos del proyecto	20 horas
1.4. Aprendizaje	25 horas
2. Análisis	**42 horas**
2.1. Captura de Requisitos	22 horas
1. Casos de uso	10 horas
2. Modelo de Dominio	12 horas
2.2. Diagrama de secuencia	12 horas
2.3. Planificación Temporal	6 horas
3. ***Diseño e implementación***	**50 horas x 4 Sprint**
3.1. Sprint	50 horas
1. Sprint backlog	2 horas
2. Diseño	12 horas
3. Implementación	30 horas
4. Documentación	6 horas
4. Documentación	**36 horas**
4.1. Redactar Memoria	30 horas
4.2. Realizar documento de presentación	6 horas
5. Cierre	**1 hora**
5.1. Entrega en secretaria de la documentación	0,5 horas
5.2. Presentación del proyecto	0,5 horas
TOTAL	**324 horas**

TABLA 1- PLANIFICACIÓN TEMPORAL

Tras ver el número de tareas y de horas que conllevará la ejecución del proyecto se ha creído oportuno que el proyecto se desarrolle en 4 sprint:

- Sprint 1:

En el primer sprint se desarrollará el módulo que permita la búsqueda en Twitter. También se desarrollará la base de datos de toda la aplicación.

A su vez se realizará la parte de la página web que permite la visualización de los tweets

- Sprint 2:

En el segundo sprint se desarrollará el módulo de tratamiento de los datos, donde se eliminará la información que no se considere relevante en los tweets.

Así mismo se añadirá a la web el apartado para la etiquetación de los tweets.

- Sprint 3:

En el tercer sprint se desplegará el módulo que generará los clasificadores que posteriormente permitirán predecir la clase de los tweets.

También se creará el apartado de la pagina web donde el usuario visualizará los resultados predichos por el clasificador para su validación.

- Sprint 4:

En el último sprint se desarrollará el módulo que permitirá la visualización en una manera clara de los datos obtenidos. También se realizarán pruebas funcionales a la aplicación y se acabará de realizar la documentación.

En la Figura 2 – diagrama de gant se muestra la planificación hecha para este proyecto:

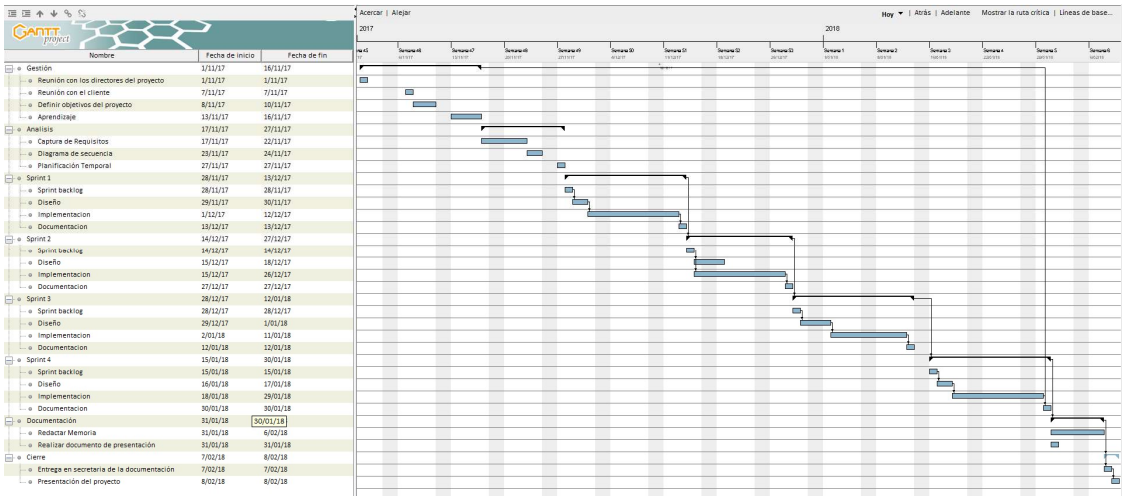


FIGURA 2 – DIAGRAMA DE GANT

2.5 Arquitectura

Para este proyecto se utilizará una arquitectura cliente – servidor. En este modelo, el cliente realiza una petición a un programa y el cliente le dará la respuesta.

La arquitectura cliente-servidor estará desarrollada en tres capas a nivel lógico, separando la base de datos del servidor web. De esta manera los equipos son más escalables a la hora de futuras ampliaciones. Aun así, con el fin de reducir el presupuesto del proyecto, a nivel físico es una arquitectura de dos capas, ya que en la misma maquina estará albergado tanto el servidor web como la base de datos. A nivel de cliente, el usuario sólo tendrá que disponer de un navegador web que le permita acceder a la página en la que esté alojada la aplicación.

Este modelo es beneficioso para este proyecto ya que, al disponer de una base de datos con gran cantidad de información y tener que realizar una gran cantidad de cálculos, en caso de necesitarlo, se podrá segregar la información en diferentes servidores, siendo esto transparente para el cliente y así realizar un balance de la carga.

Para desarrollar esta arquitectura haremos uso de la nube, como se puede observar en la Figura 3 – Arquitectura Cliente-Servidor.

En la nube todo lo que puede ofrecer un sistema informático se ofrece como servicio. Las ventajas que tiene es la escalabilidad y elasticidad que se dispone, ya que se pueden aprovisionar más de una máquina de

forma rápida. Otra ventaja es el coste, puesto que sólo se pagará por número de máquinas y el uso de los recursos que hagas de ellas. A su vez eliminamos el coste del mantenimiento de las máquinas ya que de eso se encargará el proveedor del servicio. También hay independencia entre el dispositivo y la ubicación, cualquier usuario puede acceder al sistema a través de un ordenador con acceso a internet.

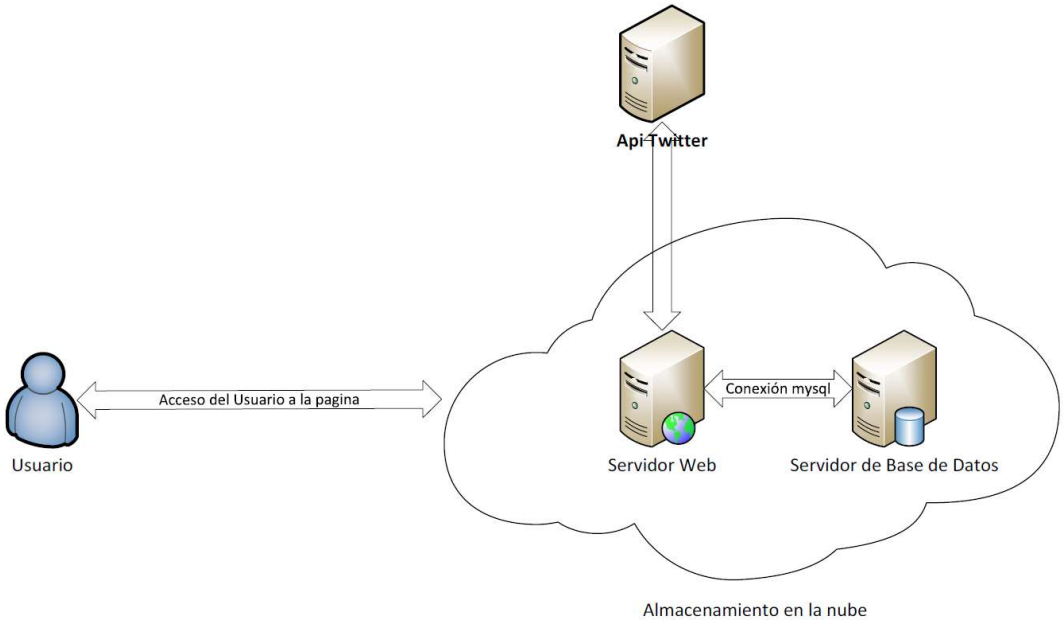


FIGURA 3 – ARQUITECTURA CLIENTE-SERVIDOR

2.6 Herramientas

En esta sección se listarán aquellas herramientas que serán utilizadas para el desarrollo de la aplicación.

1. **Visual Paradigm:** Es un programa para el desarrollo del lenguaje de modelado "UML". Se utilizará para el diseño del programa.
2. **Microsoft Office:** Suite ofimática propiedad Microsoft. Dispone de editor de texto, hojas de cálculo y creación de presentaciones entre otros. Se usará para desarrollar la documentación y la presentación final.

3. **XAMPP:** Es un servidor web de plataforma, gratuito, que consta de un sistema de gestión de bases de datos MySQL, un servidor Apache y los intérpretes de comandos para lenguajes PHP y Perl. Se utilizará para la gestión y el mantenimiento de la base de datos
4. **Tomcat:** Es un contenedor web con soporte de servlets y JSPs. Se utiliza como servidor web. Requiere Java para su funcionamiento. Se usará para la gestión de la página web.
5. **GanttProject:** Software para la administración de proyectos. Se utilizará para la realización del diagrama Gantt.
6. **Eclipse:** Plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar aplicaciones de cliente enriquecido. Se usará como herramienta para desarrollar el código de la aplicación.
7. **Weka:** Software para el aprendizaje automático y minería de datos escrito en java. Se usará como herramienta para la minería de datos.
8. **Google Cloud:** Es una plataforma que reúne todas las aplicaciones web que Google ofrece. Permite la creación de instancias de máquinas virtuales en la nube. Será el servicio dónde se hospedará la página web.

2.7 Gestión de Riesgos

En este apartado se tratará de identificar posibles incidencias que puedan ocurrir durante el transcurso del proyecto, analizando para cada caso la probabilidad de que el suceso ocurra, crear un plan de prevención para que no suceda y en caso de que ocurrir, crear un plan de contingencia.

En las siguientes tablas se mostrarán los valores orientativos que pueden tomar la posibilidad de que se sufra una incidencia y qué impacto tendrá la misma sobre el proyecto:

Probabilidad	Porcentaje
Muy poco probable	0 - 20%
Poco probable	20 - 40%
Probable	40 - 60%
Bastante probable	60 - 80%
Muy probable	80 - 100%

Impacto	Horas
Muy leve	< 8 horas
Leve	8 -16 horas
Medio	16 – 24 horas
Grande	24 - 32 horas
Muy grande	> 32 horas

1. **Baja médica**

   Descripción: El alumno se pone enfermo causando la baja durante un tiempo en el proyecto

   Prevención: Que el alumno trabaje en un entorno adecuado con una temperatura y humedad adecuada a las circunstancias y una iluminación correcta

   Plan de contingencia: Intentar recuperar el tiempo perdido trabajando más horas los días venideros.

   Probabilidad: Poco probable

   Impacto: Muy grande

2. **Cambio en la situación laboral del trabajador**

   Descripción: El alumno cambia de puesto de trabajo.

   Prevención: Escoger un empleo que se adecue a la situación del alumno y le permita compaginar ambas actividades

   Plan de contingencia: Revisar la planificación temporal del proyecto y el alcance, e intentar adecuarlo a la situación actual del alumno.

   Probabilidad: Poco probable

   Impacto: Grande

3. **Planificación incorrecta**

   Descripción: El alumno ha determinado erróneamente el tiempo a realizar de cada tarea.

Prevención: Ampliar los plazos de tal manera que haya bastante margen de tiempo entre tareas.

Plan de contingencia: Modificar la planificación original aumentando las horas de cada tarea

Probabilidad: Muy poco probable

Impacto: Muy grande

4. **Fallo del suministro eléctrico**

Descripción: Fallo en el suministro eléctrico que imposibilite la realización del proyecto

Prevención: Instalación de algún SAI (sistema de alimentación ininterrumpida) para afrontar la caída del suministro

Plan de contingencia: Ponerse en contacto con el suministrador eléctrico para informar sobre la incidencia y/o cambiar de localización para recuperar el suministro

Probabilidad: Muy poco probable

Impacto: Muy leve

5. **Fallo en el ordenador**

Descripción: Cualquier problema que deje inoperativo el ordenador con el que se está realizando el proyecto

Prevención: Disponer de más de un equipo informático para realizar el proyecto (ejem. Ordenador portátil y de sobremesa), manteniéndolos al día tanto a nivel de software como a nivel de hardware. Disponer a su vez más de una copia de seguridad del proyecto tanto en internet, con aplicaciones que permiten el almacenamiento en la nube de forma gratuita como Google drive a Dropbox, así como en almacenarlo en un pendrive.

Plan de contingencia: En caso de fallo en software intentar realizar una restauración del equipo. En caso de fallo hardware reparar la pieza que se ha averiado o comprar otro dispositivo.

Probabilidad: Muy poco probable

Impacto: Grande

6. **Caída del servidor**

   Descripción: El servidor que alberga la página web esta fuera de servicio.

   Prevención: Disponer de otro servidor redundante.

   Plan de contingencia: Migrar el servicio a otro servidor.

   Probabilidad: Muy poco probable

   Impacto: Muy grande

7. **Añadir nuevos requisitos**

   Descripción: Debido a las necesidades del cliente, este solicita la implementación de funcionalidades adicionales.

   Prevención: Realizar un diseño lo más modular posible de manera que añadir o eliminar cualquier funcionalidad no suponga un esfuerzo extra.

   Plan de contingencia: Evaluar el nuevo requisito y decidir si es viable su incorporación sin afectar al desarrollo correcto del proyecto.

   Probabilidad: Probable

   Impacto: Medio

8. **Falta de incumplimiento de alguno de los requisitos**

   Descripción: No se cumplen los requisitos iniciales que se habían previsto conseguir.

   Prevención: Ajustar los requisitos a las capacidades reales del alumno.

   Plan de contingencia: Renegociación de los requisitos.

   Probabilidad: Bastante probable

   Impacto: Muy grande

9. **Número de usuarios mayor de lo esperado**

   Descripción: El número de usuarios que utiliza la aplicación o el número de peticiones a la base de datos es mayor de lo soportado

   Prevención: Realizar un diseño adecuado de la base de datos y de las conexiones al servidor

   Plan de contingencia: Ampliación del espacio disponible, en caso de lo posible y el número de peticiones concurrentes en el servidor

   Probabilidad: Poco probable

   Impacto: Medio

10. **Falta de entendimiento con los juristas**

    Descripción: Falta de entendimiento con los juristas debido al lenguaje especifico utilizado

    Prevención: Realizar reuniones más periódicas con los juristas intentado aclarar las dudas

    Plan de contingencia: Envió de correo a los clientes para aclarar las dudas

    Probabilidad: Probable

    Impacto: Medio

11. **Falta de acuerdo entre los juristas**

    Descripción: Falta de acuerdo entre los juristas a la hora de etiquetar los tweets.

    Prevención: Aclarar el funcionamiento de la aplicación para que en caso de duda consensuen entre ellos.

    Plan de contingencia: Realizar la aplicación de manera que sólo permita una única etiqueta posible

    Probabilidad: Probable

    Impacto: Medio

12. **Fallo en alguna de las APIs de terceros**

Descripción: Fallo o problema con alguna de las API utilizadas para la realización del proyecto.

Prevención: Revisar la documentación existente y la estabilidad de la API.

Plan de contingencia: Ponerse en contacto con el proveedor para intentar resolver el fallo o intentar realizar la tarea de manera diferente.

Probabilidad: Poco probable

Impacto: Muy grande

13. **Incompatibilidad entre sistemas**

Descripción: Fallo o problema en la aplicación al haber alguna incompatibilidad entre las distintas aplicaciones que toman parte en su desarrollo.

Prevención: Revisar la documentación existente.

Plan de contingencia: Revisar posibles alternativas para suplir alguna de las aplicaciones que causan la incompatibilidad.

Probabilidad: Probable

Impacto: Grande

14. **Fallo debido a falta de almacenamiento**

Descripción: El disco duro del equipo se ve sobrepasado debido al gran volumen de datos almacenados.

Prevención: Realizar un dimensionamiento adecuado de la maquina al crearla.

Plan de contingencia: Revisar posibles alternativas para suplir alguna de las aplicaciones que causan la incompatibilidad.

Probabilidad: Probable

Impacto: Grande

2.8 Planificación económica

En este punto abordaremos la evaluación económica del proyecto.

INGRESOS

Al ser un proyecto de investigación, con una vertiente social, que se ha desarrollado junto la colaboración de la Cátedra de Derechos Humanos y Poderes Públicos de la UPV/EHU la monetización del proyecto es difícil, y sus ingresos serán de 0€. Aun así, puede tener una gran repercusión social.

Con su utilización se perseguirán actos delictivos, pudiendo llegar a ser punibles, de este modo se conseguiría hacer un uso más responsable de las redes sociales llevando ante la justicia a aquellas personas que hacen un uso indebido de éstas.

GASTOS

Para calcular el coste del trabajador, tendremos en cuenta el actual convenio para ingenierías en Bizkaia, que al no estar vigente se tendrá que aplicar el convenio a nivel estatal.

	Mes × 14	Anual
Nivel 1. Licenciados y titulados 2.º y 3.er ciclo universitario y Analista	1.687,02	23.618,28
Nivel 2. Diplomados y titulados 1.er ciclo universitario. Jefe Superior	1.253,16	17.544,24
Nivel 3. Técnico de cálculo o diseño, Jefe de 1.ª y Programador de ordenador	1.208,40	16.917,60
Nivel 4. Delineante-Proyectista, Jefe de 2.ª y Programador de maq. Auxiliares	1.107,87	15.510,18
Nivel 5. Delineante, Técnico de 1.ª, Oficial 1.ª Admtvo. y Operador de ordenador	968,23	13.555,22
Nivel 6. Dibujante, Técnico de 2.ª, Oficial 2.ª Admtvo., Perforista, Grabador y Conserje	834,17	11.678,38
Nivel 7. Telefonista-Recepcionista, Oficial 1.ª oficios varios, y Vigilante	806,20	11.286,80
Nivel 8. Auxiliar Técnico, Auxiliar Admtvo., Telefonista, Ordenanza, Personal de limpieza y Oficial 2.ª oficios varios	750,38	10.505,32
Nivel 9. Ayudante oficios varios	698,24	9.775,36

FIGURA 4 – TABLA SALARIAL DE EMPRESAS DE INGENIERÍA Y OFICINAS DE ESTUDIOS TÉCNICOS

Como se puede observar en la Figura 4 – Tabla salarial de empresas de ingeniería y oficinas de estudios técnicos, el salario mensual de un graduado informático es de 1.263,16€ mensuales, teniendo en cuenta que el proyecto se ha estimado en 324 horas, que la ley fija un máximo de 40 horas semanales para trabajar, eso da un total de 8,1 semanas de trabajo:

$$\frac{324\ hora}{40\ horas/semana} = 8{,}1 \text{ semanas}$$

Sabiendo que un mes tiene 4 semanas, este proyecto estará estimado en 2,025 meses, que multiplicándolo por el salario mensual fijado en el convenio nos da 2.537,649€:

$$\frac{8{,}1\ Semanas}{4\ Semana/mes} * 1.253{,}16\frac{€}{mes} = 2.537{,}649€$$

Por lo que el gasto en mano de obra se estima en 2.537,646€.

También se deberán tener en cuenta el software y servicios utilizados para la realización del proyecto, para ello se van a enumerar e indicar el costo:

- Google Cloud Service: Los gastos que genera un servicio en la nube es un tanto difícil de calcular, puesto que no es un gasto mensual fijo, depende de su utilización, número de conexiones al servidor, uso del procesador y del disco duro, etc.
  Es por ello que para calcular esta cifra se ha realizado una estimación utilizando la calculadora proporcionada por Google[2].
  En ella aparece un gasto mensual de 20€, que multiplicándolo por 8 meses que se ha estimado la duración del proyecto, no da un coste de 160€. Indica que para este caso no se ha tenido en cuenta las horas estimadas, puesto que al no saber en qué momento del día se va a trabajar en el proyecto, necesitamos tener el servidor disponible las 24 horas.
- Microsoft Office: El precio de una licencia de Office 365 anual es de 69€
- Microsoft Visio: La versión estándar de Visio 399€
- Microsoft Windows: Al venir incluido en la compra del portátil no se tomará en cuenta.
- El resto de elementos que no se han tenido en cuenta (Visual Paradigm, Eclipse. etc.) se debe a que su costo a sido de 0 euros, ya sea porque se ha utilizado licencias académicas o porque se haya utilizado su versión gratuita.

Fijar el coste que supone el local del trabajo junto con los gastos comunes es un tanto complicado, ya que, en caso de querer, se podría ir todos los días a una biblioteca pública y realizarlo desde ahí, siendo el coste del local, luz y agua de 0€.

Para los costes del material informático se tendrá en cuenta el uso que se le hará para este proyecto. Teniendo en cuenta que la vida útil de los equipos ronda los 8 años, calcularemos la amortización de los mismos:

[2] https://cloud.google.com/products/calculator/

$\frac{8{,}1\ meses}{12\ meses/año} = 0{,}675\ años$ de uso que se le dará al equipo para este proyecto

$\frac{0{,}675\ años}{8\ año\ de\ vida\ util} = 0{,}084375$ años de vida útil de los equipos será utilizados para este proyecto

- Ordenador portátil: Gigabyte Aero 14K. Precio:902,03€
  Amortización: 902,03*0,084375= 76,10878125€

- Ordenador de sobremesa: HP Pavilion 570-p040ns. Precio 679,15€
  Amortización: 679,15*0,084375= 57,30328125€

- Disco duro de almacenamiento: Toshiba Canvio Basics. Precio: 54€
  Amortización: 54*0,084375= 4,55625€

  Lo que sumando todo hace un total de 137,9683125€ en material informático.

  En la siguiente tabla se observan los gastos totales estimados para este proyecto:

Concepto	Gasto en €
Salarios	2.537,65
Google Cloud Service	160
Microsoft Office	69
Microsoft Visio	399
Amortización material informático	137,9683125
Total	3.303,61

TABLA 2 – GASTOS TOTALES DEL PROYECTO

3. Antecedentes

En este apartado se hará una breve introducción sobre la minería de datos, profundizando en la vertiente de minería de texto. Veremos qué es cada una de ellas, en qué se diferencian, qué tipo de clasificaciones existen, etc. También se realizará el análisis actual de las tecnologías utilizadas como el servicio en la nube o los lenguajes de programación entre otros, y se razonará cada una de las elecciones seleccionadas para la realización de este proyecto.

3.1 Minería de datos

Hoy en día se genera una gran cantidad de datos que quedan registrados en grandes bases de datos. Esto se debe, entre otras razones, por el elevado uso de dispositivos digitales. Con técnicas de análisis de datos adecuadas podemos obtener múltiples beneficios, no sólo económicos, sino también sociales.

El hecho de que el número de datos disponibles sea muy elevado, junto con su complejidad hacen difícil el procesamiento o análisis mediante tecnologías y herramientas convencionales. Es en este punto dónde se hace uso de la minería de datos. La minería de datos se define como el conjunto de técnicas y tecnologías que permiten explorar grandes bases de datos con el objetivo de encontrar patrones que nos puedan aportar información valiosa en la toma de futuras decisiones (Rodríguez Suárez & Díaz Amador, 2009). Para que esto suceda la información pasa por varios procesos, que van desde la limpieza de los datos almacenados, pasando por la minería de datos, hasta su conversión en conocimiento.

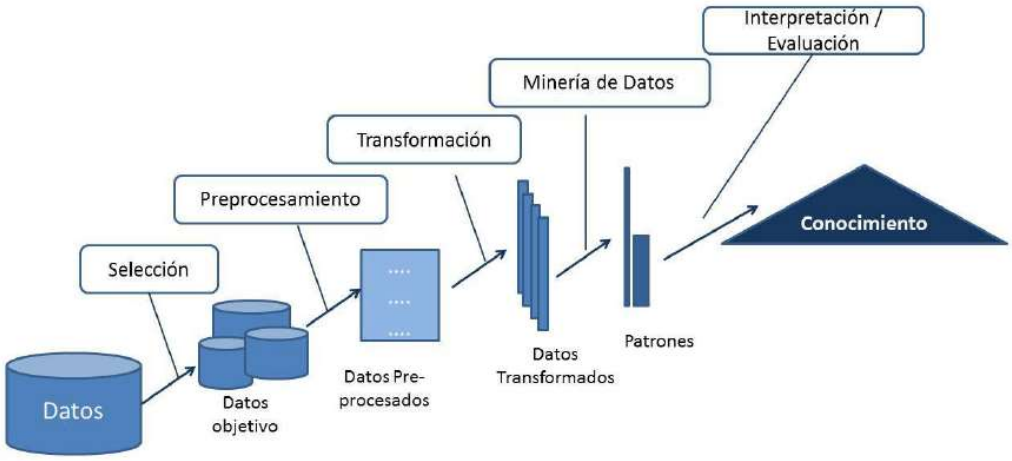


FIGURA 5 - FASES DEL PROCESO DE OBTENCIÓN DE CONOCIMIENTO A PARTIR DE DATOS

En la Figura 5 - fases del proceso de obtención de conocimiento a partir de datos, se observa los pasos que deben realizarse para que los datos se conviertan en conocimiento. A continuación, se explica brevemente cada uno de ellos:

1. **Limpieza de Datos:** En un primer paso se eliminan los datos inconsistentes obtenidos, para ello se utilizan métodos estadísticos como los histogramas para la detección de datos anómalos, la selección de datos, verticalmente eliminando atributos o horizontalmente eliminando tuplas, o la redefinición de atributos, agrupando o separándolos. En caso de encontrar datos inconsistentes se pueden usar dos estrategias, la primera es eliminar el dato en sí, y la segunda reemplazar el dato, o cambiar el atributo que le hace inconsistente.

2. **Selección de Datos:** Dado que no todos los datos almacenados son útiles a la hora de extraer información, es en este apartado donde se seleccionan los datos que se consideren relevantes para el análisis.

3. **Transformación de Datos:** Tras su selección los datos se tienen que transformar a una forma apropiada para su posterior tratamiento en la minería. En este apartado se utilizan técnicas como la reducción atributos para eliminar aquellos atributos que no aporten información. También se utilizan técnicas como la discretización, de modo que el número de valores que se pueda obtener se acote, o su inverso, que es la numeración. Estos procesos

implican la pérdida de las relaciones de integridad y la normalización en los datos.

4. **Minería de Datos:** En este paso se utilizan técnicas predictivas, como los árboles de decisión, redes neuronales, etc. O técnicas descriptivas como el *clustering*, etc. para extraer patrones de datos. Habrá que elegir los algoritmos de minería adecuados en función de los datos y del tipo de información que se desea descubrir.

5. **Evaluación de los patrones extraídos en la fase anterior:** En este apartado se utilizan las figuras de mérito como la precisión, sensibilidad o especificad para valorar el rendimiento de los algoritmos utilizado en el proceso anterior.

6. **Representación del conocimiento:** En este punto se representa visualmente, mediante gráficos, textos, sonidos, animaciones, etc. el conocimiento adquirido para que sea compresible por los expertos en la materia.

En la minería de datos se habla sobre datos, instancias, atributos, etc, términos que se requieren aclarar.

- Una instancia es cada uno de los datos de los que se disponen para hacer un análisis.
- Los atributos son los campos que describen o representan alguna característica de cada una de las instancias del conjunto de datos.
- Un modelo se puede asimilar a un filtro en el que entran datos nuevos y cuya salida es la clasificación de ese dato según los patrones que se han detectado en el entrenamiento. (Gonzalez, 2014)

Dentro de la minería de datos existen dos tipos de clasificación dependiendo de los algoritmos utilizados, la clasificación supervisada y la no supervisada. Para dar un ejemplo sobre estos dos términos, en una hoja de cálculo, cada fila sería una instancia y cada columna un atributo. La clasificación supervisada se basa en entrenar un modelo de aprendizaje por medio de diferentes datos para predecir una variable partiendo de esos mismos datos. (lalopg, 2015)

En la clasificación no supervisada los datos son clasificados en grupos que no son conocidos con anterioridad. Los valores de las variables pueden estar conectados entre si de acuerdo a vínculos desconocidos de

antemano. Este tipo de clasificación esta orientado a describir un conjunto de datos

En este proyecto se utilizará la clasificación supervisada, ya que se intenta predecir si un tweet es penalmente relevante en base a los ya etiquetados, que se utilizarán como entrenamiento para el algoritmo.

Cada clasificador tiene un funcionamiento determinado dependiendo del conjunto de datos que se le aporte. Por lo tanto, se utilizan cinco algoritmos diferentes, que tras "entrenarlos" se evalúa su calidad para determinar cuál de ellos es el mejor y utilizarlo a la hora de clasificar el conjunto no etiquetado.

Los algoritmos utilizados son: C4.5, Random forest, K-NN, SVM y Naïve bayes.

3.1.1 Algoritmos de clasificación supervisada

A continuación, se explica el funcionamiento de cada uno de los algoritmos utilizados.

1. **SVM**

   Una Máquina de Soporte Vectorial o *Support Vector Machines* (SVM) aprende la superficie decisión de dos clases distintas de los puntos de entrada. Como un clasificador de una sola clase, la descripción dada por los datos de los vectores de soporte es capaz de formar una frontera de decisión alrededor del dominio de los datos de aprendizaje con muy poco o ningún conocimiento de los datos fuera de esta frontera. Los datos son mapeados por medio de un núcleo, también llamado *kernel,* Gaussiano u otro tipo de *kernel* a un espacio de características en un espacio dimensional más alto, donde se busca la máxima separación entre clases. Esta función de frontera, cuando es traída de regreso al espacio de entrada, puede separar los datos en todas las clases distintas, cada una formando un agrupamiento.

   Una SVM primero mapea los puntos de entrada a un espacio de características de una dimensión mayor (i.e.: si los puntos de entrada están en $\Re 2$ entonces son mapeados por la SVM a $\Re 3$) y encuentra un hiperplano que los separe y maximice el margen m entre las clases en este espacio como se aprecia en la Figura 6 – cálculo del margen entre clases mediante svm

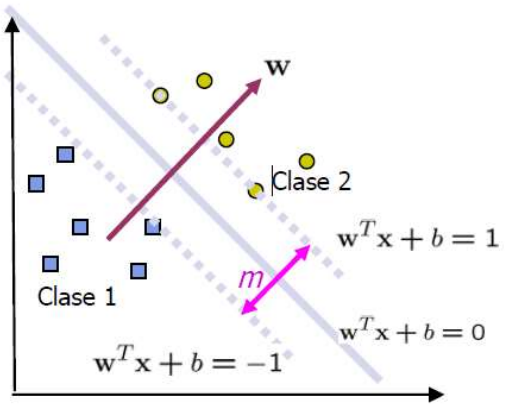


FIGURA 6 – CÁLCULO DEL MARGEN ENTRE CLASES MEDIANTE SVM

Maximizar el margen *m* es un problema de programación cuadrática (*QP*) y puede ser resuelto por su problema dual introduciendo multiplicadores de Lagrange. Sin ningún conocimiento del mapeo, la SVM encuentra el hiperplano óptimo utilizando el producto punto con funciones en el espacio de características que son llamadas *kernels*. La solución del hiperplano óptimo puede ser escrita como la combinación de unos pocos puntos de entrada que son llamados vectores de soporte.

Es un algoritmo que funciona muy bien con 2 valores para la clase. (Betancourt, 2005)

2. Naïve bayes

El clasificador naive Bayes (NB) se considera como parte de los clasificadores probabilísticos, los cuales se basan en la suposición que las cantidades de interés, se rigen por distribuciones de probabilidad, y que la decisión óptima puede tomarse por medio de razonar acerca de esas probabilidades junto con los datos observados. En tareas como la clasificación de textos este algoritmo se encuentra entre los más utilizados. En este proyecto se emplea el naive Bayes tradicional, el cual se describe a continuación.

En este esquema el clasificador es construido usando un conjunto de entrenamiento para estimar la probabilidad de cada clase. Entonces, cuando una nueva instancia i_j es presentada, el clasificador le asigna la categoría c∈C más probable por aplicar la regla:

$$c = \arg\max_{c_i \in C} P\left(c_i \middle| i_j\right)$$

Utilizando el teorema de Bayes para estimar la probabilidad se obtiene la siguiente ecuación:

$$c = \arg\max_{c_i \in C} \frac{P\left(i_j \middle| c_i\right) P\left(c_i\right)}{P\left(i_j\right)}$$

El denominador en la ecuación anterior no difiere entre categorías y puede omitirse

$$c = \arg\max_{c_i \in C} P\left(i_j \middle| c_i\right) P\left(c_i\right)$$

Teniendo en cuenta que el esquema es llamado "naive" debido al supuesto de independencia entre atributos, i.e. se asume que las características son condicionalmente independientes dadas las clases. Esto simplifica los cálculos produciendo

$$c = \arg\max_{c_i \in C} P\left(c_i\right) \prod_{k=1}^{n} P\left(a_{kj} \middle| c_i\right)$$

donde $P(c_i)$ es la fracción de ejemplos en el conjunto de entrenamiento que pertenecen a la clase c_i , y $P(a_{kj}|c_i)$ se calcula de acuerdo al teorema de Bayes. En resumen, la tarea de aprendizaje en el clasificador naive Bayes consiste en construir una hipótesis por medio de estimar las diferentes probabilidades $P(c_i)$ y $P(a_{kj}|c_i)$ en términos de sus frecuencias sobre el conjunto de entrenamiento . (Valero, 2005)

3. Random Forest

El clasificador random forest se basa en el desarrollo de muchos árboles de clasificación. Para clasificar un nuevo objeto desde un vector de entrada, ponemos dicho vector bajo cada uno de los árboles del bosque. Cada árbol genera una clasificación, en términos coloquiales diríamos que cada árbol vota por una clase. El bosque escoge la clasificación teniendo en cuenta el árbol más votado sobre todos los del bosque. Cada árbol se desarrolla como sigue:

—Si el número de casos en el conjunto de entrenamiento es *N*, prueba *N* casos aleatoriamente, pero con sustitución, de los datos originales. Este será el conjunto de entrenamiento para el desarrollo del árbol.

—Si hay M variables de entrada, un número $m<<M$ es especificado para cada nodo, m variables son seleccionadas aleatoriamente del conjunto M y la mejor partición de este m es usada para dividir el nodo. El valor de m se mantiene constante durante el crecimiento del bosque.
—Cada árbol crece de la forma más extensa posible, sin ningún tipo de poda.
Este algoritmo es uno de los más certeros disponibles, cuanto mayor sea el número datos, mejor será el clasificador. Además, no excluye ninguna variable a la hora de clasificar. (Valero, 2005)

4. K-NN

k–Vecinos más cercanos (*k-NN*, por sus siglas en inglés) es uno de los métodos de aprendizaje basados en instancias más básicos, pero con resultados aceptables en tareas que involucran el análisis de texto. En resumen, este algoritmo no tiene una fase de entrenamiento fuera de línea, por lo tanto, el principal cálculo se da en línea cuando se localizan los vecinos más cercanos. La idea en el algoritmo es almacenar el conjunto de entrenamiento, de tal modo que, para clasificar una nueva instancia, se busca en los ejemplos almacenados casos similares y se asigna la clase más probable en éstos. (Valero, 2005)
El resumen del algoritmo es el siguiente:
Entrenamiento:
-Para cada ejemplo en el conjunto de entrenamiento, agregar el ejemplo a la lista *ejemplos_entrenamiento*
Clasificación:
-Dada una instancia de prueba i_q a ser clasificada
-Sean $i_1, \ldots, i_k$ los k ejemplos de la *lista_entrenamiento* que más cercanos a i_q
-Regresar

$$c = \arg\max_{c_i \in C} \sum_{j=1}^{k} \delta\left(c_i, c_{i_j}\right)$$

Donde $\delta(a,b)$=1 si a=b y $\delta(a,b)$=0 en otro caso.

5. C4.5

El algoritmo C4.5 fue diseñado como una extensión del algoritmo ID3, éste último forma parte de los clasificadores conocidos como

árboles de decisión, los cuales son árboles donde sus nodos internos son etiquetados como atributos, las ramas salientes de cada nodo representan pruebas para los valores del atributo, y las hojas del árbol identifican a las categorías. Estos algoritmos proporcionan un método práctico para aproximar conceptos y funciones con valores discretos. A continuación, se presenta la descripción del algoritmo ID3 con el objetivo de facilitar la posterior descripción de C4.5.

Para construir el árbol, ID3 usa una aproximación descendente que da preferencia a los árboles pequeños sobre los grandes. El nodo raíz es seleccionado por encontrar el atributo más valioso en el conjunto de entrenamiento, i.e. el que mejor clasifica las instancias; la búsqueda se realiza por medio de una prueba estadística que mide cuanto de bueno es un atributo a la hora de separarlo del conjunto de entrenamiento teniendo en cuenta la clase. Una vez que la raíz es seleccionada, se agrega una rama desde la raíz para cada posible valor del atributo correspondiente, y el conjunto de entrenamiento es ordenado en los nodos apropiados, i.e. cada nodo contiene los ejemplos que cumplen la restricción de la rama anterior. Para seleccionar el atributo más valioso en cada punto del árbol, se repite el proceso completo usando el conjunto de entrenamiento asociado con el nodo. De manera que cuando una nueva instancia necesita ser clasificada, los atributos especificados por los nodos son evaluados iniciando por el nodo raíz, a continuación, de manera descendente se recorren las ramas del árbol que corresponden a los valores de los atributos en la instancia dada, el proceso se repite hasta que una hoja es alcanzada, y es en este punto donde la etiqueta asociada a la hoja es asignada a la nueva instancia como su categoría.

Finalmente, una vez introducido ID3 los pasos a seguir en C4.5 son:

1. Separar los datos en conjunto de entrenamiento y conjunto de validación.
2. Construir el árbol de decisión para el conjunto de entrenamiento (aplicar ID3).
3. Convertir el árbol en un conjunto de reglas equivalente, donde el número de reglas es igual al número de posibles rutas desde la raíz a los nodos hoja.
4. Podar cada regla eliminando precondiciones que resulten en mejorar la exactitud en el conjunto de validación.
5. Ordenar las reglas descendentemente de acuerdo a su exactitud, y usarlas en ese orden para clasificar futuros ejemplos. (Valero, 2005)

3.1.2 Técnicas de validación

Con el fin de evaluar el funcionamiento y la efectividad de un clasificador, existen unas series de medidas para validar su eficiencia. Para ello se utiliza una matriz de confusión (ver tabla 3), también conocida como matriz de error. En ella cada fila de la matriz representa las instancias de una clase pronosticada mientras que cada fila representa las instancias de una clase real, o viceversa:

		VALOR REAL	
		VERDADERO	FALSO
VALOR PREDICHO	VERDADERO	Verdadero positivo (TP)	Falso Positivo (FP)
	FALSO	Falso Negativo (FN)	Verdadero negativo (TN)

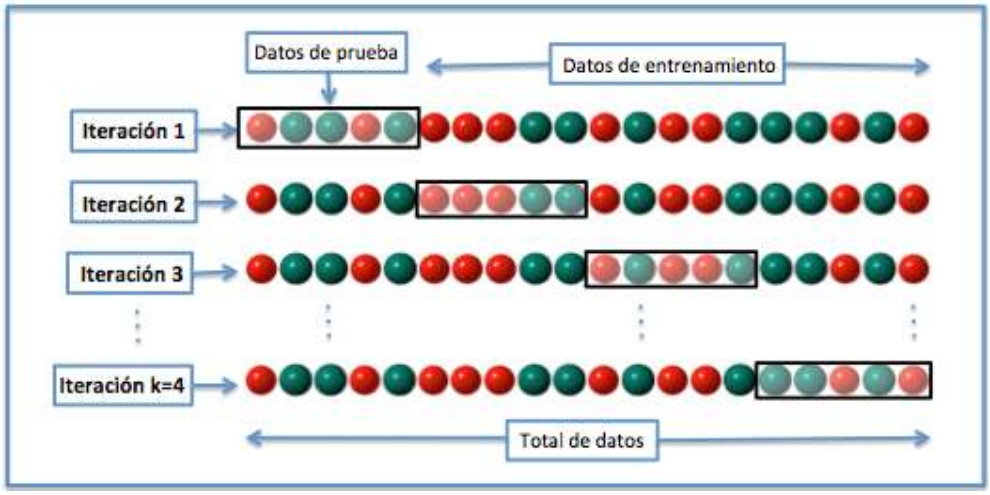
TABLA 3 - MATRIZ DE CONFUSIÓN

En la celda verdadero positivo (TP) se encuentra las instancias clasificadas como verdaderas que realmente son verdaderas. La celda falso positivo (FP) engloba las instancias clasificadas como verdaderas, pero que en realidad su clase es negativa. En la celda de falso negativo están las instancias clasificadas como negativas, pero que en realidad su clase es verdadera y por último en la celda verdadero negativo aparecen las instancias clasificadas como negativas que realmente son negativas.
Existen un gran número de técnicas de validación, sin embargo, a continuación, enumeraremos aquellas que son más relevantes para este trabajo:

- Método no honesto: En este método se utiliza la muestra completa de datos para inferir un modelo, y después se evalúa ese modelo con la misma muestra.
  Este método no es muy recomendable porque premia el sobreajuste, aportando mejores resultados de los que realmente luego se obtienen con otro conjunto distinto.

- *hold-out:* En él se tiene un conjunto de datos elevando, del que un porcentaje se utiliza para aprender el modelo de clasificación y el resto se utiliza a modo de test. En este modelo perdemos poder de conocimiento, ya que no utilizamos todos los datos disponibles para aprender él modelo. También obtenemos una variación de los resultados dependiendo los datos utilizados.

- El método más extendido es el denominado *k-fold cross-validation*. En él los datos de muestra se dividen en *K* subconjuntos. Uno de los subconjuntos se utiliza como datos de prueba y el resto (*K*-1) como datos de entrenamiento. El proceso de validación es repetido durante k iteraciones, con cada uno de los posibles subconjuntos de datos de prueba. Finalmente se realiza la media aritmética de los resultados de cada iteración para obtener un único resultado (ver Figura 7 - k-fold cross-validation). Este método es muy preciso puesto que evaluamos a partir de *K* combinaciones de datos de entrenamiento y de prueba, pero aun así tiene una desventaja, y es que, a diferencia del método *hold-out*, es lento desde el punto de vista computacional. En la práctica, la elección del número de iteraciones depende de la medida del conjunto de datos. Indicar que cuando *K*=1 sería igual que el *hold-out*.



- FIGURA 7 - K-FOLD CROSS-VALIDATION

Una vez completada la matriz de confusión se pueden obtener un conjunto de estadísticas que indicarán cómo de bueno ha sido el clasificador.
A continuación, se enumeran las más relevantes:

- ***True positive rate, recall* o sensibilidad**: Indica la capacidad de nuestro estimador para dar como casos positivos los casos que realmente son positivos; proporción de clase positiva correctamente identificadas.
  Su fórmula es $TPR = \frac{TP}{TP+F}$

- ***True negative rate* o especificidad:** Nos indica la capacidad de nuestro estimador para dar como casos negativos los casos realmente negativos; proporción de clase negativa correctamente identificados.
  Su fórmula es $TNR = \frac{TN}{TN+FP}$

  En la Figura 8-sensibilidad y especificidad se puede ver gráficamente como se calculan la sensibilidad y la especificidad.

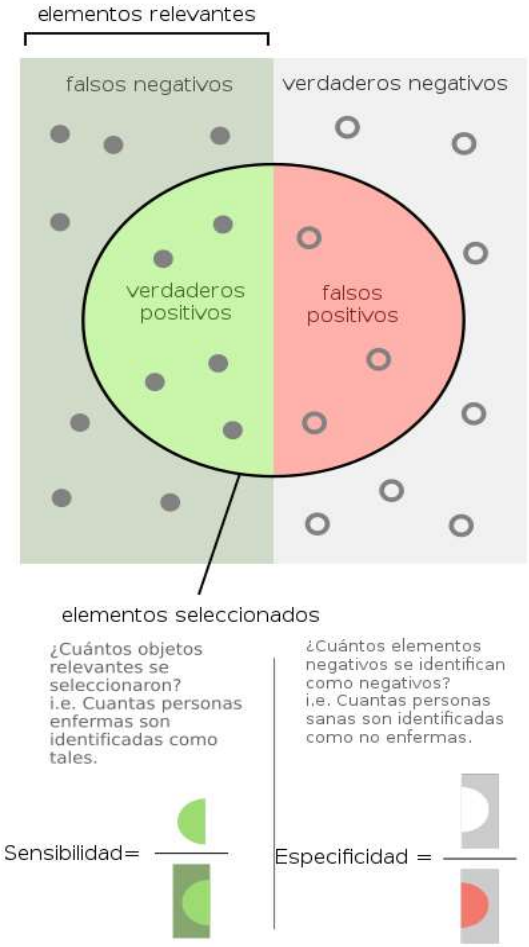


FIGURA 8-SENSIBILIDAD Y ESPECIFICIDAD

- ***False negative rate o error tipo II:*** Es el error que comente nuestro estimador al clasificar como negativo algo que realmente es positivo.
  Su fórmula es $FNR = \frac{FN}{TP+FN}$
- ***False positive rate o error tipo I:*** Es el error que comente nuestro estimador al no clasificar como negativo algo que realmente sí que lo es.

Su fórmula es $FPR = \frac{FN}{TP+FN}$

- ***Precision o Precisión:*** La precisión indica que fracción de instancias realmente positivas de todas las instancias clasificadas como positivas
  Su fórmula es $PPV = \frac{TP}{TP+FP}$
  Teniendo en cuenta la Figura **8** la precisión visualmente seria la mostrada en la Figura 9 - precisión:

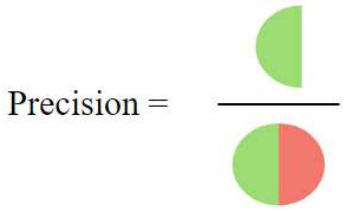


FIGURA 9 - PRECISIÓN

- ***Accuracy o exactitud:*** Indica el porcentaje correcto de clasificados, tanto positivos como negativos.
  Su fórmula es $ACC = \frac{TP+TN}{TP+TN+FP+F}$

En este proyecto se utilizan las figuras de precisión y sensibilidad como referencia. El propósito del proyecto es intentar obtener información sobre los tweets etiquetados como penalmente relevantes. Estos serán una minoría dentro del conjunto de datos, es por ello que necesitamos saber la tasa de acierto a la hora de clasificar los casos positivos y ver el porcentaje de casos realmente positivos que se han etiquetado incorrectamente.

3.2 Text-mining

El *text-mining* es un campo de la minería de datos en el cual el conjunto de datos son originalmente textos (foros, redes sociales, noticias, etc.). El objetivo del *text-mining* consiste en ser capaz de extraer conocimiento a partir de textos. El problema reside en el tratamiento de los datos, ya que no están estructurados en bases de datos, por lo que existen diferentes clases de problemas:

- La falta de estructura del texto. Puede haber textos carentes de una estructura homogénea procesable de forma automática sin que se produzca pérdida de información.
- La naturaleza heterogénea y distribuida de los documentos.
- El multilingüismo presente no solo en diferentes conjuntos, sino también dentro de una misma colección de textos.

- El análisis de texto depende del contexto y del dominio de la aplicación, lo cual implica el uso de diccionarios específicos de dicho contexto para poder llevar a cabo el procesamiento correcto del texto.

El tratamiento de datos en la minería de texto difiere de la minería de datos, como se muestra a continuación.

En cada instancia habrá un atributo de tipo texto, que tendrá asociado una clase. Para un tratamiento más sencillo y manejable se suele convertir el atributo del mensaje en un vector numérico, para ellos se utilizará una técnica llamada *bag of words* (bolsa de palabras.). Esta técnica convierte un atributo de tipo texto en un conjunto de atributos que representan la presencia de las palabras en el texto. Concretamente, la dimensión del vector será el número de palabras presentes en el texto o textos a tratar (tabla 4).

	el	perro	gato	araña	está	en	la	mesa	al	sobre
el perro está en la mesa	1	1	0	0	1	1	1	1	0	0
el gato araña al perro sobre la mesa	1	1	1	1	0	0	1	1	1	1

TABLA 4 – EJEMPLO DE *BAG OF WORDS* SOBRE DOS TEXTOS

Esta técnica para representar textos sólo toma en cuenta la presencia/ausencia de las palabras, pero no toma en cuenta el orden de esas palabras en el texto. En este punto ya no se dispone del texto con una estructura sintáctica. Por ejemplo "el pez grande se come al pequeño" tiene una representación idéntica a "el pez pequeño se come al grande" una vez utilizado el *bag of words.* Con esta transformación se pierde parte de la información, pero se consigue una representación de las instancias más simples y manejables computacionalmente. A su vez, en los textos existen las denominadas palabras vacías, que son aquellas que aparecen frecuentemente pero que no aportan significado relevante, como por ejemplo los artículos, preposiciones y conjunciones. El hecho de emplear las palabras como atributos provoca que la dimensión del espacio de atributos sea muy elevada. Se dispone de un número muy elevado de atributos para representar una instancia y el ratio de aparición de los atributos en el conjunto de instancias en muy bajo, dando lugar a dos fenómenos perjudiciales: sobre-ajuste y sesgo en la estadística. Por ello se utilizan técnicas de selección de atributos para descartar atributos redundantes o irrelevantes para el proceso de clasificación. Ayudan a simplificar la representación de las instancias reduciendo el orden de magnitud del espacio de búsqueda. Esto suele agilizar el proceso de clasificación, aunque conllevará también una

pérdida de información. En este sentido existen un gran número de técnicas de selección de atributos. A continuación, se describe la técnica TF-IDF, utilizada en este proyecto, que sirve para seleccionar los atributos más relevantes para clasificar un conjunto de textos.

- TF-IDF:

Se define la frecuencia relativa del término w_i en el documento d_j, *term frecuency* (TF), según la formula $TF(w_i, d_j) = \frac{f(w_i,d_j)}{\sum_{w_i \epsilon V} f(w_i,d_j)}$, donde:

-$f(w_i, d_j)$ representa el número de veces que aparece el termino $w_i$en el documento d_j.

-$\sum_{w_i \epsilon V} f(w_i, d_j)$ representa el número total de términos que aparecen en el documento d_j

-V es el conjunto de términos o vocabulario de la aplicación.

Cuanto mayor sea $TF(w_i, d_f)$ más característico o relevante resulta el termino w_i para describir el documento d_j. Sin embargo, los términos frecuentes como determinantes o artículos son muy frecuentes en todos los documentos, y por tanto no son buenos atributos predictores. Para atenuar la relevancia que se le asocia al termino w_i se define la frecuencia relativa de los documentos que contiene el término w_i, *document frequency* (DF), según la expresión $DF(w_i) = \frac{\sum_{d_j \epsilon D} \delta(w_i,d_j)}{|D|}$, donde:

- $\delta(w_i, d_j)$ representa la delta de Kronecker sobre la pertenencia del termino w_i en el documento d_j según la siguiente expresión:

$$\delta(w_i, d_j) = \begin{cases} 1 \; w_i \text{ está presente en el documento } d_j \\ 0 \; w_i \text{ no está presente en el documento } d_j \end{cases}$$

-$\sum_{d_j \epsilon D} \delta(w_i, d_j)$ representa el número de documentos que contienen e término w_i

-$|D|$ representa al número total de documentos

Cuanto menor se $DF(w_i, d_j)$ más ayudará el término w_i a discriminar entre los distintos documentos. A fin de determinar cuantitativamente el grado de relevancia del término w_i en el conjunto de documentos se define TF-IDF (*term frequency-inverse document frecuency*) según la expresión:

$$\text{TF-IDF}(w_i, d_j) = TF(w_i, d_j) \cdot log \frac{1}{DF(w_i)}$$

En resumen, TF es una medida para cuantificar la relevancia de un término dentro de un documento. IDF es una medida para cuantificar la relevancia de un término en un conjunto de documentos, TF-IDF es una medida que combina TF e IDF, de modo que cuantifica la relevancia de un término dentro de un documento considerando los demás documentos. (Ramirez, 2017)

3.3 Gestor de base de datos

Seleccionar el gestor de bases de datos correcto ha sido uno de los puntos críticos del proyecto, ya que es necesario tener en cuenta el gran volumen de información que se va a almacenar, además de la compatibilidad con el resto de herramientas a utilizar. Teniendo en cuenta la estructura relacional, se analizó el tipo de gestores que había en el mercado y cuáles eran sus ventajas y desventajas.

Inicialmente, se eliminaron todas aquellas opciones que suponían un coste económico, por lo que la elección se redujo a los gestores MariaDB, MySQL y PostgreSQL.

Finalmente, al utilizarse XAMPP, un paquete de software libre, que incluye entre otras aplicaciones un gestor de bases de datos con una administración vía web más intuitiva, este utiliza por defecto MariaDB, por lo que finalmente se decantó por este gestor.

Una vez seleccionado del sistema de gestión de la base de datos, hubo que elegir entre los dos tipos de motores de almacenamiento disponibles: InnoDB y MyISAM; tras revisar varios artículos disponibles en internet (Arsys, 2012)y teniendo en cuenta las restricciones de almacenamiento especificadas por el fabricante, se optó por InnoDB por los siguientes motivos (openalfa, 2013):

- Permite tener las características ACID (Atomicity, Consistency, Isolation and Durability: Atomicidad, Consistencia, Aislamiento y Durabilidad en español), garantizando la integridad de las tablas.
- Tiene restricciones de clave externa (foreign key constraints)
- Ofrece recuperación automática en caso de crash
- Dispone compresión de tablas con posibilidad de lectura/escritura
- Los datos son guardados en páginas en orden de clave primaria

- Es probable que si la aplicación hace un uso elevado de *INSERT* y *UPDATE* se note un aumento de rendimiento con respecto a MyISAM.
- El Tamaño máximo de una tabla puede llegar hasta los 64TB (Varios, 2018)

3.4 Servicio en la nube

Debido a que no se tenían conocimientos relativos a los servicios de almacenamiento online disponibles en el mercado, fue necesario realizar una pequeña investigación para conocer de primera mano la situación actual de esa área y cuáles eran las empresas más demandadas (Dignan, 2018)

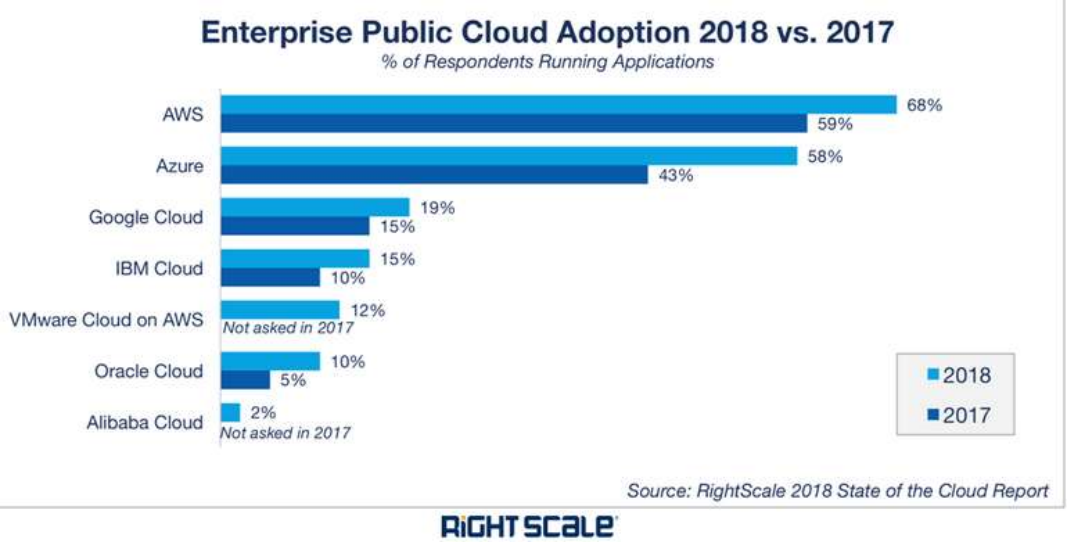


FIGURA 10 – EMPRESAS CON MAYOR PRESENCIA EN LA NUBE

Tras la revisión (Figura 10 – empresas con mayor presencia en la nube) se decidió contactar con las 3 empresas con mayor presencia en el sector para consultar el precio y las condiciones de cada una de ellas.

AWS no contesto a la misiva, Google Cloud envió un correo por defecto en el que explicaba sus condiciones, y solo Azure estableció contacto telefónico directo.

Los dos que contestaron hacían referencia a la opción de utilizar durante un periodo gratuito el sistema. Puesto que Azure ofrecía un periodo de prueba gratuito de un mes y, en cambio, la cuenta de Google

Cloud podía utilizarse durante un año (con la misma cuenta de Google, sin necesidad de registrarse nuevamente) se optó por esta última.

3.5 Lenguaje de programación

Entre los lenguajes más utilizados para el análisis de datos y su procesamiento mediante técnicas de minería de datos, destacan sobre todo Python y R (Rochina, 2016). Es por ello que en un primer momento fueron dos de las opciones que se pusieron encima de la mesa.

Los motivos por los que se decidió descartar el uso de Python son diversos. Por un lado, la falta de los conocimientos necesarios sobre el sistema impedía llevar a cabo un proyecto de esta envergadura. Por el otro, el limitado abanico de opciones para un desarrollo en un entorno web supone un proceso más tedioso que con otros lenguajes, y fue precisamente este último el factor determinante a la hora de desechar esta opción.

R se convirtió entonces en la alternativa más conveniente, puesto que dispone de todo tipo de herramientas de *text mining*, así como la posibilidad de desarrollar una aplicación web con el paquete R Shiny.
Las dificultades comenzaron a la hora de desarrollar la plataforma web con R, dado que R Shiny no permite renderizar más de una página web a la vez, de modo que hubo que buscar otro lenguaje para el desarrollo de la aplicación.
Finalmente, se optó por usar java ya que ha sido un lenguaje utilizado a lo largo de toda la carrera y del cual se tienen amplios conocimientos, además de contar con una amplia variedad de herramientas para realizar text mining y un gran número de servidores web.

3.6 Herramienta para el text mining

En un principio la herramienta se iba a desarrollar con Tensorflow, "una biblioteca de código abierto para aprendizaje automático a través de un rango de tareas, y desarrollado por Google para satisfacer sus necesidades de sistemas capaces de construir y entrenar redes neuronales para detectar y descifrar patrones y correlaciones, análogos al aprendizaje y razonamiento usados por los humanos." (Delgado, 2017) La elección de esta novedosa herramienta estuvo basada en la

potencia que ofrece, y está respaldada por la mayor compañía a nivel informático.

No obstante, el hecho de que sea una aplicación tan reciente supuso un problema en la medida de que las funciones necesarias para el desarrollo de la aplicación están aún en fase experimental, y como la propia página del producto indica, no ofrece la garantía de estabilidad requerida. Por el mismo motivo, no hay demasiada información a la que recurrir en caso de dudas o fallos en su utilización.

Todo ello supuso tener que sacrificar potencia en beneficio de más estabilidad y un mayor respaldo de la comunidad, por lo que la elección final fue utilizar Weka[3], una herramienta previamente utilizada en las asignaturas de minería de datos y sistema de apoyo a la decisión.

3.7 Servidor Web

Como ya se ha mencionado anteriormente, en un primer momento se decidió utilizar R Shiny como aplicación web, pero dada la imposibilidad de renderizar más de una página web (Kim, 2016) (algo necesario en nuestro caso ya que el usuario debe interactuar con la aplicación pasando por diferentes pantallas para ello) se descartó.

Teniendo en cuenta que Java iba a ser el lenguaje de programación a utilizar, el uso de Tomcat como servidor web pareció la opción más conveniente, dado que es una de las aplicaciones de servidores más utilizada a nivel mundial en el entorno de Java (Salnikov-Tarnovski, 2017)(Figura 11 – Cuota de mercado de aplicación de servidores java)

[3] https://www.cs.waikato.ac.nz/~ml/weka/

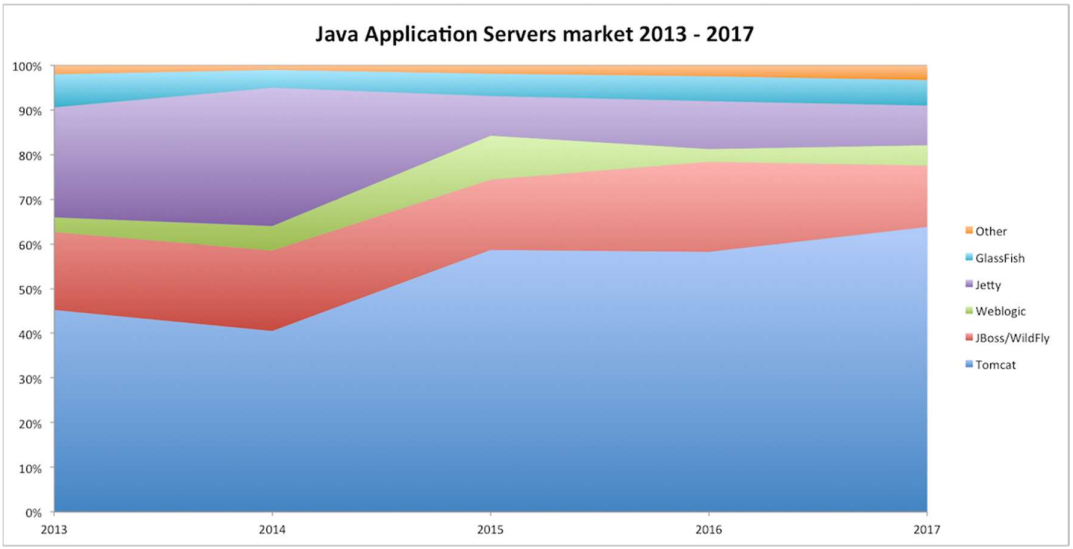


FIGURA 11 – CUOTA DE MERCADO DE APLICACIÓN DE SERVIDORES JAVA

4 Captura de requisitos

En este apartado se presentará la captura de requisitos. La captura de requisitos es el paso a seguir para poder realizar una buena aplicación y se podría considerar como uno de los pasos más importante en el proceso de realización de cualquier aplicación.

4.1 Jerarquía de actores

A continuación, se presentará el número de actores que habrá en el sistema y que rol tendrá cada uno de ellos en él.

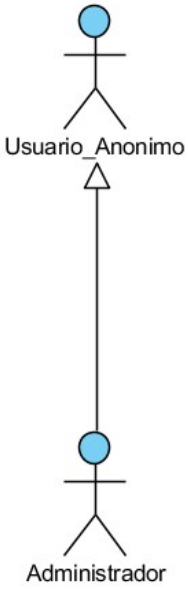


FIGURA 12 – JERARQUÍA DE ACTORES

Usuario_Anónimo
Este actor será el usuario que acaba de llegar a la página de inicio y aún no está dentro del sistema. La única acción disponible será el identificarse en la página web

Administrador
Este actor representa al usuario ya registrado en el sistema. Tendrá acceso a todas las funcionalidades del sistema

4.2 Casos de uso

A continuación, se mostrar el diagrama de los casos de uso y una descripción de ellos:

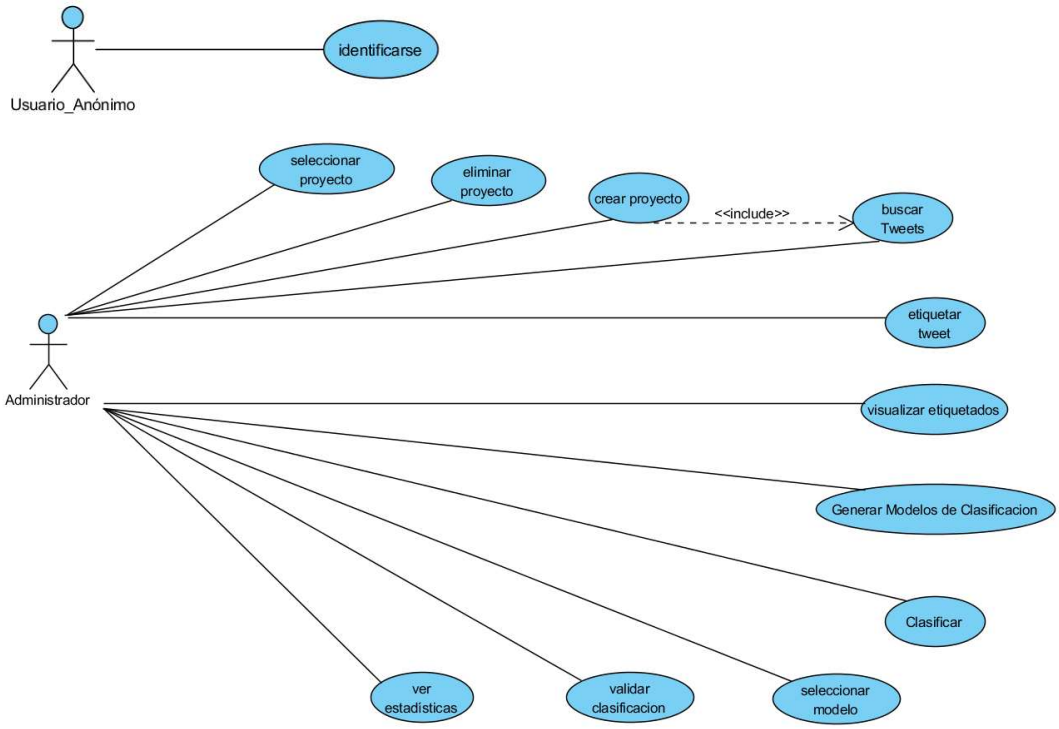


FIGURA 13 – CASOS DE USO

4.2.1 Identificarse

Permite al usuario anónimo conectarse a la aplicación, de este modo podrá acceder a todas las funcionalidades del usuario administrador

4.2.2 Seleccionar proyecto

Permite al usuario administrador seleccionar cualquiera de los proyectos anteriormente generados

4.2.3 Eliminar proyecto

Permite al usuario administrador eliminar cualquiera de los proyectos anteriormente generados

4.2.4 Crear proyecto

Permite al usuario administrador crear un nuevo proyecto en el sistema

4.2.5 Buscar tweets

Permite al usuario administrador realizar una búsqueda en Twitter

4.2.6 Etiquetar tweet

Permite al usuario administrador etiquetar los tweets indicando si son penalmente relevantes o no, y en caso afirmativo indicar que tipo de delito comenten

4.2.7 Visualiza etiquetados

Permite al usuario administrador visualizar los tweets previamente etiquetados.

4.2.8 Clasificar

Permite al usuario, seleccionar un modelo de predicción y aplicarlo sobre el conjunto de tweets no etiquetados de ese proyecto

4.2.9 Validar clasificación

Permite al usuario administrador validar la clase predicha por el modelo predictivo

4.2.10 Generar modelos de clasificación

Permite al usuario administrador generar varios modelos predictivos con los tweets previamente etiquetados

4.2.11 Ver estadísticas

Permite al usuario administrador ver las estadísticas de las palabras más repetidas de los tweets clasificados como penalmente relevantes

4.2.12 Seleccionar modelo

Permite al usuario administrador seleccionar entre los cinco modelos de clasificación disponibles para clasificar los tweets.

4.3 Modelo de Dominio

A continuación, se muestra el modelo de dominio asociado a la aplicación:

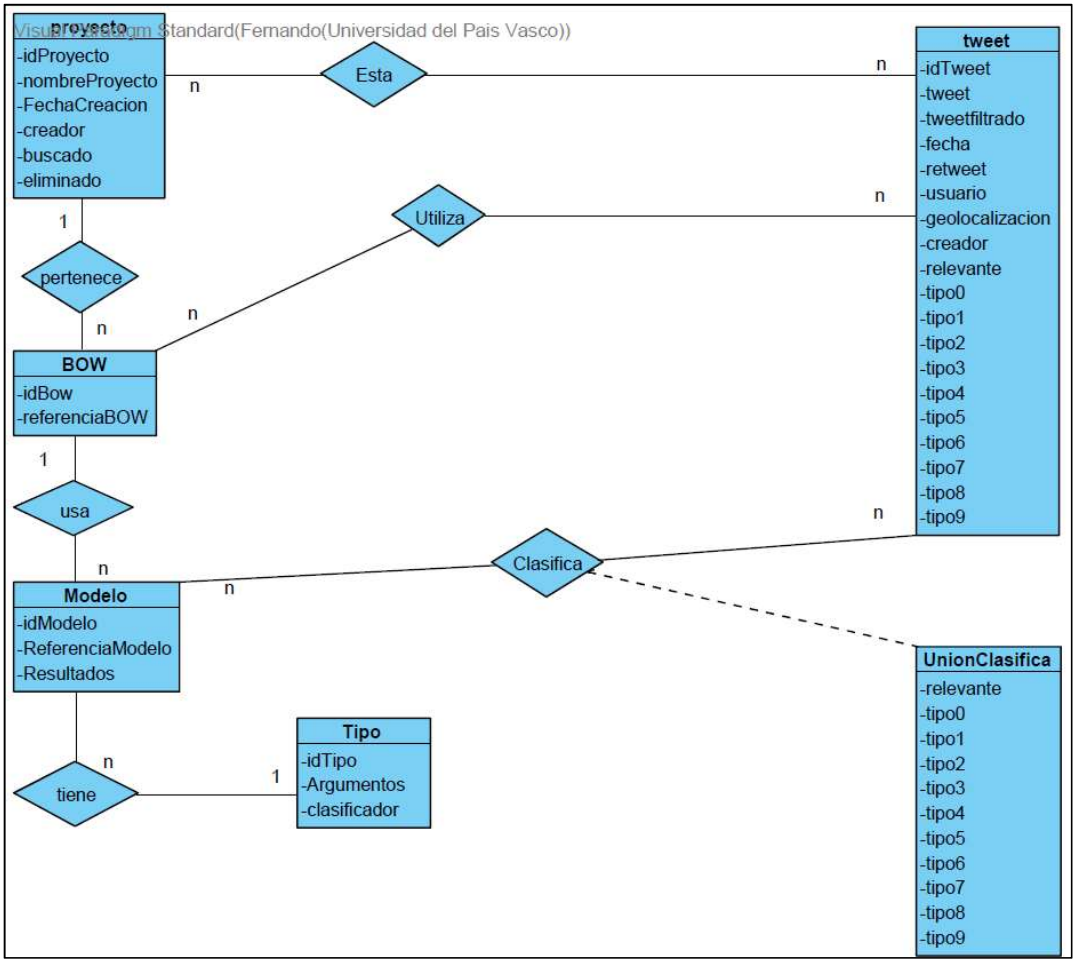


FIGURA 14 -MODELO DE DOMINIO

4.3.1 Explicación de las entidades

1. Proyecto: En la entidad proyecto se guardarán los datos correspondientes a las búsquedas realizadas por los usuarios. Se guardará el nombre que el usuario haya querido poner al proyecto, un

identificador único, su fecha de creación, el usuario que lo ha creado, el término buscado en Twitter y un atributo que con el nombre eliminado. Este último se utilizará para saber si el usuario quiere eliminar el proyecto. En caso de que lo elimine sin querer se pueda recuperar.

2. Tweet: En esta entidad se guardarán todos los datos relativos a los tweets buscados por el usuario. La clave será el propio identificador único suministrador por Twitter, además de eso se almacenará el texto del tweet sin filtrar y en otro apartado el texto del tweet filtrado (quitando referencias a enlaces externos). También se guardará la fecha de su creación, el número de retweets que tiene, quien fue el creador del tweet, y en caso de que haya sido retuiteado que usuario ha sido y su geolocalización. Tendrá también un atributo booleano llamado relevante, que, en caso de ser verdadero indicará que dicho tweet es penalmente relevante. A su vez dispondrá de diez subclases indicando conductas que incitan al odio, que, en caso de que los juristas hayan etiquetado el tweet como relevante podrán seleccionarlos.

3. BOW: La entidad BOW (Acrónimo de Bag of Word) se guardará una cadena de caracteres que hace referencia a un CSV y un identificador único para cada cadena almacenada.

4. Modelo: Esta entidad hace referencia al modelo inferido por el sistema para la clasificación. La clave es un identificador único, y con ella se almacenará también la referencia a ese objeto (el modelo generado por el sistema) y los resultados obtenidos tras su aplicación en la categorización.

5. Tipo: A la hora de realizar la minería de datos, la entidad Modelo requiere que se le pasen los parámetros que el usuario consideran óptimos para que este prediga de la manera más precisa posible. Al ser este un parámetro variable, se

almacenará bajo la entidad Tipo. Como clave dispone de un identificador.

3.1.2. Explicación de las relaciones

A continuación, se va a exponer las explicaciones entre las relaciones. Para que se puedan entender, se hará uso de la siguiente nomenclatura:

RelaciónA – RelaciónB - Nombre de la relación: Explicación de cardinalidad y razonamiento

1. **Proyecto – Tweet - Esta**: Esta es una relación N a M. En un proyecto hay gran cantidad de tweets, pero a su vez un tweet puede aparecer en más de un proyecto debido a las palabras que aparecen en él.
2. **Tweet – BOW – Utiliza**: Esta es una relación N a M. Para realizar un BOW, se utiliza todos los tweets clasificados hasta ese momento de un proyecto. A su vez, al variar el número de tweets clasificados en cada momento, un tweet aparecerá en más de una BOW, ya que al añadir nuevos tweets al sistema se genera uno nuevo.
3. **Proyecto – BOW – Pertenece**: Esta es una relación 1 a N. En un mismo proyecto podrá haber más de un BOW, al etiquetar nuevos tweets se genera un nuevo BOW. En cambio, el BOW es válido únicamente para un proyecto.
4. **BOW – Modelo – Usa**: Esta en una relación 1 a N. Para la generación de un modelo se utiliza un único BOW, pero visto que al añadir nuevos tweets al sistema se generarán nuevos BOW, esto también afectará al modelo ya que se generará uno nuevo. Esto es beneficioso ya que aumenta su probabilidad de acierto.
5. **Modelo – Tweet – Clasifica**: Esta es una relación N a M. Un modelo, clasifica todos los tweets no etiquetados de un mismo proyecto, pero dado que hay diferentes modelos cada uno con su algoritmo, un mismo tweet podrá ser clasificado de diferentes maneras dependiendo del modelo utilizado. Es por ello que en esta relación se requiera una nueva entidad llamada UnionClasifica, en

la que se almacenara la clasificación de cada tweet dependiendo del modelo utilizado. Esto a su vez ayudará para saber que tweets han sido clasificados por el sistema y cuales etiquetados por el usuario

6. **Modelo – Tipo – Tiene:** Esta en una relación 1 a N. Un modelo solo utiliza un Tipo dado que son las opciones que utilizará a la hora de clasificar los tweets. En cambio, Tipo podrá pertenecer a más de un modelo, para así cambiar las opciones de clasificación utilizadas.

5. Análisis y diseño

El siguiente apartado se centrará en exponer el diseño de las diferentes partes la aplicación.

5.1 Transformación del modelo de dominio a BBDD

La elaboración de la base de datos ha sido uno de los puntos críticos del proyecto, ya que había que tener en cuenta el gran número de datos que se iban a almacenar y a su vez tener la mayor trazabilidad posible de todos los elementos que la componen.

El primer paso fue pasar el modelo de dominio expuesta anteriormente a base de datos, quedando de la siguiente manera:

Las entidades que forman parte del modelo se transforman en tablas directamente con sus atributos como campos de la misma.

- Proyecto
- BOW
- Tweet
- Modelo
- Tipo

Las relaciones N a M también requieren de transformación

- Proyecto – Tweet – Esta: Se creará la tabla llamada esta, teniendo como clave el identificador de la tabla proyecto y el identificador tabla tweet.

- Tweet – BOW – Utiliza: Se creará la tabla llamada utiliza, teniendo como clave el identificador de la tabla tweet y el identificador de la tabla BOW.

- Modelo – Tweet – Clasifica: se creará una tabla llamada clasifica que tendrá como clave el identificador de la tabla tweet, el identificador de la tabla Modelo y como atributos, los pertenecientes a la entidad unionclasifica.

De modo que la base de datos quedará de la siguiente manera:

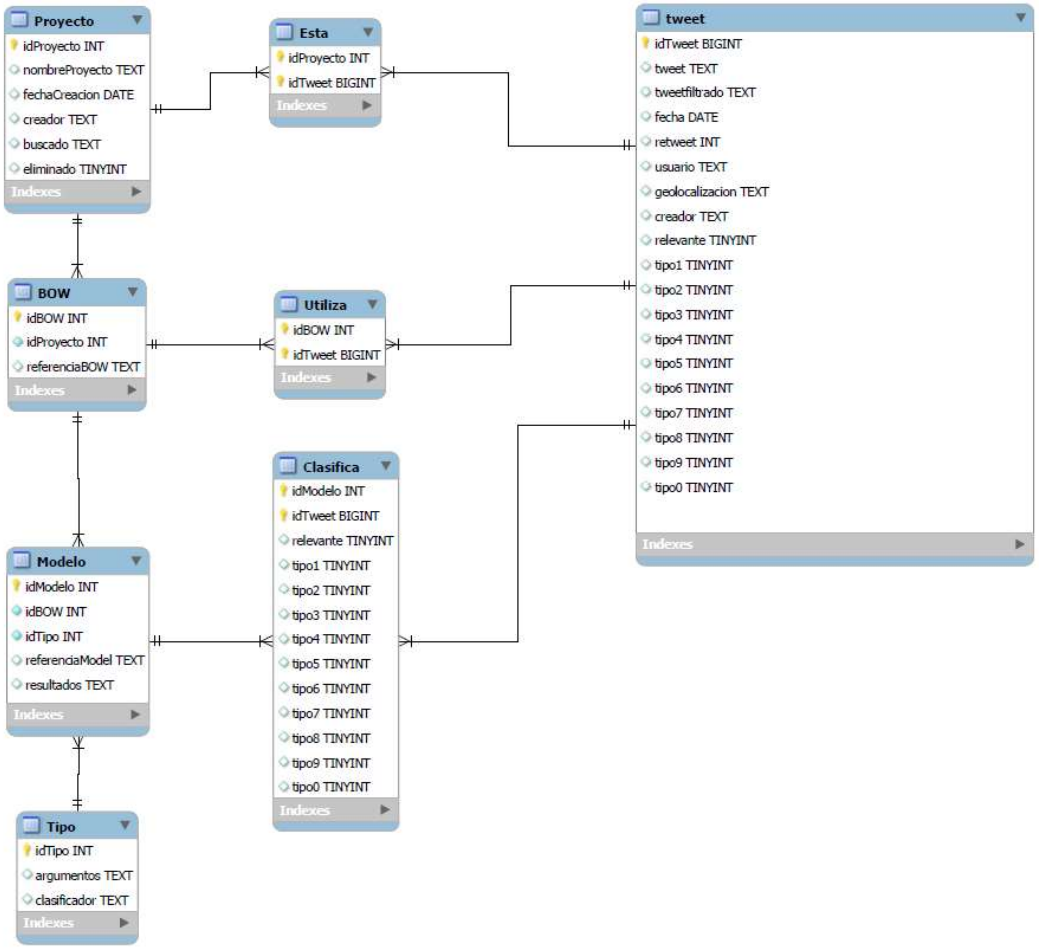


FIGURA 15 – DIAGRAMA DE LA BASE DE DATOS

5.2 Diagrama de clases

Para un mejor diseño, se decidió dividir en tres paquetes: uno llamado Weka, que es el módulo de minería de datos, otro llamado servlet para el *front-end* de la aplicación y un último llamado proyecto para el *back-end*. Se mostrarán cada uno de ellos y se dará una explicación sobre lo mas relevante de cada una de las clases.

5.2.1 Paquete Weka

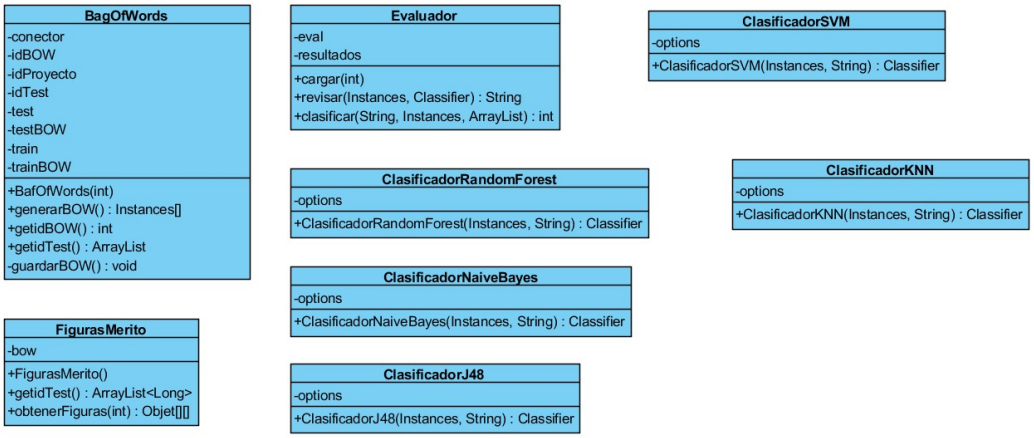


FIGURA 16 – DIAGRAMA DE CLASES WEKA

- BagofWords: Es la clase encargada de realizar el *bag of words*, requiere el identificador de un proyecto para poder realizar el bag of Word.
- Evaluador: Es la clase que se encarga de realizar los modelos de aprendizaje
- FigurasMerito: Esta clase se encarga de obtener las figuras de mérito.

Las cinco clases restantes son las que cargan el clasificador con las opciones que se decidan.

5.2.2 Paquete Proyecto

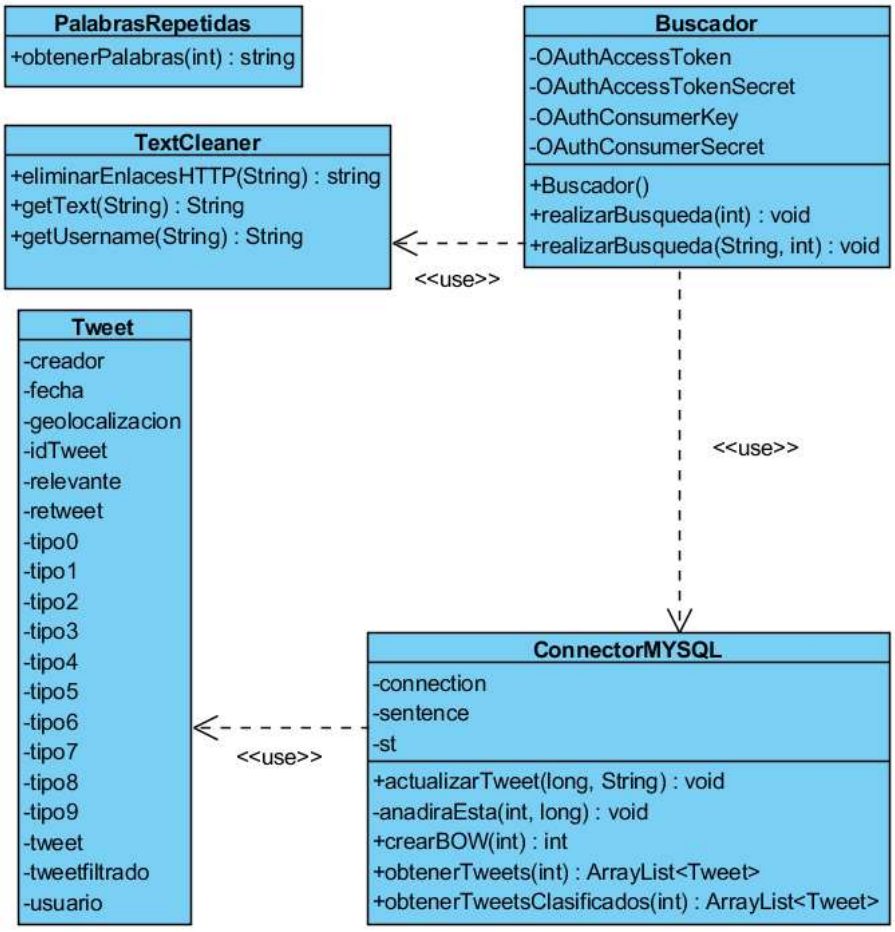


FIGURA 17 - DIAGRAMA DE CLASES PROYECTO

- PalabrasRepetidas: Esta clase devolverá un string con las palabras mas repetidas de los tweets etiquetados como penalmente relevantes de un proyecto
- Buscador: Es la clase que realiza las búsquedas en Twitter
- TextCleaner: Es la clase encargada tratar los textos, eliminar enlaces externos, selecciona los nombres o el texto en un tweet.
- Tweet: Esta clase almacena los datos de que tiene un tweet, además de los atributos que se observan tiene los setter y getter de cada uno de ellos, no se han incluido porque no impediría que se visualizase correctamente el diagrama
- ConnectorMYSQL: Es la clase encargada de realizar las consultas a la base de datos. No se han incluido todos los métodos por su gran numero. Impediría que se visualizase correctamente el diagrama

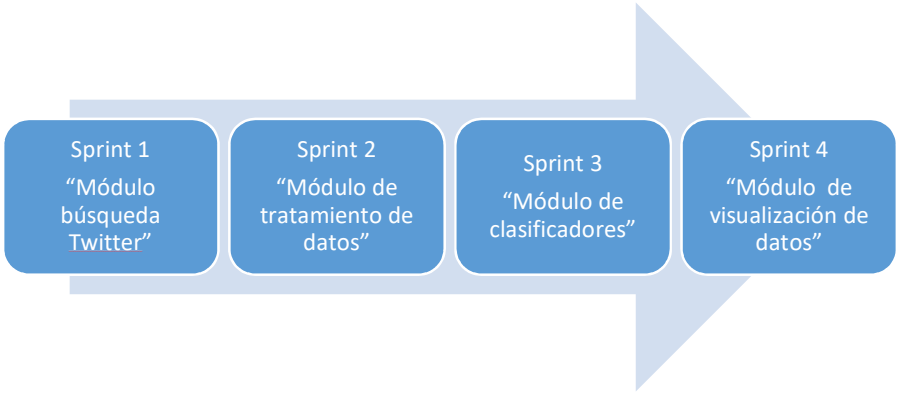
5.2.3 Paquete servlet

FIGURA 18 – DIAGRAMA DE CLASES SERVLET

La única clase reseñable en este paquete es TwitterSearcher, que es la encargada de crear los proyectos y realizar la búsqueda. El resto de clases son usadas para moverse por la página, o hacer llamadas a otras clases arriba mencionadas.

6. Desarrollo

A continuación, se detallará como ha sido el desarrollo del proyecto, separándolo en los cuatro sprint.



- Sprint 1

En el primer sprint se desarrollo el modulo que permitía realizar las búsquedas en Twitter. Dado que esas búsquedas necesitaban ser guardadas, también se desarrollo la base de datos. Finalmente se elaboro la parte web que permitiría visualizar el contenido de los tweets.
Para elaborar el módulo de búsqueda en Twitter se accedió a la propia página de la red social[4] para ver que librerías estaban disponibles para tal función en Java.
La elección fue relativamente sencilla al haber una única librería disponible en java referenciada en la página web, llamada twitter4j.
Aquí empezó uno de los primeros inconvenientes al desarrollar la aplicación, ya que, desde finales del 2017, Twitter lanzó una API de pago (Dau, 2017), dejando las funciones de la API gratuita algo mermadas, con la intención de que el usuario realizase aportaciones económicas.
Por esta razón se decidió informar al cliente de lo sucedido. Éste indicó que no tenía ningún inconveniente en realizar algún pago por la aplicación, pero la API que estaba disponible para Java no soportaba dicha opción, ya que las consultas realizadas a la página web variaban.
Finalmente se optó por utilizar exclusivamente la versión gratuita por la complejidad y el tiempo que llevaría realizar una API propia.

Uno de los principales requisitos en minería de datos es que se debe obtener el mayor número posible de información que, en nuestro caso, estará dada por tweets. Twitter permite realizar más de 50 peticiones

[4] https://developer.twitter.com/en/docs/developer-utilities/twitter-libraries.html

cada 15 minutos, devolviendo un máximo de 100 tweet por petición, siendo la respuesta máxima posible de 75.000 tweets. Además, había que considerar que era muy probable que si se realizaba una misma búsqueda sobre un mismo termino más de una vez de forma continua, el resultado fuera el mismo, al igual que sucede en páginas como Google. Es por ello que se consultó en internet algún método para intentar eludir ese tipo de restricción, y vimos que, en efecto, un usuario encontró una manera de obtener más tweets haciendo uso de los identificadores únicos de Twitter. A partir de las referencias encontradas se ha desarrollado el siguiente código:

```
do {
    result = twitter.search(query);
    List<Status> tweets = result.getTweets();

    searchResultCount = result.getTweets().size();

    for (Status tweet : tweets) {

        java.sql.Date fechaCreacion = new java.sql.Date(tweet.getCreatedAt().getTime());

        if (tweet.isRetweet()) {

            text = txt.getText(tweet.getRetweetedStatus().getText());
            creador = txt.getUsername(tweet.getText());
            mysql.añadirTweet(pIdProyecto, tweet.getId(), text, txt.elimiarEnlacesHttp(text), fechaCreacion,
                    tweet.getRetweetCount(), tweet.getUser().getScreenName(), tweet.getUser().getLocation(),
                    creador);
        } else {
            text = txt.textParser(tweet.getText());
            creador = "-";
            mysql.añadirTweet(pIdProyecto, tweet.getId(), text, txt.elimiarEnlacesHttp(text), fechaCreacion,
                    tweet.getRetweetCount(), tweet.getUser().getScreenName(), tweet.getUser().getLocation(),
                    creador);
        }

        if (tweet.getId() < lowestTweetId) {
            lowestTweetId = tweet.getId();
            query.setMaxId(lowestTweetId);
        }

    }
} while (searchResultCount != 0 && searchResultCount % 100 == 0);
```

FIGURA 19 – CÓDIGO DE BÚSQUEDA EN TWITTER

Con este código se tiene en cuenta el identificador único devuelto por Twitter, guardándose el más bajo, de modo que se van buscando desde los tweets más nuevos hasta los más viejos. También tiene en cuenta el tamaño de la búsqueda devuelta, con lo que vuelve a realizar otra búsqueda sobre sí mismo para obtener más tweets.

El desarrollo de la base de datos no tuvo grandes problemas al tener ya elaborado el diseño y utilizar un gestor vía web. La dificultad estuvo en su diseño. Había que tener en cuenta el gran número de tweets que se iban a almacenar, por lo que se llegó a pensar en la creación de tablas dinámicas. Esto último se descartó por su complejidad, pero se tuvo que revisar las restricciones de almacenamiento que había en las bases de datos disponibles para tenerlo en cuenta a la hora de su desarrollo.

En el desarrollo de la página web, se revisó qué herramientas había para visualizar el número de tweets disponibles en una tabla. Se vio que había un plug-in escrito en jquery llamado datatables que permitía la creación de tablas de una manera sencilla, por lo que se decidió utilizarla.
Al tener que adaptarla a la página, se tuvo que crear los botones de atrás y delante de forma manual.

- Sprint 2

Una vez elaborado el buscador, la base de datos y el apartado de la página web donde se visualizaban los tweets, se procedió a tratar los datos obtenidos en Twitter.
Un problema que se observó al realizar la búsqueda es que, en algunos casos, se obtenían textos incompletos, parecían acortados de manera intencionada por el programa. Tras realizar varias búsquedas por internet se observó que el problema venía dado por dos puntos:
El primero era que, si un usuario retweetea el mensaje de otra persona, ese mensaje se visualizaba de manera acortada. Es por eso que a la hora de obtener los tweets se tenía que tener en cuenta si el tweet había sido retuiteado o no y, en caso de serlo, considerarlo para un tratamiento especial, porque además de tener un usuario que lo había creado, también tenía otro usuario que lo había retweeteado:

```
if (tweet.isRetweet()) {

    text = txt.getText(tweet.getRetweetedStatus().getText());
    creador = txt.getUsername(tweet.getText());
    mysql.añadirTweet(pIdProyecto, tweet.getId(), text, txt.elimiarEnlacesHttp(text), fechaCreacion,
            tweet.getRetweetCount(), tweet.getUser().getScreenName(), tweet.getUser().getLocation(),
            creador);
```

FIGURA 20 - CÓDIGO DE OBTENCIÓN DE RETWEET

El segundo problema era que la última versión estable de twitter4j no permitía la visualización del tweet de más de 140 caracteres, al ser esta una funcionalidad añadida recientemente (EFE, 2017).
Es por ello que sacaron una versión posterior añadiendo esta funcionalidad[5]. El inconveniente que tenía es que no proporcionaban el archivo de java necesario para su funcionamiento, sino que daban el código fuente para que fuese el usuario el que lo compilase.
Una vez hecho todo lo anteriormente citado, se comprobó que todo funcionaba correctamente, y se obtenían los tweets de una manera legible.
El proceso de limpieza de texto sería un método simple que eliminaría los enlaces externos de los textos, pero como en casos anteriores hubo problemas.

[5] https://github.com/yusuke/twitter4j/

En un primer momento se pensó que con buscar toda palabra que empezase por http seria suficiente, pero tras varios intentos hubo que buscar una solución en internet. En este caso como en el anterior, había a gente que le había pasado el mismo problema (Bohemian, 2013), y les habían dado soluciones bastantes sencillas, con una simple línea de código se podría realizar dicha limpieza: *text.replaceAll("http\\S*", "");*

El segundo problema fue realizar un formulario, en el que las opciones estuviesen en separadas por columnas y que además cada fila debería de ser tratada de manera independiente.
Como en ocasiones anteriores, se lanzó una consulta en la página web stackoverflow.
En este caso se dieron varias sugerencias de cómo tratar los datos, de modo que con un solo formulario y teniendo en cuenta el identificador único de los tweets se pudiese obtener toda la información:

```
    <form action="Update" method="get">
        <input type="radio" value="No"
            name="<%=listaTweets.get(i).getIdTweet()%>" checked>No<br>
        <input type="radio" value="Si"
            name="<%=listaTweets.get(i).getIdTweet()%>">Si<br>
</td>
<td><input type="checkbox"
    name="<%=listaTweets.get(i).getIdTweet()%>" value="1">Promover
    hostilidad<br> <input type="checkbox"
    name="<%=listaTweets.get(i).getIdTweet()%>" value="2">Poseer
    material que promueve la hostilidad<br> <input
    type="checkbox" name="<%=listaTweets.get(i).getIdTweet()%>"
    value="3">Negar los delitos de Derecho Penal
    Internacional<br>
<td><input type="checkbox"
    name="<%=listaTweets.get(i).getIdTweet()%>" value="4">Vejar
    a grupos sociales<br> <input type="checkbox"
    name="<%=listaTweets.get(i).getIdTweet()%>" value="5">Enaltecer
    delitos<br> <input type="checkbox"
    name="<%=listaTweets.get(i).getIdTweet()%>" value="6">Justificar
    delitos<br>
<td><input type="checkbox"
    name="<%=listaTweets.get(i).getIdTweet()%>" value="7">Enaltecer
    el terrorismo<br> <input type="checkbox"
    name="<%=listaTweets.get(i).getIdTweet()%>" value="8">Justificar
    delitos del terrorismo<br> <input type="checkbox"
    name="<%=listaTweets.get(i).getIdTweet()%>" value="9">Vejar
    victimas del terrorismo<br> <input type="checkbox"
    name="<%=listaTweets.get(i).getIdTweet()%>" value="10">Delito
    de propaganda<br>
<td><input type="submit" value="etiquetar">
    </form>
```

FIGURA 21 - FORMULARIO PARA ETIQUETAR TWEETS

- Sprint 3

Acabados los dos primeros sprints se procedió con el tercero, el que generaría los clasificadores.

En este sprint se utilizaría la herramienta de desarrollo Weka, una herramienta ya familiar, ya que como se ha comentado previamente se había hecho uso de ella tanto en la asignatura de minería de datos como en la de sistema de apoyo a la decisión.

La realización del modulo no fue un gran problema por lo anteriormente mencionado, el problema surgió a la hora de ponerlo en marcha en el servidor web.

Por razones que se desconocían, a la hora de ejecutar la aplicación en un entorno web fallaba. En cambio, si se lanzaba como una aplicación de java normal funcionaba correctamente.

El fallo obtenido era que eclipse no encontraba las clases necesarias para realizar las funciones:

Exception in thread "main" java.lang.NoClassDefFoundError: libsvm/svm_print_interface

Se realizaron varias búsquedas en internet para ver si lo sucedido les había ocurrido a más personas, y tras comprobar que las soluciones aplicadas a ellos no funcionaban, se decidió realizar una consulta en la pagina web especializada en desarrollo informático stackoverflow y el foro oficial de Weka.

No se obtuvo ninguna respuesta de la página web stackoverflow. En cambio, del foro oficial de Weka contesto uno de sus desarrolladores, Eibe Frank[6], que indico que se siguiesen los pasos ya realizados tras la búsqueda en Google.

Se le comunicó que no funcionaba y afirmó que, según su experiencia debería de funcionar.

Dado que se estaba en un callejón sin salida, se empezaron a mirar posibles alternativas, y mientras se decidía montar un nuevo servidor para realizar pruebas, se vio que, Tomcat no utilizaba la misma ruta que eclipse a la hora de ejecutar librerías externas, se debían de definir en las propias opciones de Tomcat, como se observa en la Figura 22 – opciones de configuración del servidor tomcat

[6] https://www.cs.waikato.ac.nz/~eibe/

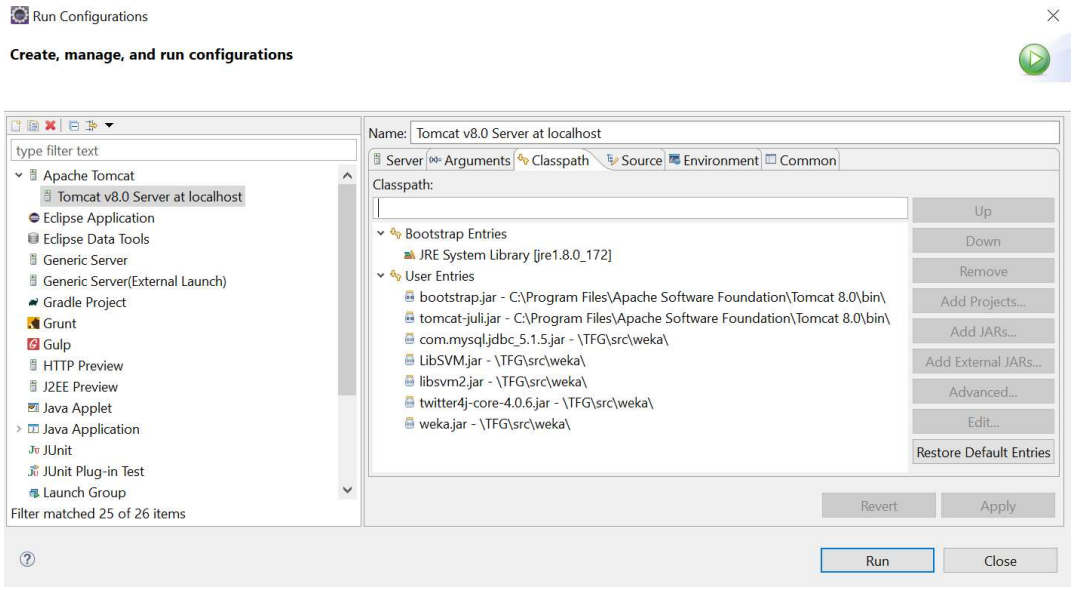


FIGURA 22 – OPCIONES DE CONFIGURACIÓN DEL SERVIDOR TOMCAT

Otra característica que no se tuvo en cuenta y que dio algún que otro quebradero de cabeza fue la realización del *Bag of Words*.

El conjunto de datos de entrenamiento y el conjunto de datos de clasificación disponen de atributos (palabras) diferentes, tanto en número como en valor. Es por ello que había que homogeneizarlo para que fuesen iguales, porque si no el clasificador no sabría qué hacer.

En nuestro caso, el conjunto que más información nos aporta es el conjunto de tweets etiquetados, por lo que no se desea perder información de ese montón en caso de haber alguna palabra que no se este disponible en el otro. Y viceversa, en caso de haber alguna palabra en el conjunto de datos no etiquetados que no aparezca en el conjunto de etiquetados, esta se podría desechar ya que no va a ser de utilidad a lo hora de clasificar.

Por ello la elaboración del *bag of word* debe de hacerse a la vez, para que tenga en cuenta en número de atributos y devuelva el mismo número:

```
public Instances[] generarBOW() {

    Instances[] devolver = new Instances[2];

    StringToWordVector filtroBOW = new StringToWordVector();
    filtroBOW.setIDFTransform(true);
    filtroBOW.setTFTransform(true);
    filtroBOW.setAttributeIndices("1");
    filtroBOW.setLowerCaseTokens(true);
    filtroBOW.setOutputWordCounts(true);
    filtroBOW.setStemmer(null);

    try {
        filtroBOW.setInputFormat(train);
        trainBOW = Filter.useFilter(train, filtroBOW);
        testBOW = Filter.useFilter(test, filtroBOW);
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    devolver[0] = trainBOW;
    devolver[1] = testBOW;

    GuardarBOW();

    return devolver;
}
```

FIGURA 23 - CÓDIGO DE GENERACIÓN DEL *BAG OF WORDS*

- Sprint 4

Debido a una mala planificación temporal, al sprint 4 no se le requirió el tiempo que hubiese sido necesario.

Es por ello que las estadísticas realizadas solo muestran las palabras mas repetidas del conjunto tweets penalmente relevantes.

Las pruebas funcionales realizadas no se han podido hacer con el cliente, algo que sería interesante para ver su opinión y realizar posibles cambios.

7. Verificación y evaluación

En este punto se detallarán el conjunto de pruebas realizados para verificar que el software cumple con los objetivos propuestos. Se describirá el proceso realizado, el resultado esperado tras su ejecución, el resultado obtenido y si se ha tenido que realizar algún tipo de corrección.

1. Introducción de nombre y/o contraseña incorrectos.
Descripción: El usuario introduce un nombre y una contraseña no válidos.
Resultado esperado: Se muestra una página de error.
Resultado obtenido: Muestra una página de error.
Acción correctora: Ninguna.

2. Introducción de nombre y/o contraseña correctos.
Descripción: El usuario introduce un nombre y una contraseña válidos.
Resultado esperado: Se redirige al usuario a la página principal y se genera una sesión con su nombre.
Resultado obtenido: Redirige al usuario a la página principal y genera una sesión con el nombre del usuario.
Acción correctora: Ninguna.

3. Pulsar sobre el botón "Ir" sin tener proyectos creados.
Descripción: El usuario pulsa sobre el botón "Ir" cuando no hay proyectos creados.
Resultado esperado: No realizar ninguna acción.
Resultado obtenido: Se obtiene error interno de Apache Tomcat intentando acceder a un recurso no disponible.
Acción correctora: Se eliminar el botón "Ir" cuando no haya proyectos creados.

4. Pulsar sobre el botón “Eliminar” sin tener proyecto creados.
Descripción: El usuario pulsa sobre “Eliminar” cuando no hay proyectos creados.
Resultado esperado: No realizar ninguna acción.
Resultado obtenido: Se obtiene error interno de Apache Tomcat intentando eliminar a un recurso no disponible.
Acción correctora: Se eliminar el botón “Eliminar” cuando no haya proyectos creados.

5. Crear dos proyectos con el mismo nombre.
Descripción: El usuario crea dos proyectos con el mismo nombre, aunque puedan tener búsquedas distintas.
Resultado esperado: Aparecen los dos proyectos, y al seleccionar cada uno aparecerán sus tweets.
Resultado obtenido: El último proyecto creado sobrescribe el anterior, aunque se muestren ambos.
Acción correctora: Se modifica la base de datos para que al generar el proyecto tenga un identificar único independiente del nombre.

6. Crear un proyecto con tildes o Ñs.
Descripción: El usuario crea un proyecto utilizando tildes y/o Ñs.
Resultado esperado: El proyecto se muestra correctamente.
Resultado obtenido: El proyecto muestra caracteres extraños en los lugares en los que debería de aparecer las tildes o las Ñs.
Acción correctora: Se modifica el código de la página web para que muestre las la codificación UTF-8 que permite visualizar caracteres como tildes y Ns.

7. Etiquetar tweets de manera incorrecta.
Descripción: El usuario indica que el tweet no es relevante, pero selecciona algún tipo de delito.
Resultado esperado: Al etiquetar el tweet solo se almacenará el tipo de delito seleccionado en caso de seleccionar la opción de que el tweet sí que es relevante.
Resultado obtenido: Al etiquetar el tweet solo se almacena el tipo de delito al indica que es un tweet relevante.
Acción correctora: Ninguna.

8. Generar más tweets.
Descripción: El usuario pulsa sobre el botón "Generar más tweets" para que realice una nueva búsqueda sobre el mismo término.
Resultado esperado: Al acabar la búsqueda aparecerán nuevos tweets sin etiquetar.
Resultado obtenido: Muestra nuevos tweets una vez a acabado la búsqueda.
Acción correctora: Ninguna.

9. Se etiquetan todos los tweets.
Descripción: El usuario etiqueta todos los tweets disponibles.
Resultado esperado: No se le mostrarán más tweets.
Resultado obtenido: No se muestra más tweets.
Acción correctora: Ninguna.

10. Los botones "siguiente" muestra los 50 siguientes tweets.
Descripción: El usuario pulsa sobre el botón "siguiente" y mostrará los 50 siguientes tweets.
Resultado esperado: La página mostrará los 50 siguientes tweets.
Resultado obtenido: La página muestra los 50 siguientes tweets.
Acción correctora: Ninguna.

11. Los botones "atras" muestra los 50 tweets anteriores.
Descripción: El usuario pulsa sobre el botón "atrás" y mostrará los 50 anteriores tweets.
Resultado esperado: La página mostrará los 50 tweets anteriores.
Resultado obtenido: Se muestra un error interno de Apache Tomcat al pulsar sobre el botón en la primera página.
Acción correctora: Se toma en cuenta la primera pagina para que no se salga del arraylist de tweets.

12. Visualizar tweets etiquetados.
Descripción: El usuario pulsa sobre el botón "ver clasificados" y le muestra los tweets ya etiquetados.
Resultado esperado: La página le muestra los tweets previamente etiquetados.
Resultado obtenido: **S**e muestran los tweets previamente etiquetados.
Acción correctora: Ninguna.

13. Reetiquetar tweets.
Descripción: El usuario pulsa sobre el botón "Reetiquetar", el tweet desaparece de la lista y aparece en la lista de tweets no etiquetados.
Resultado esperado: El tweet desaparece de la lista y aparece en la lista de tweets no etiquetados.
Resultado obtenido: El tweet desaparece de la lista y aparece en la lista de tweets no etiquetados.
Acción correctora: Ninguna.

14. Visualizar figuras de mérito.
Descripción: El usuario pulsa sobre el botón "clasificar", se redirigirá a otra página donde visualizará las figuras de merito obtenidas con cada uno de los algoritmos de clasificación.
Resultado esperado: La página mostrará las figuras de mérito obtenida de cada uno de los clasificadores.
Resultado obtenido: El clasificador SVM no muestra nada.
Acción correctora: La librería externa importada que realizaba el algoritmo SVM no era la correcta, se importa una nueva librería y se confirma que realiza el cálculo correctamente.

15. Validar tweets clasificados.
Descripción: El usuario pulsa sobre el botón "validar" el tweet se elimina de la lista y aparecerá posteriormente en la lista de tweets etiquetados
Resultado esperado: La página mostrará las figuras de mérito obtenida de cada uno de los clasificadores.
Resultado obtenido:
Acción correctora: Ninguna.

16. Ver estadísticas.
Descripción: El usuario pulsa sobre el botón "estadísticas"
Resultado esperado: La página mostrará las figuras de mérito obtenida de cada uno de los clasificadores.
Resultado obtenido:
Acción correctora: Ninguna.

8. Conclusiones y trabajo futuro

Después de la finalización del proyecto, hay que echar las vista atrás y recapitular sobre lo realizado con una mirada crítica, con la intención recoger experiencia para futuros proyectos de esta índole.

8.1 Revisión de los objetivos

Al comienzo de la memoria se definieron tres grandes módulos que a posteriori se desarrollarían. A continuación, mencionaremos si se han llegado a cumplir los objetivos para el desarrollo de esos módulos o no, con su correspondiente justificación y el correspondiente razonamiento:

1. Recuperación de tweets

Este modulo busca en Twitter el termino introducido por el usuario, almacena los tweets obtenidos y posteriormente los trata.

Como se ha mencionado anteriormente, hubo alguna que otra dificultad para desarrollar este modulo ya que Twitter acababa de cambiar la política sobre sus APIs, haciendo gran parte de sus opciones de pago.

El objetivo de buscar un término en Twitter se ha logrado, obteniendo bastante información sobre el usuario, su procedencia, número de retweets, etc., además se logró salvar una de las restricciones introducidas por Twitter que eran no obtener más de cierto número de tweets por búsqueda. Lo que no se ha podido lograr, y era una petición del cliente, era poder acotar una búsqueda por fechas o por geolocalización, ya que era una de las funcionalidades mencionadas anteriormente que habían pasado a ser de pago.

Se revisó la documentación de Twitter para ver qué posibilidades había de desarrollar una API propia, y se vio que la complejidad y sobre todo el tiempo que requería realizarla iban a ser excesivos, por lo que, teniendo en cuenta que ya se podían obtener cierto número de tweets de una manera más sencilla se decidió utilizar su versión gratuita.

2. Procesamiento, análisis y clasificación de los datos

Este modulo trata, analiza y clasifica los tweets almacenados en el módulo anterior.

El procesamiento realizado a los tweets no ha sido muy elevado. Los únicos valores eliminados han sido los enlaces externos porque se sabía de antemano que no iban a aportar ningún tipo de información adicional. También se pensó en eliminar conectores o preposiciones que no aportasen ningún valor, pero finalmente se desechó la idea porque

tampoco se sabía si eso sería correcto, se podría llegar a perder información valiosa en su eliminación. También se tuvo en cuenta la utilización del algoritmo TF-IDF porque con su utilización se penalizarían los términos muy repetidos que no aportan información.
No ha habido ningún tipo de análisis previo a la clasificación, no se han utilizado ningún filtro que facilitase la clasificación, ni que seleccionase los atributos que más información podrían llegar a aportar a la hora de generar el algoritmo clasificador.
En lo que respecta a la clasificación se han utilizado los parámetros por defecto de los clasificadores.
Debido a la falta de tiempo por problemas de diversa índole, no se ha potenciado desarrollado este módulo.
Una mejora para este módulo seria la utilización de filtros para ver su resultado a la hora de generar los modelos predictivos.
Otra mejora seria utilizar un barrido de parámetros por cada uno de los algoritmos de clasificación utilizados con la idea de obtener los parámetros óptimos y poder así realizar una clasificación más precisa.

3. Visualización de los datos

Este ha sido el modulo menos desarrollado de todos, la falta de tiempo hizo que sólo se visualizasen las palabras que más veces aparecían en los tweets que eran penalmente relevantes, por si alguna de estas palabras podría aportar algo de información.
La pagina creada para el proyecto también es algo simple, esto es algo que se podría mejorar para una visión más amigable para el usuario.

8.2 Revisión de la planificación temporal

Como se puede observar en la Tabla 5 - planificación temporal real, la planificación no se ha visto cumplida. La aparición de varios problemas, junto con la dificultad técnica del proyecto, ha hecho imposible ajustarse a lo planificado, incluso aun incumpliéndose varios de los objetivos del proyecto, lo que ha requerido 164 horas adicionales.

Tareas	Tiempo Planificado	Tiempo Real
1. Gestión	**48 horas**	**72 horas**
1.1. Reunión con los directores del proyecto	1 hora	1 hora
1.2. Reunión con el cliente	2 horas	2 horas
1.3. Definir objetivos del proyecto	20 horas	20 horas
1.4. Aprendizaje	25 horas	50 horas
2. Análisis	**42 horas**	**42 horas**
2.1. Captura de Requisitos	22 horas	22 horas
1. Casos de uso	10 horas	10 horas
2. Modelo de Dominio	12 horas	12 horas
2.2. Diagrama de secuencia	12 horas	12 horas
2.3. Planificación Temporal	6 horas	6 horas
3. ***Diseño e implementación***	**50 horas x 4 Sprint**	**78 horas x 4 Sprint**
3.1. Sprint	50 horas	78 horas
1. Sprint backlog	2 horas	2 horas
2. Diseño	12 horas	16 horas
3. Implementación	30 horas	50 horas
4. Documentación	6 horas	10 horas
4. Documentación	**36 horas**	**61 horas**
4.1. Redactar Memoria	30 horas	55 horas
4.2. Realizar documento de presentación	6 horas	6 horas
5. Cierre	**1 hora**	**1 hora**
5.1. Entrega en secretaria de la documentación	0,5 horas	0,5 horas
5.2. Presentación del proyecto	0,5 horas	0,5 horas
TOTAL	**324 horas**	**488 horas**

TABLA 5 - PLANIFICACIÓN TEMPORAL REAL

8.3 Gestión de Riesgos

La aparición de riegos es algo problemas es algo inevitable en un proyecto, pero muchas veces no se sabe la magnitud de los mismos hasta que suceden.
Uno de las situaciones que se tuvo en cuenta en la gestión de riegos, pero aun así a afectado en una medida muy gran al proyecto ha sido el cambio de la situación laboral del estudiante en dos ocasiones.
Esto supuso que el trabajo de fin de grado pasase a realizarse los fines de semana puesto que era imposible hacerlo entre semana debido al horario laboral.
También se hubo algún otro inconveniente, como el cambio de equipo, que supuso la pérdida de un día por la migración.
La utilización de APIs de terceros, junto con el cambio en las políticas de Twitter también retraso el proyecto, y dejo uno de los requisitos del cliente incumplidos, que era poder acotar en un margen temporal las búsquedas.

8.4 Trabajos futuros

Como se ha indicado antes, no se han llegado a cumplir todos los objetivos fijados para este proyecto.
La visualización de datos es un aspecto mejorable, por lo que en trabajos futuros se debería trabajar en este punto con el fin de obtener una visualización de los mismos de manera que el usuario pueda interpretar los datos de una manera lo más sencilla posible.
A su vez sería adecuado que a la hora de clasificar los tweets se hiciese un barrido de parámetros para obtener unos resultados óptimos en el momento de la clasificación.
Un problema inicial que tenía este proyecto era que los datos están desbalanceados. El numero de tweets penalmente no relevantes es mucho mayor que los que realmente son, lo que dificulta la clasificación. Esto se debería de tener en cuenta a la hora de realizar la clasificación porque un clasificador tendera a clasificar de manera más fácil los tweets penalmente no relevantes.
Hay que tener en cuenta que, al estar tratando con un proyecto innovador, pueden surgir nuevas soluciones para los problemas planteados, que habrá que tener en cuenta en el futuro.

Bibliografía y webgrafía

Arsys. (4 de Abril de 2012). *https://www.arsys.es/blog/programacion/myisam-o-innodb-elige-tu-motor-de-almacenamiento-mysql/*. Obtenido de Arsys: https://www.arsys.es/blog/programacion/myisam-o-innodb-elige-tu-motor-de-almacenamiento-mysql/

Betancourt, G. A. (2005). *LAS MÁQUINAS DE SOPORTE VECTORIAL .* Scientia et Technica .

Bohemian. (13 de octubre de 2013). *StackOverflow.* Obtenido de Removing link from Text in Java?: https://stackoverflow.com/questions/19345060/removing-link-from-text-in-java

Dau, A. (15 de Noviembre de 2017). *Twitter presentó las API Premium.* Obtenido de tecnogaming: https://www.tecnogaming.com/2017/11/twitter-presento-las-api-premium/

Delgado, D. O. (15 de Septiembre de 2017). *¿Qué es Tensorflow?* Obtenido de openwebinars.net: https://openwebinars.net/blog/que-es-tensorflow/

Dignan, L. (14 de Febrero de 2018). *zdnet*. Obtenido de Top cloud providers 2018: How AWS, Microsoft, Google Cloud Platform, IBM Cloud, Oracle, Alibaba stack up: https://www.zdnet.com/article/cloud-providers-ranking-2018-how-aws-microsoft-google-cloud-platform-ibm-cloud-oracle-alibaba-stack/

EFE. (7 de Noviembre de 2017). *Agencia EFE.* Obtenido de Twitter amplía a todos sus usuarios el límite de 280 caracteres: https://www.efe.com/efe/espana/economia/twitter-amplia-a-todos-sus-usuarios-el-limite-de-280-caracteres/10003-3431617

Gonzalez, A. (30 de Julio de 2014). *Conceptos básicos de Machine Learning.* Obtenido de cleverdata: https://cleverdata.io/conceptos-basicos-machine-learning/

Kalil, S. (13 de Marzo de 2018). *Entre trinos y 'retuits' los tuiteros celebraron su día*. Obtenido de El heraldo: https://www.elheraldo.co/entretenimiento/entre-trinos-y-retuits-los-tuiteros-celebraron-su-dia-469922

Kim, B. (26 de Junio de 2016). *Some Thoughts On Shiny Open Source: Render Multiple Pages*. Obtenido de r-bloggers: https://www.r-bloggers.com/some-thoughts-on-shiny-open-source-render-multiple-pages/

lalopg. (17 de abril de 2015). *slideshares.* Obtenido de Métodos predictivos y Descriptivos: https://es.slideshare.net/lalopg/mtodos-predictivos-y-descriptivos-minera-de-datos

Ley Orgánica 10/1995, de 23 de noviembre, De las circunstancias que agravan la responsabilidad criminal. (23 de noviembre de 1995). Obtenido de Boletín Oficial del Estado, núm. 281: https://www.boe.es/buscar/pdf/1995/BOE-A-1995-25444-consolidado.pdf

openalfa. (12 de Mayo de 2013). *Diferencias entre InnoDB y MyISAM en MySQL*. Obtenido de openalfa: https://blog.openalfa.com/diferencias-entre-innodb-y-myisam-en-mysql

Pérez, J. (16 de Febrero de 2018). *El Supremo condena a dos años y medio a un tuitero por incitar al odio contra las mujeres asesinadas por violencia machista*. Obtenido de

Publico: http://www.publico.es/sociedad/violencia-machista-supremo-condena-anos-medio-tuitero-incitar-odio-mujeres-asesinadas-violencia-machista.html

Ramirez, A. P. (2017). *Práctica 4: Text Mining: Sentiment Analysis, SPAM classification.* Bilbao.

Rochina, P. (16 de Noviembre de 2016). *Python vs R para el análisis de datos*. Obtenido de revistadigital: https://revistadigital.inesem.es/informatica-y-tics/python-r-analisis-datos/

Rodríguez Suárez, Y., & Díaz Amador, A. (11 de Junio de 2009). *Revista Cubana de Ciencias Informáticas.* Obtenido de Revista Cubana de Ciencias Informáticas: http://www.redalyc.org/pdf/3783/378343637009.pdf

Salnikov-Tarnovski, N. (23 de Mayo de 2017). *plumbr*. Obtenido de Most popular Java application servers: 2017 edition: https://plumbr.io/blog/java/most-popular-java-application-servers-2017-edition

Valero, A. T. (2005). *Extracción de Información con algoritmos de clasificacion.* Tonantzintla, Pue: Inade.

Varios, a. (20 de Junio de 2018). *Mysql*. Obtenido de Limits on InnoDB Tables: https://dev.mysql.com/doc/refman/5.5/en/innodb-restrictions.html

Vidales, Y. R. (17 de Marzo de 2017). *Un tuit sobre la tragedia de Germanwings motiva la primera condena por catalanofobia*. Obtenido de Confilegal: https://confilegal.com/20170317-un-tuit-sobre-la-tragedia-de-germanwings-motiva-la-primera-condena-por-catalanofobia/

ANEXOS I- CASOS DE USO EXTENDIDOS

Nombre: Seleccionar proyecto

Descripción: El usuario selecciona uno de los proyectos ya creados para cargar todos los tweets buscados en él.

Actores: Administrador.

Precondiciones: Debe de haber algún proyecto creado previamente.

Requisitos no funcionales: Conexión a internet

Flujo de eventos:

1. El usuario se encuentra con el desplegable que se muestra en la Figura 24 - selección de proyecto donde se muestran todos los proyectos disponibles. Debe seleccionar uno de los proyectos, por defecto está el primero creado.
2. Una vez seleccionado el proyecto debe pulsar sobre el botón de "Ir".
3. El sistema le redirigirá a una nueva página donde se muestran todos los tweets buscados y sin etiquetar.

Postcondiciones: El usuario visualiza los tweets previamente buscados.

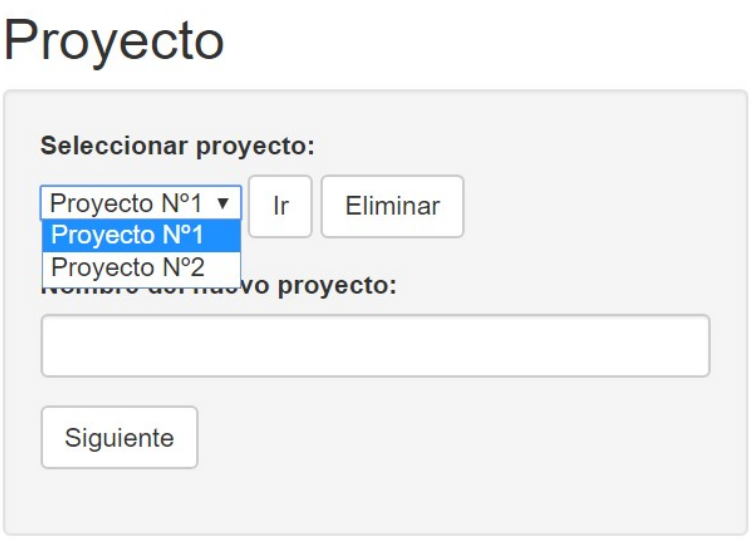
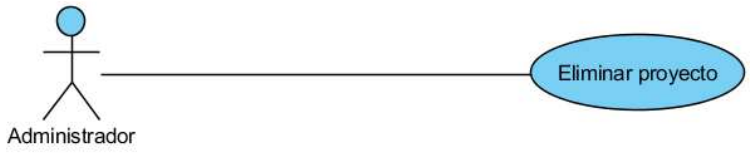


FIGURA 24 - SELECCIÓN DE PROYECTO

Nombre: Eliminar proyecto



Descripción: El usuario elimina uno de los proyectos existentes.

Actores: Administrador.

Precondiciones: Debe de haber algún proyecto creado previamente.

Requisitos no funcionales: Conexión a internet.

Flujo de eventos:

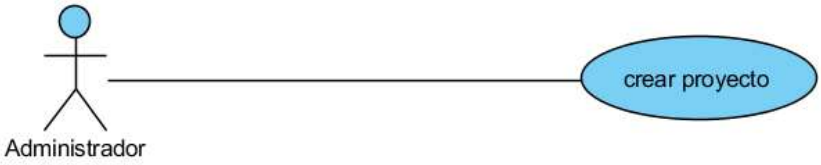
1. El usuario se encuentra con el desplegable que se muestra en la Figura 25 - eliminar proyecto 1 donde se muestran todos los proyectos disponibles. Debe seleccionar uno de los proyectos, por defecto está el primero creado.
2. El usuario pulsa sobre el botón de eliminar.
3. El usuario deja de ver el nombre del proyecto en el desplegable (Figura 26 - eliminar proyecto 2).

Postcondiciones: Se deja de ver el proyecto eliminado.

FIGURA 25 - ELIMINAR PROYECTO 1

FIGURA 26 - ELIMINAR PROYECTO 2

Nombre: Crear proyecto



Descripción: El usuario genera un nuevo proyecto.

Actores: Administrador.

Precondiciones: ninguna.

Requisitos no funcionales: Conexión a internet.

Flujo de eventos:

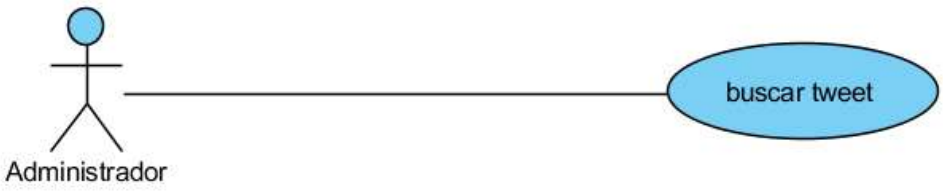
1. El usuario introduce un nombre al proyecto en el recuadro que indica "Nombre del nuevo proyecto" como se muestra en la Figura 27 - crear proyecto 1.
2. El usuario pulsa sobre el botón "Siguiente", donde le aparecerá un nuevo recuadro llamado "Busqueda de twitter".
3. En el nuevo recuadro mostrado introduce el término a buscar en Twitter como podemos ver en la Figura 28 - crear proyecto 2.
4. El sistema redirigirá al usuario a la página donde aparecen los tweets buscados por el sistema.

Postcondiciones: Se genera un nuevo proyecto con los tweets asociados.

FIGURA 27 - CREAR PROYECTO 1

FIGURA 28 - CREAR PROYECTO 2

Nombre: Buscar tweet



Descripción: El usuario selecciona uno de los proyectos ya creados para cargar todos los tweets buscados.

Actores: Administrador.

Precondiciones: Tener creado un proyecto.

Requisitos no funcionales: Conexión a internet.

Flujo de eventos:

1. El usuario pulsa sobre el botón de "Generar mas tweets" como se ve en la Figura 29 - generar más tweets.
2. Cuando la aplicación acabe de buscar, recargara la página con los nuevos tweets encontrados.

Postcondiciones: Ninguna

Tweets Sin Clasificar

Numero de Tweets: 59

clasificar | ver clasificados | Generar mas tweets

tweet	retweet	usuario	Geolocalización	creador	¿Relevante?	Tipo de Delito	Tipo de Delito	Tipo de Delito	Enviar
La primera decisión de @Sorayapp ha sido contar con @MariMarBlanco_ y con las víctimas del terrorismo, algo que muestra la importancia y el principio básico que representa todo lo que dimos en defensa de la libertad. #UnidosganaelPP @DebatAlRojoVivo https//t.co/TLW9Azo6BE	47	cespedeshuelva	Huelva, Andalucía	AlfonsoAlonsoPP	◉No ○Si	☐Promover hostilidad ☐Poseer material que promueve la hostilidad ☐Negar los delitos de Derecho Penal Internacional	☐Vejar a grupos sociales ☐Enaltecer delitos ☐Justificar delitos	☐Enaltecer el terrorismo ☐Justificar delitos del terrorismo ☐Vejar victimas del terrorismo ☐Delito de propaganda	etiquetar
@reyesdeeuropa85 @Garethito1 @realmadrid Si! Si es amargo al principio, el final debe de ser dulce!	1	Garethito1		IsherwoodBy	◉No ○Si	☐Promover hostilidad ☐Poseer material que promueve la hostilidad ☐Negar los delitos de Derecho Penal Internacional	☐Vejar a grupos sociales ☐Enaltecer delitos ☐Justificar delitos	☐Enaltecer el terrorismo ☐Justificar delitos del terrorismo ☐Vejar victimas del terrorismo ☐Delito de propaganda	etiquetar

FIGURA 29 - GENERAR MÁS TWEETS

Nombre: Etiquetar tweet

Descripción: El usuario etiqueta los tweets disponibles.

Actores: Administrador.

Precondiciones: Tener creado un proyecto.

Requisitos no funcionales: Conexión a internet.

Flujo de eventos:

1. El usuario selecciona las opciones de etiquetado disponibles para uno de los tweets como se ve en la Figura 30 - etiquetar tweet 1.
2. Una vez seleccionado pulsa sobre el botón etiquetar.
3. El tweet etiquetado desaparece de la lista, como se ve en la Figura 31 - etiquetar tweet 2

Postcondiciones: El tweet etiquetado desaparece de la lista.

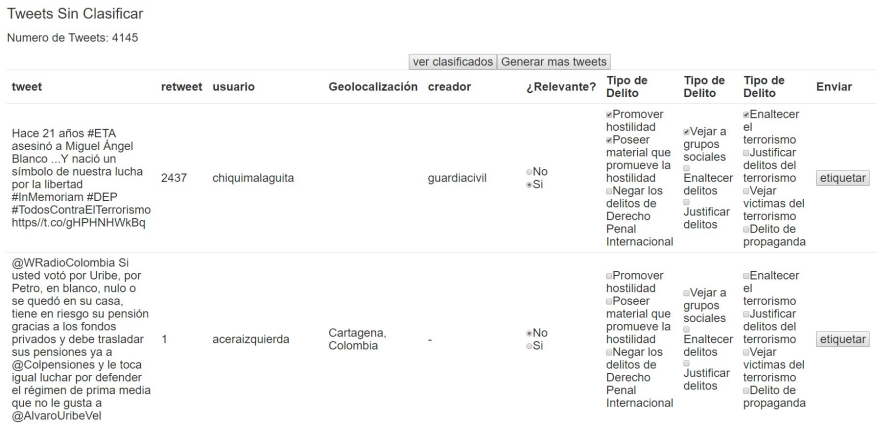


FIGURA 30 - ETIQUETAR TWEET 1

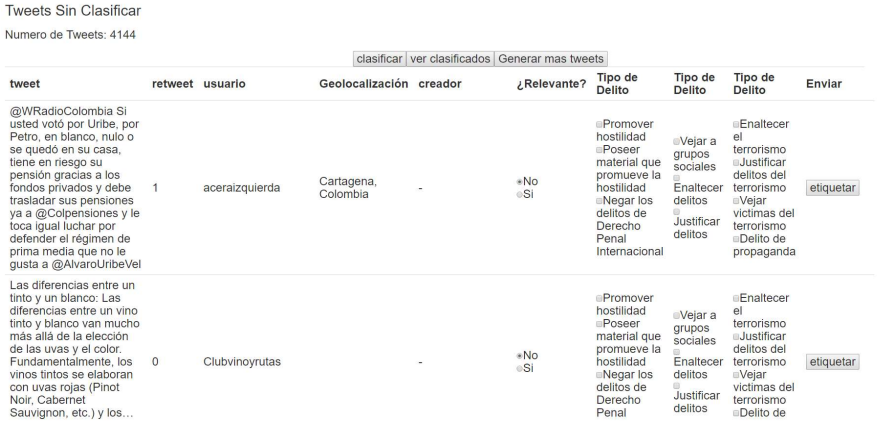
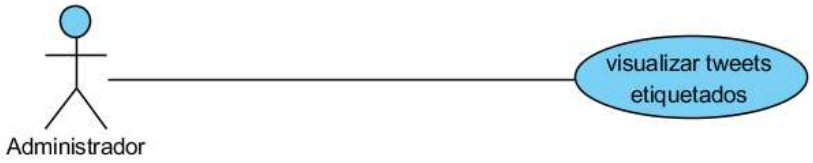


FIGURA 31 - ETIQUETAR TWEET 2

Nombre: visualizar tweets etiquetados



Descripción: El usuario visualiza los tweets previamente etiquetados.

Actores: Administrador.

Precondiciones: Tener tweets previamente etiquetados.

Requisitos no funcionales: Conexión a internet.

Flujo de eventos:

1. El usuario pulsa sobre el botón "ver clasificados". Figura 32 - ver clasificados.
2. El sistema redirige al usuario a la página donde se muestran los tweets etiquetados o validados previamente. Figura 33 - tweets clasificados.

Postcondiciones: Ninguna

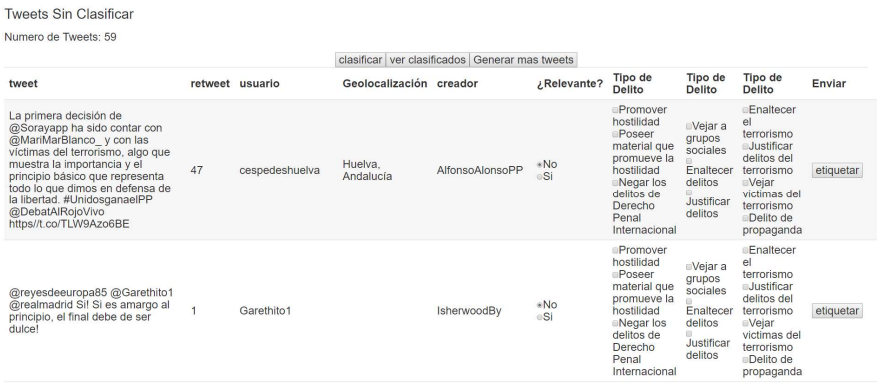


FIGURA 32 - VER CLASIFICADOS

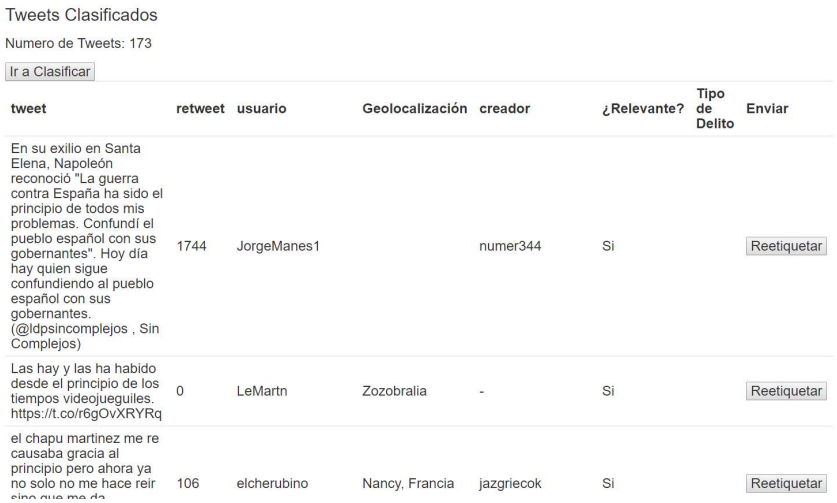


FIGURA 33 - TWEETS CLASIFICADOS

Nombre: Reetiquetar tweets

Descripción: El usuario elimina un tweet.

Actores: Administrador.

Precondiciones: Tener tweets etiquetados.

Requisitos no funcionales: Conexión a internet

Flujo de eventos:

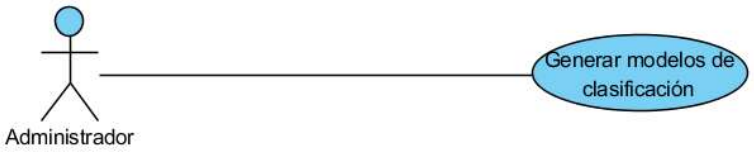
1. El usuario pulsa sobre el botón "Reetiquetar". Figura 34 - reetiquetar 1.
2. El tweet desaparece de la lista. Figura 35 - reetiquetar 2.

Postcondiciones: El tweet seleccionado aparecerá en la lista de tweets sin clasificar

FIGURA 34 - REETIQUETAR 1

FIGURA 35 - REETIQUETAR 2

Nombre: Generar modelos de clasificación



Descripción: Se generan los modelos con los que se clasifican los tweets.

Actores: Administrador.

Precondiciones: Tener tweets etiquetados.

Requisitos no funcionales: Conexión a internet.

Flujo de eventos:

1. El usuario pulsa sobre el botón "clasificar". Figura 36 – clasificar.
2. El sistema redirige al usuario donde ve las figuras de mérito de los modelos. Figura 37 - figuras de mérito.

Postcondiciones: Quedan guardados los modelos de clasificacion.

FIGURA 36 – CLASIFICAR

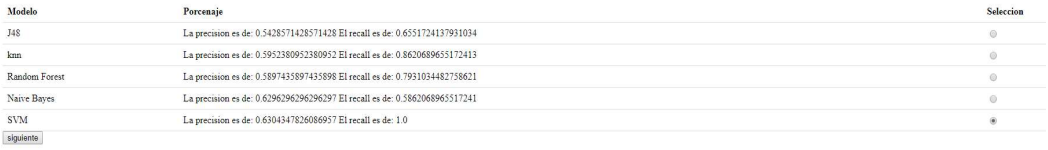
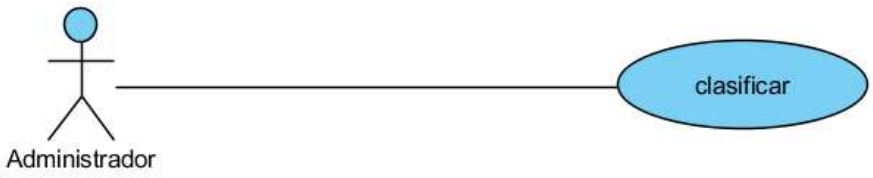


FIGURA 37 - FIGURAS DE MÉRITO

Nombre: Clasificar



Descripción: El usuario selecciona uno de los modelos disponibles para que clasifique los tweets no etiquetados.

Actores: Administrador.

Precondiciones: Tener generados los modelos de clasificación.

Requisitos no funcionales: Conexión a internet

Flujo de eventos:

1. El usuario selecciona uno de los modelos disponibles. Figura 38 - figuras de mérito.
2. El usuario pulsa sobre el botón "Siguiente".
3. El sistema redirige al usuario a una pantalla donde visualiza la clasificación hecha por el modelo. Figura 39 -validacion.

Postcondiciones: Se generan tweets clasificados pendientes de validar.

Modelo	Porcenaje	Seleccion
J48	La precision es de: 0.5428571428571428 El recall es de: 0.6551724137931034	
knn	La precision es de: 0.5952380952380952 El recall es de: 0.8620689655172413	
Random Forest	La precision es de: 0.5897435897435898 El recall es de: 0.7931034482758621	
Naive Bayes	La precision es de: 0.6296296296296297 El recall es de: 0.5862068965517241	
SVM	La precision es de: 0.6304347826086957 El recall es de: 1.0	

siguiente

FIGURA 38 - FIGURAS DE MÉRITO

Tweets Clasificados

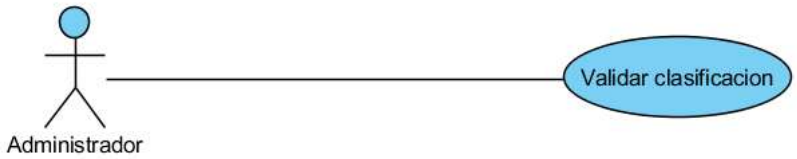
Numero de Tweets: 63

Ir a Clasificar

tweet	retweet	usuario	Geolocalización	creador	¿Relevante?	Tipo de Delito	Enviar
dejar una relación tóxica siemmmpre va a valer la pena, aunque al principio no lo veas	440	sofiifortete	Neuquén, Argentina	cynthiagarcc	Si		Validar
dejar una relación tóxica siemmmpre va a valer la pena, aunque al principio no lo veas	438	AnagMorrison		cynthiagarcc	Si		Validar
dejar una relación tóxica siemmmpre va a valer la pena, aunque al principio no lo veas	438	mililuu02	Asuncion, Paraguay	cynthiagarcc	Si		Validar
dejar una relación tóxica siemmmpre va a valer la pena, aunque al principio no lo veas	441	unaforramass_	Argentina	cynthiagarcc	Si		Validar
dejar una relación tóxica siemmmpre va a valer la pena, aunque al principio no lo veas	438	inesmoreno4	Salamanca, España	cynthiagarcc	Si		Validar

FIGURA 39 -VALIDACION

Nombre: Validar clasificación



Descripción: El usuario valida.

Actores: Administrador.

Precondiciones: Haber clasificado tweets.

Requisitos no funcionales: Conexión a internet.

Flujo de eventos:

1. El usuario pulsa sobre el botón "Validar" en caso de que este de acuerdo con la clasificación del tweet. Figura 40 - validar clasificación.

Postcondiciones: Guarda el tweet validado con la clasificación generada por el usuario.

Tweets Clasificados

Numero de Tweets: 63

Ir a Clasificar

tweet	retweet	usuario	Geolocalización	creador	¿Relevante?	Tipo de Delito	Enviar
dejar una relación tóxica siemmmpre va a valer la pena, aunque al principio no lo veas	440	sofiifortete	Neuquén, Argentina	cynthiagarcc	Si		Validar
dejar una relación tóxica siemmmpre va a valer la pena, aunque al principio no lo veas	438	AnagMorrison		cynthiagarcc	Si		Validar
dejar una relación tóxica siemmmpre va a valer la pena, aunque al principio no lo veas	438	mililuu02	Asuncion, Paraguay	cynthiagarcc	Si		Validar
dejar una relación tóxica siemmmpre va a valer la pena, aunque al principio no lo veas	441	unaforramass_	Argentina	cynthiagarcc	Si		Validar
dejar una relación tóxica siemmmpre va a valer la pena, aunque al principio no lo veas	438	inesmoreno4	Salamanca, España	cynthiagarcc	Si		Validar

FIGURA 40 - VALIDAR CLASIFICACIÓN

Nombre: ver estadísticas

Descripción: Genera u

Actores: Administrador

Precondiciones: Tener algún tweet clasificado como relevante

Requisitos no funcionales: Conexión a internet.

Flujo de eventos:

1. El usuario pulsa el botón de “” ver estadisticas”.
2. Se redirige al usuario a la pagina donde se le muestran las palabras más repetidas.

Postcondiciones:

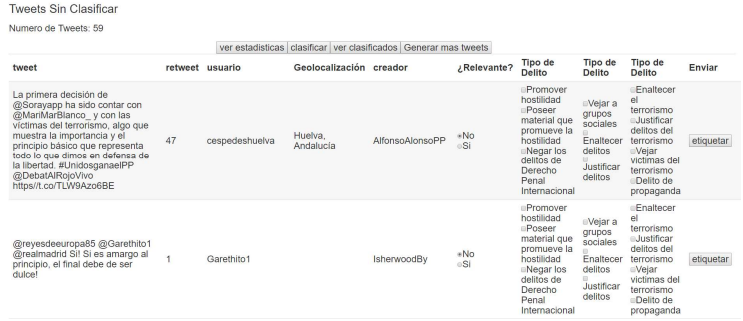


FIGURA 41 - VER ESTADISTICAS

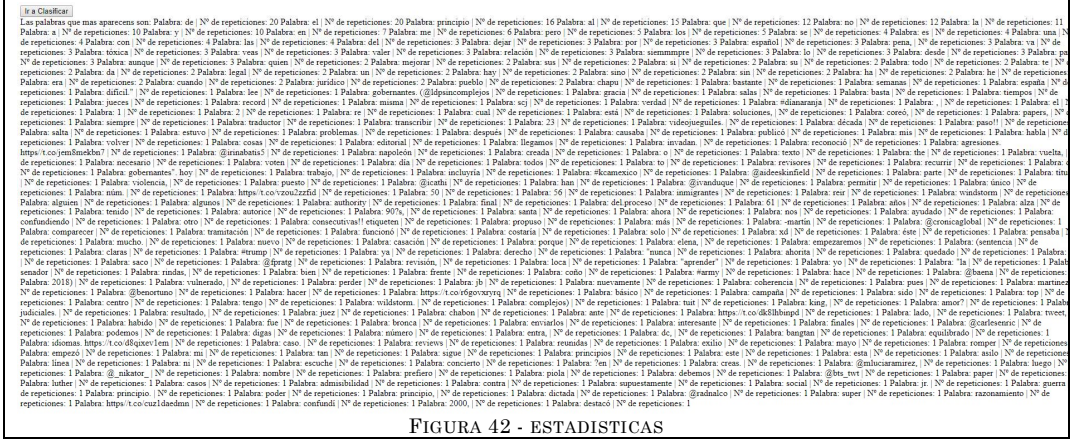


FIGURA 42 - ESTADISTICAS

Nombre: Identificarse

Descripción: El usuario

Actores: Usuario_Anónimo

Precondiciones: Ninguna

Requisitos no funcionales: Conexión a internet

Flujo de eventos:

1. El usuario anónimo desde la página de login, introduce un usuario y una contraseña Figura 43 - login:
    a. Si el usuario no es válido muestra una pantalla de error Figura 44 - error
2. Se redirige a la pantalla principal de la aplicación Figura 45 - página principal

Postcondiciones: Ninguna

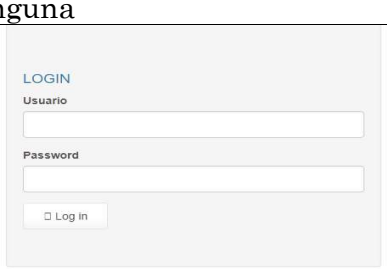


FIGURA 43 - LOGIN

FIGURA 44 - ERROR

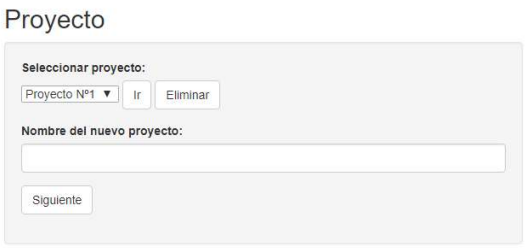
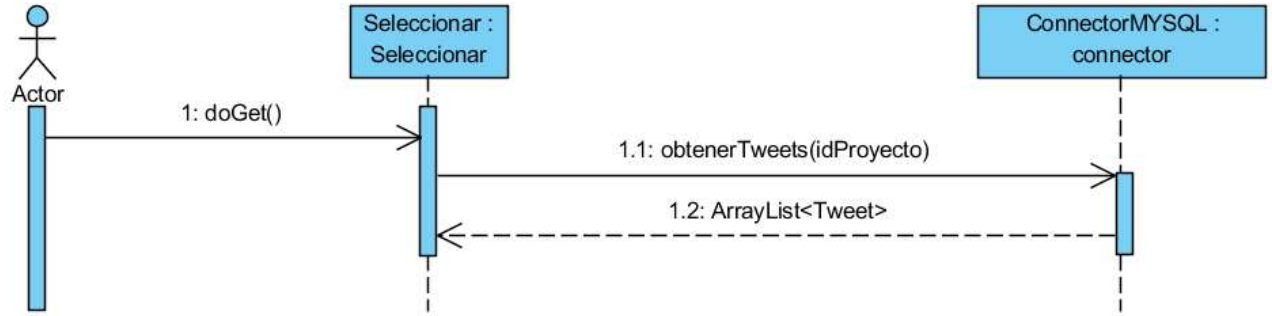


FIGURA 45 - PÁGINA PRINCIPAL

ANEXOS II- DIAGRAMAS DE SECUENCIA

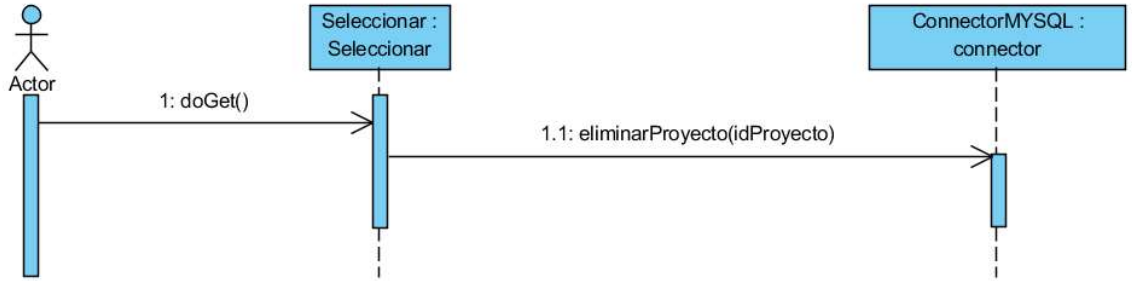
Seleccionar proyecto



- ObtenerTweets(idProyecto):

SELECT * FROM `tweet` INNER JOIN `esta` ON `tweet`.`idTweet` = `esta`.`idTweet` WHERE `esta`.`idProyecto`=IdProyecto AND `tweet`.`relevante` IS NULL

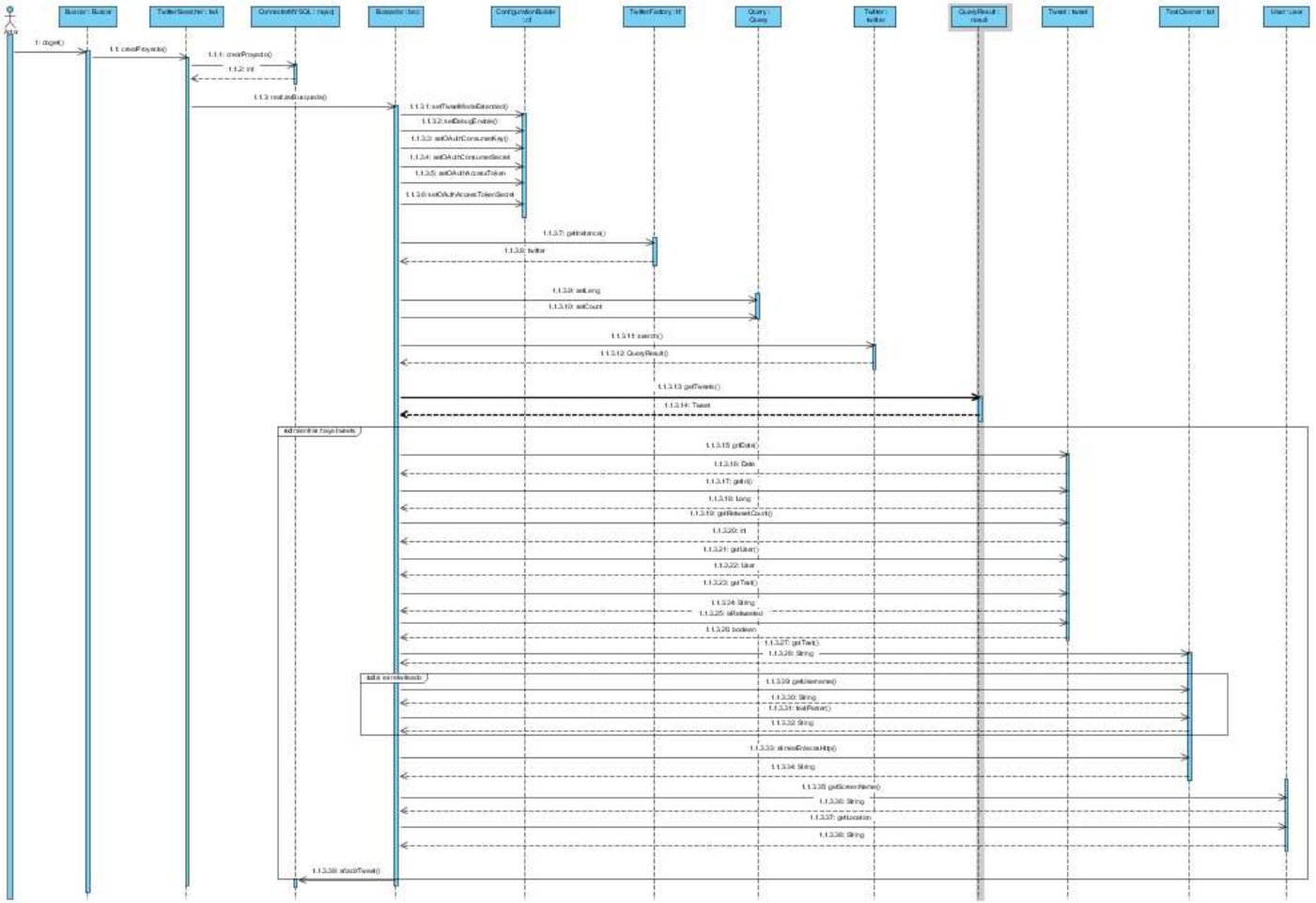
Eliminar proyecto



- EliminarProyecto(idProyecto):

UPDATE `proyecto` SET `eliminado` = '1' WHERE `proyecto`.`idProyecto` = idProyecto

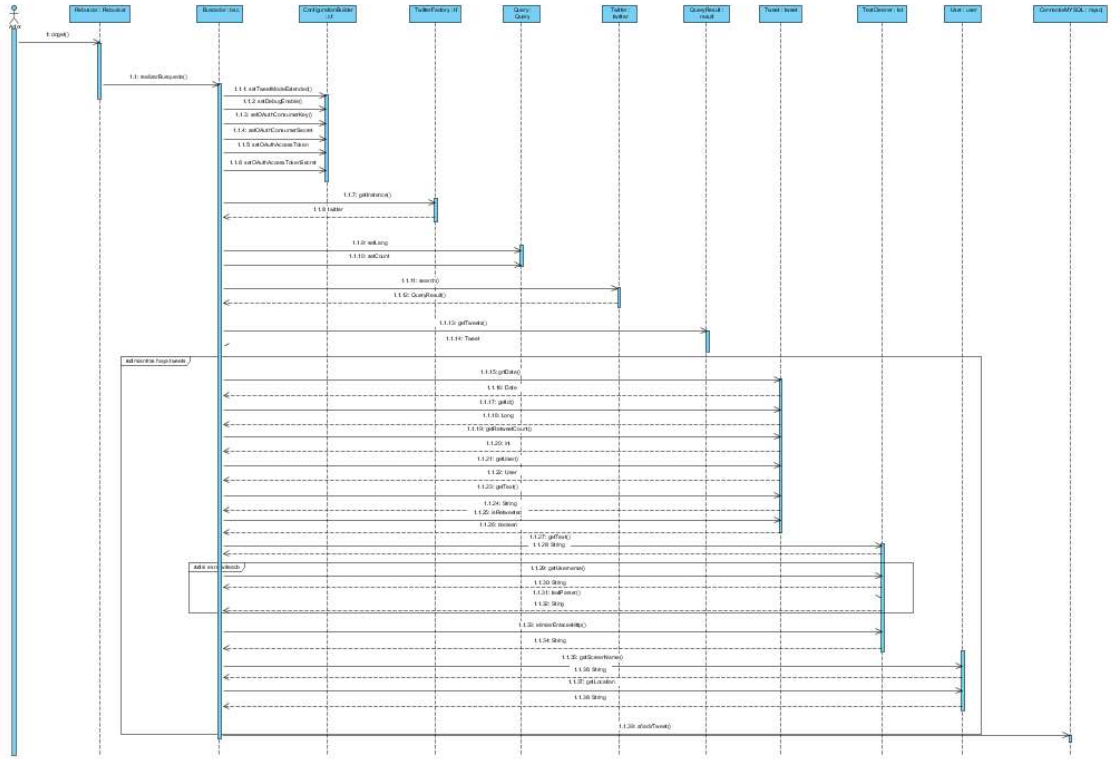
Crear proyecto



- **<u>Crearproyecto(pNombreProyecto, Creador, Buscado):</u>**
  "INSERT INTO proyecto (nombreproyecto, fechacreacion, creador, buscado,eliminado) VALUES ('"+ pNombreProyecto + "','" + date + "','" + Creador + "','" + Buscado + "','0');"

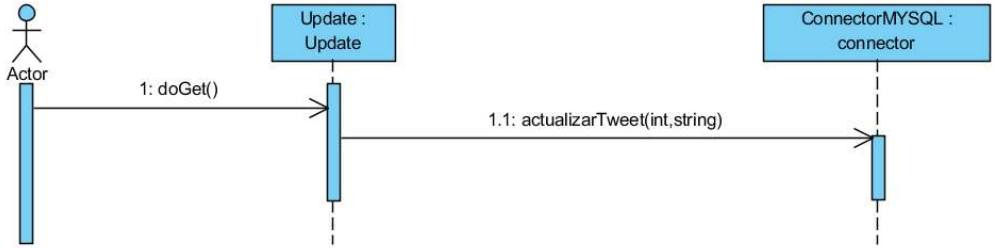
- **<u>AñadirTweet</u>**(pIdTweet, pTweet, pTweetfiltrado, pFecha,pRetweet,pUsuario,pGeolocalizacion, pCreador)**<u>:</u>**
  INSERT INTO tweet (idTweet, tweet, tweetfiltrado, fecha,retweet,usuario,geolocalizacion, creador) VALUES "+ "(" + pIdTweet + ", '" + pTweet + "', '" + pTweetFiltrado + "', '" + pFecha + "', '" + pRetweet+ "', '" + pUsername + "', '" + pGeolocalizacion + "', '" + pCreador + "')

Buscar tweet



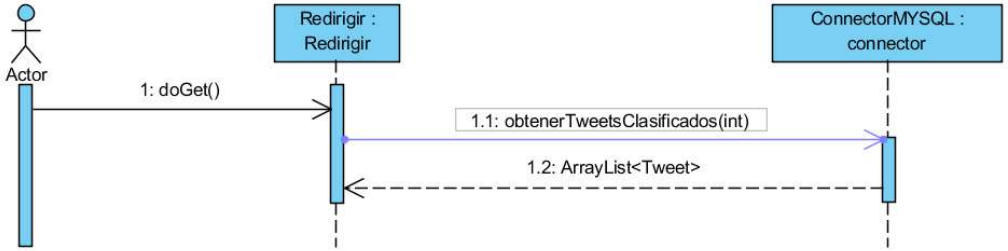
- **<u>AñadirTweet</u>**(pIdTweet, pTweet, pTweetfiltrado, pFecha,pRetweet,pUsuario,pGeolocalizacion, pCreador)**<u>:</u>**
  INSERT INTO tweet (idTweet, tweet, tweetfiltrado, fecha,retweet,usuario,geolocalizacion, creador) VALUES "+ "(" + pIdTweet + ", '" + pTweet + "', '" + pTweetFiltrado + "', '" + pFecha + "', '" + pRetweet+ "', '" + pUsername + "', '" + pGeolocalizacion + "', '" + pCreador + "')

Etiquetar tweet



- actualizarTweet(pIdTweet, pCadena):
  UPDATE `tweet` SET `relevante` = '' + pCadena + '' WHERE `tweet`.`idtweet` = '' + pIdTweet

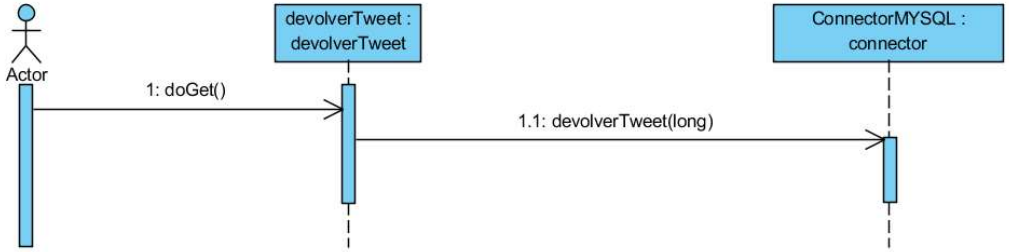
visualizar tweets etiquetados



- **obtenerTweetsClasificados(pIdProyecto):**

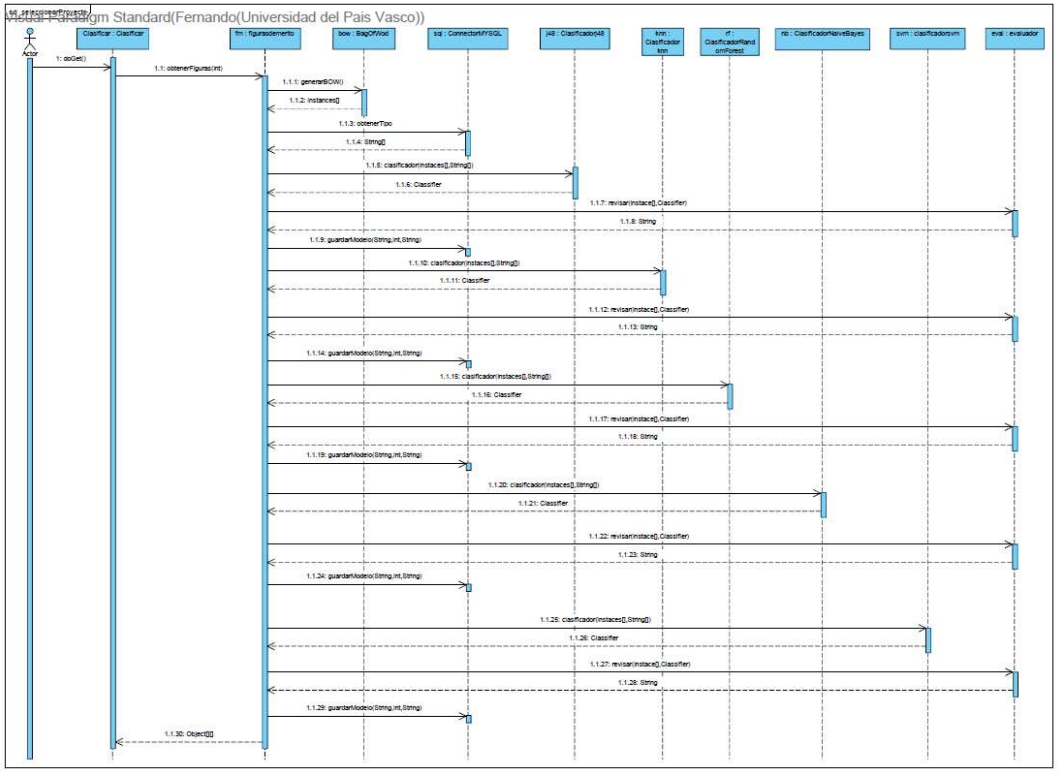
```
SELECT * FROM `tweet` INNER JOIN `esta` ON `tweet`.`idTweet` = `esta`.`idTweet` WHERE `esta`.`idProyecto`="
+ pIdProyecto + " AND `tweet`.`relevante` IS NOT NULL
```

Reetiquetar tweets



- **<u>devolverTweet(pIdTweet):</u>**
  UPDATE `tweet` SET `relevante` = NULL WHERE `tweet`.`idTweet` =" +pIdTweet+ ";

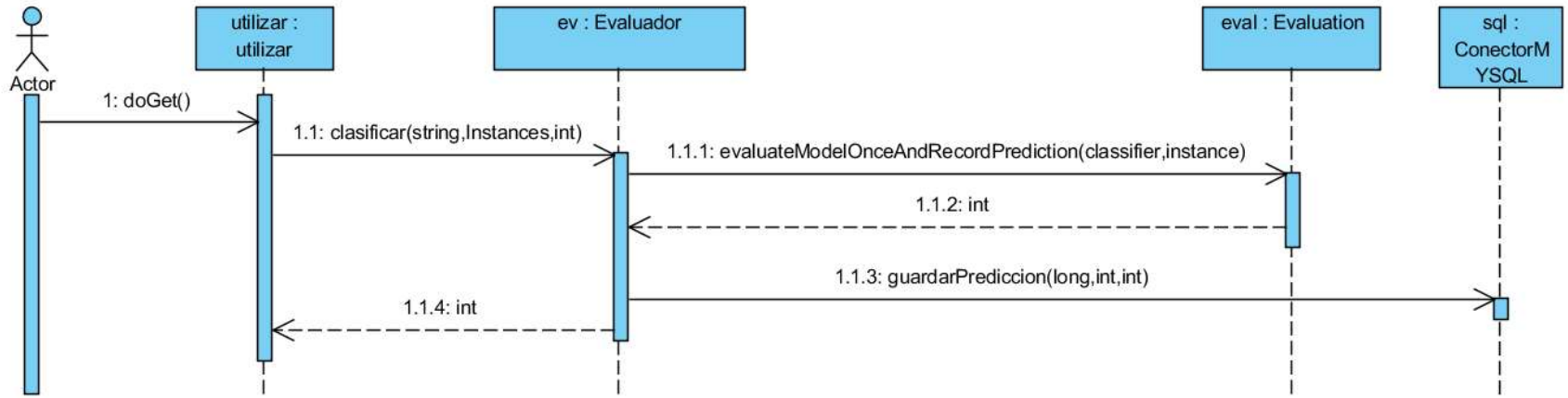
Generar modelos de clasificación



- **<u>ObtenerTipo():</u>**
  SELECT `idTipo`,`argumentos` FROM `tipo`

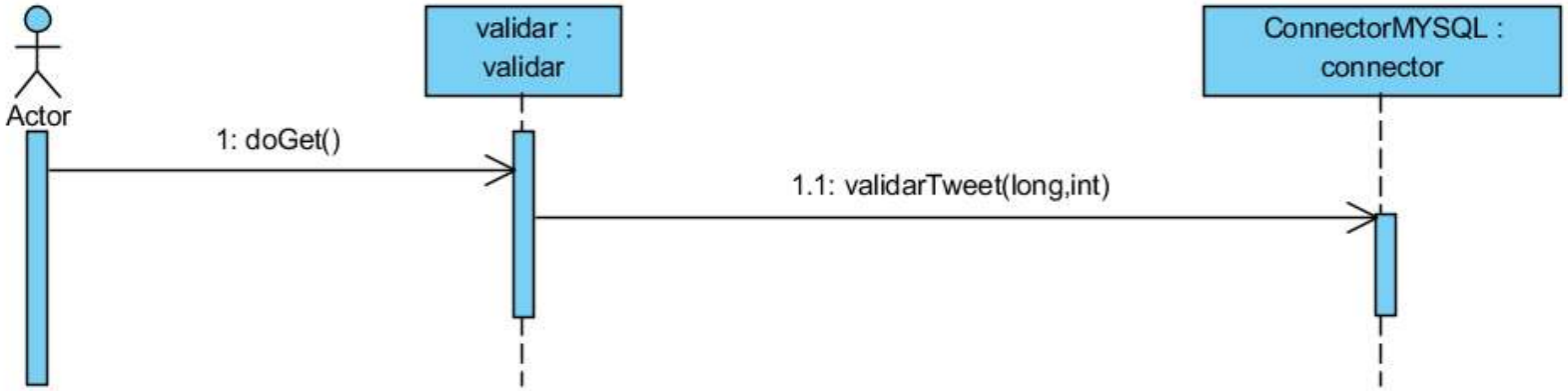
- **<u>Guardarmodelo(pIdBOW, pIdtipo, resultado):</u>**
  INSERT INTO Modelo (idBOW,idTipo,resultados) VALUES ('" + pIdBOW + "','" + pIdtipo + "','" + resultado + "');

Clasificar



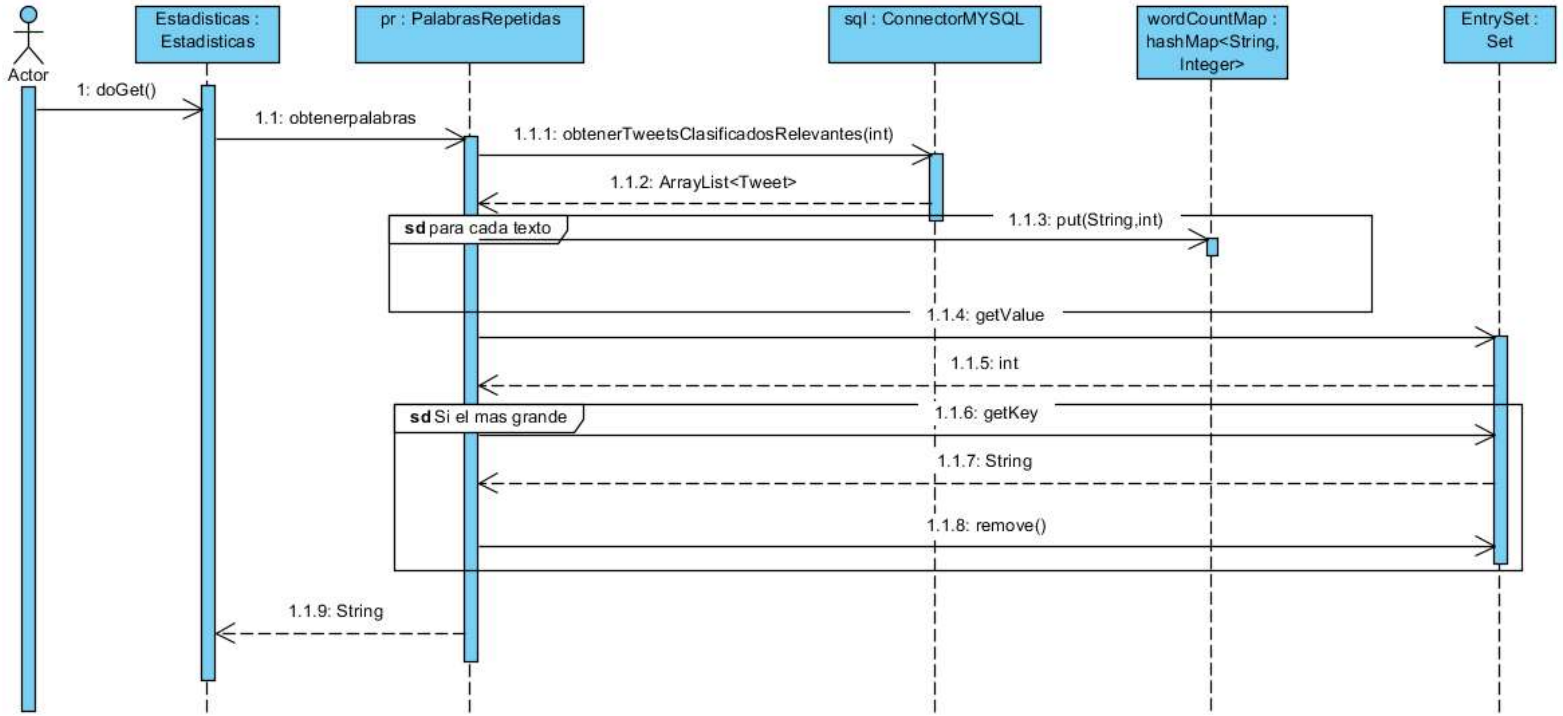
- **guardarPrediccion(pModelo,pidTweet,pRelevante):**
  INSERT INTO Clasifica (idModelo, idTweet,relevante) VALUES ('" pModelo + "', '" + pidTweet + "', '" + pRelevante + "')

Validar clasificación



- **validarTweet(long,string):**
  UPDATE `tweet` SET `relevante` = " + pCadena + " WHERE `tweet`.`idtweet` = " + pIdTweet

ver estadísticas



obtenerTweetsClasificadosRelevantes():

SELECT * FROM `tweet` INNER JOIN `esta` ON `tweet`.`idTweet` = `esta`.`idTweet` WHERE `esta`.`idProyecto`="+ pIdProyecto + " AND `tweet`.`relevante` = 1