UNIVERSIDADE DA CORUÑA

Facultad de Informática

Trabajo de fin de grado
de Ingeniería Informática

DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN WEB PARA LA RECOPILACIÓN, CLASIFICACIÓN Y ENLACE DE NOTICIAS SOBRE FUENTES DE DATOS SEMIESTRUCTURADAS.

Autor/a: SERGIO CANCELO RAMA
Director/a: FERNANDO BELLAS PERMUY
Director/a: JAVIER PARAPAR LÓPEZ

A Coruña, a 18 de junio de 2014.

Dedicado muy especialmente a mis padres y a mi novia por ese apoyo incondicional que siempre he tenido.

Agradecimientos

La ayuda proporcionada por D. Fernando Bellas Permuy y D. Javier Parapar López han sido clave para que la realización de este proyecto fuera posible. A ellos va dirigido este agradecimiento.

Resumen

El proyecto consiste en el desarrollo de una aplicación web para la visualización de noticias de interés para el usuario. Para esto es necesario, por una parte, la realización de un proceso encargado de recopilar contenido de fuentes RSS y almacenarlas en base de datos. Por otro lado, un proceso encargado de clasificar en categorías el contenido recopilado por el proceso anterior. Finalmente, la realización de una aplicación web que consuma el contenido generado, y lo estructure de forma que el usuario pueda buscar contenido, acceder a contenido actual y visualizar su contenido de forma personalizada y en función de sus intereses.

Palabras clave:

- √ Aplicación web.
- √ Ruby On Rails.
- √ Skel.js
- √ Wrapper.
- √ Inteligencia artificial.
- √ Bayes-Naïve.
- √ Support Vector Machine.

Índice general

Página

Índice de figuras

Figura **Página**

Capítulo 1
Introducción

Índice general

1.1. Contextualización

El mundo de internet ha evolucionado mucho durante los últimos años. Casi la totalidad de los medios de comunicación tienen acceso a su contenido mediante la web. La revolución de la telefonía móvil actual ha sido clave para incrementar el acceso al contenido. Los usuarios que consumen este contenido pueden ser víctimas de contenido incompleto, escueto, o de baja calidad. Como consecuencia, los usuarios se ven obligados a visitar varias fuentes distintas para leer el mismo contenido desde otra perspectiva, o simplemente para leer nuevo contenido que la anterior no contenía.

El proyecto surge como consecuencia del problema formulado en el párrafo anterior. La aplicación web resultante, será encargada de rastrear fuentes, clasificar el contenido, y mostrárselo al usuario de forma más ordenada, para que, desde una única aplicación, pueda estar informado de la actualidad.

1.2. Objetivos

El objetivo principal del proyecto es crear un enlace inteligente a noticias proporcionando al usuario una forma directa de visualizar el contenido de interés de forma estructurada, ordenada y según sus prioridades.

Desde el punto de vista del usuario, la aplicación ofrecerá las siguientes características:

- Una página principal donde se mostrarán las noticias más relevantes, que se podrán puntuar, visitar o compartir en la distintas redes sociales.
- La funcionalidad de puntuar necesita que el usuario se haya autenticado previamente, por tanto, será necesario la creación de un sistema de login y autenticación de usuarios.
- Una vez autenticado, el usuario tiene su página principal personalizada en función de sus intereses. Para mostrar el grado de interés que tiene el usuario sobre una noticia se creará un sistema de puntuación de noticias en base a distintos parámetros así como las preferencias que muestre el usuario.(Ej: el sistema puntuaría positivamente una noticia valorando las visitas, el número de veces que se ha compartido en las redes sociales y la puntuación que ha ofrecido el usuario sobre ese contenido).
- Las noticias se estructurarán en categorías y el usuario podrá personalizar el grado de interés que muestra por cada una de las categorías, eligiendo las que desea visualizar.
- En algunas ocasiones, se puede producir el caso de que el usuario visualiza contenido del que no está interesado, o solo lo está puntualmente, por lo tanto, la aplicación deberá permitir tener acceso a la totalidad del contenido de la web filtrado en base a parámetros.(Ej: un usuario tendría la opción de filtrar noticias de los últimos 3 días de la categoría deportes).
- Si el usuario necesita buscar sobre un tema en concreto de la actualidad, se facilitará un buscador de noticias.

Internamente, la aplicación ofrecerá las siguientes características:

- Un proceso que será encargado de acceder a fuentes de datos y recopilar la información necesaria sobre noticias, procesarlas y almacenarlas.
- Como cada fuente de datos tendrá su forma propia forma de estructurar el contenido, una vez extraída la información de la noticia se usará un algoritmo para clasificar automáticamente las noticias en categorías de forma homogénea.

1.3. Visión global del proyecto

La figura de la página siguiente (Figura 1.1) muestra una visión general en la que se estructura el proyecto, como ya se ha comentado en la apartado anterior, consta de una aplicación web que se encarga de interactuar con el usuario y mostrarle la información de manera ordenada. Por otro lado tenemos el wrapper, encargado de conectarse a fuentes de datos RSS y extraer su contenido, el cual, finalmente será clasificado en alguna de la categorías antes de ser insertado en la base de datos.

1.4. Estructura de la memoria

La memoria consta de los siguientes capítulos que ayudarán a comprender los diferentes aspectos y detalles del proyecto con mayor profundidad:

- **Introducción:** Permite obtener una visión global tanto de los aspectos generales del proyecto como de la memoria.
- **Fundamentos tecnológicos:** Se realiza una introducción a las diferentes tecnologías empleadas en el desarrollo de este proyecto.
- **Visión general:** Se presenta el proyecto dando una visión de alto nivel de los distintos módulos que lo componene así como sobre su estructuración y diseño global.
- **Proceso de ingeniería:** Se presenta la metodología aplicada y su adaptación en este proyecto. A mayores se presenta un diagrama sobre la planificación inicial que se ha llevado a cabo.

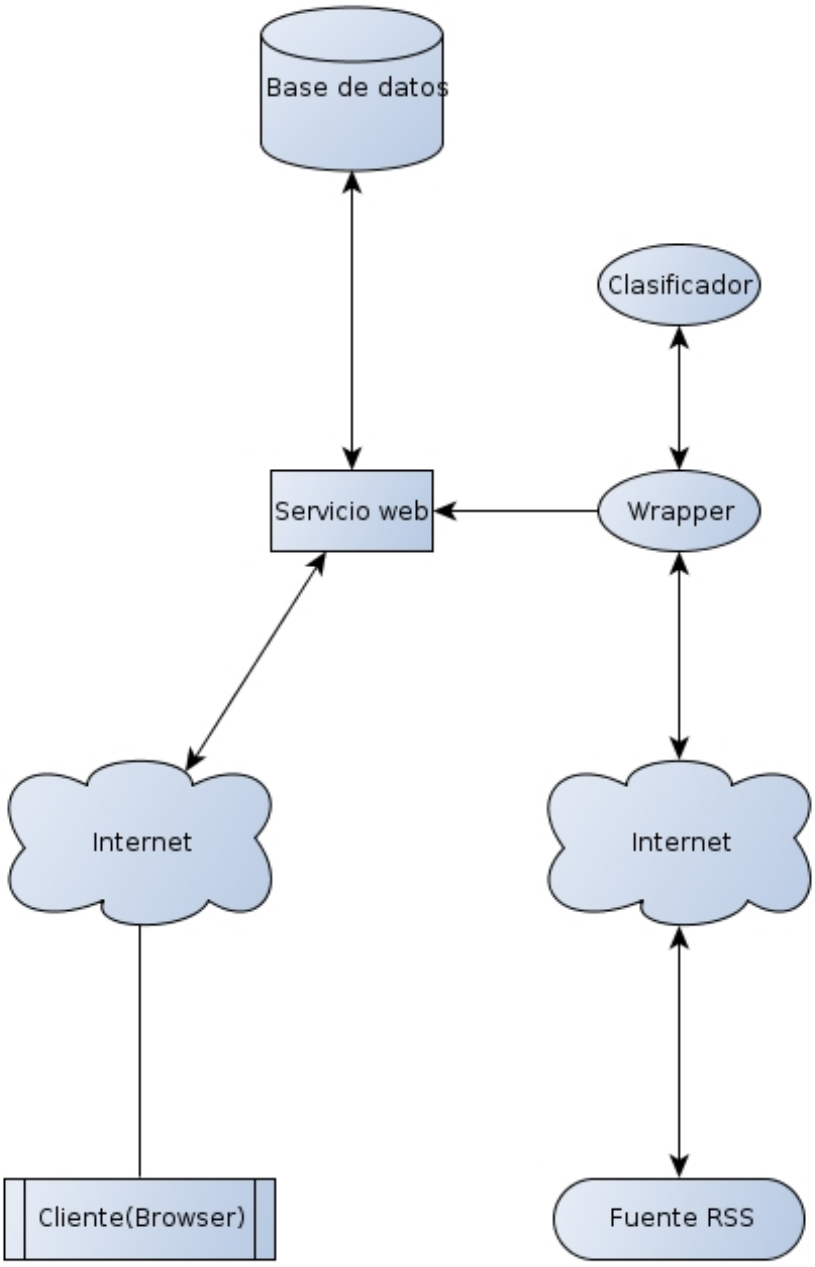

Figura 1.1: Visión general de proyecto

- **Desarrollo del Wrapper:** Se explica todo el proceso que se ha llevado a cabo para desarrollar este módulo, se comentan todos los detalles de implementación, test y problemas que surgieron.
- **Desarrollo del Clasificador:** Se explica en detalle todo el proceso que se ha llevado a cabo para el análisis de la elección del mejor clasificador así como los detalles de implementación, mejoras y métricas de comparación de los mismos.
- **Desarrollo de la Web:** Se presenta todo el proceso que se ha llevado a cabo para la realización de la aplicación web, así como la integración de los dos módulos anteriormente descritos dentro del entorno del framework web.
- **Conclusiones y trabajo futuro:** Se comenta la valoración personal sobre el proyecto y se muestra posibles mejoras que, por falta de tiempo, no han podido ser realizadas.

Capítulo 2
Fundamentos tecnológicos

Índice general

ANTES de proceder a explicar con más detalle el desarrollo del proyecto, vamos a realizar una explicación de los conceptos más relevantes utilizados en esta memoria.

2.1. Ruby on Rails

Es un framework para el desarrollo de aplicaciones web escrito en el lenguaje de programación Ruby. Sigue el paradigma del patrón arquitectónico Modelo Vista Controlador. Trata de ser simple y proporciona la posibilidad de desarrollar aplicaciones del mundo real escribiendo menos código y con menor configuración en comparación con el desarrollo en otros frameworks.

Las estructura del Modelo Vista Controlador (Figura 2.1) de Ruby on Rails es la siguiente:

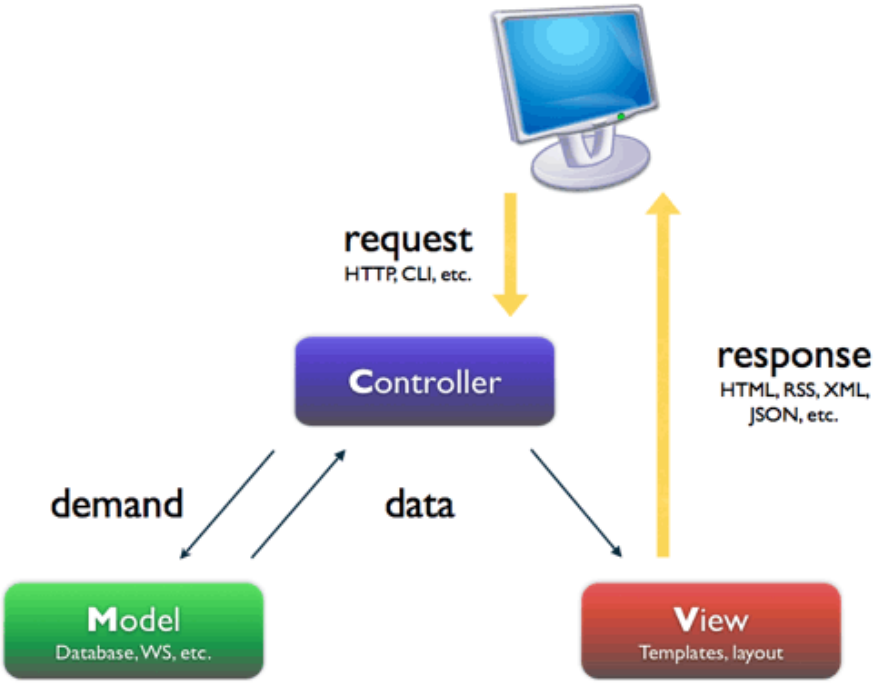

Figura 2.1: Patrón Modelo Vista Controlador

- **Modelo:** Cuando realizamos aplicaciones web en un lenguaje orientado a objetos, el modelo son clases mapeadas directamente de la base de datos, en Ruby on Rails, este tipo de clases son gestionadas por Active-Record[1].Proporciona la capa objeto-relacional que sigue rigurosamente el estándar ORM: Tablas en Clases, Registros en Objetos, y Campos en Atributos. Facilita el entendimiento del código asociado a base de datos y encapsula la lógica especifica haciéndola más fácil de usar para el programador.

- **Vista:** Las vistas es donde se visualizan los datos de las clases Controlador, consiste en una plantilla html, js, o cualquier otro formato de salida al que es embebido código en ruby incorporando los datos a mostrar en la plantilla.

  Ruby on Rails también proporciona una serie de módulos para la vista que facilitan la reutilización de código, la inclusión de otros archivos externos, y ayudantes para la creación de formularios.

[1]Patrón de diseño que resuelve el problema del acceso a base de datos desde una aplicación que quiere conectarse a ella.

- **Controlador:** Los controladores actúan entre la vista y el modelo respondiendo a la interacción del usuario y a su vez, manipulando los datos de las clases de modelo para finalmente enviárselo a la vista y que muestre los resultados al usuario nuevamente.

2.2. Skel.js

Es un framework de fron-end[2] que permite el desarrollo de aplicaciones web Responsive Design[3]. Consiste en un archivo js que proporciona al usuario herramientas muy poderosas:

- **Sistema CSS en Grid:** Un sofisticado sistema que permite organizar el sitio web como si fuera una cuadrícula, permitiendo definir los tamaño de los elementos de la plantilla de manera dinámica.
- **Definición de tamaños:** Permite definir previamente rangos de resoluciones en las que tu aplicación se va a visualizar para posteriormente adaptar el diseño correctamente.
- **Atajos de CSS:** Posee atajos para hacer frente a todo tipo de tareas comúnes en CSS.
- **Sistema de plugins:** Permite extender la funcionalidad e incorporar nuevos componentes al framework mediante este sistema.

2.3. Clasificador Bayes-Naïve

Es una clasificador probabilístico basado en el Teorema de Bayes. En términos simples, un clasificador de Bayes ingenuo asume que la presencia o ausencia de una característica particular, no está relacionada con la presencia o ausencia de cualquier otra característica, es decir, considera que cada característica contribuye de manera independiente a la probabilidad, independientemente de la presencia o ausencia de las otras características [24].

[2]Es la parte del software que interactúa con el usuario.
[3]Es una filosofía de diseño y desarrollo web que adapta el sitio web al entorno del usuario.

2.3.1. Concepto

Partiendo del teorema de Bayes:

$$p(C|F_1, \ldots, F_n) = \frac{p(C)\ p(F_1,\ldots,F_n|C)}{p(F_1,\ldots,F_n)}$$

En la práctica el denominador no es importante ya que no depende de la variable C y los valores de F_i son datos, por lo que para todas las categorías va a ser constante.

Una vez eliminado el denominador, el numerador es equivalente a una probalidad compuesta:

$$p(C)\ p(F_1, \ldots, F_n|C) = p(C, F_1, \ldots, F_n)$$

que puede ser reescrita aplicando el concepto de probabilidad condicionada:

$$p(F_1, F_2) = p(F_1)p(F_2|F_1)$$

como:

$p(C, F_1, \ldots, F_n)$
$= p(C)\ p(F_1, \ldots, F_n|C)$
$= p(C)\ p(F_1|C)\ p(F_2, \ldots, F_n|C, F_1)$
$= p(C)\ p(F_1|C)\ p(F_2|C, F_1)\ p(F_3, \ldots, F_n|C, F_1, F_2)$
$= p(C)\ p(F_1|C)\ p(F_2|C, F_1)\ p(F_3|C, F_1, F_2)\ p(F_4, \ldots, F_n|C, F_1, F_2, F_3)$

y podríamos seguir así sucesivamente. Ahora es cuando viene la independencia condicional de Naïve, asumiendo que cada una de las características F_i es independiente de cualquier otra F_j donde $i \neq j$. Lo cual, se traduce en:

$$p(F_i|C, F_j) = p(F_i|C)$$

por lo que la probabilidad compuesta puede expresarse como:

$$\begin{aligned} p(C, F_1, \ldots, F_n) &= p(C)\ p(F_1|C)\ p(F_2|C)\ p(F_3|C)\ \ldots\ p(F_n|C) \\ &= p(C) \prod_{i=1}^{n} p(F_i|C) \end{aligned}$$

2.3.2. Implementación

A nivel de implementación el clasificador obtendrá la categoría de la siguiente forma:

$$classify(word_1, word_2, \ldots, word_n) = argmax_C \; p(C) \prod_{i=1}^{n} p(word_i|C)$$

es necesario realizar un cambio más en la fórmula para evitar el Floating Point Underflow[4] tomando logaritmos:

$$log_a(u \cdot v) = log_a u + log_a v$$

por tanto, el resultado final será:

$$classify(word_1, word_2, ..., word_n) = argmax_C \; log_e(p(C)) \sum_{i=1}^{n} log_e(p(word_i|C))$$

2.4. Máquina de soporte vectorial

Son un conjunto de algoritmos de aprendizaje supervisado desarrollados por AT&T que están propiamente relacionados con problemas de clasificación y regresión [25].

2.4.1. Concepto

Simplificando el problema de dos dimensiones, dado un conjunto de puntos, que es subconjunto de un conjunto mayor (el plano), en el que cada uno de ellos pertenece a una de las dos categorías, un algoritmo basado en SVM[5] es capaz de construir un modelo que predice si un nuevo punto, cuya categoría es desconocida, pertenece a una de las dos categorías.

SVM recibe como entrada un vector de dimensión n y busca un hiperplano de forma que separe los puntos de ambas clases. El concepto de "separación óptima.[es] donde reside la característica fundamental de SVM, que se encarga de buscar el hiperplano que tenga la máxima distancia con los puntos que estén mas cerca de él mismo, por este motivo también recibe el nombre de *clasificador de margen máximo*.

[4]Es una condición en un programa donde le resultado del cálculo es más pequeño de lo que la computadora puede almacenar.

[5]Support Vector Machine

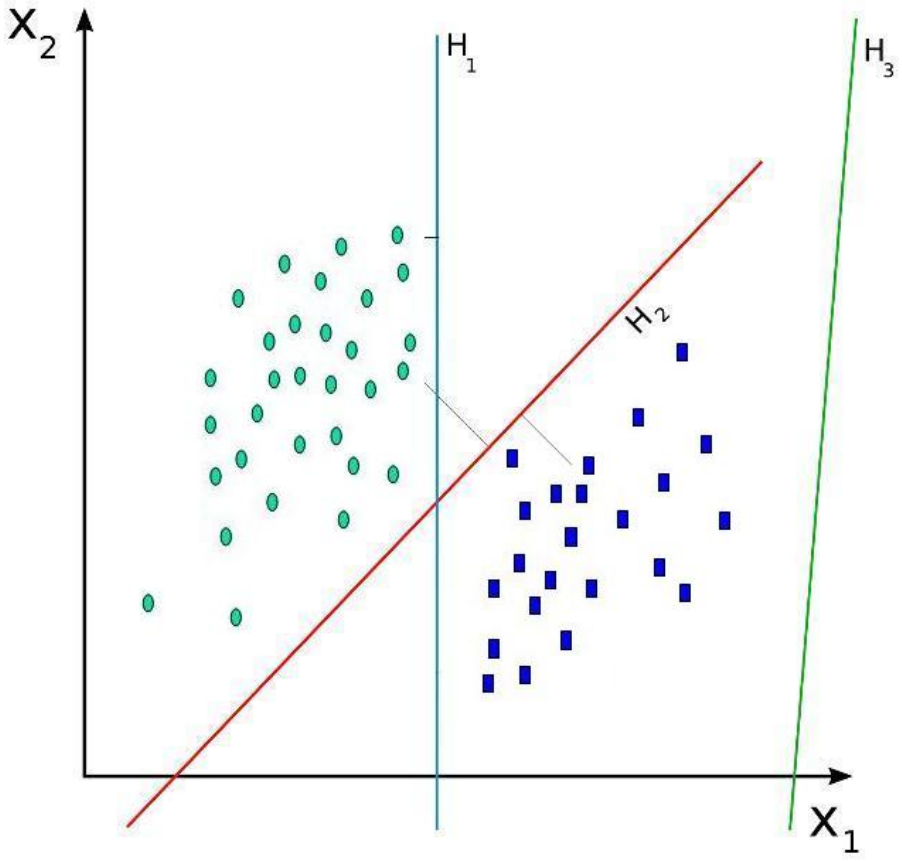

Figura 2.2: Principio básico de SVM

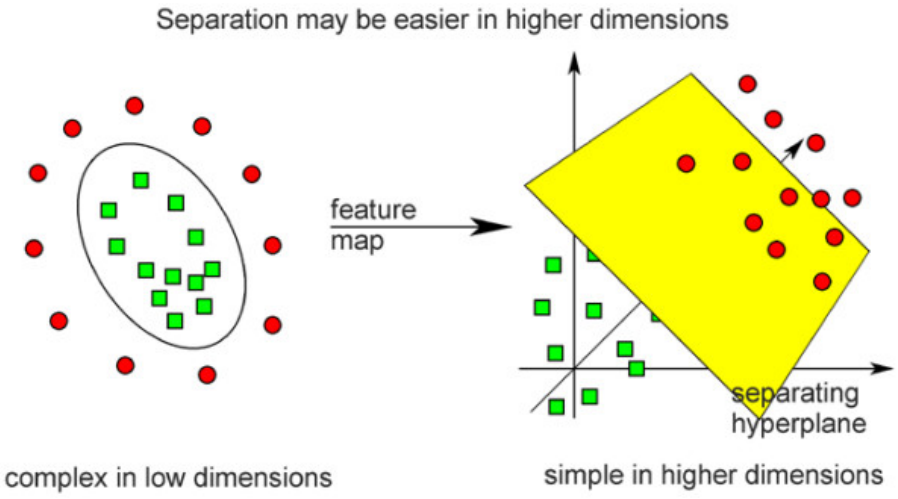

Figura 2.3: Vista en 3D

Capítulo 3

Visión general

Índice general

El proyecto que se ha desarrollado está compuesto de:

- Un Wrapper[1] para conectarse a una serie de fuentes RSS y extraer la información necesaria de las noticias que posteriormente será almacenada en una base de datos.
- Un algoritmo de clasificación de noticias que es usado dentro del Wrapper antes de que las noticias sean almacenadas en base de datos, y que permite dar una estructuración al contenido en categorías.
- Una aplicación web que consume el contenido almacenado por el Wrapper y es la encargada de mostrar el contenido al usuario de manera sencilla, limpia y ordenada.

[1]Es un programa que extrae un determinado tipo de información de una o más fuentes de datos.

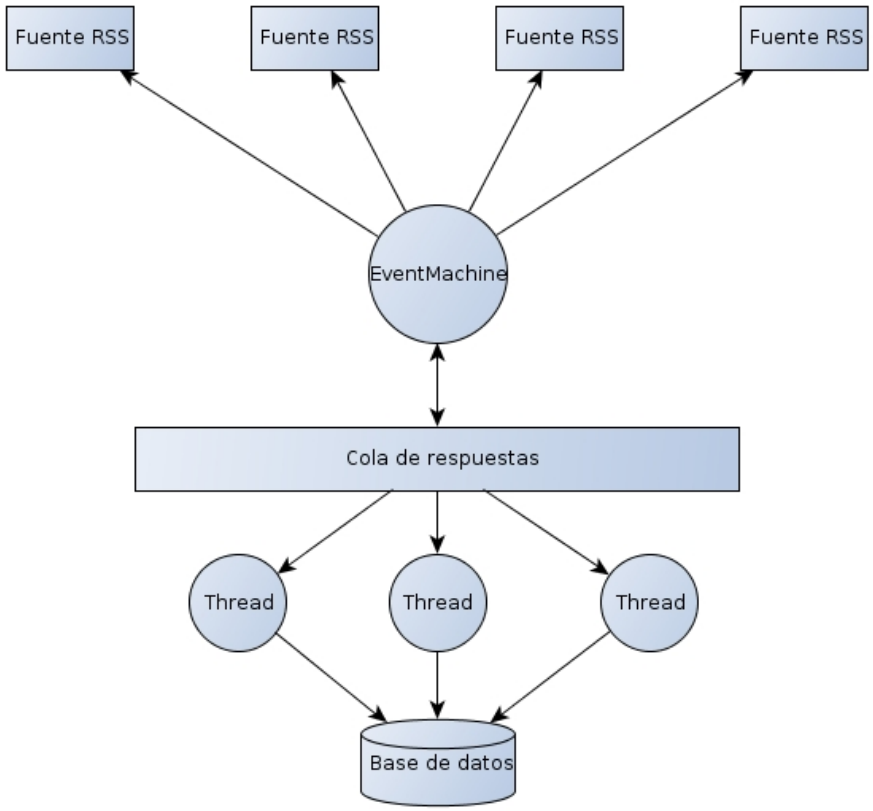

Figura 3.1: Arquitectura del Wrapper

3.1. Wrapper

El Wrapper ha sido desarrollado en ruby, empleando técnicas, tecnologías y patrones de diseño que permiten mejorar su rendimiento a gran escala. El proceso una vez iniciado envía peticiones asíncronas, utilizando una librería llamada EventMachine, a las distintas fuentes que es necesario procesar, cuando estas responden, se procesa la información en paralelo y se almacena en base de datos[Figura 3.1].

Por tanto, el único caso de uso que contempla este proceso es:

- Extraer el contenido de un lista de fuentes RSS dadas.

La información de las fuentes RSS que se procesan se obtiene de la base de datos y la elección de las mismas la realiza el administrador.

3.2. Clasificador

Una vez realizado un estudio de varios clasificadores que en apartados posteriores será explicado con más detalle, el que se ha utilizado finalmente fue la Máquina de Soporte Vectorial o SVM.

La implementación elegida de SVM, LibSVM, está escrita en C y proporciona un binding[2] para ruby mediante la Gem[3] Rb-Libsvm. Para poder utilizar el clasificador es necesario realizar una serie para de previos para que el clasificador pueda funcionar correctamente:

- Insertar los datos de entrenamiento en una base de datos.
- Ejecutar el proceso que convierte los datos de entrenamiento en vectores de características para que SVM pueda recibirlos como entrada.
- Entrenar el clasificador generando un modelo listo para predecir.

Por tanto los casos de uso que el clasificador requiere son los siguientes:

- Convertir los datos de entrenamiento a vectores de características.
- Entrenamiento.
- Clasificación.

En la figura 3.2 se presenta la arquitectura de entrenamiento y en la figura 3.3 la arquitectura de clasificación. En ningún momento se refiere a la arquitectura interna del propio clasificador ya que no es objeto es este documento siendo una librería externa utilizada.

3.3. Aplicación web

La aplicación web utiliza dos frameworks para su desarrollo, en el front-end utliza Skel.js mientras que en el back-end utiliza Ruby on Rails.

Los casos de uso que se contemplan en esta parte del proyecto son los siguientes:

[2]Interfaz que permite enlazar código entre distintos lenguajes de programación
[3]Nombre que recibe las librerías o módulos reutilzables en ruby.

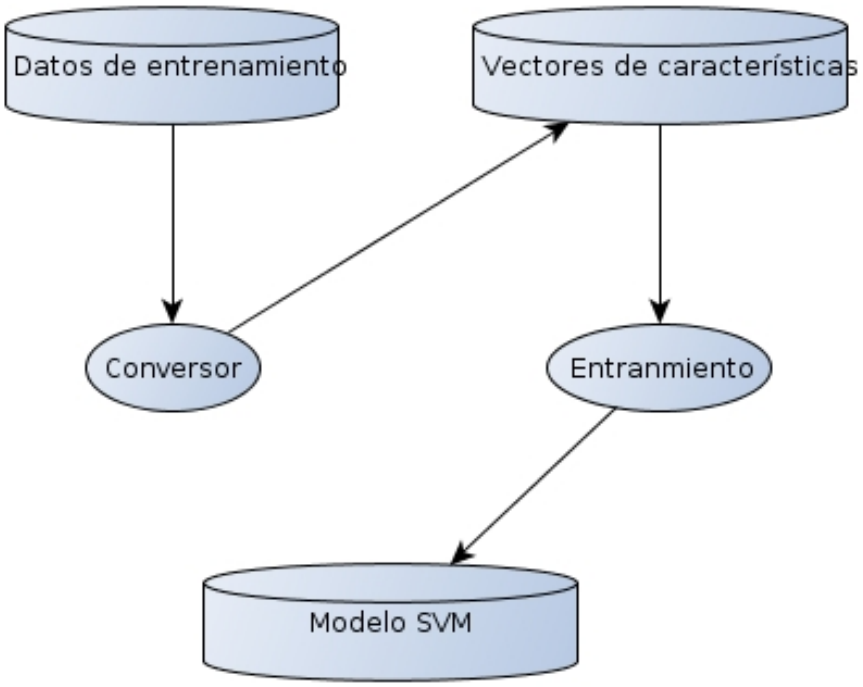

Figura 3.2: Arquitectura de entrenamiento

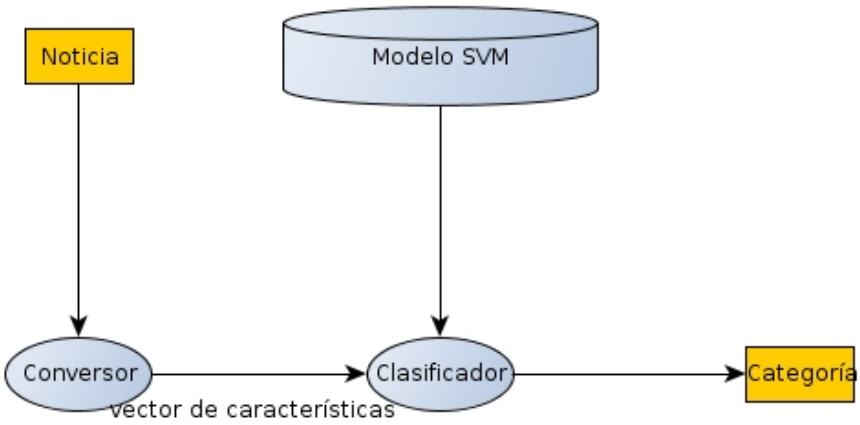

Figura 3.3: Arquitectura de clasificación

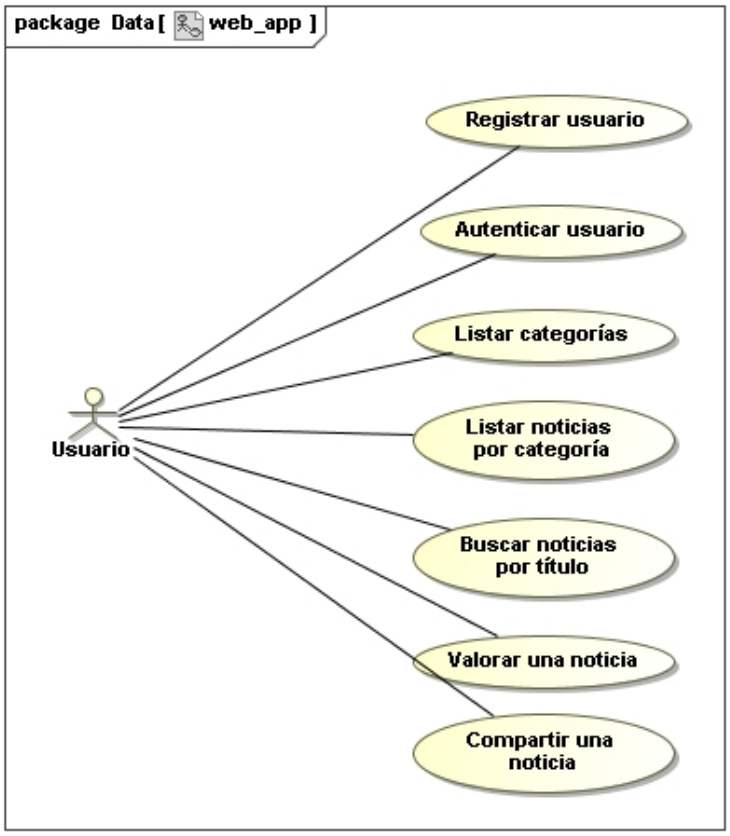

Figura 3.4: Casos de uso

- Registrar un usuario.
- Autenticar un usuario.
- Listar categorías.
- Listar noticias por categoría.
- Buscar noticias por título.
- Valorar una noticia.
- Compartir una noticia en la redes sociales.

El modelo de datos resultante de los casos de uso es el mostrado en la figura 3.5.

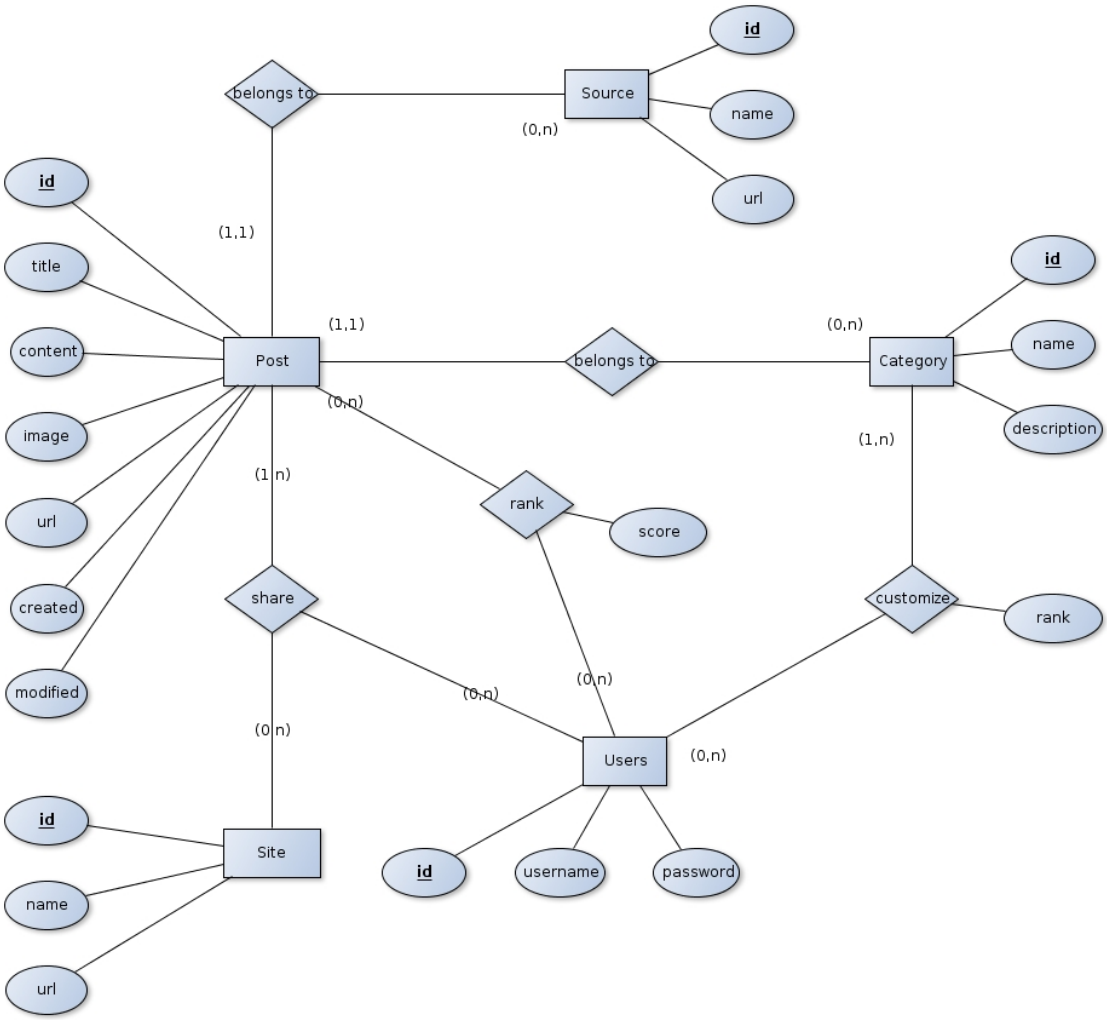

Figura 3.5: Modelo de datos

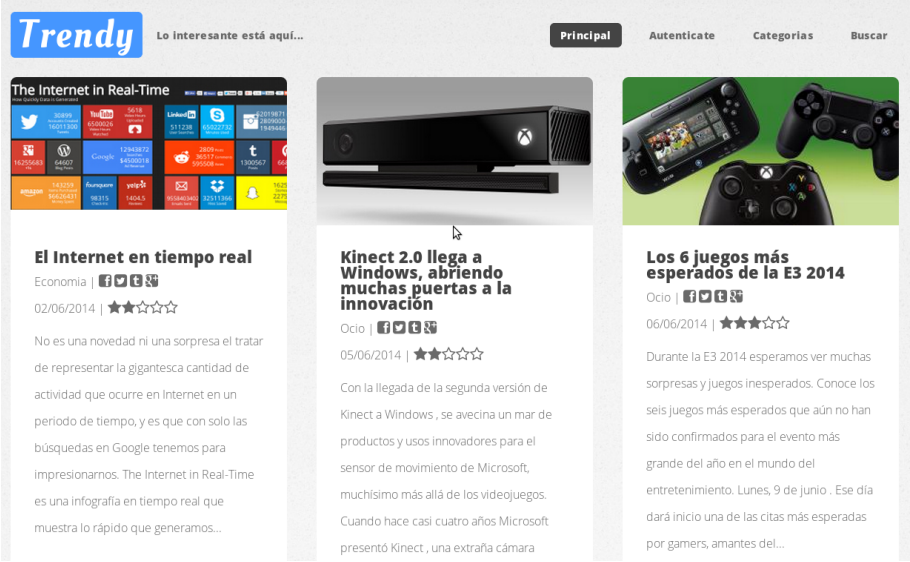

Figura 3.6: Aplicación versión de Personal Computer.

Skel.js adapta la interfaz de la aplicación web a cualquier tipo de pantalla, para este caso, exiten 2 modelos de presentación de los datos como se muestran en las figuras 3.6 y 3.7.

3.4. Cronología

La figura 3.8 muestra una línea temporal de como se ha ido desarrollando y publicando las versiones de los distintos módulos que componen este proyecto.

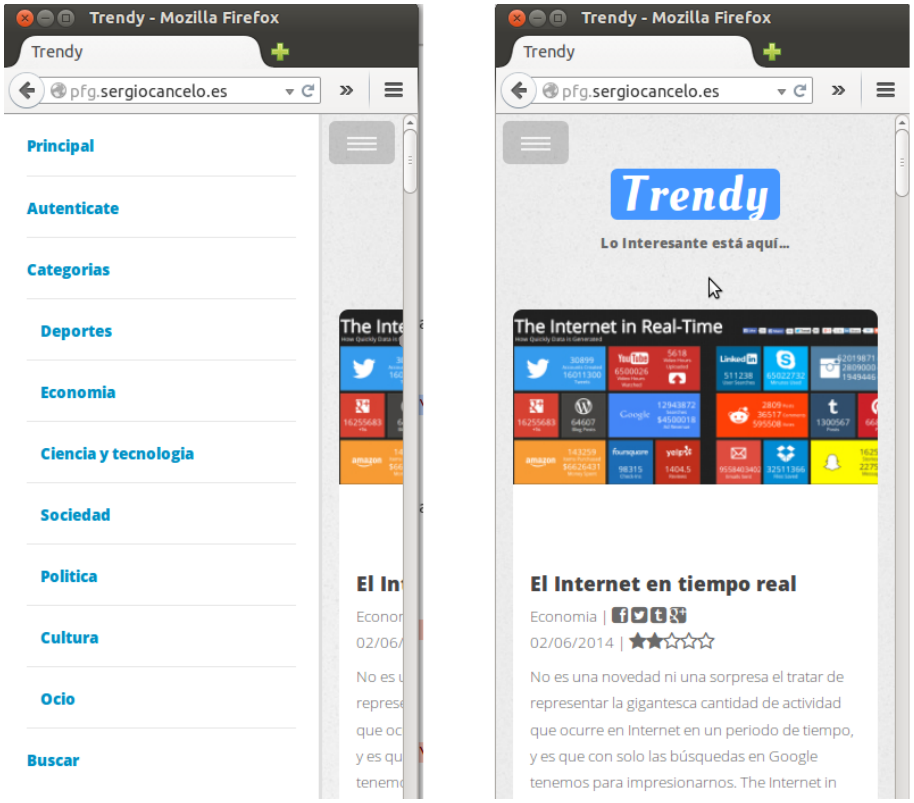

Figura 3.7: Aplicación versión de Móbil.

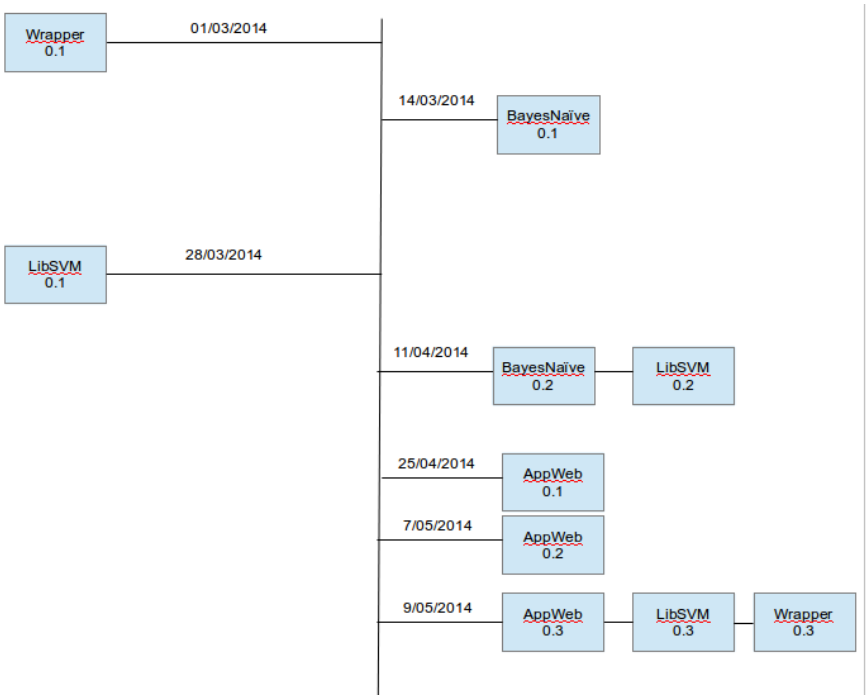

Figura 3.8: Cronología de las versiones del proyecto

Capítulo 4
Proceso de ingeniería

Índice general

En este capítulo se describe la matodología de trabajo y la planificación seguida durante el proceso de ingeniería.

4.1. Metodología aplicada

4.1.1. Metologías ágiles

Las metodologías ágiles se aplican en proyectos donde los requisitos no se encuentran bien definidos o es posible que cambien una vez iniciado el proyecto. Estas metodologías se adaptan mejor a pequeños equipos que resuelven problemas concretos. Las principales características de este tipo de metodologías son:

- Capacidad de respuesta a cambios en los requisitos del proyecto.
- Entrega continua y en pequeños plazos de partes pequeñas del proyecto pero que son funcionales permitiendo construir el proyecto de forma incremental.

- Trabajo conjunto entre el cliente y el equipo de desarrollo.
- Mejora continua tanto de los procesos como del equipo de desarrollo.

4.1.2. SCRUM

Es una metodología que proporciona un marco de desarrollo iterativo e incremental. Nos permite rápidamente y en repetidas ocasiones inspeccionar software real de trabajo. Los equipos se auto-organizan a fin de determinar la mejor manera de entregar las funcionalidades de más alta prioridad. Las iteraciones en SCRUM se denominan sprint, la duración de los sprint es constante y por tanto conduce a un mejor ritmo de trabajo y, en cada uno de dichos sprint, el producto es diseñado, codificado y testeado.

Cada sprint, dura un periodo entre una y cuatro semanas, esta duración es establecida por los miembros del equipo, donde se crea un incremento de software potencialmente entregable.

La funcionalidades que se van a implementar en cada sprint están definidas en el Product Backlog[1], que es un conjunto de requisitos de alto nivel priorizados que definen el trabajo a realizar.

Los elementos del Product Backlog que forman parte del sprint se determinan durante la reunión de Sprint Planning[2]. Durante esta reunión, el Product Owner[3] identifica los elementos del Product Backlog que quiere ver completados y los hace del conocimiento del equipo. Entonces, el equipo determina la cantidad de ese trabajo que puede comprometerse a completar durante el siguiente sprint. Durante el sprint, nadie puede cambiar el Sprint Backlog, lo que significa que los requisitos están congelados durante el sprint.[Figura 4.1]

Cada día de un sprint, se realiza la reunión sobre el estado de un proyectoque comienza siempre a la misma hora. Todos son bienvenidos, pero sólo los involucrados en el proyecto pueden hablar. La reunión tiene una duración de 15 minutos, en esta se realizan las siguiente preguntas:

- ¿Qué has hecho desde ayer?
- ¿Qué es lo que harás hasta la reunión de mañana?

[1]Lista de todas la funcionalidades que se espera que se implementen del producto.
[2]Planificación de la iteración.
[3]Cliente al que se le realiza el proyecto

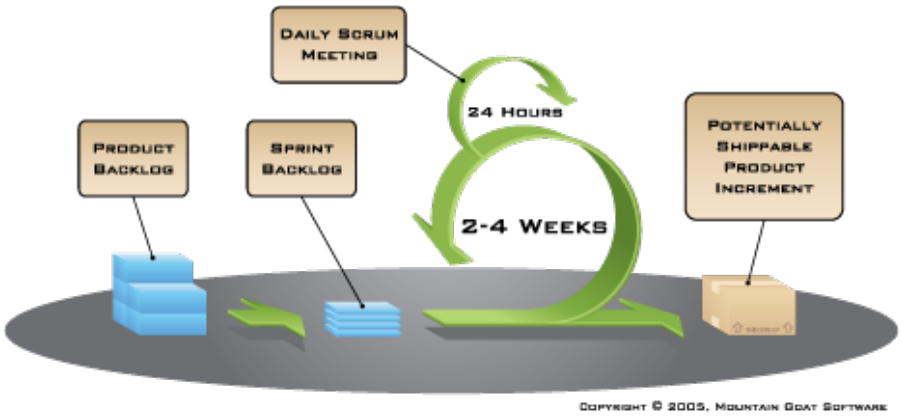

Figura 4.1: Metodología SCRUM

- ¿Has tenido algún problema que te haya impedido alcanzar tu objetivo?

Al final de cada sprint se realiza otra reunión donde se revisa si el trabajo ha podido completarse, se presenta el trabajo a los interesados, y, como máximo, tiene una duración de cuatro horas.

4.1.3. Adaptación de SCRUM al proyecto

Durante la realización de este proyecto se ha tratado de seguir en la mayor medida de lo posible las prácticas descritas anteriormente de esta metodología como pueden ser el desarrollo en iteraciones y las reuniones al final de cada sprint.

Se realizan iteraciones de dos semanas al final de las cuales sale una versión del producto entregable que contiene mayor funcionalidad que la iteración anterior.

Una vez elegidos los elementos del Product Backlog que van a formar parte de un determinado sprint, se realiza el diseño, implementación y test de los mismos.

Al final de cada iteración se realiza una reunión con el director del proyecto para que realice un seguimiento del trabajo que se ha realizado lo valore, o en caso de ser una valoración negativa, aporte nuevas instrucciones para realizar en un nuevo sprint.

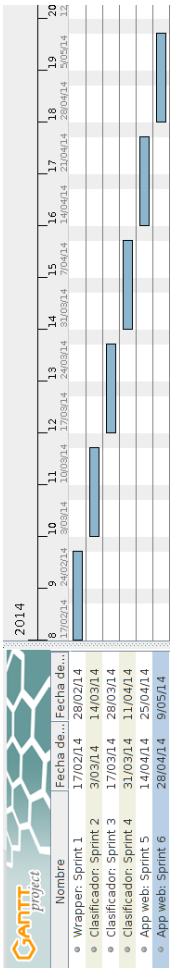

Figura 4.2: Planificación del proyecto

4.1.4. Planificación del proyecto

La planificación del proyecto se ha realizado teniendo en cuenta que es un proyecto que hay partes que presentan bastante incertidumbre sobre el tiempo que puede llevar realizarlas, así como también hay incertidumbre sobre las horas diarias de dedicación.

Capítulo 5

Desarrollo del Wrapper

Índice general

El proceso wrapper es el encargado de rastrear las fuentes RSS existentes y extraer el contenido necesario para que la aplicación pueda funcionar correctamente, por tanto, necesita una gran rapidez haciendo las peticiones y una gran eficiencia parseando el contenido necesario.

El desarrollo del wrapper se ha realizado en un único sprint que se detalla en los siguientes apartados.

5.1. Primer sprint: Rastreado de fuentes

Este primer sprint se realiza en dos partes, una primera parte que se encarga de gestionar las peticiones, y una segunda parte que se encarga de procesar los datos y la persistencia. El modelo general es el de la figura 5.1

Para realizar el rastreado de fuentes es necesario realizar muchas peticiones a la red, para ello hay que tener en cuenta la latencia de la red y el tiempo que tarda el servidor en responder. Necesitamos una librería que permita:

- Gestionar peticiones HTTP de manera concurrente.

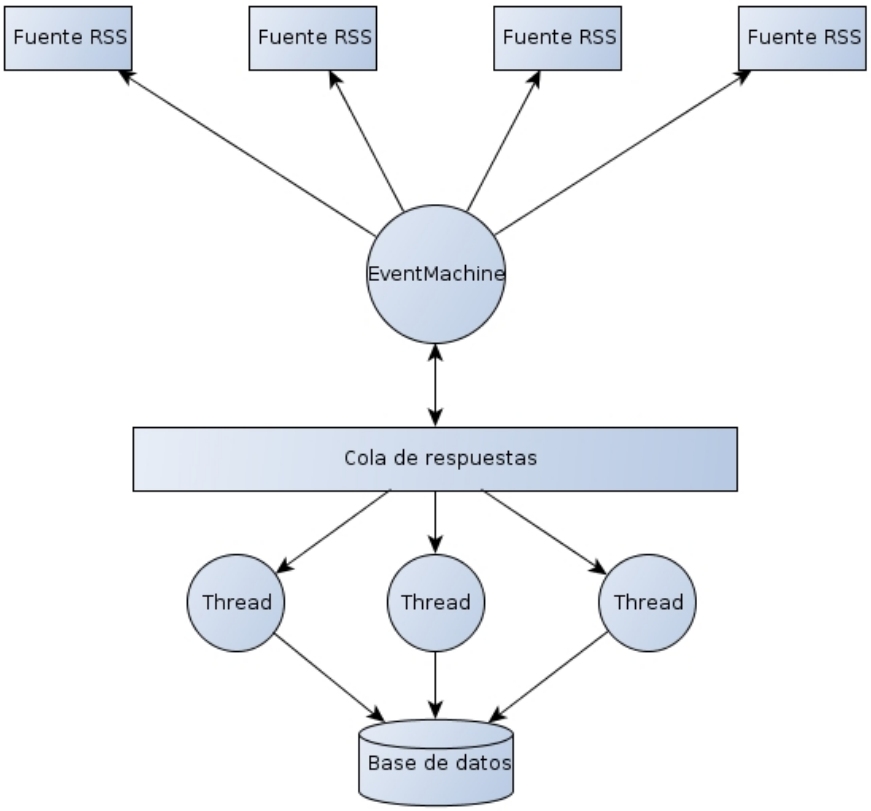

Figura 5.1: Arquitectura del Wrapper

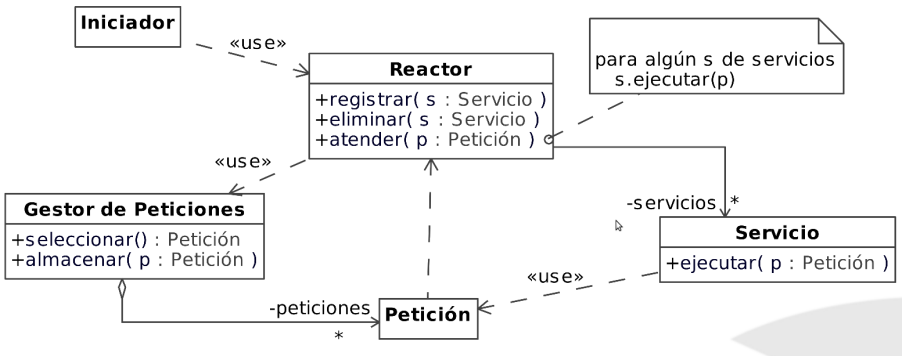

Figura 5.2: Patrón reactor

- Gestionar peticiones asíncronas.

Esta funcionalidad es proporcionada por la gem Event Machine HTTP Request, que es una librería que extiende ligeramente en funcionalidad a Event Machine, un software diseñado para ruby que provee una interfaz para gestionar peticiones de manera concurrente que nos ofrece un alto rendimiento, su arquitectura se basa en el patrón Reactor [Figura 5.1], esta librería nos permite gestionar peticiones asíncronas de manera eficiente.

5.1.1. Desarrollo

En esta primera parte se realiza una implementación de patrón de concurrencia productores-consumidores [Figura 5.3], en la que se lanzan todas las peticiones a la vez de manera asíncronas con EventMachine, estas peticiones se encolan en el momento que se recibe la respuesta, y a su vez, en paralelo, los elementos de la cola son consumidos por un pool de threads que, en esta primera parte, no procesa la información.

La segunda parte del sprint consiste en darle forma a la información generada anteriormente, para ello en este sprint se realiza una función que utiliza la gem Nokogiri[1] para extraer la información de las fuentes RSS y ActiveRecord[2] para almacenarla en base de datos. Además de ello, se utiliza activerecord para cargar fuentes de ejemplo y hacer la primera prueba de funcionamiento del wrapper.

[1]Librería para parseo de RSS.
[2]Mapeador Objeto-Relacional.

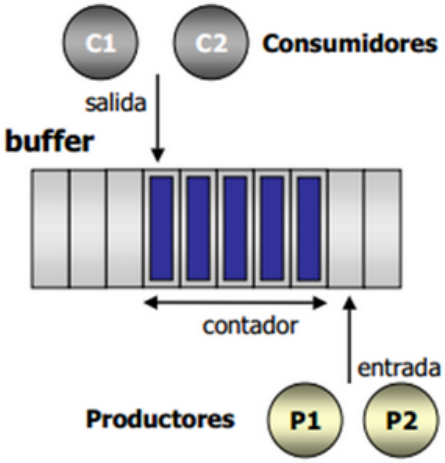

Figura 5.3: Productores consumidores

Los pasos comentados anteriormente se resumen en el diagrama de secuencia de la figura 5.4.

5.1.2. Test

Para la realización de los test del wrapper se ha proporcionado varias plantillas RSS en local a las que el proceso accede y parsea. Si no se han producido errores, al guardarlo en base de datos, se testea que los resultados creados por el proceso sean iguales a los resultados de las plantilla.

5.1.3. Problemas encontrados

Los problemas encontrados fueron los siguientes:

- Problemas con RSS mal formados: En este caso el problema se ha resuelto capturando da Exception que proporciona la librería.
- Problemas de sincronización entre productores y consumidores: Este tipo de problemas sucedían cuando la cola que une a ambos subprocesos se vacía. Se soluciona haciendo que los consumidores esperen por más elementos.
- Problema de la parada: Este problema es derivado del anterior, sucedía cuando ya se terminaban todas las fuentes y los consumidores seguían espe-

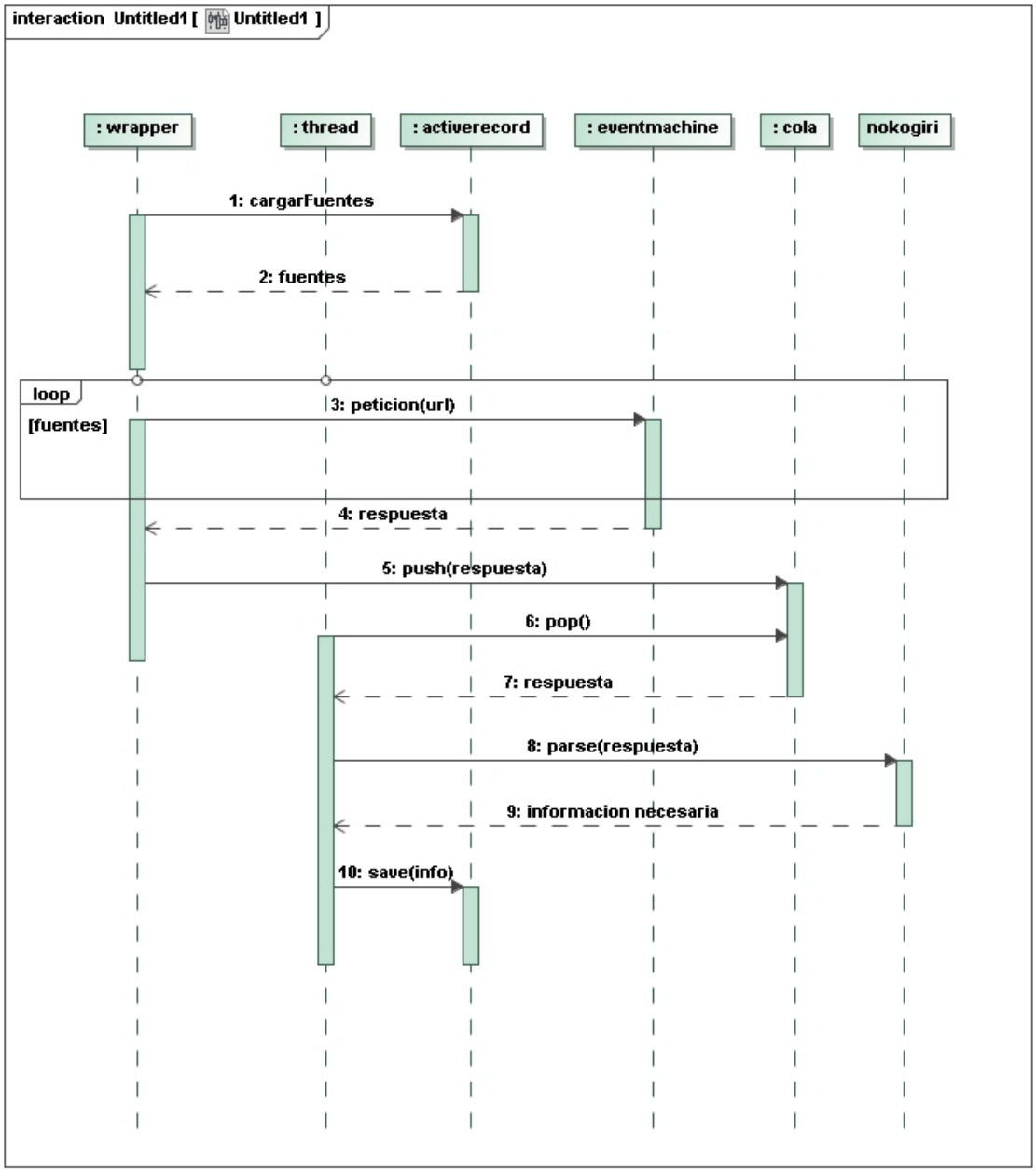

Figura 5.4: Diagrama de secuencia del Wrapper

rando a que la cola se llenara con mas elementos haciendo que el programa nunca termine. Se soluciona con una variable de finalización.

Capítulo 6

Desarrollo del Clasificador

Índice general

En este capítulo se va a describir todo el proceso que se ha llevado a cabo para la implementación del clasificador que mejor se adapta a los requerimientos del proyecto.

Se va a realizar una modificación de uno de ellos, concretamente de Bayes-Naïve, ya que no se ha encontrado ninguna librería que implemente este clasificador para contenido en Español. Del segundo, se van a implementar una serie

de scripts necesarios para su funcionamiento y también se aplicarán mejoras para que soporte multiclase. Finalmente, se mostrarán unos gráficos comparativos para justificar la elección de uno de ellos.

Por tanto, proceso de desarrollo del clasificador se realiza en 3 sprints y que son los siguientes:

- Adaptación de Bayes-Naïve al castellano.
- Desarrollo del entorno necesario para SVM.
- Comparación de ambos clasificadores.

6.1. Segundo sprint: Apaptación de Bayes-Naïve

Inicialmente partimos de una implementación de este clasificador [01] y para adaptarlo al español es necesario realizar una serie de cambios que se exponen a continuación:

- El clasificador posee una blacklist de palabras que no son relevantes en el documento como pueden ser las preposiciones, conjunciones, artículos, etc. por lo que es necesario eliminarlas del texto, este es el primer paso que realiza. El problema reside en la blacklist que tiene una lista de palabras en inglés, por lo que es necesario adaptarlas al español.
- El clasificador usa Stemming[1], de este modo mejoramos la inteligencia del mismo ya que es capaz de detectar, por ejemplo, el plural de las palabras. En este caso el stemmer que está utilizando también es esclusivo para Inglés, por lo que es necesario cambiarlo por uno que soporte Español.

6.1.1. Desarrollo

El diseño que presenta el clasificador se puede visualizar en la figura 6.1.

Por una parte, se ha hecho una recopilación de totas la palabras que se pueden introducir dentro de la blacklist, concretamente en la constante CORPUS_SKIP_WORDS.

[1]Procedimiento que consiste en dejar solo la raíz de la palabra. Ej: coches -¿coch, coche -¿coch

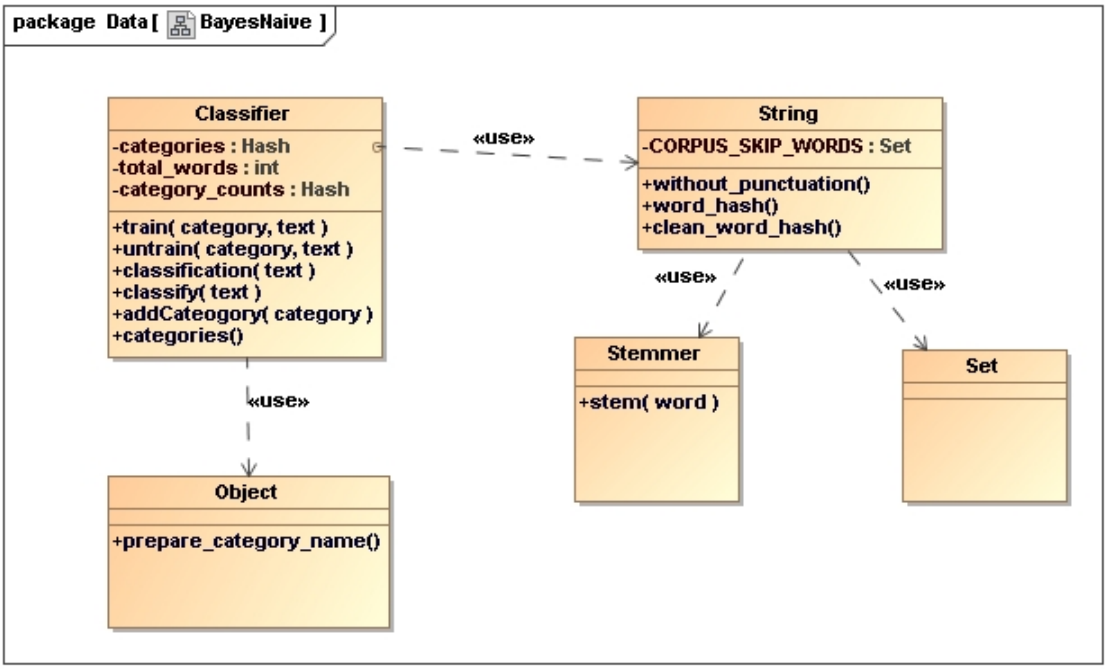

Figura 6.1: Digrama de clases del clasificador

Por otro lado, se han barajado alternativas al stemmer actual, finalmente se ha optado por ruby-stemmer, una librería de stemming escrita en C que tiene un binding para ruby y, además, tiene soporte para una gran cantidad de idiomas además del Español.

El diseño de clasificador a nivel de clases sigue siendo el mismo.

6.1.2. Test

Inicialmente el clasificador ya aportaba una batería de test, pero ha sido necesario cambiar la totalidad de ellos al español y hacer nuevos test para comprobar que la integración del nuevo stemmer era la correcta.

6.2. Tercer sprint: Desarrollo del entorno SVM

Para el desarrollo de este clasificador se parte de la gem rb-libsvm, pero para que el clasificador funcione correctamente es necesario realizar una serie de pasos previos:

- Convertir los datos de entrenamiento a vectores de características.
- Implementar un algoritmo para cargar los vectores y entrenar a SVM.
- Implementar una mejora para que SVM soporte multiclase[2].

6.2.1. Conversión de datos

Tenemos un conjunto de entrenamiento, en este caso son noticias recogidas de fuentes RSS, por tanto es texto, pero este clasificador lo que necesita es un vector de pesos en cada uno de las posiciones representando una serie de características.

Los pesos de los vectores se calculan mediante la medida TF-IDF[3] . Es una medida numérica que expresa cuán relevante es una palabra para un documento en una colección. El valor TF-IDF aumenta proporcionalmente al número de veces que una palabra aparece en el documento, pero es compensada por la frecuencia de la palabra en la colección de documentos, lo que permite manejar el hecho de que algunas palabras son generalmente más comunes que otras.

La fórmula aplicada para calcular ele TF-IDF es la siguiente:
Sea f(t,d) = número de veces que aparece la palabra t en el documento d.

$$\text{TF} = 1 + log(f(t,d)) \text{ (y 0 si f(t,d) = 0)}$$

Sea D = número total de documentos de la colección.
Sea g(T) = número de documentos en los que aparece la palabra T.

$$\text{IDF} = \frac{D}{1+log(g(T))}$$

Por lo tanto:

$$\text{TF-IDF} = \text{TF} * \text{IDF}$$

Tenemos dos documentos A y B con una determinadas cadenas de texto [Figura 6.2]. Para convertir las cadenas de texto en vectores se hace lo siguiente:

- Generamos un diccionario con todas las palabras de todos los documentos existentes y le asignamos a cada palabra nueva un número cualquiera y en cualquier orden tal y como muestra la figura 6.2, es decir, en el diccionario no existen palabras repetidas.

[2]Permita clasificar en más de dos categorías.
[3]Term frequency – Inverse document frequency

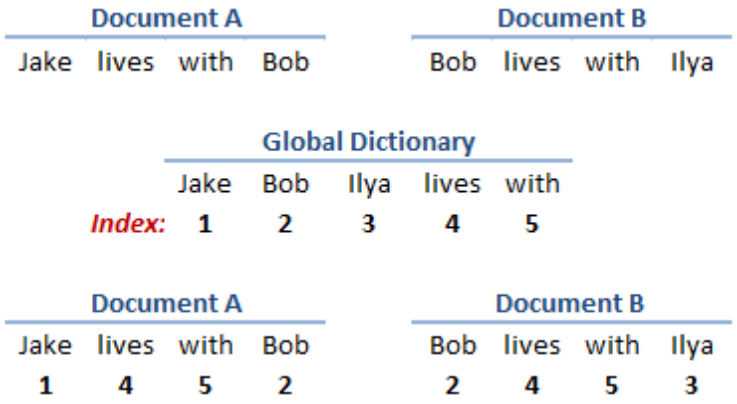

Figura 6.2: Generación de vectores del clasificador

- Con el diccionario presente, a cada palabra del documento le asignamos el número del diccionario.
- Calculamos el TF-IDF para cada una de las palabras del documento.
- Generamos un vector del tamaño del diccionario para cada documento, rellenando con el TF-IDF la posición que corresponde a cada palabra, el resto de las posiciones se rellena con 0. Así para el documento A el vector resultante sería:

$$[\ \text{TFIDF(Jake)}\ ,\ \text{TFIDF(Bob)}\ ,\ 0\ ,\ \text{TFIDF(lives)}\ ,\ \text{TFIDF(with)}\ ]$$

El algoritmo implementado para la conversión de datos hace exactamente lo que se ha mostrado en esta explicación. Realiza la carga de los documentos de base de datos, aplica el proceso de conversión y almacena los vectores en archivos de texto para su posterior carga.

Una aproximación a su diseño es el mostrado en la figura 6.3.

6.2.2. Entrenamiento de SVM

Una vez convertidos los datos, el proceso de entrenamiento es muy sencillo, simplemente se realiza un algoritmo que carga los datos del fichero vectores, genera una instancia de Libsvm y llama al método train pasándole la totalidad de los vectores. SVM genera un modelo que finalmente se almacena en un archivo listo para ser cargado de nuevo y poder realizar predicciones sobre nuevos documentos.

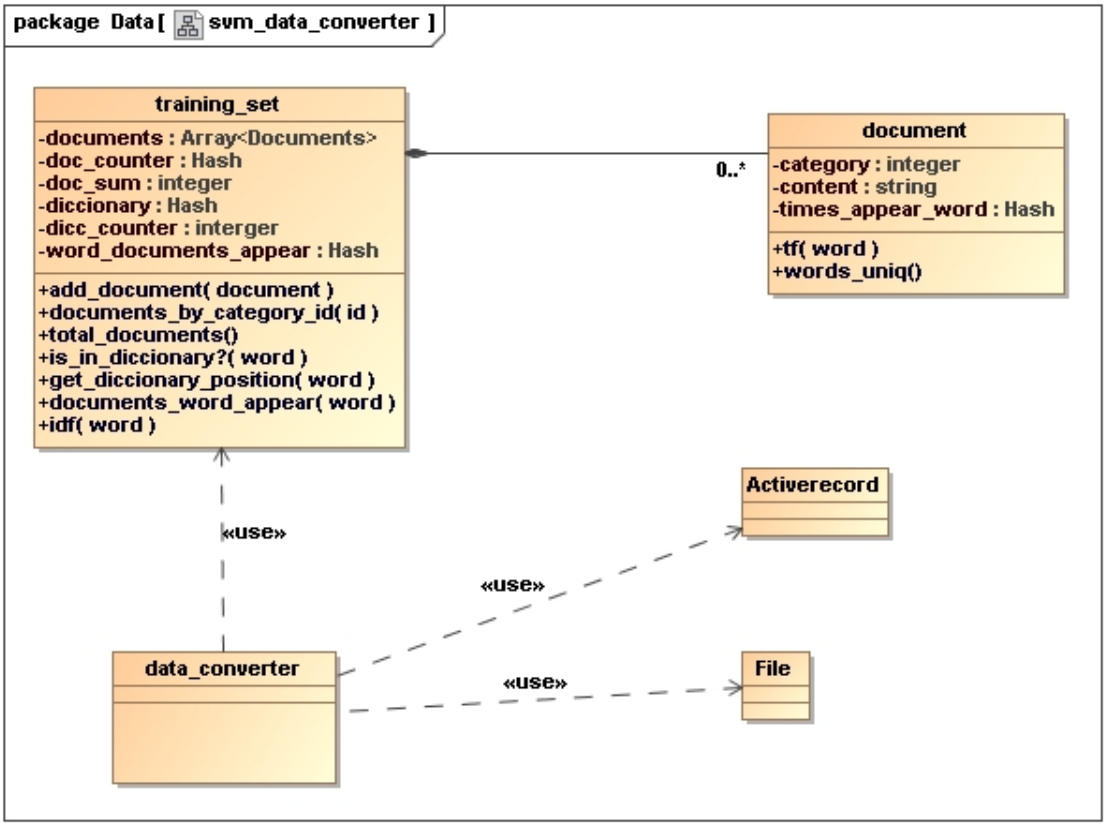

Figura 6.3: Diseño del conversor de datos

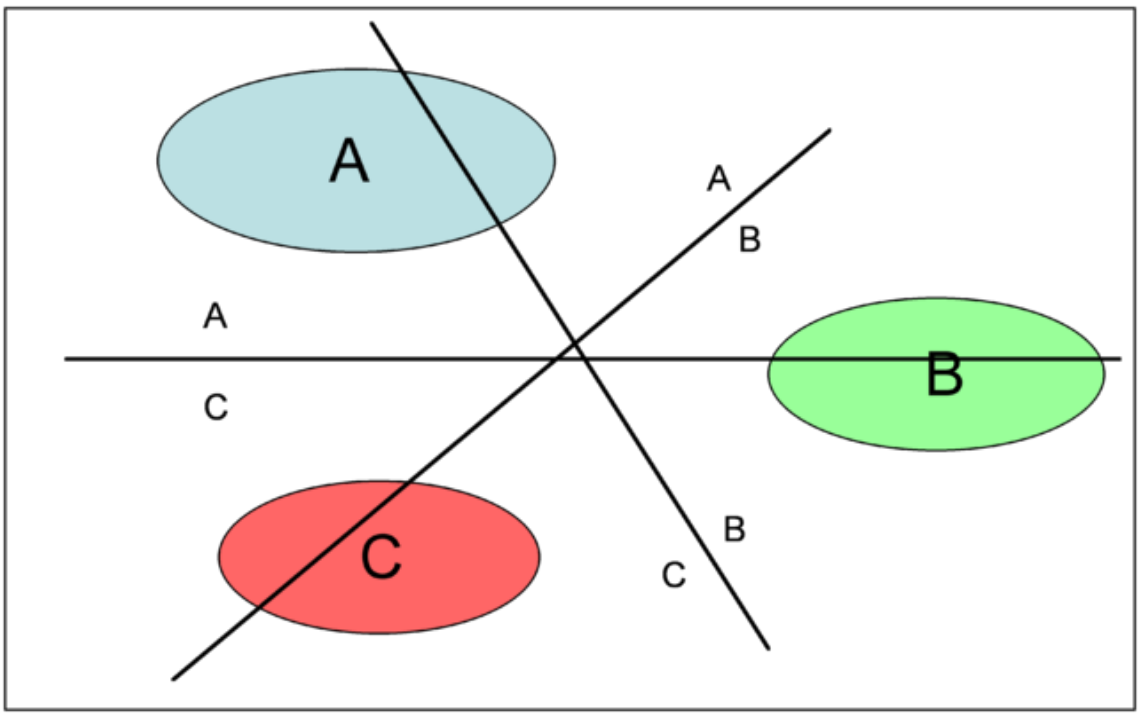

Figura 6.4: Esquema pairwise

6.2.2.1. Problemas

El principal problema encontrado en esta parte de proceso es que esta implementación de libsvm no obtiene buenos resultados cuando se realiza una clasificación de más de dos categorías, el motivos es que esta librería funciona con una versión de libsvm que ella misma integra, y esta, no trae un buen soporte para multiclase. Como consecuencia de ello, en el siguiente apartado implementaremos una mejora aumentar el poder de predicción de esta librería en el caso de multiclase.

6.2.3. Pairwise

Pairwise es un método que se utiliza en SVM para mejorar el soporte multiclase [02]. SVM obtiene un alto rendimiento cuando clasificamos dos clases. Si tenemos un problema en el que queremos clasificar n clases, generamos $\sum_1^n i$ máquinas donde cada máquina predice un par de categorías, es decir, generamos un conjunto de máquinas combinando todas las categorías por pares. De este modo conseguimos obtener un esquema como el mostrado en la figura 6.4.

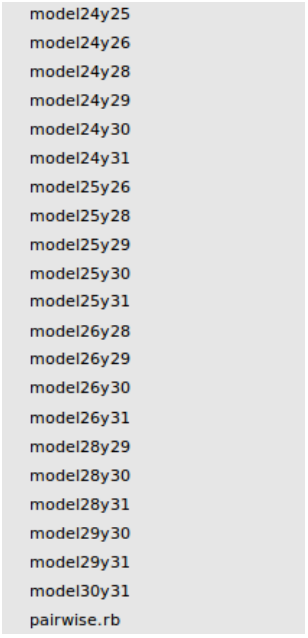

Figura 6.5: Modelos generados

Con esto podemos conseguir que una librería que aparentemente no tiene soporte para más de dos clases pueda tenerlo.

6.2.3.1. Implementación

La implementación de este método no cambia el diseño del conversor de datos, ni del algoritmo de entrenamiento, los únicos cambios a realizar son los siguientes:

- El conversor de datos, carga pares de categorías hasta combinarlas, y almacena tantos ficheros de datos para cargar como máquinas SVM tenemos [Figura 6.5].
- Para el entrenamiento aplicamos el mismo proceso n veces, hasta entrenar una máquina con cada modelo.

6.3. Cuatro sprint: Comparativa de clasificadores

En esta parte se realiza la elección del mejor algoritmo, para ello se realizan una serie de medidas de ambos algoritmos con el mismo conjunto de entrena-

miento. Para realizar el análisis es necesario implementar para cada uno de los clasificadores una técnica de validación llamada cross-validation y analizar los resultados midiendo tasa de acierto, precision y recall.

6.3.1. Creación del conjunto de Entrenamiento

Para construir un buen conjunto de entrenamiento necesitamos que este sea heterogéneo, es decir, que abarque la mayor variedad de los temas que puede tener una clase. Además es conveniente que esa recopilación se realice variando lo máximo posible las fuentes para evitar que el clasificador recuerde patrones concretos de redacción de una determinada fuente y esto provoque en él una influencia negativa.

Para ello se se recopilan 2000 noticias para cada una de las categorías extrayendo contenido los más variado posible y alternando muchas fuentes de datos hasta alcanzar un total de 14.000 noticias para las 7 categorías que se van a utilizar.

6.3.2. Cross-validation

La validación cruzada es una técnica utilizada para evaluar clasificadores. Consiste en repetir y calcular la media aritmética obtenida de las medidas de evaluación sobre diferentes particiones. Los datos de muestra se dividen en K subconjuntos. Uno de los subconjuntos se utiliza como datos de prueba y el resto como datos de entrenamiento. El proceso de validación cruzada 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. Este método es muy preciso puesto que evaluamos a partir de K combinaciones de datos de entrenamiento y de prueba [Figura 6.6] [24].

6.3.3. Métricas

Antes de introducir las medidas a utilizar es necesario dar cuatro definiciones:

- True Positives (TP) para la clase C: son instancias pertenecientes a la clase C que se clasifican correctamente en la clase C

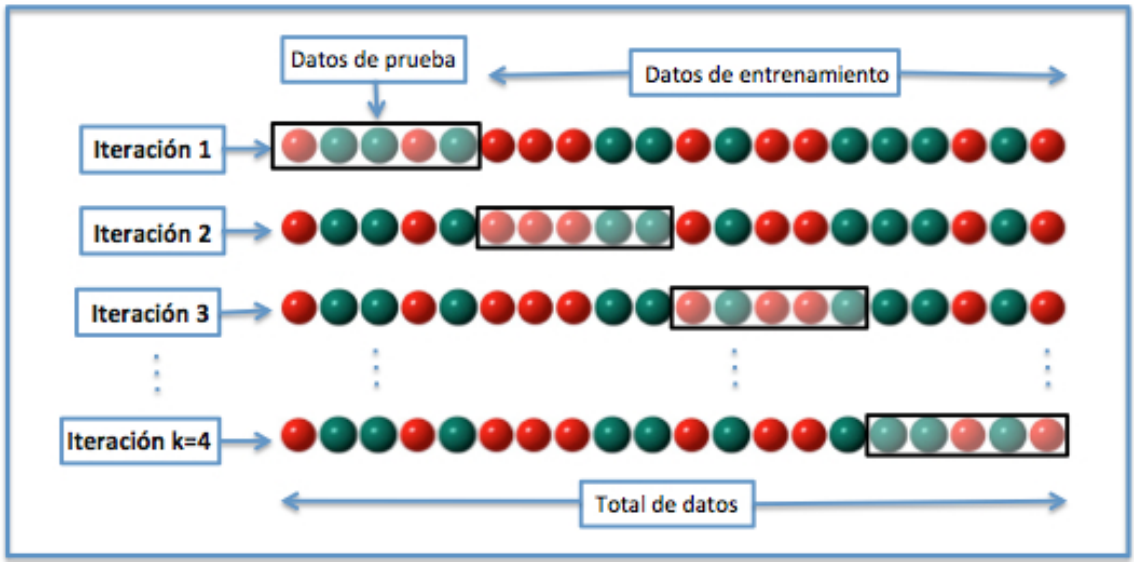

Figura 6.6: Validación cruzada

- True Negatives (TN) para la clase C: son instancias no pertenecientes a la clase C y que no se clasifican como clase C
- False Positives (FP) para la clase C: son instancias no pertenecientes a la clase C pero que se clasifican como clase C
- False Negatives (FN) para la clase C: son instancias pertenecientes a la clase C pero que no se clasifican como clase C

Para realizar la medición y poder hacer una evaluación mas precisa de los clasificadores se utilizan las siguientes medidas:

- Tasa de acierto: Esta medida es muy sencilla y ofrece una visión aproximada sobre la efectividad de los clasificadores. Nos proporciona el porcentaje de documentos que han sido categorizados correctamente sobre el total.

$$\text{Tasa de acierto} = \frac{Documentos\ clasificados\ correctamente}{Total\ documentos\ a\ clasificar} * 100$$

- Precision para la clase C: Mide que las instancias clasificadas como clase C sean realmente de la clase C, aunque haya instancias de la clase C que se clasifiquen como otra clase. Es un valor entre 0 y 1. Su valor aumenta cuando hay pocos falsos positivos [Figura 6.7].

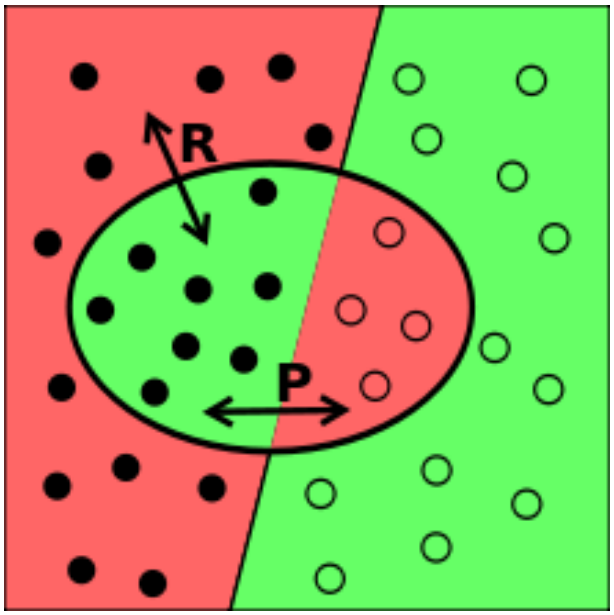

Figura 6.7: Precision y recall

$$\text{Precision} = \frac{TP}{TP+FP}$$

- Recall para la clase C: Mide que las instancias de la clase C se clasifiquen como clase C, aunque otras instancias también se clasifiquen como clase C sin serlo. Es un valor entre 0 y 1. Su valor aumenta cuando hay pocos falsos negativos [Figura 6.7].

$$\text{Recall} = \frac{TP}{TP+FN}$$

6.3.4. Análisis

En nuestro caso vamos a aplicar 5-fold-cross-validation, es decir, se crea un algoritmo que divide el conjunto de datos en 5 partes, y coge $\frac{1}{5}$ para test y $\frac{4}{5}$ para training. Finalmente se aplica la media sobre las 5 iteraciones de los resultados obtenidos.

La primera métrica que se ha analizado es precision [Figura 6.8], como se puede observar en los resultados obtenidos, SVM obtiene una mejor precision para todas las categorías excepto para la categoría Deportes, aunque ambos resultados son realmente buenos en la misma.

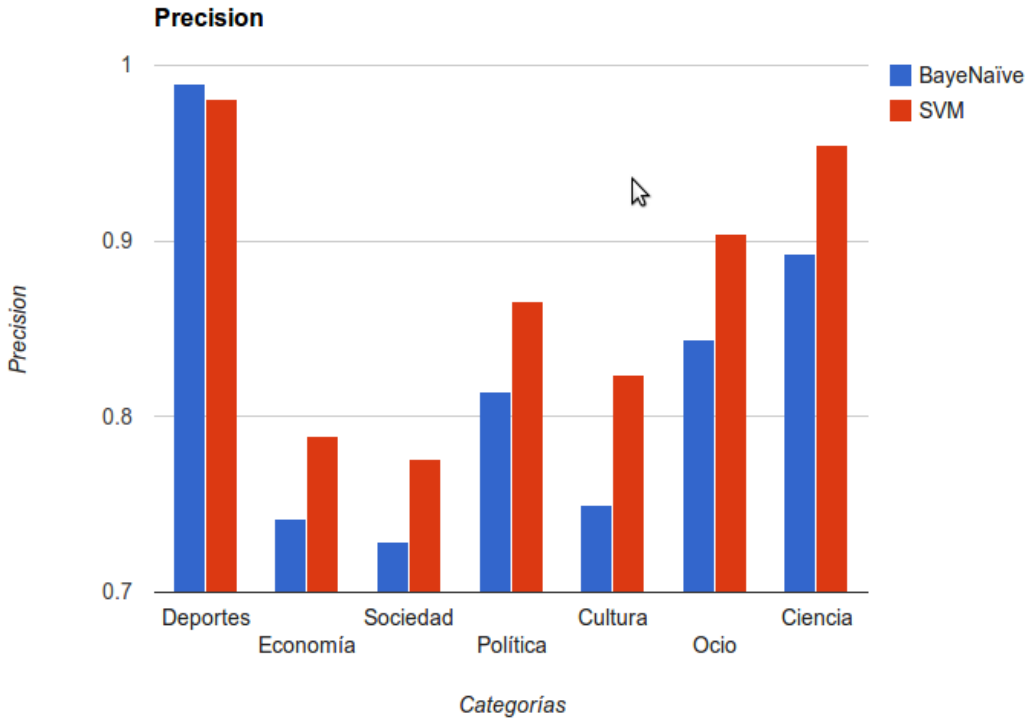

Figura 6.8: Análisis de precision

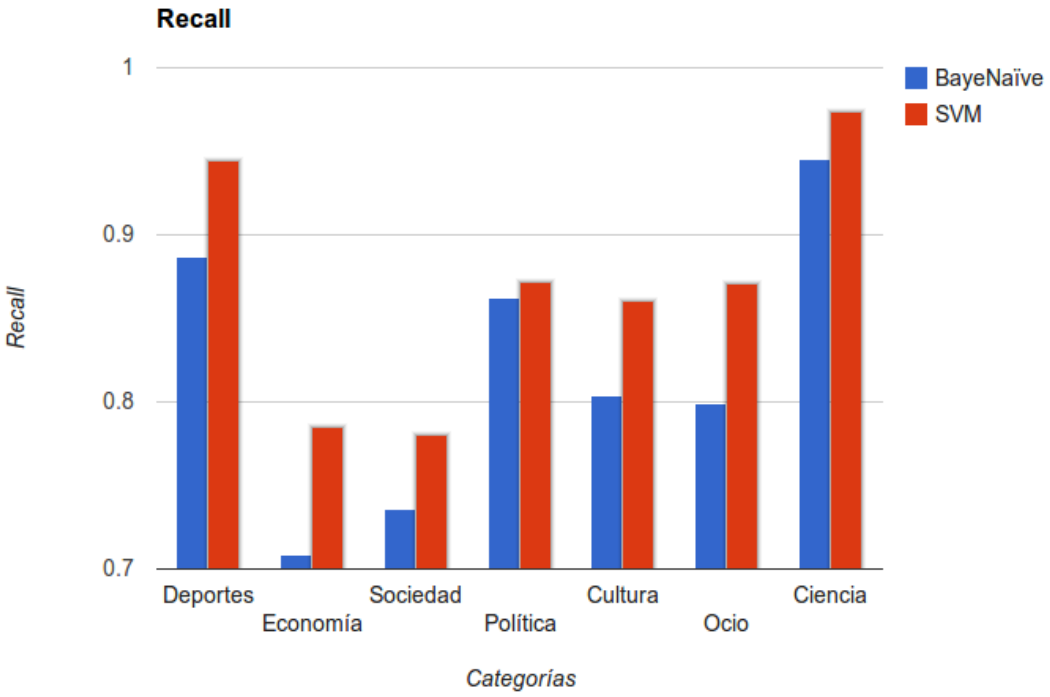

Figura 6.9: Análisis de recall

Las segunda métrica que analizada es recall [Figura 6.9], a diferencia de la precision, en este caso SVM obtiene mejores resultados en la totalidad de las categorías. Cabe destacar en este caso las categorías como Economía o Sociedad en las que el clasificador Bayesiano obtenía resultado poco satisfactorios, SVM compensa ligeramente esa carencia.

Finalmente la última métrica que se ha analizado es la tasa de acierto [Figura 6.10]. Esta medida permite ver claramente que clasificador obtiene mejores resultados y nos ofrece una visión global de este análisis un poco más intuitiva.

Por lo tanto, el clasificador que se ha decidido elegir después de este estudio realizado ha sido Support Vector Machine.

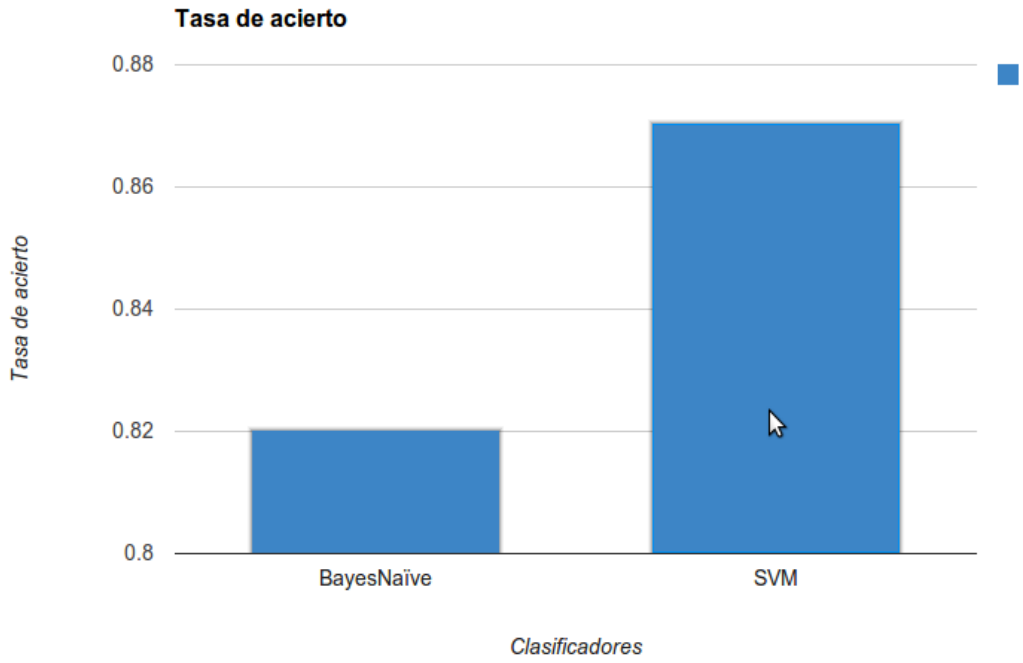

Figura 6.10: Análisis de tasa de acierto

Capítulo 7

Desarrollo de la Web

Índice general

En este capítulo se va a describir todo el proceso que se ha llevado a cabo para el desarrollo y la implementación de la aplicación web.

En un primer lugar de desarrolla en front-end de la aplicación, así como toda parte que corresponde con la gestión de contenidos (listar contenidos, gestión de usuarios y buscador). En una segunda parte se implementan las funcionalidades de valorar y compartir contenidos, así como mostrar la portada personalizada para cada usuario. En esta parte también se integran los dos módulos anteriormente descritos: wrapper y clasificador.

Por tanto, esta parte del proyecto de desarrolla en dos sprints, que son los siguientes:

- Front-end y gestor de contenidos.
- Funcionalidades restantes e integración de módulos.

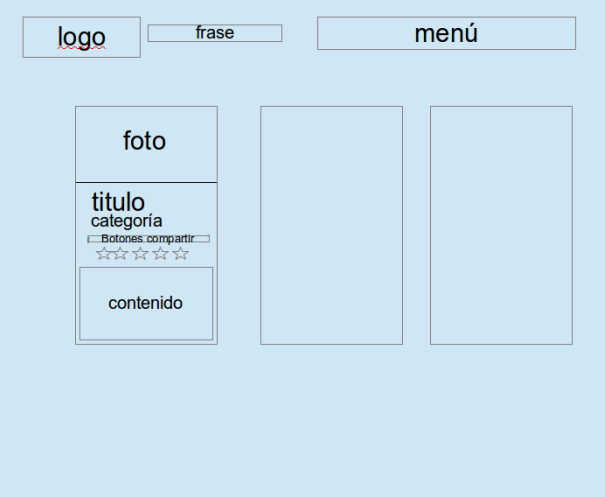

Figura 7.1: Mock-up 1

7.1. Quinto sprint: Front-end y gestor de contenidos

En este sprint se va a realizar el diseño gráfico de la interfaz web así como todo lo relativo a gestión de las noticias y de los usuarios.

7.1.1. Front-end

Inicialmente se comienza la iteración esbozando algunos mock-ups para intentar visualizar una idea sobre como podría ser el resultado final de la aplicación, tratando sobretodo, el tema de la usabilidad. Se trata de diseñar una página que sea atractiva para el usuario, y sobretodo, que tenga muy buena facilidad de uso.

Los mock-ups resultantes se pueden ver reflejados en las figuras 7.1 7.2 y 7.3.

Como se puede comprobar, desde un principio se ha procurado la adaptación de la página a los distintos dispositivos que puede haber en el mercado.

- La figura 7.1 muestra una imagen de como sería el resultado de la plantilla que será encargada de visualizar las noticias por páginas, esta plantilla es reutilizable tanto para la portada como para visualizar las noticias por categorías. En la cabecera se muestra el logo, una frase que pueda resumir brevemente la temática de la web y un menú de navegación; a continuación de presentan las noticias en tres columnas.

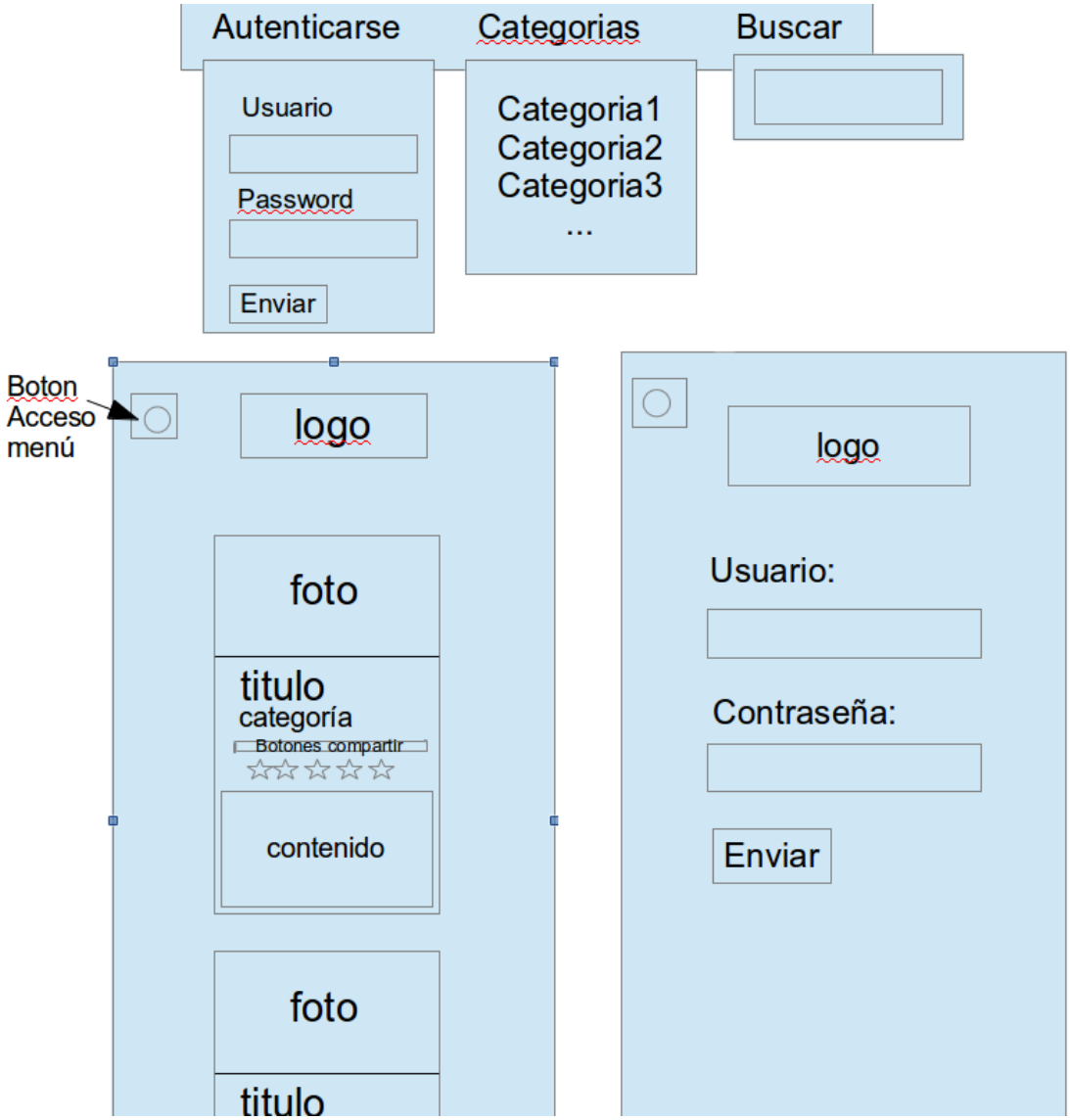

Figura 7.2: Mock-up 2

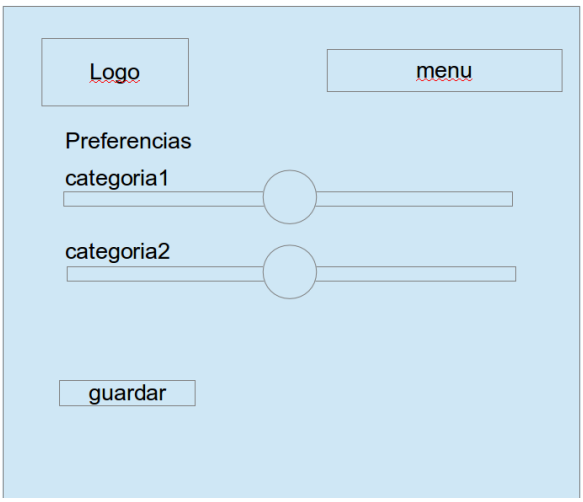

Figura 7.3: Mock-up 3

- En la figura 7.2 se puede ver como se ha pensado el menú inicialmente, cuenta con un apartado para que el usuario pueda autenticarse, una sección para navegar a mostrar las noticias de una categoría, y un buscador para realizar búsquedas concretas. Además del menú, se presentan dos representaciones de como sería la versión adaptada para teléfonos móviles.

- La figura 7.3 está relacionado la personalización de la portada, esta pantalla corresponde a la de ajustes del usuario. El usuario puede elegir mediante un control de tipo range"[1] que grado de interés se tiene por cada una de las categorías.

Una vez realizados los bocetos se procede a la implementación de la interfaz utilizando el framework skel.js para la definición del la estructura global.

Ya terminada la maquetación se procede a aplicar los estilos CSS, para ello se hace una recopilación de ellos en páginas que proporcionan recursos web, así como otros que se visualizan por la web y se intentan copiar.

El resultado final se puede ver en la figura 7.4, figura 7.5 y figura 7.6. Una plantilla estática que carece de funcionalidad, pero nos proporciona un gran prototipo para poder diseñar el back-end.

[1]Control HTML que consiste en una rueda que se desliza de izquierda a derecha y viceversa

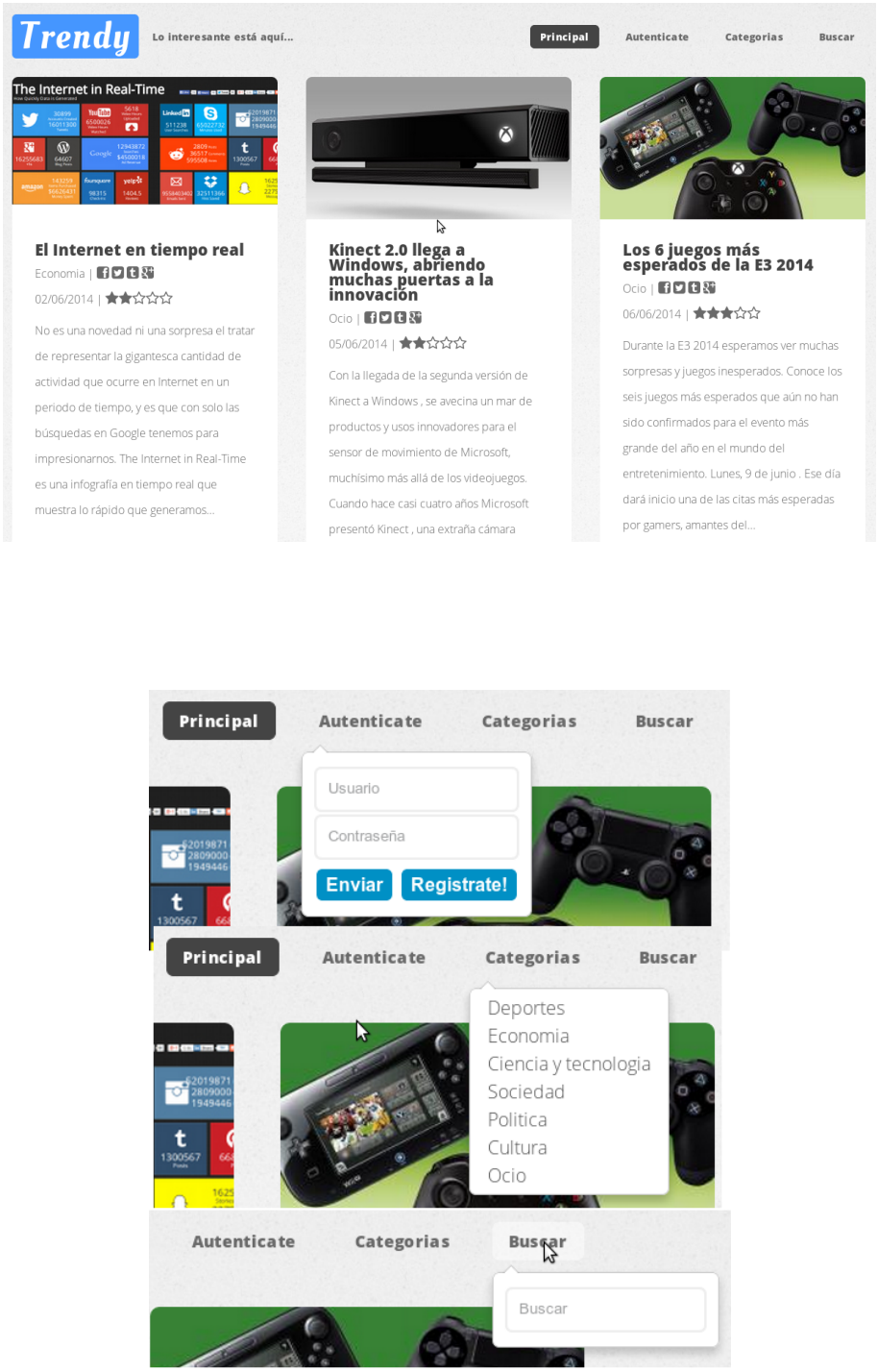

Figura 7.4: Diseño de personal computer

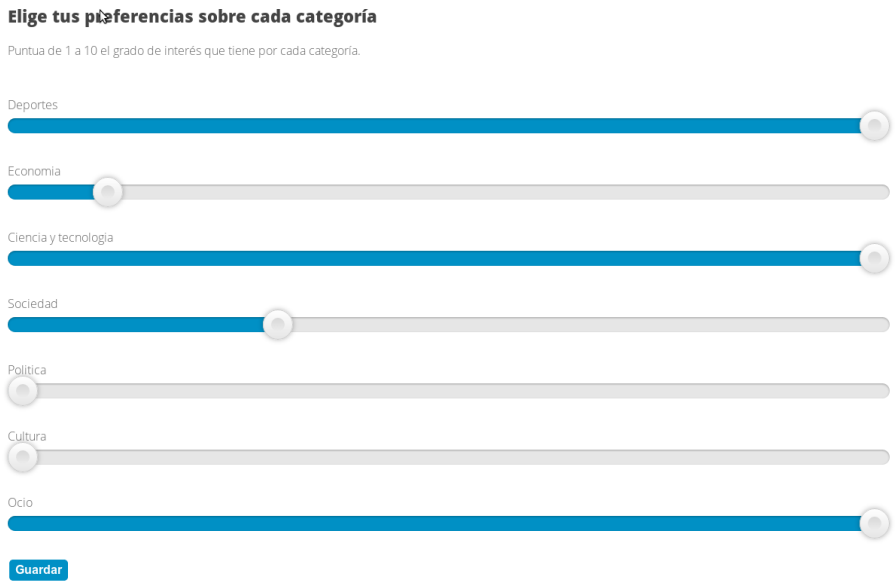

Figura 7.5: Ajustes de categorías

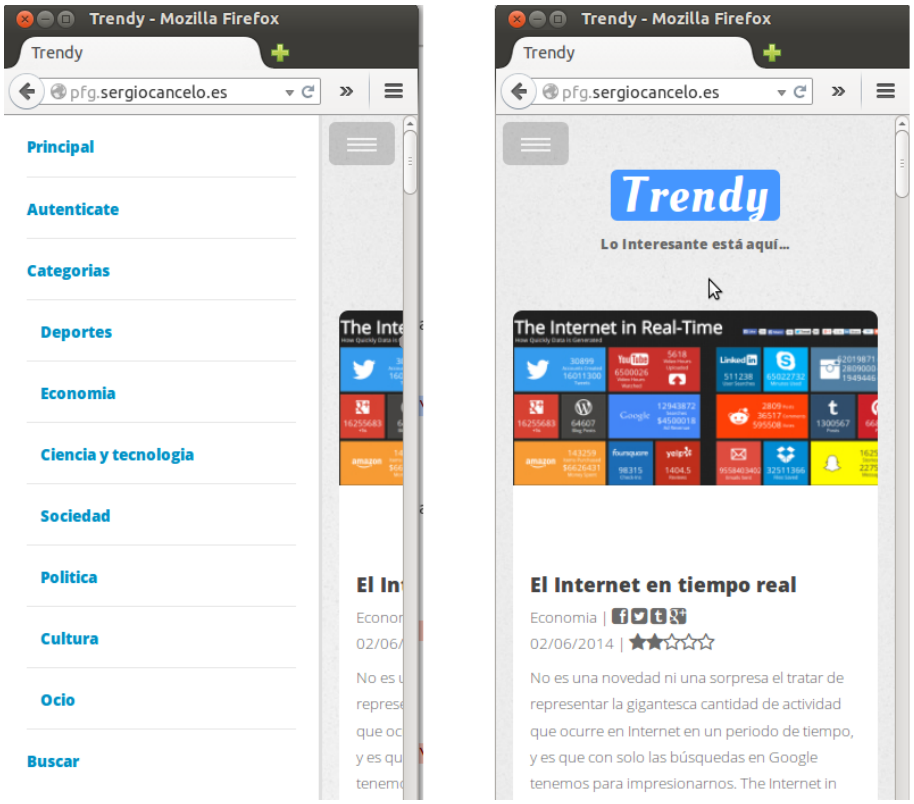

Figura 7.6: Diseño de móbil

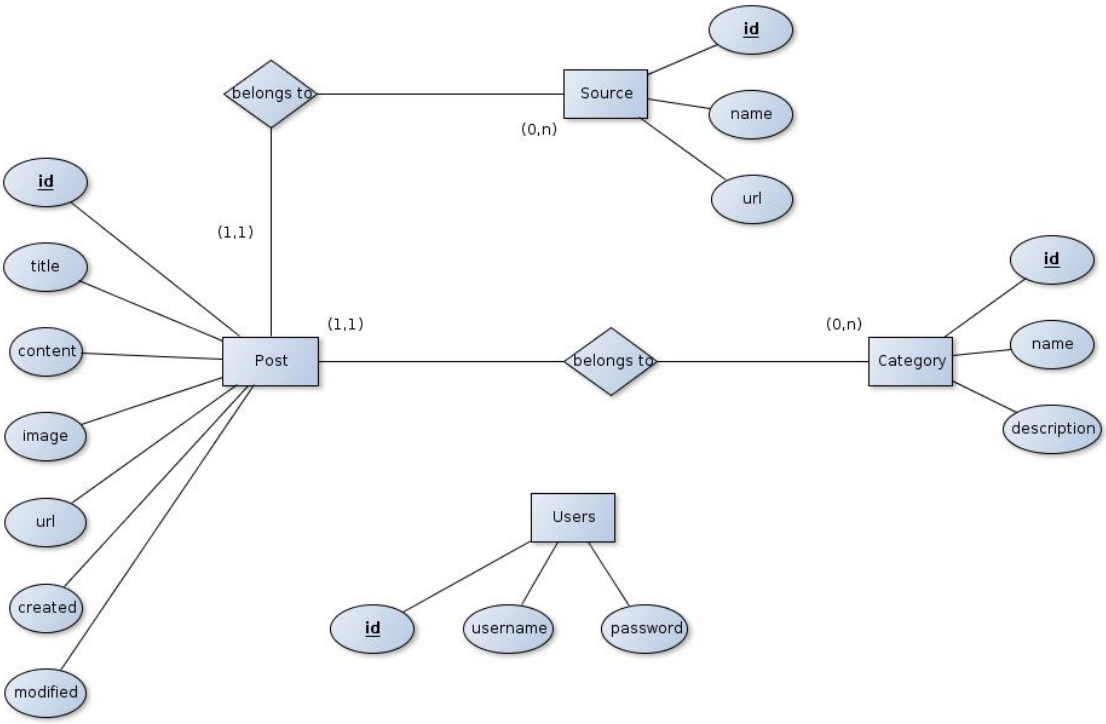

Figura 7.7: Entidad-relación

7.1.2. Gestor de contenidos

7.1.2.1. Desarrollo del modelo

Una vez definidas las plantillas podemos deducir qué datos necesita exactamente el gestor de contenidos, por tanto esbozamos un diagrama entidad-relación que se puede visualizar en la figura 7.7.

Nótese que este modelo no va a ser convertido directamente a SQL en la base de datos, sino que vamos a utilizar una herramienta que nos proporciona ActiveRecord, cuyo no es ActiveRecord migrations.

Cuando creamos una migración [Figura 7.8] estamos creando una clase ruby que hereda de Migration y nos proporciona independencia de la base de datos que vamos a utilizar, mas tarde ejecutando la migración, se genera automáticamente el sql necesario para crear la tablas. De este modo se construye todo el modelo de la base de datos.

El siguiente paso es crear los modelos de activerecord, para que el ORM pueda hacer el mapeo de las tablas de la base de datos a clases correctamente. Para ello

```
class CreatePosts < ActiveRecord::Migration
  def change
    create_table :posts do |t|
      t.string :title
      t.text :content
      t.string :image
      t.string :url
      t.references :category
      t.references :source

      t.timestamps
    end
    add_index :posts, :title, unique: true
    add_foreign_key :posts, :categories
  end
end
```

Figura 7.8: Migración

se crean las clases correspondientes a la figura 7.9.

Dentro de estas clases que representan la base de datos se definen las relaciones como están especificadas en el entidad-relación y la validación de datos.

La validación de datos solo se realiza en clases donde se producen inserciones o actualizaciones, en este caso, las clases Source y Category no es necesario realizar validaciones, las restantes, se validan del siguiente modo:

USER

- username: no nulo y ser único.
- password: no nulo

POST

- título: no nulo y ser único.
- contenido: no nulo y ser mayor de 20 caracteres.
- imagen: no nulo y ser una url.
- url: no nulo y ser una url.
- category_id: no nulo, y debe ser numérico.
- source_id: no nulo, y debe ser numérico.

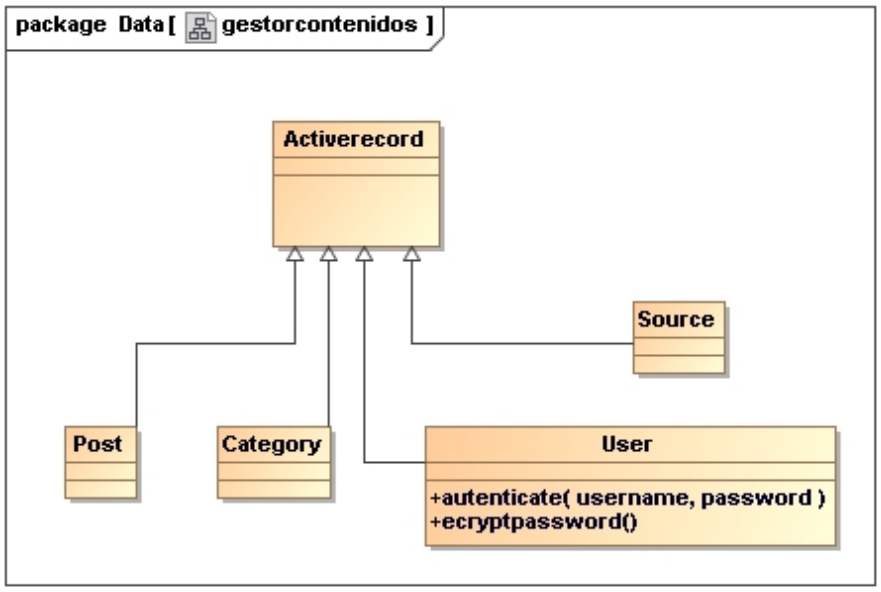

Figura 7.9: Modelo

7.1.2.2. Test del modelo

Para la realización de los test previamente es necesario crear fixtures, que no son mas que archivos que contienen datos de prueba que se insertan en base de datos antes de la ejecución de los test. Los casos de test que se testean en estos casos son test que comprueban que las reglas de validación han sido insertadas correctamente y, a mayores, también es necesario testear las funciones que se han creado en los modelos, en este caso, solo se ha creado la función autenticate, por tanto solo se realiza un caso de prueba a mayores de los anteriores para esta función.

7.1.2.3. Desarrollo de los controladores

Se implementan los siguiente controladores[Figura 7.10]:

- users_controller: es el encargado de gestionar las operaciones relacionadas con los usuarios, en este caso los usuarios solo se pueden dar de alta, por tanto es la única operación que implementa.

- sessions_controller: es que gestiona el manejo de las sesiones, posee dos operaciones, autenticar usuario y desautenticar usuario, este controlador se

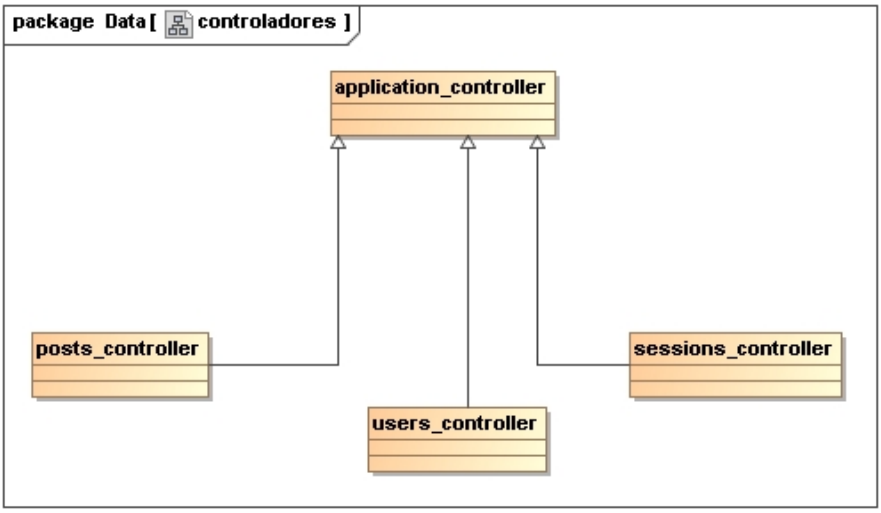

Figura 7.10: Controladores

apoya en el modelo de User, ya que necesita realizar comprobaciones de datos como el usuario y el password.

- posts_controller: Es el encargado de gestionar la visualización e las noticias, las operaciones que implementa en este caso solo las de listar noticias por categoría, y buscar noticias por título.

7.1.2.4. Test de los controladores

Al igual que en los modelos, para testear los controladores, se testean aquellas operaciones que realizan lógica mas allá de simples delegaciones en los modelos, por tanto se realizan los siguientes test.

- Para el controlador de sesiones se testean las funciones de autenticación y desautenticación de usuarios, se comprueba que la variable que controla las sesiones tome el estado correcto.
- Para el controlador de posts se testea la función de buscar por título y se testea que todas las noticias extraídas coinciden con los criterios de búsqueda.

7.1.2.5. Adaptación de la interfaz

Testeados modelos y controladores, lo siguiente que hacemos es generar un layout[2] como base para posteriormente situar cada una de las vistas, teniendo la interfaz gráfica diseñada con antelación nos facilita mucho la labor, simplemente se cambian los datos de ejemplo estáticos y se aplica la impresión de los datos en la plantilla con codigo ruby, la ruta de los archivos CSS y JS generados también se genera con una función que proporciona Rails, esto es debido a cuestiones de enrutamiento.

Las vistas necesarias son las siguientes:

- Listar noticias por categoría
- Mostrar resultados de una noticia buscada.
- Registrar un usuario.
- Autenticar usuario: esta vista solo se genera para ser utilizar en la versión móvil, debido a que en la versión de escritorio el formulario de autenticación se muestra mediante un cuadro desplegable, esto no es posible visualizarlo en la versión de móvil por lo que es necesaria su creación.

Al igual en el caso de los layouts, en el caso de las vistas solo es necesarios imprimir los datos, eliminar los datos estáticos y enrutar los formularios a la ruta correcta del controlador para que se envíen los datos.

7.2. Sexto sprint: Funcionalidades restantes e integración de módulos

7.2.1. Funcionalidades restantes

Las funcionalidades restantes por implementar de la aplicación web son las siguientes:

- Votar una noticia.
- Compartir en redes sociales.

[2]Plantilla base en la que se cargan las vistas, proporciona la base de la web

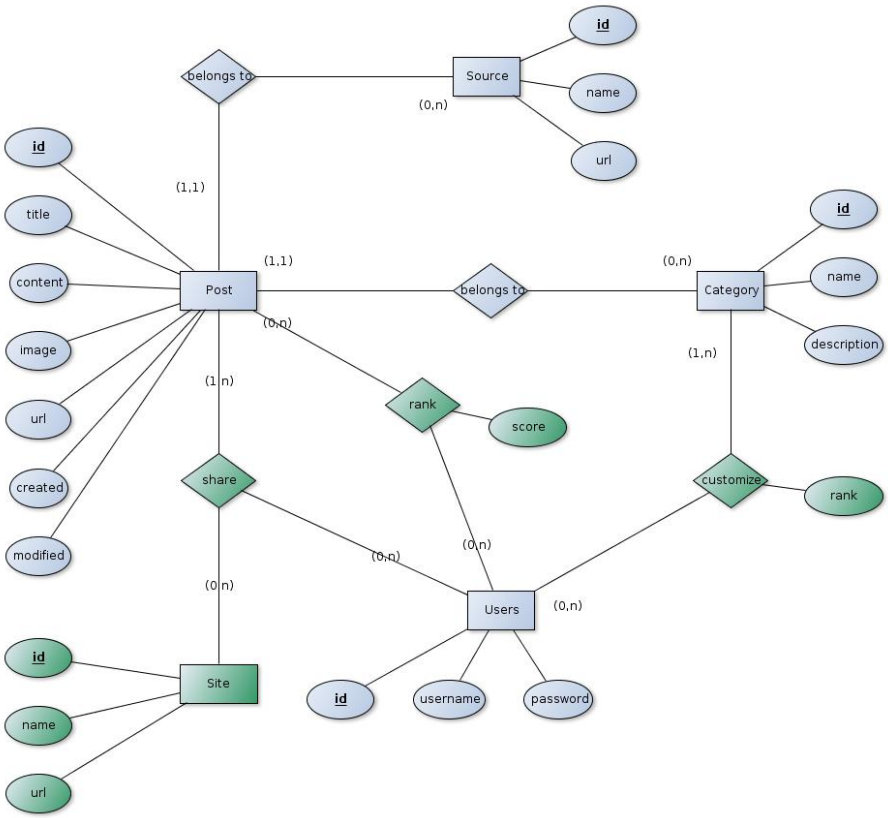

Figura 7.11: Entidad-Relación

- Mostrar las noticias de forma personalizada al usuario.

7.2.1.1. Desarrollo del modelo

Para la implementación de estas funcionalidades es necesario añadir a nuevas migraciones, concretamente las tablas mostradas de color verde de la figura 7.11

Una vez aplicadas las nuevas tablas creamos las nuevas entidades como objetos de activerecord para ser mapeadas. El resultado obtenido se puede ver en la figura 7.12

Con respeto a la validación de datos, en este caso es necesario realizar validación en las nuevas entidades, excepto en la entidad Site, ya que no se realizarán inserciones ni actualizaciones. Las validaciones son las siguientes:

SHARE

- post_id: no nulo, debe ser un número.

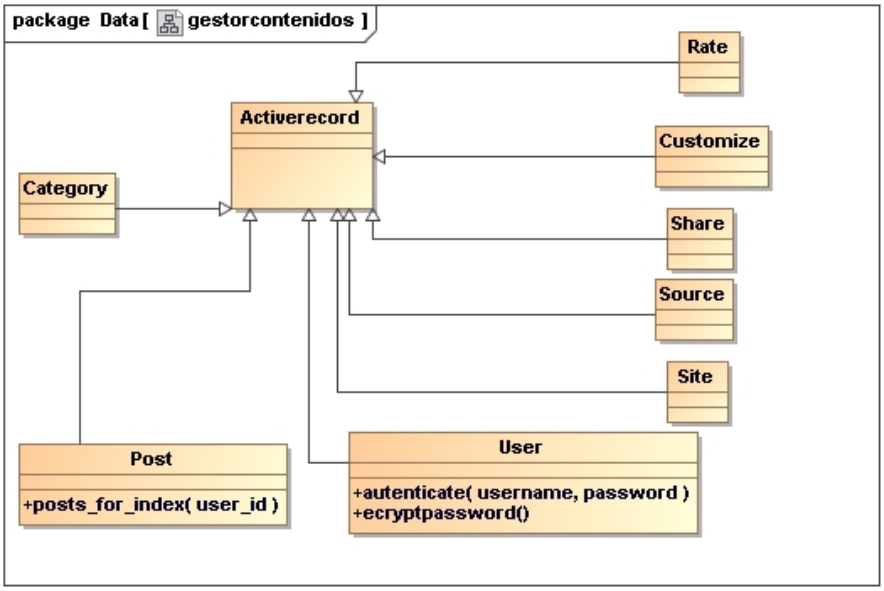

Figura 7.12: Modelo

- user_id: no nulo, debe ser un número.
- site_id: no nulo, debe ser un número.
- Los tres campos anteriores deben ser únicos en conjunto.

RATE

- post_id: no nulo, debe ser un número.
- user_id: no nulo, debe ser un número.
- score: no nulo, debe ser un número entre 1 y 10.
- post_id y user_id deben ser únicos en conjunto.

CUSTOMIZE

- category_id: no nulo, debe ser un número.
- user_id: no nulo, debe ser un número.
- rank: no nulo, debe ser un número entre 1 y 10.

A continuación se crea una consulta en el modelo Post para extraer las noticias de la portada en función de los intereses del usuario.

La consulta cuenta para cada categoría las noticias que más veces han sido compartidas en las redes sociales así como también la media de las valoraciones que tenga la noticia y las ordena por, número de comparticiones en las redes sociales + media de valoraciones en la noticia, una vez obtenido esto se cargan las preferencias que tiene el usuario en cada categoría categoría, y se muestran tanta noticias en la portada como porcentaje de interés tenga el usuario en esa categoría, es decir, si en la portada se muestran 15 noticias y el usuario elige que quiere ver 70 % deportes 30 % Ciencia y tecnología se mostrarán 10 noticias de deportes y 5 de Ciencia y tecnología.

7.2.1.2. Test del modelo

Como en sprints anteriores, se testean que las validaciones estén correctas y entre la información a la base de datos de manera consistente. En este caso, es necesario crear un nuevo este para la entidad Post para la nueva consulta añadida y se comprueba que obtiene los resultados deseados.

7.2.1.3. Desarrollo de los controladores

Se implementan los siguientes controladores:

- shares_controller: Es el encargado de controlar los shares en las redes sociales. En este caso solo tiene la función de crear un share.
- rates_controller: es el que gestiona la votación de noticias, al igual que el anterior solo implementa la operación de inserción.
- customizes_controller: Se encarga de controlar la creación de preferencias sobre el grado de interés que el usuario tiene en cada categoría, y de su edición, por tanto, implementa las operaciones de crear y editar.

Los controladores shares y rates no devuelven datos en sus respuestas, solo códigos de error, esto es debido a que las peticiones entrantes son mediante AJAX[3].

[3]Advanced Javascript And XML

7.2.1.4. Test de los controladores

Se van a testear aquellas operaciones que realizan lógica más allá de simples delegaciones en la capa modelo, por lo tanto se realizan los siguientes test:

- Para el controlador de shares y rates se comprueba que se inserte correctamente y además se realiza un test para cada unos de los códigos de error que devuelve.
- EL controlador customizes se testea que las opciones se puedan crear y editar correctamente, y además se comprueba de que la transacción que utiliza realiza la operación en un único bloque.

7.2.1.5. Adaptación de la interfaz

Para esta funcionalidades las únicas vistas es es necesario crear son las siguientes:

- Mostrar noticias personalizadas en la portada.
- Mostrar la preferencias del usuario y permitir actualizarlas: La visión y actualización de estas se realiza en una única vista.

Además de esto se crean dos archivos javascript que hacen los siguiente:

- Para la votación captura el evento click en las estrellas de valoración y envía la petición de votación junto con el id del post y el id del usuario. También es encargado de procesar el mensaje de error en pantalla.
- Para compartir en redes sociales, inicialmente envía una petición al servidor contando que se ha compartido esa noticia y, una vez es aceptada por el servidor, abre una ventana que nos lleva a la url de la red social donde va a ser compartido el contenido.

7.2.2. Integración de módulos

Una vez terminada la implementación de la aplicación web es hora de integrar todos los componentes. La integración se realiza en los siguientes pasos:

- Se integra el clasificador SVM con el modelo de datos de la aplicación web para que el conversor de noticias a vectores de características pueda acceder a la tabla del conjuntos de entrenamientos.
- El conversor de datos a vectores de caraterísticas, guarda ficheros de los vectores, en este caso, se cambian las rutas anteriores por otras relativas dentro de Rails. Se crea una tarea de rake, que es el sistema de ejecución de tareas que posee rails, para convertir los datos, de esto modo se arranca usando *rake classifier:dataconvert*
- Se crea una tarea para la entrenar el clasificador. *rake classifier:train*
- El clasificador se integra dentro de wrapper, cuando la noticia es procesada para su inserción en base de datos se aplica una instancia de SVM para predecir la categoría en la que se situará la noticia.
- El wrapper se integra con el modelo de la aplicación web para que guarde las noticias en base de datos. También se crea una tarea rake para arrancarlo, de este modo es posible arrancar el proceso wrapper ejecutando: *rake wrapper:start*.

Capítulo 8

Conclusiones y trabajo futuro

Índice general

8.1. Conclusiones

El objetivo de proyecto era la creación de un enlace de noticias que permitiera al usuario tener un espacio simple y reducido donde estar informado de las noticias de la actualidad. La metodología aplicada y el uso de patrones arquitectónicos como MVC, facilitaron en gran medida la ampliación de nuevas funcionalidades a la web y se reducen los errores en los componentes gracias a los test.

Inicialmente cuando se realizó la planificación de proyecto había mucha incertidumbre por mi parte sobre la clasificación de noticias, ya que solo tenía conocimientos básicos sobre técnicas de clasificación y se convertía en una situación a ciegas.

Ahora, finalizado el proyecto, puedo decir que este proyecto ha superado con creces todas mis expectativas, especialmente en la parte del clasificador, conseguir un 87 % de tasa de acierto ha sido un camino duro y largo de recorrer pero, a la vez, una experiencia muy enriquecedora.

He adquirido muchos conocimientos de Inteligencia Artificial gracias a este proyecto, también, estoy realmente sorprendido de la velocidad de desarrollo que

se puede llegar a alcanzar utilizando el framework web Ruby on Rails.

Después de un largo tiempo sin utilizar algunas de estas tecnologías que ya conocía con anterioridad, este proyecto ha supuesto una reintegración para mi en este mundo, y he podido comprender la gran evolución que ha sufrido el mundo web desde entonces, así como las nuevas tecnologías que surgieron.

8.2. Trabajo futuro

La idea que pensé cuando comencé este proyecto le quedan, por cuestiones de tiempo, algunas funcionalidades por desarrollar.

Las funcionalidades que se proponen, y que pueden mejorar todavía más esta aplicación, son las siguientes:

- Por una parte, podría ser conveniente que el wrapper abarque más tipos fuentes de datos a mayores de RSS, como por ejemplo fuentes HTML. Los principales inconvenientes que presenta el formato RSS en internet no es el formato en sí, sino que las fuentes no presentan el mismo tipo de información vía RSS y HTML, en ocasiones, la información en formato en RSS de algunas fuentes es más pobre que en su versión HTML, lo que produce una debilidad en el proyecto. Para solventar este problema, sería necesario añadir una funcionalidad más a wrapper que permita extraer contenido de fuentes HTML mediante patrones y guardar el contenido en base de datos.

- Por otro lado, el conjunto de entrenamiento proporcionado para el clasificador puede que se quede obsoleto y en un futuro pierda eficiencia por diversos motivos: La forma de redacción evoluciona, los intereses de la gente cambian, aparecen nuevos conceptos que actualmente no están en el mercado. Por estos motivos, sería conveniente añadir una nueva funcionalidad que se integra en la aplicación web y en el clasificador, que permitiría al usuario darle realimentación al clasificador para su mejora continua. Para esta realimentación se integraría un sistema que permitiese al usuario clasificar una noticia correctamente si ésta no lo estuviese, con esto conseguimos que el clasificador reaccione a fallos de clasificación y mejore con el tiempo adaptándose a nuevas circunstancias.

- Finalmente, el sistema de mostrar las noticias de forma personalizada en función del grado de interés de usuario puede ser substituido por un sistema de recomendación de contenidos. Este nuevo sistema va más allá que el anterior observando que contenidos consume el usuario diariamente, a partir de ellos es capaz de extraer información de todo ese contenido y sugerirle nuevo contenido al usuario que le pueda interesar. Para ello, es necesario hacer un estudio sobre que algoritmos de filtrado de información, filtrado colaborativo y recomendación basada en contenido, sería conveniente aplicar a este problema.

Bibliografía

[01] Repositorio de Bayes-Naïve: *https://github.com/cardmagic/classifier*

[02] Método Pairwise para Support Vector Machine
http://courses.media.mit.edu/2006fall/mas622j/Projects/aisen-project/

[03] API Ruby on Rails *http://api.rubyonrails.org/*

[04] Event Machine API *http://rubydoc.info/gems/eventmachine/1.0.3/frames*

[05] Event Machine HTTP Request API
http://rubydoc.info/gems/em-http-request/1.1.2/frames

[06] Nokogiri API *http://nokogiri.org/Nokogiri/*

[07] Repositio de Rb-LibSVM *https://github.com/febeling/rb-libsvm*

[08] Documentación y repositorio de Ruby-stemmer
https://github.com/aurelian/ruby-stemmer

[09] Documentacion y repositorio de Fast-stemmer
https://github.com/romanbsd/fast-stemmer

[10] Documentación y repositorio de Foreigner
https://github.com/matthuhiggins/foreigner

[11] Documentación y repositorio de Will Paginate
https://github.com/mislav/will_paginate

[12] Documentación de Bcrypt *https://rubygems.org/gems/bcrypt-ruby*

[13] Documentación de Postgresql *http://www.postgresql.org/docs/*

[14] SCRUM *http://www.mountaingoatsoftware.com/agile/scrum*

[15] Cascade Style Sheet Reference *http://www.w3schools.com/cssref/*

[16] HTML Reference *http://www.w3schools.com/tags/*

[17] JSON Reference *http://json.org/*

[18] HTTP Error Code Reference *http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html*

[19] Documentación de Bundler *http://bundler.io/*

[20] Documentación de Rake *https://github.com/jimweirich/rake*

[21] *Programming Ruby 1.9 - Dave Thomas*

[22] *Agile Web Development with Rails 4 - Sam Ruby, Dave Thomas y David Heinemeler Hansson*

[23] *Algorithms of the Intelligent Web - Haralambos Marmanis y Dmitry Babenko*

[24] *Data mining - Ian H. Witten & Eibe Frank*

[25] *Collective Intelligence in Action - Satnam Alag*

Apéndice A

Glosario de acrónimos

SVM *Support Vector Machine.*

LIBSVM *Support Vector Machine Library.*

RoR *Ruby On Rails.*

MVC *Model View Controller.*

ORM *Object-Relational Mapping.*

CSS *Cascade Style Sheet.*

AJAX *Advanced Javascript And XML*

JS *JavaScript*

RSS *Really Simple Syndication.*

TF *Term Frecuency*

IDF *Inverse Document Frecuency*

Apéndice B

Glosario de términos

ActiveRecord Patrón de diseño arquitectural que indica que un objeto incluye operaciones CRUD en su interfaz, se utiliza habitualmente para persistencia de objetos.

API (Aplication Program Interface) Ofrece un componente software para interactuar con el.

CRUD (Create Read Update Detele) Operaciones básicas de creación, lectura actualización y eliminación de datos.

CSS (Cascade Style Sheet) Lenguaje empleado para definir los estilos de la presentación de un documento estructurado en HTML o XML.

JSON Formato de serialización de datos.

MVC (Model View Controller) Patrón de diseño arquitectural que define la separación entre la estructura de los datos (Model), la lógica de negorcio (Controller) y las representación de la información(Vista).

ORM (Mapeador Objeto-Relacional) Es una técnica que convierte los datos entre el sistema de tipos de un programa desarrollado en orientación a objetos y una base de datos relacional.

Ruby Es un lenguaje de programación interpretado, reflexivo y orientado a objetos. Combina una sintaxis inspirada en Python y Perl con características de programación orientada a objetos similares a Smalltalk.

SVM (Support Vector Machine) Librería utilizada en problemas de recuperación de la información. Esta está incluida dentro de la categoría de clasificadores.

Wrapper Software encargado de extraer información particular de una o más fuentes de datos.