UNIVERSIDAD
REY JUAN CARLOS

MÁSTER OFICIAL

EN SISTEMAS TELEMÁTICOS E INFORMÁTICOS

Curso Académico 2008/2009

Trabajo de Fin de Máster

FAVS, UNA HERRAMIENTA PARA LA CREACIÓN Y GESTIÓN DE PLANETAS DE BLOGS DOCENTES

Autor : Jesús Moreno León

Tutor : Gregorio Robles Martínez

Resumen

Las bitácoras (o blogs) han supuesto una revolución en Internet, y también en las aulas de todo el mundo. Blogs institucionales, profesionales, de estudiantes o de aula son una herramienta utilizada en todos los niveles educativos.

En los últimos tiempos han surgido sitios web que enlazan historias de varios blogs, popularmente denominados planetas, y otras herramientas que permiten que aparezcan artículos, noticias e historias publicadas en diferentes bitácoras en una única página web. De esta manera, se pueden encontrar planetas de las más diversas temáticas (p.ej., sobre la crisis financiera) o de colectivos (p.ej., personas que trabajan en la misma empresa).

Profesores de todo el mundo han comenzado a utilizar estas aplicaciones para sus clases, intentando aprovechar las posibilidades que ofrecen como herramientas didácticas. Sin embargo, todas estas aplicaciones presentan ciertas limitaciones para ser usadas en un aula docente, ya que no se crearon pensando en esta labor. Algunas requieren conocimientos sólidos de informática, otras exigen mucho tiempo para su puesta en marcha y mantenimiento, e incluso, algunas de ellas, requieren contar con un servidor web.

En este proyecto se presenta FAVS, una aplicación libre, pensada y diseñada para que cualquier docente, sin apenas conocimientos informáticos, pueda crear, mantener y gestionar un sitio web donde aparezcan los artículos escritos por sus estudiantes en sus bitácoras. Los alumnos podrán inscribirse en el planeta de la asignatura, leer los escritos de sus compañeros, votar las historias más interesantes -de manera que, si así lo deseara el docente, pudiera utilizar esta información para evaluar esta actividad (heteroevaluación)- y consultar las estadísticas de votos emitidos y recibidos.

Esta aplicación puede ser usada en cualquier asignatura de cualquier disciplina, ya que propiciará la creación y el intercambio de conocimiento, potenciará las habilidades escritoras de los estudiantes y motivará una sana competición por escribir los artículos más interesantes.

Índice general

Capítulo 1

Introducción

En este primer capítulo se introducirán los conceptos de blog y planeta de blogs, y se analizará su potencial en el ámbito educativo. A continuación se expondrán experiencias docentes desarrolladas con estas aplicaciones, las dificultades experimentadas durante su realización y los conocimientos y medios técnicos necesarios para su puesta en marcha.

Posteriormente, dado que el proyecto se distribuye con una licencia libre, se realizará una breve introducción al software libre, explicando sus características fundamentales. Se incluye, aunque de forma escueta, una descripción del concepto de comunidad virtual y se resume la forma en que los proyectos de software libre son gestionados habitualmente.

Se explicarán, finalmente, las características más importantes de Ruby on Rails, la plataforma de desarrollo en la que se ha realizado el proyecto.

1.1. Los blogs en la educación

Las bitácoras o blogs (que proviene de *web log*) son sitios web donde un autor, o varios autores, publican distintos textos (anécdotas, opiniones, artículos o comentarios). Los textos publicados aparecen ordenados en el blog de forma cronológica, siendo el primer texto el publicado más recientemente.

Los primeros blogs nacieron a mediados de los años 90, pero no se popularizaron hasta el año 1999, cuando se lanzaron herramientas gratuitas para su creación y mantenimiento. A partir de ese momento, los blogs han ido ganando protagonismo hasta convertirse en uno de los servicios más populares de Internet. Cantantes, periodistas, políticos, profesores, empresas

internacionales... los blogs se han extendido por toda la sociedad y han supuesto una revolución creativa en Internet[1].

Hasta el lanzamiento de este servicio, el mantenimiento y la actualización de una página web era una tarea ardua que requería ciertos conocimientos técnicos, lo que propiciaba que, prácticamente, sólamente profesionales de la informática se encargaran de aportar contenido a la web. Con esta nueva herramienta el panorama cambió por completo, ya que cualquier persona con acceso a Internet podía mantener su propio blog en el que podía escribir sobre cualquier temática. Surgió una relación entre iguales en la que todos los usuarios podían participar construyendo y compartiendo conocimiento.

Los planetas de blogs son sitios web que enlazan las historias publicadas en un conjunto de bitácoras que suelen compartir una temática común. Los artículos publicados en los distintos blogs enlazados en el planeta aparecerán en una única página web siguiendo un orden cronológico, de manera que la última historia publicada será la que aparezca en primer lugar.

Los blogs y los planetas presentan varias características que hacen de ellos una herramienta con un gran potencial educativo, por lo que profesores de todo el mundo han comenzado a utilizarlos en su labor diaria[2, 4]:

- Permiten que los estudiantes puedan construir su propio conocimiento de forma colaborativa, lo que se adapta a la perfección a las metodologías comunicativa y constructivista.
- Generan conocimiento libre, posibilitando el intercambio de ideas y los debates.
- Potencian las habilidades de escritura de los estudiantes.
- Permiten un aprendizaje continuado durante toda la vida, ya que es posible seguir aprendiendo fuera del aula, buscando y creando conocimiento en cualquier momento, lo que los convierte en una gran herramienta para la enseñanza a distancia.

1.2. Experiencias educativas con blogs

Los blogs permiten llevar a cabo actividades de e-learning que pueden desarrollarse en asignaturas de titulaciones técnicas y no técnicas en cursos universitarios de grado y post-grado, en las que los propios estudiantes participan en la evaluación de los contenidos aportados por los compañeros.

En la Tercera Jornada de Innovación Pedagógica del Proyecto ADA Madrid, profesores de la Universidad Rey Juan Carlos presentaron la siguiente experiencia docente de e-learning basada en el uso de blogs que habían llevado a cabo en varias asignaturas [3]:

Cada estudiante se creará un blog donde escribirá entradas relacionadas con la temática de la asignatura (aclaraciones, comentarios de noticias, anécdotas...). Todos los blogs serán enlazados desde el planeta de la asignatura. Los estudiantes podrán puntuar positivamente las entradas de los compañeros que consideren más interesantes.

El alumnado deberá escribir periódicamente en el blog (semanal o bisemanalmente), de forma que al final del cuatrimestre se tenga al menos una docena (o media docena) de artículos por alumno. Los blogs han de tener contenidos propios, y no se permitirán las entradas sin valor añadido que simplemente copien noticias de Internet.

Para la evaluación de la actividad se tendrá en cuenta los votos recibidos por los compañeros en el blog (por escribir) y los votos emitidos (por leer); por tanto, los estudiantes estarán motivados para escribir buenas entradas y para leer y votar las historias publicadas por el resto de alumnos. Los profesores de la asignatura podrán valorar positivamente otros elementos adicionales.

1.2.1. Infraestructura informática necesaria

Para que los alumnos pudieran darse de alta en la actividad, se diseñaron unos formularios en los que cada estudiante debía proporcionar su nombre, su dirección de correo eléctronico (que debía coincidir con la dirección usada en la plataforma Moodle de la asignatura) y el feed de su blog. Un feed es un medio de redifusión de contenido web que se utiliza para suministrar información actualizada del contenido de un sitio web.

Era necesario contar con un servidor web donde instalar el programa *Planet*[1] que se encargaba de enlazar las historias de los distintos blogs. El profesor debía construir un fichero de texto con las direcciones de los feed de los alumnos dados de alta y lo pasaba al formato del fichero de configuración de Planet.

Se modificó ligeramente la plantilla original que mostraba el planeta de manera que se mostrara un enlace para poder votar las historias. Este link enlazaba a una página de la Universidad donde se procedía a votar. Los datos de los votos se almacenaban en un fichero de texto que

[1] http://www.planetplanet.org/

eran leídos por una página en HTML que mostraba las estadísticas de votos emitidos y recibidos, como se muestra en la figura 1.1.


Blogs más votados

http://societatdelainformacio.blogspot.com 74
http://pmiturain.blogia.com 74
http://prensafuturo.blogspot.com 72
http://rgarciarui.blogspot.com 69
http://emaringo.blogspot.com 64
http://eudald-escriba.blogspot.com 63
http://lidiamunoz.wordpress.com 51
http://guillem-ferrer.blogspot.com 49
http://codolar.wordpress.com 46
http://feeds.feedburner.com 44
http://www.toni-felguera.es 41
http://fontesic.blogspot.com 39
http://evagarciatob.blogspot.com 39
http://duckmaster.wordpress.com 39
http://netorganic.wordpress.com 38
http://funtecsi.blogspot.com 36
http://unafinestraobertaalmon.wordpress.com 35
http://joanfundamentosuoc.blogspot.com 35
http://desconocimiento.wordpress.com 35
http://y-sociedadinformacion.blogspot.com 34
http://mrbala.blogspot.com 34

Figura 1.1: Página original con las estadísticas de votos emitidos y recibidos

1.2.2. Conclusiones y problemas detectados

Esta actividad se llevó a cabo en el transcurso de seis asignaturas de grado y post-grado de titulaciones no técnicas. De estas seis asignaturas, una era presencial y cinco a distancia -una de ellas dentro del programa ADA Madrid-, y todas ellas relacionadas directa o indirectamente con la informática y las nuevas tecnologías. El número de estudiantes total que participó en las mismas se encuentra en torno a los 250.

El resultado de la actividad fue bastante satisfactorio, ya que el alumnado participó, en líneas generales, de forma activa y los objetivos de creación de contenido e intercambio de ideas fueron alcanzados.

Respecto a los problemas detectados, por un lado, es evidente que los requisitos técnicos para preparar la actividad son bastante altos (conocimientos de administración de sistemas GNU/Linux, programación HTML, necesidad de contar con un servidor web), por lo que sólo profesores de carreras técnicas, con los conocimientos y el tiempo necesarios, podrían llevarla a cabo.

Por otra parte, el hecho de que los estudiantes tuvieran que identificar y proporcionar el URL (Uniform Resource Locator, es decir, localizador uniforme de recurso, o simplemente dirección web) del feed de su blog resultaba bastante problemático, ya que la mayoría no cursaban carreras técnicas y no se encontraban familiarizados con estos conceptos.

Con el objetivo de que cualquier profesor pudiera realizar esta actividad, y que la aplicación resultara los más amigable posible para estudiantes y profesores, nació el proyecto que aquí se

presenta.

1.3. Software libre

Todo programa que sea considerado software libre debe ofrecer una serie de libertades que se resumen en [5]:

- Libertad de usar el programa con cualquier fin, sin necesidad de comunicarlo a los desarrolladores.
- Libertad de estudiar el código fuente del programa y modificarlo adaptándolo a nuestras necesidades, sin necesidad de hacer públicas las modificaciones.
- Libertad de distribuir copias, tanto binarios como código fuente, modificadas o no, gratis o cobrando por su distribución.
- Libertad de modificar el programa y publicar las mejoras para beneficio de la comunidad.

Estas libertades conllevan una serie de factores que están cambiando el desarrollo de software tradicional. Por ejemplo, permiten que se pueda reutilizar gran cantidad de software que se encuentra disponible en repositorios públicos de Internet. Y no sólo eso, ya que también pueden consultarse las listas de correo y los foros utilizados por los desarrolladores del proyecto, donde puede encontrarse gran cantidad de información muy valiosa. De esta forma, el desarrollo de software se acerca a los procesos que se siguen en la investigación científica en general.

Así, el proyecto que aquí se presenta no comenzó su desarrollo desde cero sino que se apoyó en otros proyectos libres existentes, reutilizando partes de su código y añadiendo las funcionalidades necesarias para conseguir los objetivos propuestos. Como veremos en el capítulo 3, el proyecto no habría sido terminado sin esta posibilidad; los recursos (tanto económicos como humanos) necesarios para su desarrollo habrían impedido su finalización en los plazos previstos si se hubiera comenzado desde cero.

1.3.1. Gestión de proyectos de software libre

Las libertades que ofrecen los programas libres hacen que a su alrededor se creen comunidades de usuarios y desarrolladores que colaboran en la detección y corrección de errores,

en la solicitud y programación de nuevas funcionalidades y en otros muchos aspectos como la traducción de la interfaz o la documentación del proyecto a nuevos idiomas [9].

El objetivo de cualquier nuevo proyecto libre es conseguir una comunidad de usuarios y desarrolladores entorno al mismo que lo ayuden a crecer, a desarrollarse, de manera que el nivel de actividad llegue a ser tal que el desarrollo del proyecto sea autocatalítico, es decir, que la propia comunidad resuelva las necesidades que se plantean por sí misma.

Cómo conseguir esta comunidad no es un problema trivial y, aunque actualmente esta labor se lleva a cabo sin usar procesos ingenieriles, existen algunas buenas maneras de proceder para alcanzar este ansiado éxito.

En el capítulo tercero justificaremos las decisiones tomadas y explicaremos la labor realizada respecto a los siguientes aspectos:

- Elección de la licencia.
- El sitio web del proyecto. Estructura, idioma y contenido.
- Promoción del proyecto. Conferencias, artículos, blogs.

1.4. Ruby on Rails

Ruby on Rails[2], también conocido como RoR o Rails, es un framework de desarrollo de aplicaciones web de código abierto, escrito en el lenguaje de programación Ruby que sigue el paradigma de la arquitectura Modelo Vista Controlador [6].

Rails ofrece a los desarrolladores clases que implementan fuciones comunes que se usan en la mayoría de las aplicaciones web, encargándose de muchos detalles de bajo nivel que pueden resultar repetitivos y aburridos de programar, permitiendo al desarrollador concentrarse en la funcionalidad de la aplicación:

- Abstracción en el acceso a la base de datos, asegurando que las consultas funcionarán sin importar el Sistema Gestor de Base de Datos con el que se esté trabajando.
- Plantillas, reutilizando código de la capa de presentación por toda la aplicación
- Gestión de las sesiones de usuarios

[2] http://rubyonrails.org/

- Generación de URLs limpias y amigables para los buscadores

1.4.1. Principios fundamentales

Rails se apoya en varios principios que lo hacen distinto a otros frameworks de desarollo y que consiguen que los desarrolladores ahorren tiempo y esfuerzo:

- **Convención sobre configuración**. Este principio se refiere al hecho de que Rails asume un buen número de opciones por defecto respecto a la forma en la que debería construirse una aplicación web. Cuando la convención utizada es suficiente para conseguir los objetivos es innecesario realizar aquellas tareas para las que se ha definido un comportamiento. Sólo cuando la convención no se adapta a las necesidades de la aplicación, el desarrollador puede alterar el comportamiento por defecto y adaptarlo para lograr el deseado. Ejemplos de convenciones usadas en Rails son la nomenclatura de los objetos relacionados con la Base de Datos y el proceso por el que los controladores encuentran sus correspondientes modelos y vistas.

- **No te repitas** (Don't repeat yourself, DRY). Cuando un desarrollador decide cambiar el comportamiento de una aplicación que sigue el principio DRY, no debería modificar el código en más de un sitio. En lugar de copiar y pegar código con la misma funcionalidad en distintas partes de la aplicación, con Rails, esta funcionalidad se almacena una vez en un repositorio central y se referencia desde cada parte de la aplicación que lo necesita. De esta forma, se reduce la dificultad en los cambios y evolución del proyecto.

- **Desarrollo Ágil**. El desarrollo ágil usa un enfoque adaptativo. Pequeños grupos de desarrolladores van completando pequeñas partes del proyecto. Antes de comenzar cada iteración, el equipo reevalúa las prioridades para la aplicación que está siendo construida; estas prioridades pueden haber cambiado durante la última iteración, así que podrían necesitar algún tipo de ajuste. Este enfoque se adapta a la perfección al desarrollo de un proyecto de software libre como el que se presenta en esta memoria: los usuarios de la aplicación van informando de fallos o solicitando nuevas funcionalidades que el equipo de desarrollo se encargará de atender.

1.4.2. Ruby

Ruby[3] es un lenguaje de programación libre, interpretado y orientado a objetos, creado por el programador japonés Yukihiro Matsumoto en el año 1995. Sus creadores lo definen como *un lenguaje de programación dinámico y de código abierto enfocado en la simplicidad y productividad. Su elegante sintaxis se siente natural al leerla y fácil al escribirla.*

Como lenguaje interpretado, Ruby no requiere de una fase de compilación, sino que usa un intérprete (un programa que se ejecuta en el servidor web) que traduce el código fuente en lenguaje máquina sobre la marcha. Cada vez que se invoca un código (es decir, cada vez que se sirve una página web) el código es traducido.

En Ruby todo es un objeto. Todos los tipos de datos son un objeto y todas las funciones son métodos, y se le pueden asignar propiedades y acciones a toda información y código.

Ruby es considerado un lenguaje flexible, ya que permite a sus usuarios alterarlo libremente. Sus partes esenciales pueden ser quitadas o redefinidas a placer. Se puede agregar funcionalidad a partes ya existentes.

Desde su liberación pública en 1995, este lengaje ha ido atrayendo cada vez más desarrolladores y, actualmente, cuenta con grupos de usuarios activos en las ciudades más importantes del mundo. Ruby-Talk, la lista te correo principal del lenguaje, recibe 200 mensajes diarios. El índice TIOBE, que mide el crecimiento de los lenguajes de programación, ubica a Ruby en la posición número 13 del ranking mundial.

1.4.3. El patrón de diseño MVC

Un patrón de diseño en ingeniería del software es un modelo formal que es aplicable a diferentes dominios. Existen problemas que, aunque pertenecen a distintos ámbitos, son semejantes desde el punto de vista de la estructura lógica de la solución. El patrón de diseño es esta estructura común que tienen diversas aplicaciones.

El patrón que sigue el framework Ruby on Rails, y que se ha utilizado en la construcción de este proyecto, es el patrón Modelo Vista Controlador, conocido como patrón MVC. Este patrón divide la aplicación en los siguientes componentes:

- **Modelo**. Para manejar los datos y la lógica de negocio.

[3] http://www.ruby-lang.org/es/

- **Controlador**. Se encarga de redirigir un procesamiento determinado por cada petición recibida. Gestiona la lógica de la aplicación.
- **Vista**. Maneja los objetos gráficos de la interfaz de usuario y se encarga de la lógica de la presentación.

Esta separación implica que una petición de un usuario sea procesada como sigue:

1. El navegador, desde el lado del cliente, envía la petición de una página al servidor, que será atendida por el controlador apropiado.
2. El controlador recupera los datos necesarios del modelo para responder la petición.
3. El controlador facilita dichos datos a la vista.
4. Se genera la vista correspondiente y se envía al cliente para que se muestre en el navegador.

El proceso se ilustra en la siguiente figura:

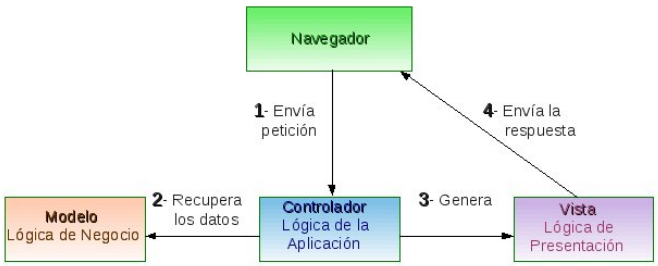


Figura 1.2: Procesando una petición de una página en la arquitectura MVC

Dividir una aplicación software en estos tres componentes implica varias ventajas que pueden resumirse en los siguientes puntos:

- Mejora la escalabilidad.
- El mantenimiento es más sencillo
- Promueve la reutilización

Capítulo 2

Objetivos

Una vez presentado el marco general de desarollo de este proyecto y las tecnologías utilizadas, en este capítulo abordaremos los objetivos que se han perseguido durante la elaboración del mismo. Se realizará una descripción detallada del problema, se resumirá, posteriormente, el estudio de las distintas alternativas planteadas para su resolución y se mostrará, finalmente, la metodología seguida en su desarrollo.

Los objetivos principales del proyecto son:

- El desarrollo de una aplicación libre que permita a cualquier docente, sin apenas conocimientos informáticos, crear, mantener y gestionar un sitio web donde aparezcan los artículos escritos por sus estudiantes en sus bitácoras.

- Formar una comunidad virtual alrededor del proyecto para conseguir que su desarrollo sea autocatalítico.

2.1. Descripción del problema

Como se ha visto en el capítulo de introducción, los blogs y los planetas de blogs presentan una serie de características que hacen de ellos aplicaciones con un gran potencial educativo. Es por ello que multitud de profesores están intentando utilizarlos como herramienta didáctica para sus clases, aprovechando al máximo estas características, pero sufriendo, por otra parte, el hecho de que estas aplicaciones no fueron diseñadas pensando en la labor de los docentes.

Se ha presentado una experiencia docente de uso de blogs, llevada a cabo en diferentes

asignaturas de la Universidad Rey Juan Carlos, en la que los propios estudiantes participan en la evaluación de los contenidos aportados por sus compañeros, y se han explicado los problemas detectados que impiden que esta actividad pueda ser realizada por profesores de carreras no relacionadas con la informática:

- Es necesario contar con un servidor web donde instalar el paquete *Planet*.
- Son imprescindibles conocimientos de administración de sistemas GNU/Linux, y conocimientos de programación en HTML.
- El alumnado debe enfrentarse al concepto de feed y ser capaz de identificar su URL.

Estas son las razones principales por las que se decide comenzar este proyecto, que nace con el objetivo fundamental de desarrollar una aplicación que permita que cualquier docente pueda realizar la actividad presentada. Los requisitos que se identificaron inicialmente para la primera versión de la aplicación fueron los siguientes:

- La aplicación debe permitir la creación y gestión de un planeta que recoja los artículos publicados en los blogs de los estudiantes de una asignatura.
- Los alumnos podrán darse de alta ellos mismos en el planeta.
- Los estudiantes podrán valorar positivamente las historias más interesantes.
- Debe implantarse algún mecanismo que evite la suplantación de votos.
- Las estadísticas de votos emitidos y recibidos deben estar disponibles.
- La instalación debe ser sencilla y la interfaz será lo más simple y amigable posible. La aplicación no debe requerir apenas conocimientos informáticos para su puesta en marcha.
- Se limitarán al máximo los medios técnicos necesarios para su desarrollo.
- Se distribuirá con una licencia libre.

2.2. Estudio de alternativas

Desde el comienzo del proyecto desechamos la idea de comenzar un desarrollo desde cero. Dado que se deseaba implementar una aplicación libre, se decidió aprovechar el trabajo realizado por otros programas libres que incluyeran funcionalidades similares a las de nuestro proyecto. Ésta es una de las ventajas del software libre; podíamos estudiar el código de otras aplicaciones similares, coger ideas de cómo implementar ciertas partes de nuestro proyecto o, incluso, reutilizar partes enteras del código de otros proyectos.

Veamos, por tanto, los diferentes proyectos que fueron estudiados, con sus ventajas e inconvenientes.

2.2.1. Planet

Este programa muestra en una única página web (el planeta) una referencia a las noticias publicadas en los sitios web que se agreguen. Utiliza el *Universal Feed Parser* de Mark Pilgrim para descargar contenido desde fuentes RDF, RSS o feeds de Atom y proporciona gran cantidad de plantillas que permiten que se personalice el planeta con diferentes fuentes y colores.

Como se ha comentado, este programa, ligeramente modificado, era el que se estaba utilizando para realizar la actividad. La primera idea fue modificar de alguna forma este software para que permitiera el registro del alumnado en el planeta y que incluyera la posibilidad de las votaciones, y distribuir el nuevo programa como un paquete Debian para facilitar su instalación.

El problema fundamental de esta solución es que el docente que quisiera utilizarlo debería contar con una máquina conectada a Internet donde pueda instalar nuestro programa. Esto choca, por una parte, con el requisito de no exigir conocimientos informáticos y, por otra parte, con la intención de no requerir hardware para su puesta en marcha. De hecho, aunque el docente pudiera contar con una máquina conectada a Internet, es probable que no cuente con derechos de administración sobre la misma, por lo que no podría instalar nuestro software. Esto ocurre, por ejemplo, en los centros educativos públicos no universitarios de Andalucía, donde los ordenadores son administrados de forma centralizada por el Centro de Gestión Avanzado.

2.2.2. Drupal

Drupal[1] es un Sistema Gestor de Contenido libre, que permite publicar, gestionar y organizar todo tipo de contenido (artículos, imágenes, u otros archivos) en un sitio web y ofrece servicios añadidos como foros, encuestas, votaciones, blogs y administración de usuarios y permisos.

Drupal cuenta con una amplia comunidad de usuarios y desarrolladores, entre los que se ha formado un grupo de profesores que han puesto a disposición de la comunidad una distribución de Drupal lista para instalarla en un servidor, con todos los módulos y extensiones necesarios para que sea personalizada y se pueda crear un Planeta de Blogs. Algunas de las características de esta distribución que lo convertían en un gran candidato son:

- Instalación sencilla.
- Plantilla o tema visual fácilmente personalizable gracias a su selector de color y al sistema de bloques y menús de Drupal.
- Documentación de ayuda incorporada y un foro para soporte.
- Sistema anti-spam en formularios, comentarios, etc.
- Módulo para la votación de los artículos por los usuarios.
- Comunidad de usuarios y desarrolladores activa.

Figura 2.1: Planeta de blogs con Drupal

[1] http://drupal.org/

A pesar de sus ventajas, presentaba un inconveniente que chocaba con los requisitos de no requerir conocimientos técnicos ni hardware para la puesta en marcha de la actividad: es necesario contar con un servidor web donde instalar la distribución de Drupal.

2.2.3. Feevy

Feevy[2] es una aplicación web libre que permite a sus usuarios crear su propio *blogroll* (colección de enlaces de blogs) de forma dinámica, mostrando el contenido de los blogs que se agreguen a través de sus correspondientes feeds.

Su funcionamiento es muy sencillo:

1. El usuario se registra en el sitio web de Feevy.
2. Añade la URL de los blogs que desea agregar.
3. Feevy genera una etiqueta HTML que el usuario debe pegar en su blog o en su sitio web.
4. A partir de ese momento se mostrará el título y un fragmento de los últimos artículos de los blogs agregados en una columna, que se actualizará dinámicamente colocando arriba el más reciente.

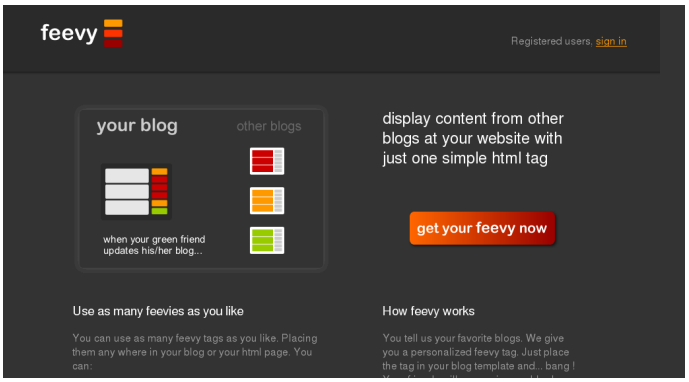


Figura 2.2: Sitio web de Feevy

La idea de Feevy nos cautivó desde un principio: el usuario no necesita contar con ninguna máquina para la actividad; tan sólo debe registrarse en un sitio web y contar con un blog donde pegar la etiqueta HTML generada.

[2]http://feevy.com

Además, el proyecto presentaba una serie de características que hacían que pudiéramos reutilizar gran parte de su código:

- Se distribuye con la licencia libre GPL (GNU General Public License).
- Se trata de una aplicación web que se encuentra en producción. Muchas funcionalidades podían ser reutilizadas, como, por ejemplo, el registro de usuarios o la agregación de un blog.
- Es capaz de leer los feed de los blogs gratuitos más populares.
- Ha sido programada utilizando el framework Ruby on Rails siguiendo el patrón MVC.
- Para añadir un blog el usuario no necesita conocer la URL del feed.

Por otra parte, estudiando los usuarios de Feevy, se observó que existían muchos docentes que habían comenzado a utilizarlo para crear un planeta con los blogs de sus alumnos, y que podían encontrarse en Internet turoriales que explicaban cómo ponerlo en marcha.

Por estas razones se decidió que Feevy sería el punto de partida para nuestra aplicación. Descargaríamos su código fuente y lo estudiaríamos para ver qué cosas podíamos hacer como ellos y qué cosas debíamos modificar para añadir las funcionalidades deseadas.

La decisión estaba tomada, ofreceríamos un servicio web, gratuito y libre, que permitiera que cualquier docente pudiera crear y mantener un planeta de blogs (con soporte para votaciones y estadísticas) que le permitiera desarrollar la actividad presentada en el primer capítulo.

2.3. Metodología empleada

El objetivo de aplicar una metodología o modelo de desarrollo a la construcción de una aplicación software es convertir el desarrollo en un proceso formal, con resultados predecibles, que permitan obtener un producto final de alta calidad, que satisfaga las necesidades y los requisitos identificados.

El objetivo de este proyecto era lanzar una primera versión de la aplicación que cumpliera con todos los requisitos planteados y conseguir una comunidad de usuarios que fuera aportando información al proyecto. Por un lado, los posibles fallos o vulnerabilidades del programa serían

descubiertos y, por otra parte, los usuarios irían demandando nuevas funcionalidades para hacer de la aplicación un programa más completo.

Tras estudiar distintos modelos de desarrollo se decidió que se seguiría, de forma parcial, la metodología ágil conocida como Programación Extrema (XP, *Xtreme Programming*)[8]. Esta metodología se adapta considerablemente al desarrollo de un proyecto como el que se pretende implementar, ya que hace énfasis en la adaptabilidad y la comunicación, y promueve el desarrollo de pequeñas iteraciones con nuevos requisitos.

En este capítulo se pretende exponer, por tanto, las bases de esta metodología y cómo se ha adaptado la misma para el desarrollo de este proyecto.

2.3.1. Programación Extrema

La Programación Extrema es una de las metodologías ágiles más existosas de los últimos tiempos. Nació hace unos 8 años, y ha sido probada con éxito en multitud de compañías de todos los sectores y tamaños por todo el mundo.

El éxito de XP radica en que se centra en la satisfacción del cliente. Esta metodología está diseñada de manera que el cliente recibe el software que necesita en el momento que lo necesita. XP anima a los desarrolladores a cambiar los requisitos sin importar el momento del ciclo de vida del producto.

Esta metodología también se basa en el trabajo en equipo. Los gestores, los clientes y los desarrolladores forman parte de un equipo dedicado a crear un software de calidad. XP introduce una forma sencilla, pero efectiva, de lograr un estilo de desarrollo que implique a todo el equipo.

XP se apoya en cinco principios fundamentales: simplicidad, comunicación, retroalimentación (feedback), valentía y respeto.

- Simplicidad:

  La simplicidad es la base de la programación extrema, y propone dos máximas que recogen este principio: *Haz la cosa más simple que pueda funcionar* y *No lo vas a necesitar*.

  Al realizar modificaciones en el código por parte de diferentes desarrolladores la complejidad aumenta severamente. Para mantener la simplicidad es necesaria la refactorización del código, es decir, modificar el código sin alterar el comportamiento. También se aplica

la simplicidad en la documentación; en el código sólo se comentarán las partes que no varíen (como el objetivo de un método), y se apostará por la autodocumentación, eligiendo adecuadamente los nombres de las variables, métodos y clases.

- Comunicación:

  Muchos de los problemas que aparecen en los proyectos se deben a una mala comunicación. La programación extrema busca conseguir una comunicación constante.

  Por una parte, la comunicación con el cliente es fluida ya que pertenece al equipo de desarrollo. El cliente debe decidir las características y funcionalidades que tienen prioridad y siempre debe estar disponible para solucionar dudas. Por otro lado, el código es otro instrumento de comunicación entre los programadores. Cuánto más simple sea el código mejor será la comunicación. Por último, las pruebas unitarias son otra forma de comunicación ya que describen el diseño de las clases y los métodos al mostrar ejemplos concretos de cómo usar su funcionalidad.

- Retroalimentación (feedback):

  La retroalimentación ayuda a mantener el proyecto en el rumbo adecuado, por ello XP propone técnicas para conseguir feedback rápido y frecuente.

  La opinión del cliente respecto al estado del proyecto se conoce en tiempo real, ya que forma forma parte del equipo de desarrollo. La generación constante de nuevas versiones, con ciclos muy cortos, que muestran resultados de las nuevas características ayuda a no tener que deshacer parte del trabajo y volver a comenzar por cambios en los requisitos o malentendidos con el cliente. Las pruebas unitarias proporcionan una gran retroalimentación, informando de la corrección del proyecto descubriendo posibles fallos.

- Valentía:

  La valentía es imprescindible para mostrar el sistema al cliente, con todos sus errores, para que pueda orientarnos y modificarlo a su gusto. También lo es para modificar el código que funciona con el objetivo de hacerlo más simple.

- Respeto:

Todos los miembros del equipo de desarrollo deben respetarse y colaborar comportándose como uno sólo. Los programadores deben respetar el trabajo de otros programadores, y deben respetar la opinión del cliente y de los usuarios.

Para concretar estos proncipios generales, XP propone una serie de prácticas de las que mostramos, a continuación, las más importantes:

- Trabajar con el cliente: el cliente debe formar parte del equipo de desarrollo.
- Planificación: la planificación se realiza en base a ciclos de desarrollo cortos (2 o 3 semanas). Al comienzo de cada ciclo el cliente puede cambiar los requisitos, por lo que la planificación se revisa continuamente.
- Versiones pequeñas: al final de cada ciclo hay que ofrecer al cliente una nueva miniversión que muestre algo útil al usuario final y no trozos de código que no pueda ver funcionando.
- Diseño simple: hacer siempre lo mínimo imprescindible de la forma más sencilla posible, sin incluir partes que podamos necesitar en el futuro, se programa para hoy.
- Metáforas: los nombres de las clases, los métodos y las variables deben elegirse de manera que sólo con los nombres se pueda uno hacer una idea de qué es lo que hace cada parte del programa.
- Pruebas automáticas: las pruebas automáticas son una excelente herramienta de comunicación y feedback por lo que deben convertirse en una actividad básica.
- Integración continua: cada modificación debe incluirse en el sistema final y probarse de forma conjunta.
- Propiedad del código colectiva: todos los miembros del equipo pueden y deben modificar y conocer cualquier parte del código.
- Normas de codificación: debe haber un estilo común de codificación de forma que parezca que ha sido realizado por una única persona.

La siguiente figura recoge todas las características mencionadas y resume cómo se integran los aspectos fundamentales de la programación extrema:

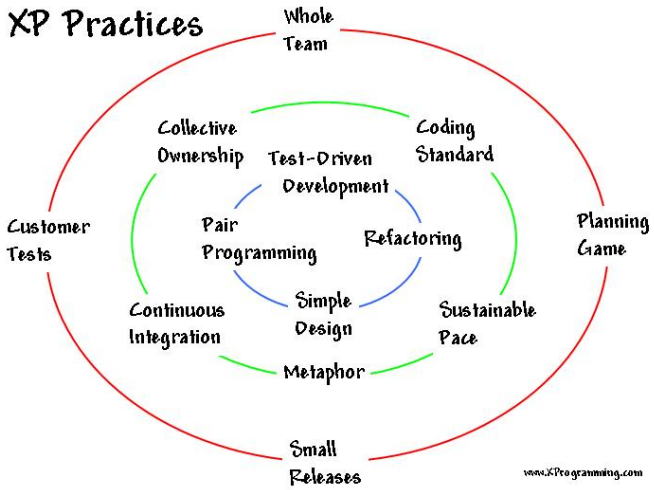


Figura 2.3: eXtreme Programming

2.3.2. Aplicación de la Programación Extrema al proyecto

Esta metología se adapta considerablemente al proyecto que se presenta en esta memoria. Como ya se ha comentado, tras conseguir una primera versión funcional de la aplicación, el objetivo era publicarla y promocionarla para conseguir una comunidad virtual a su alrededor que ayudara al proyecto a crecer. La comunidad de usuarios de la aplicación (que, como veremos, está formada por docentes de varias universidades e institutos) se irían encontrando con pequeños errores, dificultades o nuevas nececesidades de las que informarían a través de las herramientas preparadas para tal efecto, y el equipo de desarrollo se pondría a trabajar para solucionarlo.

De esta forma, la **comunicación** entre el cliente (los usuarios) y el equipo de desarrollo tomará un papel fundamental que irá marcando la evolución de la aplicación. No existe un plan a priori de las nuevas funcionalidades a desarrollar o de los errores que corregir, estos van siendo informados sin previo aviso, por lo que la **adaptabilidad** que propugna XP se convierte en un factor imprescindible para nuestro proyecto.

Así, la idea de publicar cada pocas semanas nuevas **mini-versiones** (una versión de la aplicación con pequeños cambios que ofrecen algo útil al usuario) se adapta a la perfección a nuestra situación.

De igual forma, el concepto de **equipo total** defendido por XP puede aplicarse a la comunidad del proyecto.

Por último, cuando el objetivo de crear una comunidad que incluya varios desarrolladores sea alcanzado, el **trabajo en equipo** y el **respeto** entre los miembros del equipo de desarrollo que propone XP será fundamental para que el proyecto pueda seguir creciendo. Cualquier desarrollador podrá modificar cualquier parte del código, aunque, como también se defiende en XP, habrá que seguir unas **normas de codificación** de manera que se mantenga un estilo común en todo el código.

Capítulo 3

Descripción Informática

En este capítulo se pretende mostrar, desde un punto de vista más técnico que el seguido hasta el momento, la estructura y el funcionamiento de la aplicación desarrollada.

Comenzaremos el capítulo con una sección que describe la arquitectura utilizada para la puesta en producción de la aplicación, la organización del código siguiendo el patrón MVC, el modelo de la Base de Datos y, finalmente, algunos datos de interés sobre la aplicación.

Posteriormente, con el objetivo de acercar al lector al proceso de desarrollo seguido en su implementación, se incluyen 3 secciones que representan las diferentes etapas del ciclo de vida del proyecto:

- Fase 1: Implementación de una versión funcional de la aplicación.
- Fase 2: Publicación de la aplicación y promoción del proyecto.
- Fase 3: Gestión y evolución del proyecto.

3.1. Estructura general de la aplicación

Como ya se ha explicado en los capítulos anteriores, tras estudiar varios proyectos libres se decidió seguir la línea del proyecto Feevy, adaptando su aplicación para que cumpliese nuestros requisitos.

De esta manera, se comenzó a desarrollar una aplicación web que permitiera que cualquier docente, sin apenas conocimientos informáticos, puediera llevar a cabo la actividad descrita en el primer capítulo.

En el Anexo A puede encontrarse un manual de usuario en el que se detalla el funcionamiento de la aplicación, que se resume en los siguientes pasos:

1. El docente se registra en nuestro sitio web.
2. Selecciona el idioma y personaliza la apariencia.
3. La aplicación genera una etiqueta HTML.
4. El docente pega la etiqueta en su blog o en su página web.
5. A partir de ese momento, los estudiantes pueden agregar su blog al planeta.
6. Los artículos escritos en los blogs agregados serán mostrados en el planeta.
7. Los estudiantes podrán votar los posts más interesantes.
8. Las estadísticas de votos emitidos y recibidos podrán ser consultadas.

3.1.1. Arquitectura del entorno en producción

La infraestructura necesaria para poder ofrecer el servicio descrito requiere contar con un equipo conectado a Internet en el que se instale un servidor web, un servidor de aplicaciones Rails y una Base de Datos.

Las posibilidades de elección son numerosas y, tras estudiar diferentes opciones, nos decidimos por la siguiente arquitectura:

- Como sistema operativo, se utilizaría Debian GNU/Linux.
- Como servidor web, Apache.
- Como servidor de aplicaciones Rails, mongrel.
- Como SGBD, MySQL.
- Como sistema de memoria cache, para reducir los accesos a la Base de Datos, nos decidimos por memcached.

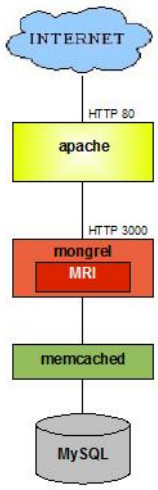


Figura 3.1: Arquitectura del entorno de producción

3.1.2. Patrón Modelo Vista Controlador

Como se explica en el capítulo de introducción, el modelo de desarrollo seguido responde al patrón MVC. Así, la estructura de directorios que marca el Framework de desarrollo Ruby on Rails separa el código de cada uno de estos elementos almacenándolos en diferentes ficheros de directorios separados. Puede observarse esta situación en la figura 3.2

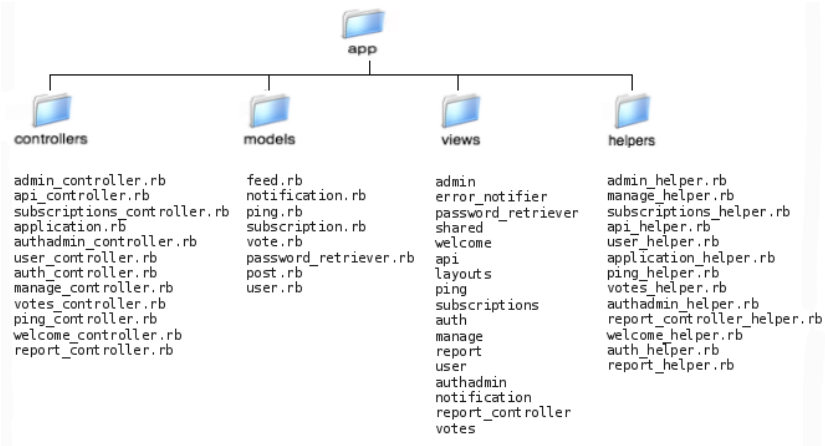


Figura 3.2: Estructura de directorios siguiendo el Patrón MVC

En el subdirectorio *controllers* se almacenan los ficheros que se encargan de gestionar las peticiones recibidas. Si para atender una petición es necesario acceder a la información almacenada en la Base de Datos, el controlador implicado solicitará los datos al modelo apropiado (los

modelos están almacenados en el subdirectorio *models*). En ese momento se generará la vista oportuna (subdirectorio *views*) que devolverá la página correspondiente.

En el subdirectorio *helpers* se guardan pequeños trozos de código que serán reutilizados en diferentes partes de la aplicación. Un helper suele contener lógica de presentación relativamente complicada, de manera que este código se saca de las vistas manteniendo su código más sencillo.

3.1.3. Modelo E/R de la Base de Datos

La figura 3.3 resume el modelo Entidad-Relación de la Base de Datos y muestra las entidades y las relaciones fundamentales de la aplicación:

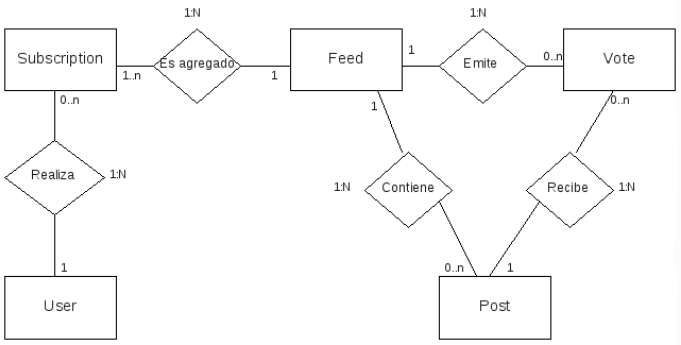


Figura 3.3: Modelo E/R de la Base de Datos

La entidad *Feed* representa los blogs de los estudiantes y, como veremos en detalle en el esquema, algunos de sus atributos son el título, la url o el nombre del estudiante que lo mantiene. La información de los profesores se recoge en la entidad *User*; y los blogs agregados a un planeta de un profesor se almacenarán en la entidad *Subscriptions*.

La información relativa a los artículos publicados en cada blog se recoge en la entidad *Post*, y la entidad *Vote* representa cada voto emitido por un estudiante a favor de un post.

En la figura 3.4 pueden estudiarse en profundidad los atributos identificados en cada una de las entidades utilizadas por la aplicación:

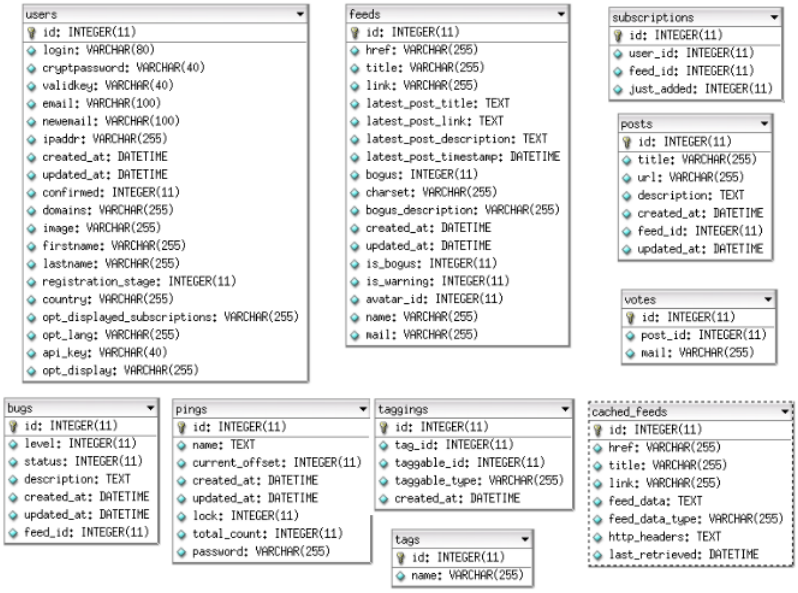


Figura 3.4: Atributos de cada entidad

3.1.4. Otros datos de interés

Los datos que se muestran en la figura 3.5 se han obtenido usando la herramienta SLOCCount[1], una aplicación libre que permite analizar proyectos software. Para ello, el programa cuenta las líneas físicas de código, separándolas por lenguajes, y realiza una estimación del coste y del esfuerzo necesario para su desarrollo basándose en el modelo COCOMO (Modelo Constructivo de Costes).

Número de líneas de código total:	7027
Lenguaje más utilizado:	Ruby, 94.85%
Meses necesarios estimados para el desarrollo:	7,59
Coste total estimado necesario:	$ 209291
Estimación de número de desarrolladores:	2,45

Figura 3.5: Datos obtenidos con SLOCCount

Llama la atención el dato relativo al coste estimado necesario para el desarrollo del proyecto, que supera los 209.000 dólares, alrededor de 150.000 euros. La herramienta estima que hubiera sido necesario el trabajo de 2,45 desarrolladores durante más de 7 meses y medio para poder completar el trabajo.

Por esta razón se afirmaba en el capítulo de introducción que el proyecto no habría podido llevarse a cabo en el tiempo previsto si no hubiéramos podido reutilizar el código y el trabajo de

[1] http://www.dwheeler.com/sloccount/

otras aplicaciones libres. Ésta es una de las grandes ventajas del software libre: la innovación. Gracias a que el proyecto Feevy fue liberado, basándonos en su aplicación y realizando algunas modificaciones, hemos puesto a disposición de docentes de todo el mundo una herramienta que puede ayudarles en su trabajo diario. Además, nuestro proyecto también es libre, por lo que, quizás, alguien, en cualquier parte del mundo, podrá reutilizar nuestro trabajo adaptándolo a sus necesidades y desarrollar una nueva herramienta que estará, de nuevo, a disposición de toda la comunidad.

3.2. Fase 1: Implementación de una versión funcional de la aplicación

En esta sección pretenden mostrarse las tareas realizadas y las decisiones tomadas desde que se comienza con el desarrollo de la aplicación hasta que se consigue una versión beta que satisface los requisitos iniciales.

Tras tomar la decisión de aprovechar el trabajo realizado por el equipo de desarrollo de Feevy, la siguiente tarea fue realizar una instalación de la aplicación en un servidor propio para estudiar detenidamente su funcionamiento y su código.

En la web del proyecto aparecía un enlace al código fuente y unas instrucciones que, tras varios intentos de instalación frustrados, demostraron su invalidez. Se estableció contacto con David de Ugarte, uno de los líderes del proyecto, el cual nos remitió a Alexandre Girard, el desarrollador principal de Feevy. Alexandre, amable y pacientemente, nos ayudó en el proceso de instalación, guiándonos en la solución de los problemas con los que nos encontramos. En el Anexo B de esta memoria se han incluido algunos de los correos electrónicos enviados que fueron intercambiados durante el proceso de instalación.

Algunos de los errores obtenidos se debían a que el código disponible contenía errores que habían sido modificados posteriormente pero no se habían incluido en el tarball descargable, por lo que, tras ser descubiertos, el equipo de desarrollo de Feevy tomó nota y actualizó el código.

Una vez finalizado el proceso de instalación de Feevy en un servidor propio, y tras estudiar el código fuente y la estructura de la Base de Datos, podíamos comenzar su modificación de manera que cumpliera con los requisitos de nuestra aplicación.

3.2.1. Modificación de la aplicación para el cumplimiento de los requisitos

El primer paso consistió en la eliminación de las funcionalidades que no iban a ser utilizadas. Por ejemplo, en Feevy, al tratarse de una aplicación de ocio utilizada fundamentalmente por adolescentes, cada blog se identifica con un avatar. En nuestra aplicación no se iban a usar estos elementos, de manera que se procedió a su eliminación en el código y en la Base de Datos.

Tras la supresión de todo lo que no sería usado, teníamos una aplicación sobre la que comenzar a añadir las nuevas funcionalidades. Se resumen a continuación las tareas realizadas para implementar las principales características.

Soporte para las votaciones

La aplicación debía ser modificada para que admitiera la posibilidad de que los estudiantes valorasen positivamente las entradas de sus compañeros. Se identificaron los siguientes requisitos:

1. Cuando un usuario desee votar un artículo, deberá pinchar sobre el enlace 'vota este post' que aparecerá junto al mismo.

2. El usuario será dirigido a una página en la que se mostrará el título del artículo que va a votar y se le solicitará que introduzca su correo electrónico.

3. La aplicación enviará un correo electrónico al usuario informándole de que se ha recibido un voto en su nombre, de manera que se puedan detectar posibles suplantaciones de votos.

4. Se impedirá que un usuario se vote a sí mismo, que vote más de una vez un mismo artículo, o que un usuario vote por un blog que no pertenece a su asignatura.

Para añadir esta funcionalidad se realizaron las siguientes tareas:

- Se creó la tabla *votes* en la Base de Datos.

- Se diseñó la vista *new.rhtml*, que permite a los estudiantes emitir un voto.

- La vista *notification/newvote.rhtml*, que se generaría al crear un voto para que se enviara un correo electrónico al usuario que estaba votando, fue diseñada.

- Posteriormente, se creó el controlador *votescontroller.rb* que se encargaría de gestionar las peticiones de creación de un nuevo voto. Se añadió el método público *create* y el método protegido *check-vote*, que se encarga de comprobar que se cumplen los requisitos establecidos para poder enviar un voto. Se muestra a continuación el código de este método:

```
def check_vote
  ## Check that pupil is not voting himself
  feed_id = Post.find_by_id(params[:id]).feed_id
  if Feed.find_by_id(feed_id).mail == params[:mail]
    return 2
  end
  ## Check that pupil is not voting a post again
  if Vote.count(:conditions =>
                ["mail = ? and post_id = ?",
                params[:mail],params[:id]]) > 0
    return 3
  end
  ## Check that mail belongs to a verified student
  subscription = Subscription.find(:first,
                       :conditions => ["feed_id = ?",
                       feed_id])
  user_id = subscription.user_id
  subs = []
  subs = Subscription.find(:all,
                           :conditions => ["user_id = ?",
                           user_id])
  subs.each do |s|
    if s!=nil and s.feed !=nil and
       s.feed.mail == params[:mail]
          return 1
    end
  end
  return 4
  end
```

- Para que apareciera el enlace 'vota este post' junto a los artículos publicados, hubo que modificar el modelo *user.rb* y las diferentes vistas que muestran el contenido de los blogs agregados en función de los colores y el idioma elegido.

Soporte para las estadísticas

Uno de los motivos por los que la actividad había cosechado tan buenos resultados en las asignaturas donde se había probado, era que los estudiantes podían consultar las estadísticas de votos recibidos, de manera que se podía comprobar qué entradas del blog resultaban más atractivas para el resto de compañeros. Del mismo modo, no había muchos estudiantes a los que les resultara atractivo el hecho de aparecer en los últimos puestos del ranking, por lo que la motivación para escribir buenos artículos aparecía de forma (casi) espontánea. Además, los profesores utilizaban la información de votos emitidos y recibidos para evaluar parte de la actividad.

Los requisitos identificados respecto a esta funcionalidad fueron los siguientes:

- En cada planeta aparecerá un enlace que llevará hasta la página con las estadísticas.
- Esta página de estadísticas mostrará los votos recibidos por cada blog, los votos recibidos por cada post y los votos emitidos por cada alumno.

Las tareas llevadas a cabo para lograr este objetivo fueron los siguientes:

- Se modificó el controlador *usercontroller.rb* y las vistas que muestran los planetas de forma que se mostrara un link que enlazara con la página de estadísticas asociada a ese planeta.
- Se implementó el método *stats* en el controlador *usercontroller.rb* que se encargaría de gestionar las peticiones de las páginas de estadísticas, del que se muestra el código:

```
def stats
  @user = User.find(params[:id])
  @pupils = @user.generate_stats_pupils(params[:tags])
  @posts = @user.generate_stats_posts(params[:tags])
  @blogs = @user.generate_stats_blogs(params[:tags])
  partial_badge = "user/badge/content/stats"
  partial_style = "user/badge/style/light"
  partial_badge += "_" + @user.opt_lang.gsub('-','_')
  logger.debug "partial_style: #{partial_style}"
  logger.debug "partial_badge: #{partial_badge}"
  @style =
     render_to_string(:partial => partial_style,
                      :locals => { :id => params[:id]})
  @style = "" if params[:style] == "open-css"
```

```
    @content =
       render_to_string(:partial => partial_badge,
                        :locals => { :id => params[:id],
                        :posts => @posts,
                        :blogs => @blogs,
                        :pupils => @pupils} )
    @stats = [@content, @style]
    if request.env['HTTP_ACCEPT'] =~
      /(application|text)\/(html|xhtml)/
      logger.debug @content.to_s
      render :layout => true
    end
  end
```

- En el modelo *user.rb* se añadieron los métodos que realizan las consultas necesarias en la Base de Datos para obtener la información de los votos emitidos y recibidos.

- Se diseñaron las vistas en los distintos idiomas para mostrar estas estadísticas, como puede verse en la figura 3.6. Se decidió que se dividiría la página en tres columnas que mostrarían los votos que había recibido cada blog, los votos recibidos por cada artículo y los votos emitidos por cada alumno, respectivamente, y que se ordenaría cada columna de mayor a menor número de votos.

Figura 3.6: Nueva página con las estadísticas de votos emitidos y recibidos

Los estudiantes deben poder agregar sus blogs al planeta

En Feevy, cada usuario debe ir añadiendo uno a uno todos los blogs que quiera incluir en su planeta. En nuestra aplicación esta situación convertiría la creación de un planeta en una

tarea un tanto ardua si el número de alumnos de la asignatura fuera elevado: el profesor debería que recibir la información de las bitácoras de cada estudiante (la url del blog, su nombre y su correo electrónico) y tendría que ir añadiéndolos uno a uno, aumentando las posibilidades de que produjera algún error.

Con el objetivo de facilitar la puesta en marcha de la actividad, se decidió modificar nuestra aplicación de forma que permitiera que los estudiantes pudieran agregar su blog sin intervención del docente.

Se modificaron las vistas de los distintos idiomas de manera que incluyera el texto *Si eres alumno de esta clase quizás quieras dar de alta tu blog*, añadiendo el enlace correspondiente.

Se implementó el método *new* del controlador *subscriptions-controller.rb* que se encargaría de gestionar las peticiones de agregación de un blog.

Finalmente, se diseñó la vista *views/subscriptions/new.rhtml*, mostrada en la figura 3.7, que permite que los alumnos puedan proporcionar los datos de su bitácora.

Se decidió incluir una nota que anima a los estudiantes a utilizar una licencia Creative Commons para los contenidos publicados en sus bitácoras, de forma que los aspectos relacionados con la propiedad intelectual y el conocimiento libre se introducirán en el aula de forma natural.

Figura 3.7: Agregando un blog al planeta

Configurar FAVS como un planeta tradicional

En Feevy tan sólo se muestra el último artículo publicado en cada bitácora agregada y, por

tanto, hubo que modificar la aplicación de manera que se comportara como un planeta tradicional mostrando todos los posts publicados en cada blog.

De hecho, aprovechando la funcionalidad ya implementada en Feevy, se decidió que se daría la posibilidad de elegir al docente el número de artículos mostrados por blog.

Figura 3.8: Eligiendo el número de artículos mostrados por blog

3.2.2. Desarrollo del sitio web del proyecto

El sitio web es el elemento que proporciona la primera impresión sobre el proyecto y de su diseño depende gran parte de su éxito. Debe proporcionar visibilidad, es decir, debe dar a conocer el proyecto a los usuarios y explicar el funcionamiento del servicio ofrecido[9]. Dado que será el primer contacto de cualquier persona con el proyecto, debe tener una estructura adecuada y clara[7].

Identificamos varias secciones que la página principal debía contener: descripción del servicio, ejemplos de FAVS en acción y ayuda.

En este paso también se tuvo que tomar otra decisión importante para el proyecto: ¿qué idioma se usará en el sitio web del proyecto? Aunque la primera idea fue realizar una web en castellano, finalmente se decidió, pensando en las posibilidades potenciales de atracción de usuarios y desarrolladores, que la web estaría en inglés.

Se realizaron pruebas de uso con profesores de distintos niveles educativos y con diferentes dominios del idioma inglés que fueron muy satisfactorias. Se prepararon manuales de usuario tanto en español como en inglés que incidían en los aspectos que habían resultado más problemáticos en las pruebas.

La figura 3.9 es una captura de la página principal del sitio web desarrollado.

Figura 3.9: Página principal del sitio web de FAVS

Uno de los objetivos principales de la aplicación ha sido desde el principio el de la usabilidad[11]; queríamos que cualquier docente pudiera usar nuestro servicio aunque no tuviera muchos conocimientos de informática. Por ello, además de explicar con un texto el funcionamiento de la aplicación, se diseñaron otras páginas, como la página *faq* o la página *scheme*, que intentan describir de una forma sencilla y gráfica los pasos necesarios para disfrutar del servicio ofrecido.

El primer objetivo del sitio web estaba cubierto: resultaba muy útil para un usuario que quisiera comenzar a utilizar la aplicación, ya que contaba con documentación y páginas gráficas que explicaban su uso y funcionamiento. Sin embargo, no era suficiente para aquellos usuarios que quisieran informar de incidencias o solicitar nuevas características, ni ofrecía soporte para los posibles colaboradores.

Con el objetivo de dotar al proyecto de las herramientas necesarias para crear y potenciar una comunidad a su alrededor se decidió registrar el proyecto en Sourceforge. Sourceforge.net es el sitio web de desarrollo de software libre más grande del mundo, albergando en la actualidad

más de 230.000 proyectos y más de 2.000.000 de usuarios.

Este sitio permite que el proyecto cuente con todas las herramientas necesarias: foros, listas de correo, sistema de seguimiento de errores, sistemas de control de versiones, descarga de software, útimas noticias y solicitud de nuevas funcionalidades.

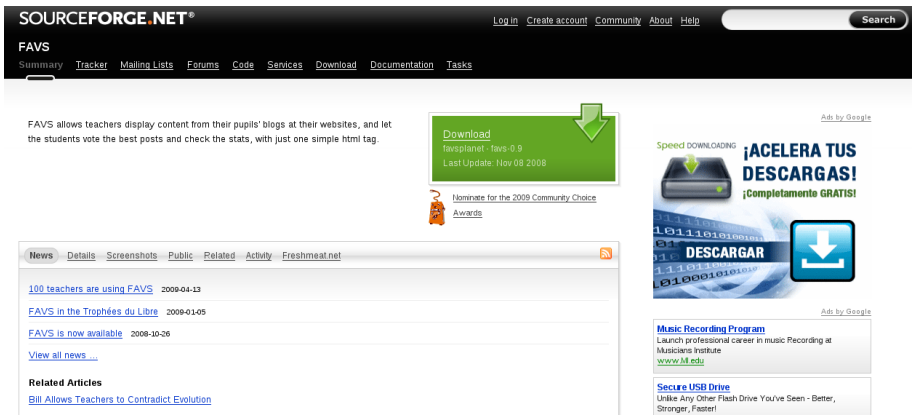


Figura 3.10: El sitio de FAVS en SourceForge

3.2.3. Elección de la licencia

La licencia de una aplicación establece las normas con las que se regirá su distribución, uso, modificaciones e incluso las redistribuciones del software.

Feevy fue liberado bajo la licencia GPL[2] (GNU General Public License). Esto implica que podíamos estudiar el código, hacer modificaciones y redistribuirlas, pero obliga a que estas modificaciones se distribuyan también bajo los mismos términos.

Es ésta la principal característica de la GPL, y del resto de licencias conocidas como robustas o *copyleft*, ya que aseguran que la naturaleza del software creado siempre será libre, evitando que el autor de un programa libre pueda tener que competir con versiones modificadas privativas su propio trabajo.

3.2.4. Puesta en marcha de la aplicación

La Universidad Rey Juan Carlos puso a nuestra disposición una máquina con conexión a Internet mediante una dirección IP estática.

[2]http://www.gnu.org/copyleft/gpl.html

Se realizaron las instalaciones y configuraciones oportunas para conseguir la infraestructura presentada en la sección 3.1.1 y se registró el dominio *favs.homelinux.org*, redirigiendo el tráfico a la dirección IP del ordenador de la Universidad.

Se instaló el servidor de correo electrónico sendmail necesario para el correcto funcionamiento de la aplicación, realizando las configuraciones oportunas.

La versión beta estaba lista para comenzar a ser utilizada.

3.3. Fase 2: Promoción del proyecto

Tras haber implementado la versión beta de la aplicación y haber preparado una máquina para ofrecer el servicio, comenzaba una nueva etapa en el ciclo de vida del proyecto: había que promocionar la aplicación para conseguir una comunidad de usuarios que nos informase de las dificultades, problemas o mejoras que detectaran o, incluso, nuevos desarrolladores que quisieran colaborar con el proyecto. El objetivo de esta nueva etapa era conseguir el fenómeno conocido como *efecto en red*[10]: conseguir usuarios y desarrolladores suele ser el primer paso para conseguir más usuarios y más desarrolladores.

Esta fase puede resumirse como una campaña publicitaria para dar a conocer nuestro proyecto. Las tareas realizadas se detallan a continuación:

- Comunicamos a David de Ugarte, uno de los líderes del proyecto Feevy, que ya teníamos lista una primera versión. La reacción de David fue más que satisfactoria para nosotros. Él había publicado en su blog[3] un artículo titulado *Por qué liberar Feevy no aporta gran cosa*, en el que afirmaba que estaba convencido de que liberar el código de Feevy no iba a suponer algo positivo para la comunidad. Tras recibir la noticia del nacimiento de FAVS, David nos pidió permiso para publicar el correo electrónico que le enviamos y publicó un nuevo artículo en su blog titulado *Liberar software siempre paga*, que comienza así:

> Feevy ha tenido un cachorrito orientado al trabajo escolar. Se llama FAVS -como nuestra gran amiga curitibense- y está escrito por Jesús Moreno. Feevy comienza a dar frutos. Frutos para el dominio público. Para todos.
> Jesús Moreno, nos escribía esta mañana:

[3] www.deugarte.com

```
Como os comenté este verano, para mi proyecto fin de máster
he comenzado a desarrollar una aplicación web basada en
feevy. Se llama FAVS, y es una aplicación que está pensada
para profesores. La idea es que el profe pueda tener en su
blog o en su sitio web las entradas que publican sus
alumnos en sus bitácoras.
```

- Dimos de alta el proyecto en Freshmeat[4], un portal dedicado a informar de las últimas noticias, actualizaciones y versiones de programas libres.

- Nos pusimos en contacto con los responsables de la sección de software libre de Educared[5], una fundación que impulsa el uso de Internet en la educación, quienes publicaron un reportaje sobre la aplicación.

Figura 3.11: Reportaje sobre FAVS en Educared

- Enviamos una petición de participación a la *Conferencia sobre Software de Libre en Educación Superior* de Valencia y fuimos seleccionados para impartir una de las conferencias. Nuestra participación consiguió los objetivos propuestos, ya que, gracias a la ponencia[6] impartida, el Departamento de Filología Inglesa de la Universidad de Valencia se interesó por nuestra aplicación y nos solicitó información y ayuda para poner en marcha la actividad descrita en el siguiente curso académico.

[4] http://freshmeat.net/
[5] www.educared.net/
[6] http://aulavirtual.uv.es/ficheros/view/slides/presentacion.pdf

- Se envió un artículo sobre la aplicación al espacio *Innovando con las TIC*[7] de la Junta de Andalucía.

- Nos registramos en el sitio Aulablog, una comunidad de profesores que usan las TIC en su práctica diaria, y envíamos un artículo[8] que ha tenido más de 750 lecturas.

Figura 3.12: Artículo sobre FAVS en AulaBlog

- Se creó la página FAVS en la Wikipedia, y se incluyeron enlaces a nuestro sitio web en páginas relacionadas (como la página que habla de los planetas de blogs).

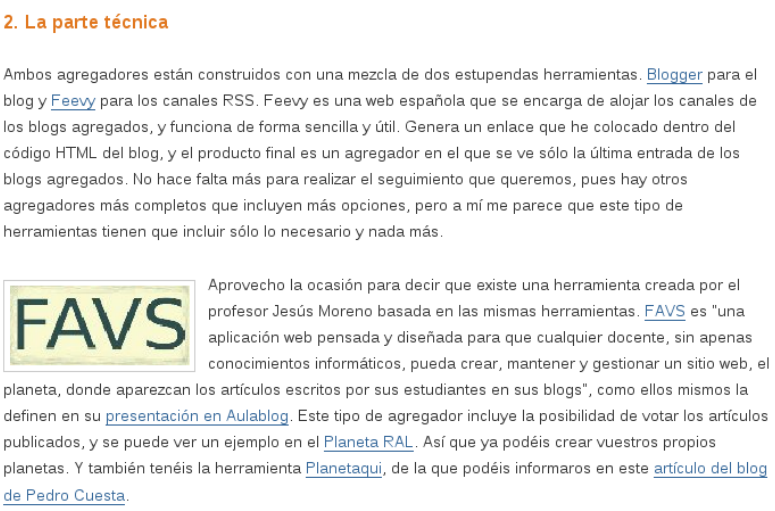
- Se escribieron reseñas y se enviaron artículos a otras webs como Menéame, Internet en el Aula o Profeblog, y se enviaron correos electrónicos a profesores que imparten cursos sobre las nuevas tecnologías aplicadas a la educación en facultades o centros de formación del profesorado.

Como consecuencia de esta campaña de promoción del proyecto los primeros usuarios se registraron en nuestro sitio. ¡El efecto en red había comenzado!

Una muestra es el artículo *Participa en los planetas de blogs de aula*[9], escrito por Alejandro Valero, en su bitácora *Apuntes sobre blogs*. En él, el autor explica qué son los planetas de blogs y presenta distintas herramientas para su creación, entre las que destaca a nuestro proyecto.

Alejandro Valero es autor del taller *Creación y uso educativo de blogs* que se llevó a cabo en el marco del último Congreso Nacional Internet en el Aula.

[7] http://www.juntadeandalucia.es/averroes/espaciotic/spip.php?article46
[8] http://www.aulablog.com/favs-planetas-de-blogs-docentes
[9] http://avalerofer.blogspot.com/2009/04/participa-en-los-planetas-de-blogs-de.html



2. La parte técnica

Ambos agregadores están construidos con una mezcla de dos estupendas herramientas. Blogger para el blog y Feevy para los canales RSS. Feevy es una web española que se encarga de alojar los canales de los blogs agregados, y funciona de forma sencilla y útil. Genera un enlace que he colocado dentro del código HTML del blog, y el producto final es un agregador en el que se ve sólo la última entrada de los blogs agregados. No hace falta más para realizar el seguimiento que queremos, pues hay otros agregadores más completos que incluyen más opciones, pero a mí me parece que este tipo de herramientas tienen que incluir sólo lo necesario y nada más.

FAVS

Aprovecho la ocasión para decir que existe una herramienta creada por el profesor Jesús Moreno basada en las mismas herramientas. FAVS es "una aplicación web pensada y diseñada para que cualquier docente, sin apenas conocimientos informáticos, pueda crear, mantener y gestionar un sitio web, el planeta, donde aparezcan los artículos escritos por sus estudiantes en sus blogs", como ellos mismos la definen en su presentación en Aulablog. Este tipo de agregador incluye la posibilidad de votar los artículos publicados, y se puede ver un ejemplo en el Planeta RAL. Así que ya podéis crear vuestros propios planetas. Y también tenéis la herramienta Planetaqui, de la que podéis informaros en este artículo del blog de Pedro Cuesta.

Figura 3.13: Artículo de Alejandro Valero sobre Planetas de blogs

Otro ejemplo es el artículo *Planeta FAVS - vota el mejor post*[10] escrito por la profesora Amelia Tierno tras haber comenzado a utilizar FAVS para sus clases en el IES Ignacio Ellacuría:

AULA DE TECNOLOGÍAS

BLOG de Tecnologías, Informática, Educación y Web 2.0

domingo 10 de mayo de 2009

Planeta FAVS - Vota el mejor post

FAVS es un sitio para crear planetas de blogs. Está basado en Feevy pero ha sido diseñado pensando en los profesores. Para mí el gran atractivo es que permite que los alumnos voten los artículos de los compañeros que les resulten más interesantes, además de que se pueden consultar las estadísticas de votos enviados y recibidos.

Otra ventaja, permite que cada alumno dé de alta su propio blog en el planeta, lo que evita la engorrosa tarea en Feevy de ir añadiendo una a una la URL de cada blog.

¡Problema solucionado, voto por FAVS!

FAVS

Una vez dicho esto, vamos a lo que nos interesa. Este es nuestro "*planeta*":

Planeta Ellacuría 2009 blogs de alumn@s

Acerca de

Blog de consulta para l@s alumn@s de Tecnología e Informática de Amelia Tierno.
Contacta: ameteino@gmail.com

Etiquetas

1ºE.S.O. TECNOLOGÍAS (18)
4ºESO INFORMÁTICA (37)
ANIMACIONES (4)
blog (9)
EDUCACIÓN (2)
Electricidad (2)
Electrónica (4)

Figura 3.14: Artículo de Amelia Tierno sobre FAVS

3.4. Fase 3: Gestión y evolución del proyecto

Tras la campaña publicitaria realizada, los usuarios comenzaron a registrase en el sitio web y a utilizar la aplicación para sus clases. En el capítulo 4 se hace un estudio más detallado de esta comunidad de usuarios, pero podemos adelantar que está compuesta por docentes de varias universidades españolas e institutos de diferentes paises.

En su labor diaria, estos profesores se iban encontrando con problemas, errores o carencias de la aplicación, y nos las fueron haciendo llegar, fundamentalmente, a través de correos

[10] http://auladetecnologias.blogspot.com/2009/05/planeta-favs-vota-el-mejor-post.html

electrónicos. Esta última sección recoge las tareas realizadas para corregir los errores (o *bugs*) que presentaba nuestro servicio y las nuevas funcionalidades y características que se añadieron tras ser demandadas por la comunidad.

3.4.1. Bugs

Enlaces erróneos

Aunque se habían realizado bastantes pruebas con muchos tipos de blogs, cuando comenzó a utilizarse en producción, los usuarios nos hicieron saber que los enlaces a los artículos publicados en ciertos blogs no eran correctos.

Tras estudiar el caso descubrimos que todos los blogs implicados se habían creado en Blogger[11], una herramienta de publicación de blogs gratuita de Google. Sin embargo, no todos los blogs de Blogger presentaban este problema.

Tras mucho investigar descubrimos que el problema estaba con el formato de redifusión Atom utilizado por Blogger. Hasta el final del verano había funcionado correctamente, por eso las pruebas realizadas habían sido satisfactorias. Tras el verano se habían producido algunos cambios y hubo que modificar la url del feed que ofrecía la gema rfeedfinder (que se encarga de buscar los feeds):

En lugar de usar la url *http://sw-libre.blogspot.com/atom.xml* había que utilizar la dirección *http://sw-libre.blogspot.com/feeds/posts/default?alt=rss*

Programamos un script (del que mostramos el código a continuación) que se encargó de modificar los datos de los blogs ya almacenados en la Base de Datos y modificamos el código del método *create-feed* del modelo *feed.rb* para que guardara la url correcta.

```
#!/usr/bin/env ruby
RAILS_ENV = 'production'
require File.dirname(__FILE__) + '/../config/boot'
require "#{RAILS_ROOT}/config/environment"

feeds = []
feeds = Feed.find(:all)
blogger = "blogspot"
feeds.each do |feed|
```

[11] http://www.blogger.com

```
  if feed.link.include? blogger
    feed.link = feed.href + "feeds/posts/default?alt=rss"
    feed.save
  end
end
```

Artículos repetidos en el planeta

El error más molesto para los usuarios y que más dificultades ha presentado para su solución ha sido que, en ciertas ocasiones, en algunos planetas comenzaban a repetirse dos artículos de un blog, y se almacenaban una y otra vez en la base de datos, llenando el planeta de posts repetidos.

La primera vez que ocurrió, cuando el usuario nos avisó del problema, se detuvo la actualización del planeta para que no siguiera almacenando los artículos y se borraron de la base de datos todos los posts repetidos.

Para actualizar los artículos publicados en los blogs hay un script que se ejecuta constantemente, el script *update-feeds*, que realiza una llamada al método *refresh* del modelo *feed.rb*. Este método obtenía los datos del último post publicado (a través del método *read-first* del modelo *Rfeedreader*) y comparaba su título con el del último post de ese blog almacenado en la base de datos. Si eran distintos, creaba un nuevo post con los datos del Rfeedreader.

Se modificó este método para que no sólo comprobara el título, sino que también comparara la url de los posts en cuestión.

Se lanzó de nuevo el script *update-feeds* y todo comenzó a funcionar correctamente.

Pensábamos que habíamos solucionado el problema, pero algún tiempo después otro usuario nos avisó de que se le había surgido la misma incidencia

Comparamos los blogs que habían presentado esta incidencia pero habían sido creados con diferentes herramientas por lo que sospechamos que el problema estaba en nuestro programa. Finalmente, el método refresh fue modificado para que comprobara los datos obtenidos del Rfeedreader con los de todos los posts de ese blog almacenados en la base de datos, y el código queda como se presenta a continuación:

```
def refresh(forced=false)
  historias = []
  encontrado = 0
```

```
  begin
    Timeout::timeout(30) do
      entry = Rfeedreader.read_first(link).entries[0]
      if !entry.nil? and !entry.link.nil?
        if (forced or latest_post.nil?)
          Post.create(:url => entry.link,
                      :title => entry.title,
                      :description => entry.description,
                      :feed_id => id)
        else
          historias = posts
          historias.each do |post|
            if entry.title == post.title
               or entry.link == post.url
              encontrado = 1
              break
            end
          end
          if encontrado == 0
            Post.create(:url => entry.link,
                        :title => entry.title,
                        :description => entry.description,
                        :feed_id => id)
          end
        end
    end
  rescue Timeout::Error
  rescue => err
  end
end
```

3.4.2. Petición de nuevas características

Al utilizar la aplicación los usuarios se han ido encontrado con ciertas carencias o con algunas funcionalidades no implementadas que les ayudarían en su labor diaria. La figura 3.15 es una captura de la sección de petición de nuevas características (Feature Requests) del proyecto en SourceForge:

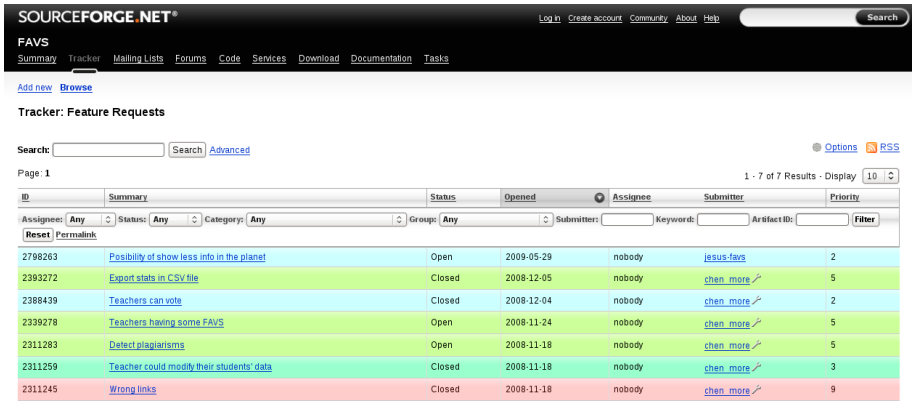


Figura 3.15: Feature Requests

Los docentes necesitan editar los datos de sus alumnos

La primera característica que fue solicitada era que los docentes pudieran editar los datos de los alumnos que se han inscrito en el planeta. Si un alumno introducía su correo electrónico con algún error (o con un espacio al final, por ejemplo), cuando intentaba votar y se le solicitaba su e-mail, la aplicación no le permitía emitir el voto.

Se modificó el controlador *manage-controller* añadiendo el método *edit-student* que se encarga de actualizar los cambios realizados en los datos de los alumnos (la url del blog, su correo y su nombre) llamando al método *update-attributes* del modelo *feed.rb*.

Se diseñó la vista *edit-student.rhtml* que muestra los datos del estudiante seleccionado para su posible modificación, de la que se muestra una captura en la figura 3.16.

Figura 3.16: Editando los datos de un estudiante

Los *profes* quieren votar

En un principio sólo los estudiantes que habían agregado su blog al planeta podían votar las

entradas del resto de compañeros. Sin embargo, algunos profesores nos hicieron saber que les gustaría poder emitir votos para premiar los posts que les resultaban interesantes.

Se modificó el método *check-vote* del controlador *votes-controller* para permitir esta situación añadiendo una nueva comprobación:

```
subscription = Subscription.find(:first,
                                 :conditions =>
                                  ["feed_id = ?", feed_id])
user_id = subscription.user_id
if User.find_by_id(user_id).email == params[:mail]
  return 5
end
```

Descargar las estadísticas de votos

Las páginas diseñadas para mostrar las estadísticas resultan muy útiles para consultar las historias más votadas o los estudiantes más activos, pero cuando los docentes querían descargarlas para utilizarlas en la evaluación de la actividad se encontraban con ciertos impedimentos.

Se decidió añadir la posibilidad de que las estadísticas fueran descargadas en un fichero CSV, del inglés *comma-separated values*, un tipo de documento muy sencillo para representar datos en forma de tabla y que puede ser editado con las hojas de cálculo más populares.

Se modificaron las vistas de las páginas de estadísticas de los distintos idiomas para incluir este enlace y se implementó el controlador *report-controller.rb* que incluye el método *users* que se encarga de gestionar estas peticiones:

```
def users
  @user = User.find(params[:id])
  @blogs = @user.generate_stats_blogs(params[:tags])
  stream_csv do |csv|
    csv << ["student","mail","votes_received", "votes_cast" ]
    @blogs.each do |u|
      csv << [u [:pupil],u [:mail],u [:votes], u[:votes_cast]]
    end
  end
end
```

3.4.3. Un nuevo desarrollador

Desde el mes de abril, Jesús Sánchez, un estudiante del Ciclo Formativo de Grado Superior de Administración de Sistemas Informáticos del IES Gonzalo Nazareno de Dos Hermanas (Sevilla), comenzó a colaborar con FAVS en el marco del desarrollo de su Proyecto Integrado (el proyecto final del ciclo).

Comenzaba una nueva etapa para FAVS; el equipo de desarrollo contaba con un nuevo developer por lo que había que dotar al proyecto de nuevas herramientas:

- Hasta la fecha, aunque el código fuente de la aplicación estaba disponible para su descarga, no se había configurado un CVS público donde escribir los cambios. En la nueva situación esta herramienta resultaba imprescindible para que el equipo de desarrollo pudiera trabajar de forma conjunta.

- Se creó una lista de correo, favs-developers, en la que se llevarían a cabo los intercambios de ideas entre los desarrolladores de manera que toda la información contenida en los debates no se perdiera y quedara disponible para la comunidad.

Tras instalarse en una máquina propia un entorno como el que se describe en la sección 3.1.1, la primera tarea que llevó a cabo fue la de poner a disposición de los usuarios más idiomas en los que mostrar los planetas. En un principio, tan sólo podía trabajarse con castellano e inglés, y Jesús preparó las vistas necesarias para ofrecer soporte, también, en francés e italiano.

En la actualidad Jesús se encuentra trabajando en la implementación de una nueva funcionalidad que ha sido solicitada por un profesor de un instituto de Sevilla. Este profesor nos contaba en un correo electrónico que él ha montado un planeta en el que incluye todos los artículos de sus estudiantes, pero que le gustaría que en una columna de la página principal de la asignatura aparecieran tan sólo el nombre de cada blog y el título del último post publicado.

Puede seguirse la evolución de la implementación de esta nueva característica en la lista de correo de desarrolladores del proyecto.

Capítulo 4

Conclusiones

Una vez descritos los objetivos del proyecto y analizada la solución desarrollada, esta memoria termina resumiendo los principales logros alcanzados y proponiendo posibles trabajos futuros.

4.1. Logros alcanzados

A partir del desarrollo de este proyecto se ha puesto a disposición de los docentes de todo el mundo una herramienta libre que permite, sin exigir apenas conocimientos informáticos, que cualquier profesor pueda crear, mantener y gestionar un planeta que enlace las historias publicadas por sus alumnos en sus blogs. Los estudiantes podrán valorar las historias más interesantes y las estadísticas de votos emitidos y recibidos estarán disponibles.

Consideramos que esta aplicación puede ser utilizada por docentes de todos los niveles educativos, ya que introduce en el aula aspectos relacionados con la propiedad intelectual y el conocimiento libre, propicia la creación y el intercambio de conocimiento y potencia las habilidades de redacción de los estudiantes motivando una sana competición por escribir los artículos más interesantes.

Se ha conseguido crear una comunidad de usuarios del proyecto que informa de los problemas y carencias que detectan en la aplicación y ha comenzado el proceso de adhesión de nuevos desarrolladores, por lo que todos los objetivos descritos en el capítulo 2 han sido alcanzados con éxito.

La lista que se muestra a continuación incluye las universidades, institutos, escuelas o enti-

dades que, por haber establecido contacto con nosotros, o por el correo electrónico registrado en la base de datos, tenemos constancia de que están usando nuestro servicio:

- Universitat Oberta de Catalunya.
- Universidad Rey Juan Carlos.
- Universidad de Valencia.
- IES María Moliner, Sevilla.
- IES Gonzalo Nazareno, Dos Hermanas, Sevilla.
- IES Las Galletas, Arona, Tenerife.
- IES Ignacio Ellacuría, Madrid
- IES Eduard Fontserè, L'hospitalet de Llobregat, Barcelona.
- Colegio Nacional Mariano Melgar, Perú.
- Escuela Especial nº 1383, Rosario, Argentina.
- Planeta Educativo
- Profeblog

4.2. Posibles trabajos futuros

Aunque los objetivos descritos en el capítulo segundo han sido alcanzados y, por tanto, el desarrollo de este trabajo final de máster se da por finalizado, estamos seguros de que el proyecto va a seguir creciendo y evolucionando. La comunidad de usuarios que se ha formado a su alrededor seguirá detectando nuevas necesidades que, esperamos, serán implementadas por el equipo de desarrolladores, añadiendo funcionalidad y nuevas características.

Planteamos algunas posibles líneas de desarrollo futuro:

1. Dotar a la aplicación de la capacidad de detectar votaciones engañosas. Un ejemplo de votación engañosa se da si un estudiante vota 10 artículos en 1 minuto, ya que parece

difícil que los haya leído. O, por ejemplo, cuando 2 estudiantes se votan constantemente el uno al otro.

2. Detectar plagios de forma automática antes de añadir un nuevo artículo en la Base de datos.

3. Facilitar la gestión de los distintos planetas de un docente. Aunque esta posibilidad existe en la actualidad, como se explica en el Anexo A, resulta un tanto incómoda si el número de estudiantes es elevado.

4. Permitir la posibilidad de que los docentes que no tengan blog o sitio web puedan elegir algún tipo de plantilla y que nuestro sitio albergue sus planetas.

Anexo A

Breve manual de usuario de FAVS

Para crear un planeta con FAVS, el profesor tan sólo debe registrarse en la web http://favs.homelinux.org, eligiendo un nombre de usuario y una contraseña y proporcionando su dirección de correo electrónico.

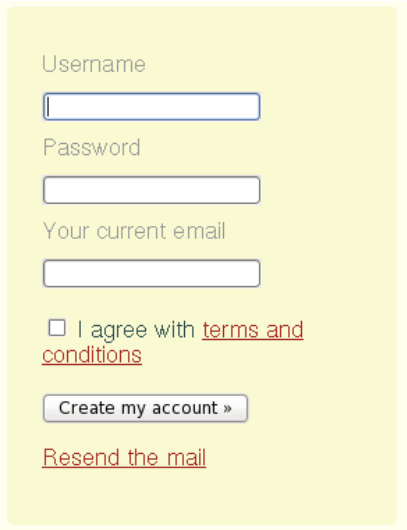


Figura A.1: El docente se registra

En ese momento FAVS genera una etiqueta HTML que el profesor colocará en su blog o en su página web. A continuación, el planeta puede personalizarse, eligiendo el idioma (inglés o español), los artículos mostrados (decidiendo si se mostrarán todos los post de cada alumno o sólo el último) y el estilo y los colores.

Figura A.2: FAVS genera la etiqueta HTML

Una vez que el docente ha pegado la etiqueta HTML en su sitio web, los estudiantes de la asignatura podrán darse de alta en el planeta pinchando en el enlace que aparecerá en el planeta.

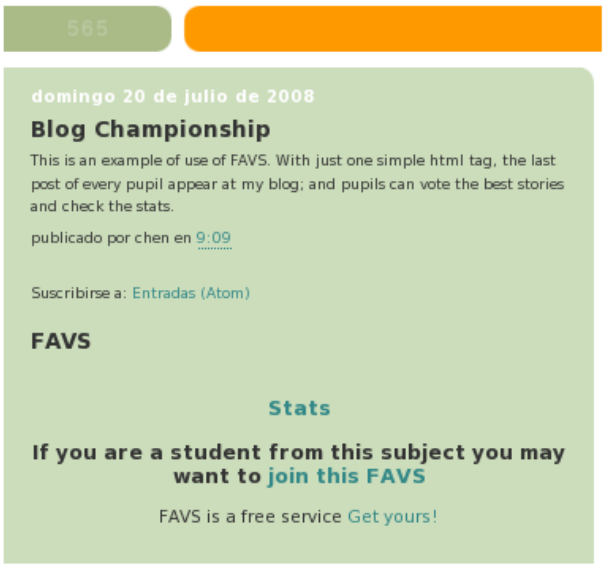


Figura A.3: El blog del docente al pegar la etiqueta HTML

Los datos que se solicitan al estudiante son la URL de su blog, su nombre y su correo electrónico.

Figura A.4: Agregando un blog al planeta

Al registrar su blog, FAVS pregunta al alumno si ha elegido una licencia para los contenidos de su bitácora, y lo anima a utilizar una licencia Creative Commons. A partir de ese momento, en la web del profesor aparecerá un resumen de los artículos que escriba cada estudiante en su bitácora, junto con su fecha de publicación, un enlace para leer el post completo y un enlace para votar esa historia.

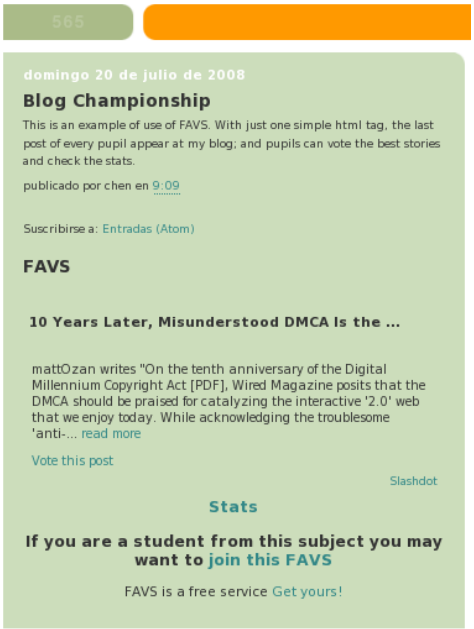


Figura A.5: Los posts publicados en los blogs de los estudiantes se enlazan en el planeta

Cuando un estudiante decide votar una historia, FAVS solicita su correo electrónico y comprueba que el alumno pertenece a esa asignatura, que no ha votado ya por ese artículo y que no se está votando a sí mismo. Con el objetivo de detectar posibles votos falsos, la aplicación envía

un comprobante por correo electrónico en el que se indica que se acaba de votar una noticia y se insta a contactar con los profesores en el caso de que no sea así.

Figura A.6: Votando un artículo

Tanto los estudiantes como el docente pueden consultar en cualquier momento las estadísticas de votos acumuladas. Estas estadísticas muestran los votos recibidos por cada blog, los votos recibidos por cada artículo y los votos emitidos por cada alumno.

Figura A.7: Estadísticas de votos recibidos y emitidos

Aunque los estudiantes pueden agregar sus bitácoras al planeta, el docente también tiene la posibilidad de añadir los blogs que desee. Para ello tan sólo debe seguir el enlace *add another blog* del panel de gestión de FAVS tras identificarse en la aplicación.

Figura A.8: Los docentes pueden añadir blogs al planeta

En el panel de gestión de FAVS los docentes pueden ver los blogs que han sido agregados a su planeta y tienen la posibiidad de editar los datos de cada bitácora o, incluso, de eliminar algún blog que no forme parte de la asignatura. Para ello deben usar los enlaces *edit student's data* o *delete* del blog correspondiente.

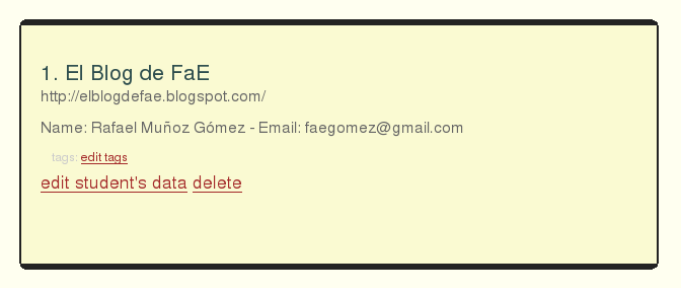


Figura A.9: Borrar o editar los datos de un blog

Si un profesor quiere tener varios planetas distintos puede hacerlo sin tener que crearse diferentes usuarios haciendo uso de las etiquetas o tags. Para ello, desde el panel de gestión FAVS, debe pinchar sobre el enlace *edit tags* y escribir la etiqueta deseada. En el sitio web

donde el profesor tiene el código de FAVS, debe reemplazar el atributo *src* siguendo el siguiente formato para filtrar los blogs mostrados:

src=”http://www.FAVS.com/code/your-id/tag/subject-name”

De esta forma tan sólo se mostrarían los blogs etiquetados con el tag *subject-name*.

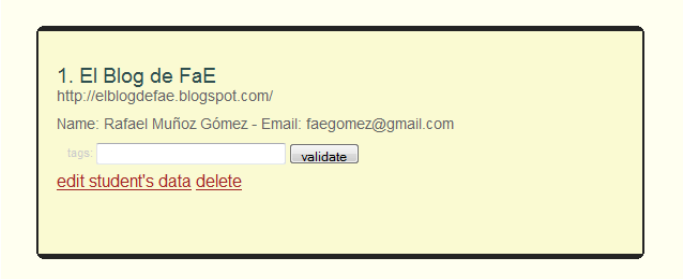


Figura A.10: Utilizando las etiquetas

Anexo B

Instalando Feevy. Ayuda recibida.

Se recogen en este anexo los correos electrónicos intercambiados con los desarrolladores del proyecto Feevy en los que nos ayudaron a instalar en un servidor propio su aplicación:

```
Hola [líder del proyecto Feevy],

Antes de nada me gustaría felicitaros por vuestro trabajo,
y por la idea de liberar Feevy.

Para mi Proyecto Fin de Carrera estoy desarrollando una
aplicación que va a utilizar Feevy, así que estoy intentando
instalar Feevy en mi servidor, para jugar un poquillo y
realizar algunos cambios, pero estoy teniendo problemas.

Siguiendo las instrucciones que publicásteis en la bitacora,
hay algunas gemas que no puedo instalar: open-uri y timeout
no  se encuentran.
Además al intentar realizar rake db:schema:dump para
construir la BBDD siempre obtengo un error con el fichero
Rakefile.

¿A qué pueden deberse estas situaciones? Estoy empezando a
trabajar con rails y aún soy novatillo, pero sospecho que
puede deberse a las versiones que uso. He instalado las
últimas  versiones de rails, ruby y gems. ¿Puede ser ése
el problema? ¿Qué versiones serían necesarias para poder
instalarlo?

Muchas gracias. Un saludo,
```

Jesús Moreno.

Hola Jesús!

Que bueno lo de tu proyecto! Gracias por usar feevy!

Sobre tu problema, me da que estás utilizando la segunda
versión con las instrucciones de instalación de la primera...

Te respondo CC a [desarrollador de Feevy] y a
[desarrolladora de Feevy]. Seguro que [desarrollador de Feevy]
te puede ayudar, porque yo hice una instalación de la primera
pero no de la segunda y creo que hay mejoras sustanciales :-)

Un abrazo grande
[líder del proyecto Feevy]

Hola Jesus,

Graias para tu mensage :)

Tienes el codigo de github? Si no, puedes cogerlo allí:
http://github.com/alx/feevy/tree/master

Hace un monton que no he instalado Feevy del principio,
problamente es una cosa de Rake que ha desaparecido en
una nueva version.
Puedes enviarme el mensage que tienes, con el mas
informaciones posible? tiendras mas informaciones
con: rake --trace

Muchas gracias,

[desarrollador de Feevy]

Hola [desarrollador de Feevy],

Gracias por tu interés. Me he descargado la última versión
del repositorio, pero al instalar las gemas necesarias

obtengo esto:

```
Successfully installed rails-2.1.0
Building native extensions.  This could take a while...
Successfully installed hpricot-0.6.161
Successfully installed simple-rss-1.1
ERROR:  could not find gem open-uri locally or
in a repository
ERROR:  could not find gem timeout locally or
in a repository
Successfully installed cached_model-1.3.1
Building native extensions.  This could take a while...
Successfully installed mongrel-1.1.5
Successfully installed htmlentities-4.0.0
Successfully installed rfeedfinder-0.9.12
Successfully installed rfeedreader-1.0.13
Successfully installed rfeedfinder-0.9.12
9 gems installed
```

Como ves, no puedo instalar dos de las gemas.

He configurado el fichero database.yml con las
credenciales necesarias y al ejecutar
rake db:schema:dump --trace:

```
(in /var/rails/feevy2)
rake aborted!
undefined method `require_gem' for main:Object
/var/rails/feevy2/config/boot.rb:28

....

/usr/local/bin/rake:19:in `load'
/usr/local/bin/rake:19
```

Quizás es por la versión de rails que uso? es la 2.1.0.

Muchas gracias!

Hola Jesus,

```
Probando ayer con Feevy, tenia el mismo problemo con
Rake. Necesitas cambiar "require-gem" con solo "gem"
en config/boot.rb

No sé para las gemas, ceo que timeout y open-uri son
dentro la libreria estandar de ruby, entonces no debres
tener más problemas con eso.

En config/environment.rb, la versión de rails está
bloqueada a la 1.2.3. Creo que la ultimá vez que he mirado
un poco Feevy, fue experimantando haciendo upgrade hasta
rails2, y no tenia muchas problemas.
El mejor sere que haces un fork de Feevy en Github, donde
podrias poner todos tus cambios, eso foncionaba bien
con rFeedReader y rFeedFinder para tener ayuda de otros
desalloradores :)

Suerte, estoy esperando para tus avancadas, y te ayudaré
si tienes algun pregunta :)

[desarrollador de Feevy]

---------------

Hola [desarrollador de Feevy],

perdona por molestarte de nuevo :-(

Ahora obtengo otro error, parece que no encuentra
libgd.so.2, que está instalado y se encuentra
en /usr/local/. Te dejo la salida:

rake aborted!
libgd.so.2: cannot open shared object file:
No such file or directory

...

/usr/local/lib/ruby/gems/1.8/gems/rake-0.8.1/bin/rake:31
/usr/local/bin/rake:19:in `load'
/usr/local/bin/rake:19


Alguna idea? Muchas gracias de nuevo por tu ayuda!

---------------
```

```
Hola Jesús,

Ah, la famosa libgd :)
Me ha dado tanto susto el año pasado que tengo ese codigo
en mi buzon:

Patch gd2.rb with following line to link to correct
libgd.so:

def self.gd_library_name
    case Config::CONFIG['arch']
    when /darwin/
      'libgd.2.dylib'
    when /mswin32/, /cygwin/
      'bgd.dll'
    else
      '/usr/local/lib/libgd.so.2'
    end

Tienes que hacer ese patch sobre la gema de gd,
dentro gd2.rb, y digame si fonciona ñejor :)

Hasta ahora,

--------------

Hola [desarrollador de Feevy]! Ya he solucionado el
problema con gd2, eres un artista! :-D

Ahora obtengo un error al migrar la BBDD,

 DefaultSubAvatar: migrating ================
-- change_column(:subscriptions, :avatar_id,
:integer, {:default=>1}) rake aborted!
Mysql::Error: Unknown column 'avatar_id' in 'subscriptions':
ALTER TABLE subscriptions CHANGE
avatar_id avatar_id int(11) DEFAULT 1


Saludos!

-------------

Hola [desarrollador de Feevy],
```

mirando la BBDD he visto que la tabla subcriptions no
tiene un campo avatar-id, (de hecho en el migrate
017-migrate-avatar.rb aparece comentada la línea
que crea ese campo)

¿No debería aparecer?

Hola Jesus,

he subido una correcion para la migración 017 en un
nuevo commit en github, podría ayudarte la próxima vez :)

http://github.com/alx/feevy/commit/f558...1ff7f6cfc9c1e0ae

Un saludo,

[desarrollador de Feevy]

Hola [desarrollador de Feevy],

Muchas gracias por toda la ayuda!!

Por fin tengo una versión de Feevy corriendo en mi servidor.

Ya puedo comenzar a estudiarla y modificarla, os iré contando
mis avances.

Un saludo y muchas gracias de nuevo!

Jesús.

Bibliografía

[1] Orihuela, José Luis. *La revolución de los blogs*. La Esfera de los Libros, 2006.

[2] Ganley,Barabara. *Blogging as a dynamic, transformative medium in an American liberal arts classroom*. BlogTalks 2.0: The European Conference on Weblogs, Vienna, Austria, 2004.

[3] Robles, Gregorio, González Barahona, Jesús María y de las Heras Quirós, Pedro. *Experiencia de uso de blogs en e-learning*. Tercera Jornada de Innovación Pedagógica del Proyecto ADA Madrid, Fuenlabrada, España, 2008.
`http://moodle.upm.es/adamadrid/file.php/1/web_III_jornadas_ADA/Comunicaciones/Robles&20Martinez.pdf`

[4] Downes,Stephen *Educational Blogging*. EDUCAUSE Review, vol. 39, no. 5 (September/October 2004): pp 14–26.
`http://www.educause.edu/ir/library/pdf/ERM0450.pdf`

[5] González Barahona, Jesús, Senoane Pascual, Joaquín y Robles Martínez, Gregorio *Introducción al Software Libre*. UOC, 2007.

[6] Lenz, Patrick. *Simply Rails 2 (2nd Edition)*. SitePoint, 2008.

[7] Ruby, Sam, Thomas, Dave y Hansson, David. *Agile Web Development with Rails (3rd Edition)*. Pragmatic Programmers, 2008.

[8] Kent, Beck y Andres, Cynthia. *Extreme Programming Explained: Embrace Change (2nd Edition)*. Addison-Wesley Professional, 2004.

[9] Amor, Juan José, Herraiz, Israel y Robles, Gregorio. *Desarrollo de proyectos de software libre (segunda edición)*. UOC, 2007.

[10] DiBona, Chris, Ockman, Sam y Stone, Mark. *Open Sources: Voices from the Open Source Revolution*. O'Reilly Open Source, 1999.

[11] Nielsen, Jakob y Loranger, Hoa. *Prioritizing Web Usability*. Pearson, 2006.