Proyecto Fin de Carrera

Desarrollo de una aplicación web de una revista social orientada a la docencia

Autor: Alberto Fernández Zazo
Tutor: Gregorio Robles Martínez

La defensa del presente Proyecto Fin de Carrera se realizó el día de de , siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de

A mis padres

"Sólo con el corazón se puede ver bien;

lo esencial es invisible para los ojos"

Antoine de Saint-Exupéry. El Principito.

Agradecimientos

Ciertos profesores de la carrera, muchos, se esfuerzan y buscan de verdad enseñar. Gracias a todos ellos.

Gracias a mi tutor por haberme dado la idea de este proyecto y la libertad para llevarlo a cabo a mi gusto, ya que he tenido la gran suerte de disfrutar desarrollándolo.

Gracias a la salsa y a las salseras, por ser lo que necesitaba cuando lo necesitaba. En especial a una tan loca, tan rara y tan buena, una que me ha mostrado una perspectiva nueva y maravillosa.

Gracias a esos poquitos amigos del colegio a los que apenas veo, pero que son muy importantes para mí. Sé que estáis y estaréis ahí.

Gracias por supuesto a la "Euromanía", al "Come y bebe sin límites" y a los cumples VIP, por dar lugar a los recuerdos inolvidables que me quedarán de esta etapa. Recuerdos que lo serán por vivirlos junto a los amigos de la universidad.

Gracias a mis hermanos, de los que tanto he aprendido, y por último, y más importante, gracias a mis padres, que siempre me han dado la educación, la libertad, la confianza y el apoyo para poder elegir y seguir mi camino, por el que llevo andando 24 felices años hasta aquí.

Resumen

Este proyecto pretende ser una herramienta útil y moderna para la actividad docente, promoviendo la interacción y participación de los alumnos y otorgando herramientas de calificación a los profesores. Por ello, el proyecto consiste en una aplicación web que facilita la inclusión de noticias relacionadas con una o varias asignaturas, de forma que en su conjunto forman una revista online y colaborativa. Las noticias se muestran con los titulares que la aplicación detecte y acompañadas de una imagen.

Para promover la interacción de los alumnos, la aplicación cuenta con elementos comunes en redes sociales, como son los comentarios en las noticias, el botón de "Me gusta" y el uso de etiquetas. También se incluye un ranking de usuarios por puntos, que se consiguen participando en la aplicación. En cuanto a los profesores, para facilitar la calificación y el seguimiento de los alumnos, pueden ver todas las acciones relevantes realizadas por cada uno de los usuarios de la aplicación.

Se pretende publicar este proyecto con una licencia de software libre, por lo que se ha elegido una tecnología acorde con ese entorno. Dicho esto, el *famework* elegido es Django, se ha desplegado en una máquina virtual de la Universidad Rey Juan Carlos sobre servidores Apache y Nginx, empleando una base de datos relacional sqlite3, mostrando páginas web con HTML5 para el contenido y CSS3 para el estilo, y con funciones de AJAX para el contenido dinámico con la ayuda de la librería jQuery.

Índice general

Índice de figuras

List of Tables

Acrónimos

GPL	General Public License
GNU	GNU is Not Unix
IDE	Integrated Develepment Environment
PHP	Hypertext Pre-Processor
HTML	HyperText Markup Language
XML	eXtensible Markup Language
W3C	World Wide Web Consortium
WHATWG	Web Hypertext Application Technology Working Group
CSS	Cascade Style Sheet
REST	Representational State Transfer
RSS	Rich Site Summary
AJAX	Asynchronous JavaScript and XML
SQL	Structured Query Language
WSGI	Web Server Gateway Interface
ASCII	American Standard Code for Information Interchange
UTF-8	UCS Transformation Format
UCS	Universal Character Set

Parte I

Introducción y objetivos

Capítulo 1

Introducción

"Comenzar bien no es poco, pero tampoco es mucho"

Sócrates

1.1. Estructura de la memoria

La estructura que sigue esta memoria se acerca a la planificación cronológica del proyecto y es la que sigue:

Primero, en el capítulo de la introducción, se habla de las tecnologías relevantes y las aplicaciones existentes similares al proyecto, para luego tratar los objetivos que va a tener el proyecto.

A continuación, en el capítulo dos, se describe toda la fase de diseño de la aplicación. Esto es, las tecnologías que se van a utilizar, los directorios y ficheros fundamentales que va a tener el proyecto, una descripción de las funcionalidades que tendrá la aplicación y un boceto general de su interfaz. En el siguiente capítulo se trata toda la implementación en código de lo diseñado, desde las bases de datos hasta el algoritmo empleado para extraer los datos de las noticias, y además una sección de problemas donde se explican algunas de las dificultades encontradas en esta fase.

Por último, se tratan las pruebas y los resultados. Primero se explican los pasos del despliegue de la aplicación para poder realizar las pruebas, cuyos resultados se exponen, y por último se extraen conclusiones, tanto de los resultados como del proyecto en general.

1.2. Estado del arte

Para tratar el estado del arte se estudiará, por un lado, el estado de las tecnologías actuales a tener en cuenta para el diseño del proyecto. Por otro lado, se analizarán algunas páginas web similares a la desarrollada.

1.2.1. Estado de las tecnologías

- Software libre

  Aquel software cuyo código fuente es accesible a cualquier usuario, y que además puede ejecutar, modificar y distribuir con total y absoluta libertad [1]. Para proteger estas reglas, a menudo el software contiene una licencia de copyleft que otorga ciertos derechos a los autores sin afectar a las libertades de los usuarios. Estos derechos pueden ser exigir que las modificaciones sean también software libre, que se identifiquen las modificaciones realizadas, que se cambie el nombre, etc.

  Existen multitud de licencias, y la más importante con diferencia es la licencia GPL de GNU.

- Aplicaciones REST

  Estas aplicaciones interactúan mediante peticiones HTTP entre cliente y servidor que, entre otras características, no requieren del uso de estados [2]. Esto es, toda la información necesaria se aporta con la petición y cada recurso disponible está claramente identificado.

- Entornos de desarrollo

  Un entorno de desarrollo integrado o IDE es un programa informático que proporciona un marco de trabajo amigable con herramientas

para trabajar con uno o varios lenguajes de programación. Contienen herramientas de depuración, ejecución, árbol del proyecto, detección y resolución de errores sintácticos en tiempo real, etc.

Aunque hay muchos, los que destacan sobre los demás por ser software libre y adecuado para el desarrollo de aplicaciones web son:

- Netbeans

  Desarrollado inicialmente para programación en Java, funciona con gran cantidad de lenguajes y proporciona multitud de herramientas [3].

- Eclipse

  Similar a Netbeans, puede trabajar y aportar herramientas para casi cualquier lenguaje y función mediante la instalación de módulos específicos [4].

- *Framework* de desarrollo web

  La pieza elemental para el desarrollo de una aplicación web es una plataforma que agiliza y simplifica enormemente la tarea. Proporcionan un marco de trabajo estructurado donde tratar fácilmente con bases de datos, con plantillas web, control de sesiones, seguridad, etc [36]. Por lo general siguen una estructura Modelo Vista Controlador que favorece la encapsulación en el desarrollo de aplicaciones web. De los orientados a aplicaciones web, los dos más importantes son:

  - Django: creada en el lenguaje Python [5].
  - Ruby on rails: similar a Django, pero para el lenguaje Ruby, que también es similar a Python [6].

- Lenguajes de programación

  Para desarrollar la aplicación se pueden emplear uno o varios, y a continuación se detallan algunos de los más apropiados para la programación web ya que, como puede verse en el cuadro 1.1, se encuentran entre los diez lenguajes de programación con más crecimiento en la actualidad:

- Ruby

  Lenguaje libre y orientado a objetos, fue publicado en 1995 y debe gran parte de su éxito a la ya mencionada plataforma de desarrollo web Ruby on Rails [7].

- Pyhton

  Python es un lenguaje libre, interpretado, de tipado dinámico y capaz de trabajar con distintos paradigmas (orientado a objetos, entre otros) [8]. Es muy intuitivo, rápido y eficaz para la programación web.

- Java

  Lenguaje creado en 1995 a partir de C pero a más alto nivel y orientado a objetos, con la novedad de que era válido para cualquier lugar porque se ejecutaba el mismo código en máquinas virtuales adecuadas a la plataforma y arquitectura [9][10].

Position May 2013	Position May 2012	Delta in Position	Programming Language	Ratings May 2013	Delta May 2012	Status
1	1	=	C	18.729%	+1.38%	A
2	2	=	Java	16.914%	+0.31%	A
3	4	↑	Objective-C	10.428%	+2.12%	A
4	3	↓	C++	9.198%	-0.63%	A
5	5	=	C#	6.119%	-0.70%	A
6	6	=	PHP	5.704%	-0.07%	A
7	7	=	(Visual) Basic	4.656%	-0.80%	A
8	8	=	Python	4.322%	+0.50%	A
9	9	=	Perl	2.276%	-0.53%	A
10	11	↑	Ruby	1.670%	+0.22%	A
11	10	↓	JavaScript	1.536%	-0.60%	A
12	12	=	Visual Basic .NET	1.131%	-0.14%	A
13	15	↑↑	Lisp	0.894%	-0.05%	A
14	18	↑↑↑↑	Transact-SQL	0.819%	+0.16%	A
15	17	↑↑	Pascal	0.805%	0.00%	A
16	24	↑↑↑↑↑↑↑↑	Bash	0.792%	+0.33%	A
17	14	↓↓↓	Delphi/Object Pascal	0.731%	-0.27%	A
18	13	↓↓↓↓↓	PL/SQL	0.708%	-0.41%	A
19	22	↑↑↑	Assembly	0.638%	+0.12%	B
20	20	=	Lua	0.632%	+0.07%	B

Cuadro 1.1: Índice TIOBE de la comunidad de programadores Mayo 2013 [11].

- HTML

  HTML es el lenguaje más utilizado en internet para describir una página web. Se forma a partir de marcas o etiquetas (que pueden anidarse y formar un árbol) de la forma <etiqueta> para declarar el comienzo y de la forma </etiqueta> para declarar su fin, y entre medias suele introducirse el contenido. Estas etiquetas predefinidas tienen distintas propiedades y funciones y además pueden contener "atributos" que proporcionan información adicional [12].

  De esta manera, se puede dar formato al contenido, introducir enlaces, usar formularios, incluir imágenes, crear tablas, etc.

  Aunque tiene estrictas reglas, no siempre las páginas de internet forman un documento HTML válido, y los navegadores y tecnologías que trabajen con estos documentos tienen que tratar de interpretarlo correctamente de todas formas.

  La primera versión data de 1991, creada por Tim Berners-Lee, y desde entonces ha ido evolucionando en varias versiones hasta las dos que se utilizan actualmente: HTML 4.0.1 (1999) y HTML5 (en desarrollo) [13].

  La versión HTML5 está siendo desarrollada por el W3C (World Wide Web Consortium) y el WHATWG (Web Hypertext Application Technology Working Group), y la mayoría de los navegadores se actualizan para soportarla. Uno de sus objetivos fundamentales es ser independiente del dispositivo en el que se muestre. Además, reduce la dependencia de pluggins, introduce nuevos elementos y atributos, facilita la inclusión de audio y vídeo, los gráficos 2D y 3D (mediante canvas), permite el almacenamiento local de información, etc [14].

- CSS

  Desarrollado para funcionar con HTML 4, los ficheros CSS se encargan de definir el estilo de una página (principalmente en HTML) desde un archivo externo enlazado. Es posible describir el estilo dentro de la página pero no es recomendable salvo para algunas excepciones. El objetivo

de estos ficheros es separar el contenido de la forma en las páginas web [15].

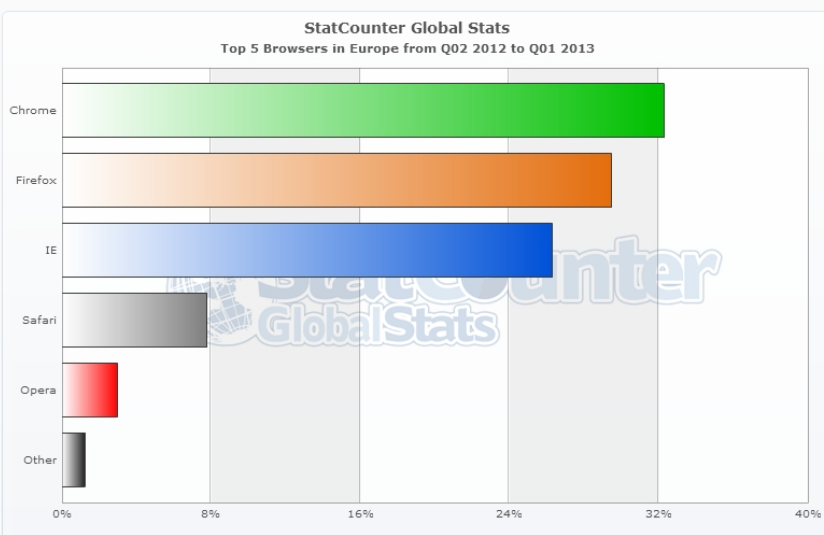
Actualmente la versión más reciente es CSS3, la cual añade bastantes propiedades, principalmente orientadas a objetos de dos y tres dimensiones.

- JavaScript

  Lenguaje de programación creado para ejecutarse principalmente en el navegador del cliente y aportar dinamismo a la web [16]. Es sencillo y funciona orientado a eventos que suceden en el navegador, como movimientos del ratón o clicks, ejecutando funciones que modifican contenido o estilo de la página.

- Navegadores

  Actualmente en Europa, como puede verse en el cuadro 1.2, cuatro navegadores engloban más del 95 % del mercado: Mozilla Firefox, Google Chrome, Safari e Internet Explorer. Por lo tanto, un requisito de la aplicación será funcionar correctamente al menos en estos navegadores, luego se seleccionarán tecnologías compatibles con ellos.



Cuadro 1.2: Comparativa de navegadores por su uso. 1 Mayo 2012 - 30 Abril 2013 [17]

- Parseadores

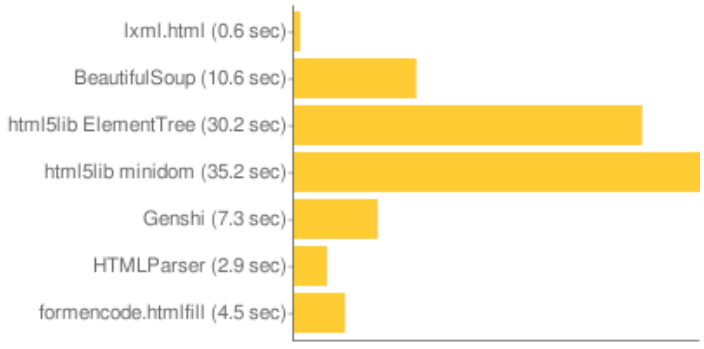
  Existen multitud de parseadores para recorrer las páginas web con noticias que reciba la aplicación y extraer la información deseada, y a menudo también se encargan de introducirla. Los dos más populares son:

  - Beautifoulsoup

    Librería para Python, es bastante sencillo especialmente en la tarea de extraer datos de los ficheros, y destaca en su capacidad para trabajar con ficheros HTML mal formados, los cuales desgraciadamente son muy comunes en la red [18].

  - lxml

    Desarrollado también en Python [19], es el más rápido y más potente, aunque por lo tanto también algo complejo



Cuadro 1.3: Comparativa de rapidez de parseadores de HTML [20]

1.2.2. Antecedentes

Existen multitud de aplicaciones en la red con similitudes con el objetivo de este proyecto, y por lo general se pueden englobar en dos grupos:

- Revistas colaborativas

  Estas aplicaciones consisten en revistas orientadas a la publicación de noticias relacionadas con algún tema en concreto o con una serie de secciones, pero escritas por colaboradores, usuarios, o cualquier visitante externo, como la revista hala [23].

  Un caso a destacar es el de Magazine Factory [24], que provee de multitud de herramientas a los colegios principalmente europeos para que creen sus revistas. A veces se agrupan colegios de varios países para conformar la revista, cuyas noticias son creadas por los alumnos.

  Por lo general, el aspecto de estas aplicaciones es bastante sencillo, con mucho espacio en blanco y a veces con pocas fotos, ya que las noticias son más bien artículos.

- Redes sociales de noticias

  Estas aplicaciones tienen una orientación completamente diferente. Las noticias son oficiales, externas a la aplicación, pero las revistas son individuales. Esto es, cada usuario incluye en su revista las noticias que desee y puede "seguir" las publicaciones que incluyen otros usuarios en sus respectivas noticias.

  Los dos ejemplos más claros de este tipo de aplicaciones son:

  - Scoop.it

    Las noticias se pueden incluir en la revista desde la revista de otro usuario o directamente introduciendo su url, de forma que la aplicación extrae con gran precisión su titular principal y los secundarios, además de una imagen [26]. Además, permite editar el texto y elegir la imagen entre las capturadas[25].

  - Paper.li

    En este caso, las noticias se extraen de fuentes conocidas como redes sociales y RSSs, y permite multitud de herramientas de edición del aspecto de la noticia, como colores, tamaño de la fuente, etc [27]

El aspecto en este caso suele ser más parecido a un periódico online, con una estructura de las noticias muy bien elaborada y un aspecto muy cuidado [28].

Por lo tanto, la idea de este proyecto es nueva, ya que los antecedentes que publican noticias que parsean de la web como Scoop.it no están orientados a la creación de una única revista ni a a ser una herramienta docente, y las que si están orientadas a eso están formados por artículos escritos por los usuarios y no por noticias publicadas en fuentes externas.

1.3. Objetivos

El objetivo principal de este proyecto es la creación de una aplicación web orientada al uso docente mediante su análisis, diseño, implementación, despliegue y depuración. La idea general es la de crear una revista colaborativa y social entre alumnos y profesores, formada por noticias relacionadas con una asignatura.

Para ello, la aplicación debe proporcionar tres funcionalidades fundamentales:

- Una correcta captura y síntesis de noticias externas, de cualquier origen, de forma que se pueda mostrar lo más relevante de la noticia. Esto es, titulares y una imagen o un vídeo, si los hubiere.
- Elementos atractivos para los alumnos, para que se interesen y disfruten con la revista.
- Herramientas disponibles para los profesores que faciliten la calificación y seguimiento de los alumnos.

Además, el proyecto se englobará dentro de un entorno de software libre, y a su conclusión se liberará con alguna licencia de copyleft a decidir.

Parte II

Diseño e implementación

Capítulo 2

Diseño

"Have the courage to follow your heart and intuition. They somehow already know what you truly want to become"

Steve Jobs

Antes de comenzar con la implementación es fundamental establecer claramente todos los detalles, la estructura, el funcionamiento y las herramientas del proyecto.

2.1. Tecnologías elegidas

A partir del estudio de las tecnologías existentes desarrollado en el punto 1.2.1 y de los objetivos de la aplicación destacados en el punto 1.3, se ha decidido emplear los siguientes lenguajes, programas y tecnologías:

Django y Python

Se ha optado por Django principalmente porque ya trabajé anteriormente con él y fue una buena experiencia, mientras que no tenía experiencia alguna con Ruby on Rails y poca programando en Ruby. Concretamente, se ha trabajado con la última versión oficial al inicio del proyecto: la versión 1.4.5.

Esta decisión está relacionada directamente con el lenguaje a programación, que por lo tanto será Python. Concretamente la versión 2.7, ya que las versiones siguientes no son compatibles con la versión 1.4.5 de Django. Como ya se ha explicado anteriormente, es un lenguaje muy importante y muy usado para la programación web.

Editor de textos

Una desventaja de Django es que no incluye un editor de textos con el que escribir el código necesario, por lo que ha sido necesario elegir uno.

En un principio se optó por utilizar un IDE como Eclipse o Netbeans, que aportan mucho más que un editor de textos, pero al final se escogió uno específico para Python, y además preparado para proyectos en Django: PyCharm. Sin embargo, la experiencia como usuario no fue satisfactoria en muchos aspectos, y además el programa no era software libre, por lo que tras el mes de prueba se abandonó.

A continuación, puesto que las dimensiones del proyecto no obligaban al uso de un IDE, se optó por el editor de textos por defecto en Ubuntu: gedit. Desgraciadamente, no aporta casi ventajas más allá del uso de colores para diferenciar el tipo de texto en el código, por lo que se buscó otra herramienta mejor.

El editor de textos definitivo ha sido Sublime Text 2. Mejora sustancialmente a gedit e incluso a cualquier IDE como editor, aporta una sencilla gestión de proyectos, y la experiencia como usuario ha sido realmente buena, aunque no aportara corrección de sintaxis.

BeautifulSoup

Inicialmente se decidió trabajar con lxml por su notable superioridad en velocidad a sus compañeros como se vió en el cuadro 1.3. Sin embargo, tras un estudio y unas pruebas iniciales, el comportamiento no era el deseado, principalmente por la falta de conocimientos y complejidad del programa. Por

lo tanto, se optó por cambiar a BeautifulSoup por su capacidad para la extracción de datos incluse en páginas mal formadas y por el conocimiento y experiencia previa que se tenía de éste.

Front end

A pesar de que existen multitud de programas que facilitan la creación de páginas web con interfaz gráfica y herramientas, se ha rechazado utilizarlos porque la aplicación es compleja, y es aconsejable trabajar directamente con el código para obtener los resultados óptimos. Por lo tanto, en el proyecto se ha trabajado directamente con las siguientes tecnologías en la interfaz:

HTML5 Se ha optado por esta versión porque, aunque no sea un estándar definitivo, uno de sus objetivos fundamentales es ser independiente del dispositivo en el que se muestre, lo cual es un requisito indispensable para el proyecto.

CSS3 Se ha elegido esta versión por ser la más moderna y la que más posibilidades ofrece, y por supuesto porque separar contenido y forma hace el trabajo mucho más sencillo.

AJAX Para aportar dinamismo a la página y comodidad es necesario aprovecharse de las funciones de JavaScript. De esta manera, se pueden ejecutar funciones como introducir contenido nuevo, ocultar elementos, modificar el estilo, etc, que se activan por eventos tales como hacer “click”, colocar el ratón sobre un elemento, etc.

Para programar las funciones dinámicas que harán falta para mejorar la experiencia del usuario se optará por usar AJAX, que utiliza varias tecnologías, entre ellas técnicas de JavaScript junto a peticiones al servidor de XML [16]. Gracias a que sus peticiones son asíncronas, la interfaz no se queda bloqueada mientras se solicitan peticiones al servidor y se sustituyen partes de la página sin recargarla [22]. Además, se hará uso de la librería de JavaScript jQuery por ser una herramienta muy útil que simplifica el trabajo con AJAX.

En la siguiente imagen se muestra de forma gráfica las tecnologías involucradas en el empleo de AJAX.

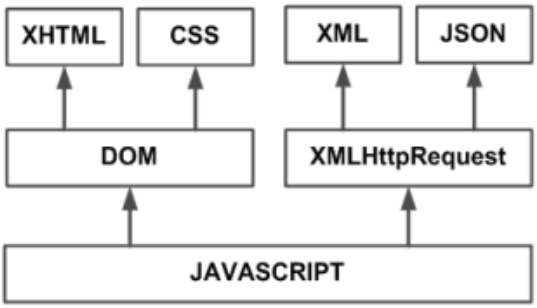


Figura 2.1: Gráfico de las tecnologías implicadas en AJAX [21]

Chrome

Para poder depurar la parte del programa que concierne a la interfaz, estos son, los .html, el .css y el .js, es muy útil e incluso necesario usar algunas herramientas de depuración específicas para desarrolladores. Los dos navegadores que mejor proveen estas herramientas son Firefox mediante su pluggin "Firebug" [29], y las herramientas que ya tiene por defecto Google Chrome ("Chrome developer tools") [30]. Inicialmente se optó por utilizar Firefox sencillamente por ser el navegador por defecto. Sin embargo la experincia no fue del todo satisfactoria y, tras un tiempo, se instaló y comenzó a utilizar Google Chrome para la depuración de los ficheros de la interfaz.

Back end

Django permite trabajar con una pequeña variedad de bases de datos: PostgreSQL, MySQL y SQLite. Se ha escogido SQLite3 porque es muy sencilla y eficaz, y el proyecto no exige una compleja base de datos con gran variedad de campos [31].

Despliegue

El despliegue de la aplicación podría realizarse sobre multitud de sitios web que proveen este servicio de manera sencilla, eficiente, y a veces incluso gratis con algunas restricciones. Sin embargo, un despliegue "manual" parece una mejor manera de finalizar el proyecto y además permitirá desplegarlo en algún servidor de la propia universidad.

Para llevar esto acabo, Django facilita varias maneras, pero la de usar WSGI es la estándar para Python y la más recomendada en los foros especializados [32]. En concreto, recomiendan el módulo de Apache mod_wsgi, así que es el que se empleará. Además, ya que Django no sirve archivos, recomiendan otro servidor para servirlos, por lo que también se empleará Nginx para esa tarea.

2.2. Estructura del proyecto

Al crear un proyecto en Django, automáticamente crea un directorio con ficheros y carpetas, muchos con contenido que no hace falta modificar, y otros que o están vacíos o requieren modificaciones. A continuación se detallan las características de los ficheros creados o editados durante el proyecto.

settings.py

Este fichero es el encargado de la configuración general del proyecto. A continuación se enumeran la configuración modificada o añadido que se ha realizado sobre el fichero por defecto creado al crearse el proyecto:

- Establecer la codificación UTF-8 para todo el proyecto. Por defecto, la codificación empleada es ASCII pero durante el desarrollo de la aplicación, como se explica más adelante en el apartado de Pruebas, aparecieron problemas al tratar con datos de la red, especialmente con caracteres especiales no cubiertos por ASCII.
- Establecer el tipo y localización de la base de datos a emplear, que en este caso es sqlite3.

- Modificar la “TIME_ZONE”, que por defecto es la estadounidense, por la local: “Europe/Madrid”.
- Modificar también el idioma, que por defecto es el inglés, por el español: “es”.
- Añadir la dirección de los directorios donde residen los componentes estáticos, tales como ficheros de hojas de estilo, funciones JavaScript, imágenes, etc. a la variable “STATIC_ROOT” y “STATIC_URL”.
- Añadir también la localización de las plantillas, agrupadas dentro del directorio “templates”.
- Establecer una redirección hacia la página principal tras la autenticación.
- Establecer un permiso a la tabla “Usuario” de la base de datos para enlazarla con la “User” por defecto que usa Django, como se explicará en el punto 3.1 de la base de datos.

urls.py

En este fichero se indican mediante una serie de reglas las vistas a ejecutar dependiendo de la URL recibida. Además, le puede pasar como parámetros parte de la dirección recibida. Las reglas están codificadas mediante expresiones regulares.

Puesto que la aplicación sigue una estructura REST, para mostrar la vista de un comentario el enlace sería /comentario/<<id comentario>>. Aquí también se especifican los enlaces de las funciones AJAX como el de borrar una noticia: /deleteNews/<<id noticia>>.

views.py

Este fichero contiene la programación de la aplicación. Recibe las llamadas de urls.py con al menos un parámetro: “request”, que contiene información de la petición y del usuario que lo ha solicitado, y se encarga de realizar todas las

tareas necesarias y de devolver un contenido html, normalmente por medio de las plantillas.

En este proyecto, todas las peticiones POST se envían a la página de inicio, por lo que van a una única función en views.py que las va separando en función de sus parámetros. En cuanto a las peticiones GET, por comodidad todas tienen direcciones distintas y tienen sus propias funciones.

El fichero, con casi 1000 líneas de código, se organiza de la siguiente manera:

1. Importación de librerías necesarias en el código.
2. Definición de constantes, como el tiempo límite para editar o el número de comentarios máximo que se debe mostrar.
3. Procesos secundarios que ayudan a la escalabilidad y la eficiencia por ser necesarios en varias partes del código.
4. Procesos que trabajan con la base de datos
5. Funciones llamadas por urls.py
    - Peticiones POST
    - Peticiones GET o directamente relacionadas con la interfaz, incluyendo al final las peticiones provenientes de jQuery.

admin.py

Este documento indica los elementos de la base de datos que pueden ser accesibles desde el menú del administrador. Esta es una herramienta facilitada por Django para manejar los datos, crear usuarios, configurar sus permisos, etc. accesible para el administrador (o cualquier usuario que haya recibido permisos). En este caso, los datos que se han considerado que deben ser accesibles gráficamente desde ese menú son Noticias, Comentarios y Usuarios. Gracias a esto se le pueden dar permisos a los usuarios pertenecientes al grupo de

los Profesores para crear nuevos usuarios mediante una interfaz gráfica, entre otras cosas.

Aún así, todos los elementos de la base de datos son accesibles y editables a través del terminal para quien tenga permisos.

models.py

Fichero encargado del diseño de la base de datos, que en este caso es SQLite3. Mediante clases crea tablas de datos con diferentes parámetros, que pueden ser de varios tipos de campo. Estos campos pueden ser una fecha, una url, un entero, etc, y también pueden ser relaciones a otras clases. Estas relaciones pueden ser 1:1 (OneToOneField), 1:N (ForeignKey) o M:N (ManyToManyField).

También permite crear tipos de formularios, pero dado que en este proyecto hay pocos formularios y son muy parecidos y sencillos se ha decidido prescindir de esta herramienta.

templates

Directorio en el cual se guardan las plantillas del proyecto, de forma que se pueden reutilizar. Estas plantillas contienen la extensión .html y, como se explica más adelante, pueden heredar unas de otras.

revistaSocial.sql3

Es la base de datos como tal, donde reside toda la información, pero no se trabaja con ella directamente sino a través de la aplicación y de las herramientas que facilita el entorno de desarrollo.

El resto de ficheros, salvo los correspondientes al despliegue en servidores que se explicarán más adelante, no se han editado, sino que han sido creados por defecto por el entorno de desarrollo y por lo tanto no se detalla su funcionalidad ni su contenido.

2.3. Funcionalidades

Los usuarios pueden realizar una serie de acciones en la aplicación web, y éstas deben estar bien definidas y diseñadas antes de implementarlas.

En la siguiente imagen se puede observar el diagrama general de casos de uso de la aplicación, donde se muestran estas funciones:

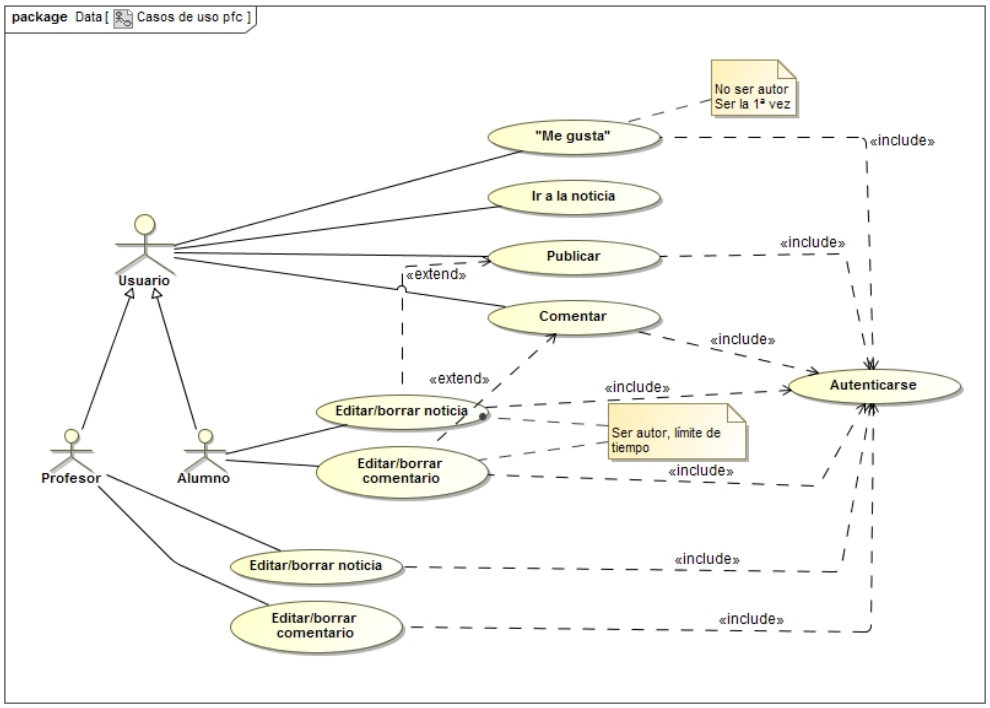


Figura 2.2: Diagrama general de casos de uso

Se puede ver que hay dos tipos de usuarios registrados: alumnos y profesores, que heredan de usuario. Por lo tanto, ambos pueden realizar las funciones a las que está asociado.

Aparte, ambos pueden editar y borrar comentarios y noticias, pero los alumnos tienen restricciones. Éstas vienen indicadas en el cuadro de texto que tienen asociado: ser los autores del comentario/noticia que van a borrar/editar, y hacerlo antes de un límite de tiempo. Además, como tienen que ser los autores, esas funciones se "extienden" de publicar/comentar. Esto es, publicar es re-

quisito previo para borrar/editar noticias, y comentar lo es para borrar/editar comentarios.

Por último, las líneas discontinuas "include" nos indican que todas las funciones, a excepción de la de ir a la noticia, requieren que el usuario esté autenticado en la aplicación.

Como se puede observar, no aparece la funcionalidad "Registarse". Puesto que el uso de la aplicación está orientado a un grupo limitado y conocido de usuarios, se ha considerado que el registro de usuarios debe llevarse a cabo por el administrador o un profesor desde el menú de administración que facilita Django.

Restricciones

Con el fin de evitar acciones que afecten significativamente a la aplicación, debe existir un tiempo límite para los alumnos para editar o borrar una noticia o un comentario (siendo ellos mismos los autores). De esta forma, se evita que una noticia antigua con comentarios y votos pueda borrarse por el alumno que la publicó. Otras aplicaciones web, como es el caso de Moodle, aplican exitosamente esta técnica en los foros.

Por otro lado, siempre podrán producirse errores por parte de un alumno y por un descuido no verlo antes del tiempo límite; para solucionar esto, y un posible uso malintencionado, los profesores no tienen restricciones, y pueden editar y borrar sin ser autores y sin límite de tiempo.

Para poder pinchar en el "Me gusta" de una noticia, las restricciones son obvias: no ser el autor de la noticia y no haberlo hecho ya.

Especificaciones

Las noticias y comentarios que se "borran" no lo hacen literalmente. Sólo dejan de mostrarse en la revista, y sólo pueden acceder a ellos los profesores. Esto se hace por dos motivos:

1. Para poder perseguir acciones malintencionadas como publicar noticias poco adecuadas o comentarios ofensivos para borrarlo antes del límite de tiempo.

2. Para hacer más robusto el historial de acciones.

A continuación, se muestra el diseño y objetivo de cada una de las funcionalidades. Para analizarlas y diseñarlas se pueden emplear multitud de diagramas cuyo grado de complejidad y precisión va en aumento. Se ha optado por mostrar directamente los diagramas de secuencia, que son los definitivos antes de la implementación y definen cada uno de los pasos para llevar a cabo una funcionalidad en orden secuencial y distinguiendo los distintos elementos involucrados. En este caso, serán el usuario (Usuario), la interfaz de la aplicación (Interfaz), que media en la comunicación entre el usuario y el gestor de la aplicación (Aplicación), y las distintas tablas de las bases de datos. Para simplificar, se han generalizado todas las tablas en una (Base de datos), sólo se ha representado el camino exitoso más completo posible (y no las múltiples alternativas), y se ha obviado la comprobación de las restricciones.

2.3.1. Publicar noticia

Proporciona una versión reducida de la información contenida en un enlace facilitado por un usuario, de manera que pueda mostrarse junto a otras noticias en la página principal.

La aplicación recibe el enlace a una página web; obtiene la información más relevante encontrada y las imágenes relacionadas con la noticia (si las hay); devuelve un formulario con los titulares e imágenes encontradas (más una imagen general de la página); el usuario elige una imagen, incluye un comentario y edita los titulares; se genera y guarda la noticia final y se muestra la revista al usuario.

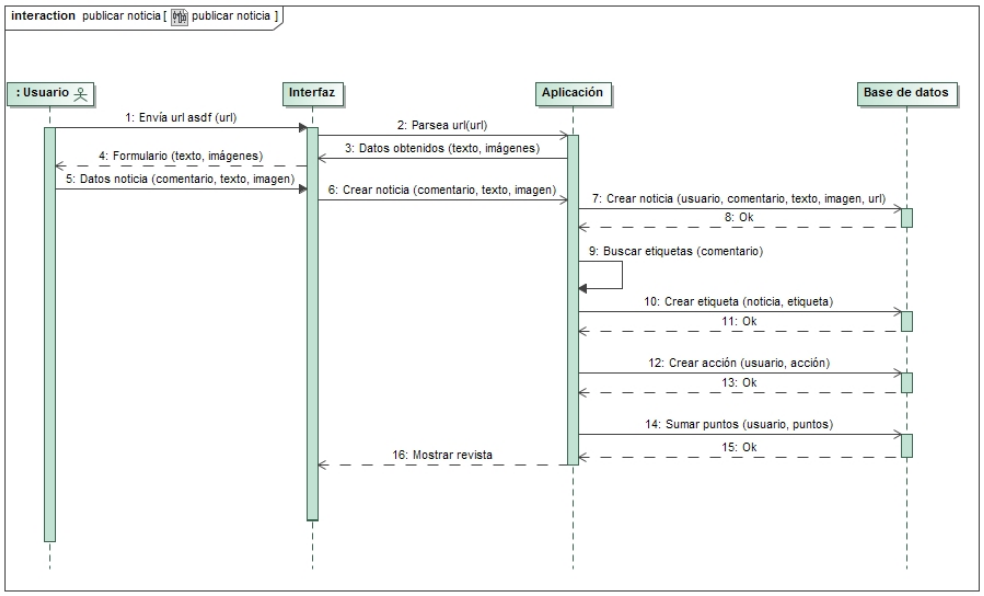


Figure 2.3: Diagrama de secuencia de “Publicar noticia”

2.3.2. Editar noticia

Permite modificar el comentario, los titulares y la imagen de la noticia.

La aplicación devuelve el mismo formulario de “Publicar”, pero con el campo del comentario incluyendo el antiguo comentario. El usuario vuelve a elegir imagen y puede editar titulares y comentario, y la aplicación guarda los nuevos valores de la noticia, volviendo a mostrar la revista.

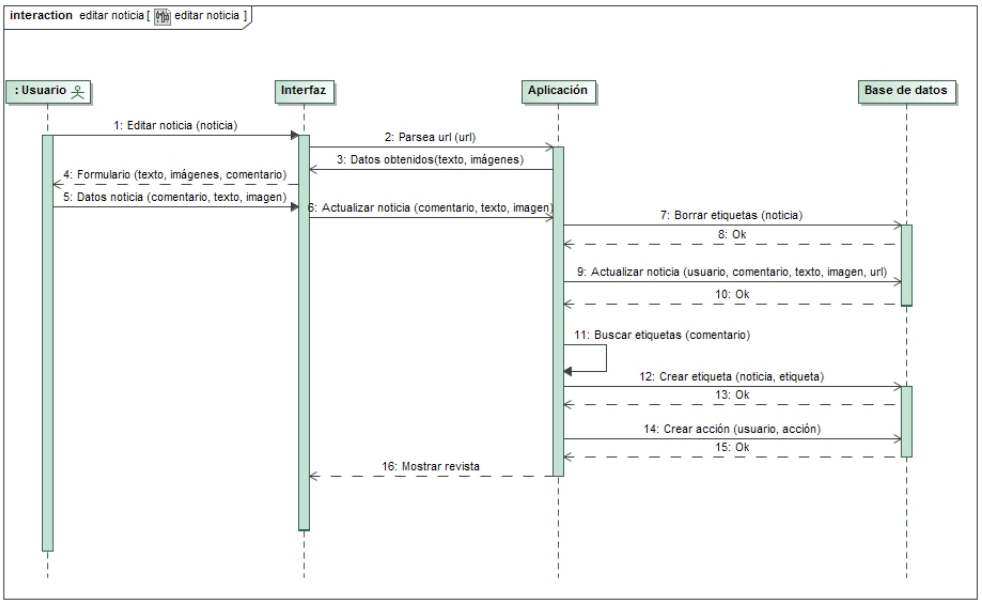


Figura 2.4: Diagrama de secuencia de “Editar noticia”

2.3.3. Borrar noticia

La aplicación borra tanto la noticia como los comentarios y etiquetas asociadas a ambos, y hace desaparecer de la página ese contenido.

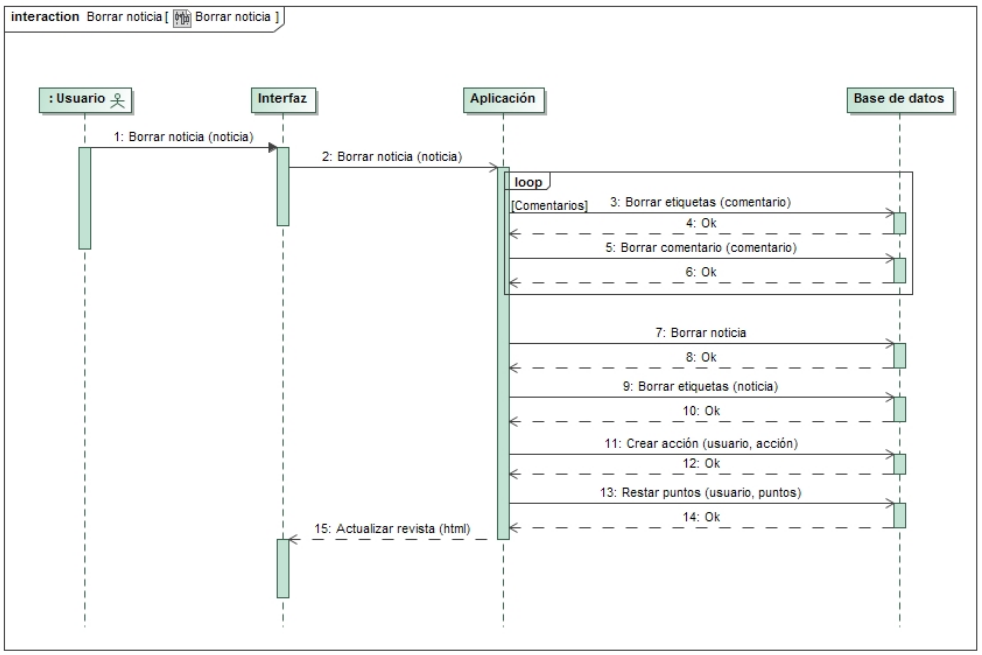


Figura 2.5: Diagrama de secuencia de “Borrar noticia”

2.3.4. Comentar

Añade, junto a la noticia, un texto breve que puede contener etiquetas.

La aplicación recibe el texto por un formulario, genera y guarda el comentario y muestra la revista al usuario.

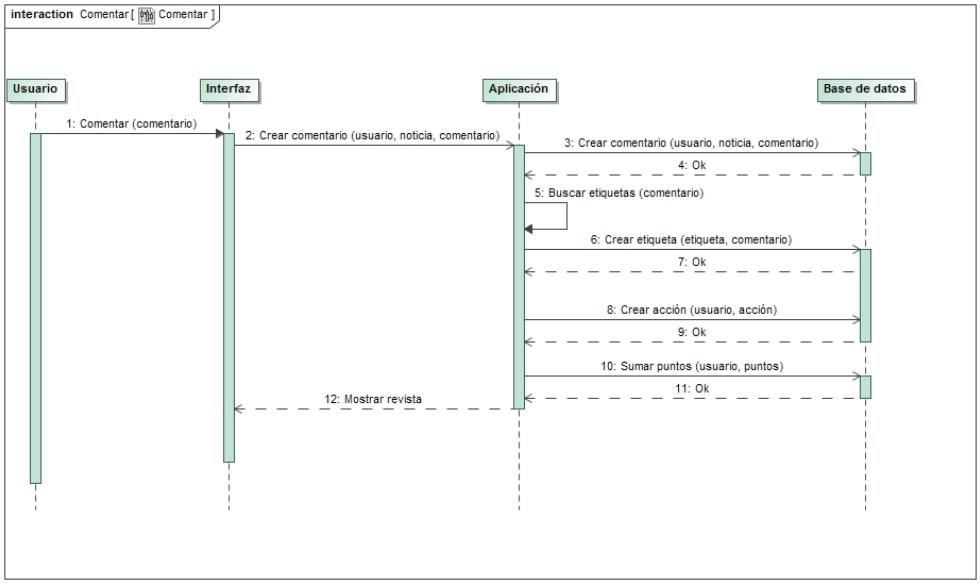


Figura 2.6: Diagrama de secuencia de “Comentar”

2.3.5. Editar comentario

Modifica el texto de un comentario.

Tras recibir la petición, la aplicación devuelve un formulario con un campo de texto editable y conteniendo el texto del comentario; al enviarlo se actualiza el comentario y se muestra la revista.

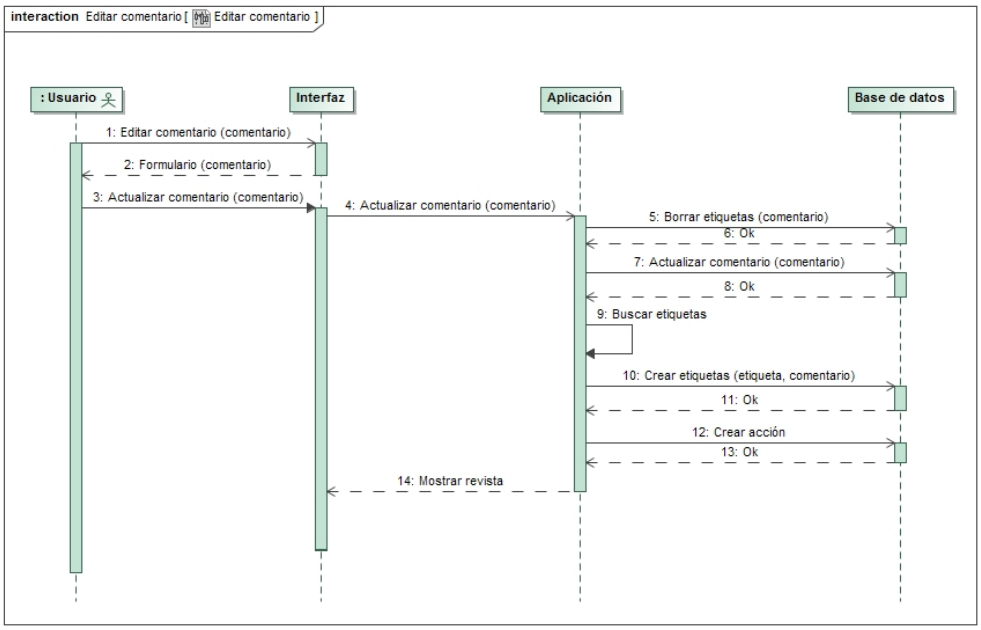


Figura 2.7: Diagrama de secuencia de "Editar comentario"

2.3.6. Borrar comentario

La aplicación borra el comentario con sus posibles etiquetas, y lo hace desaparecer de la página.

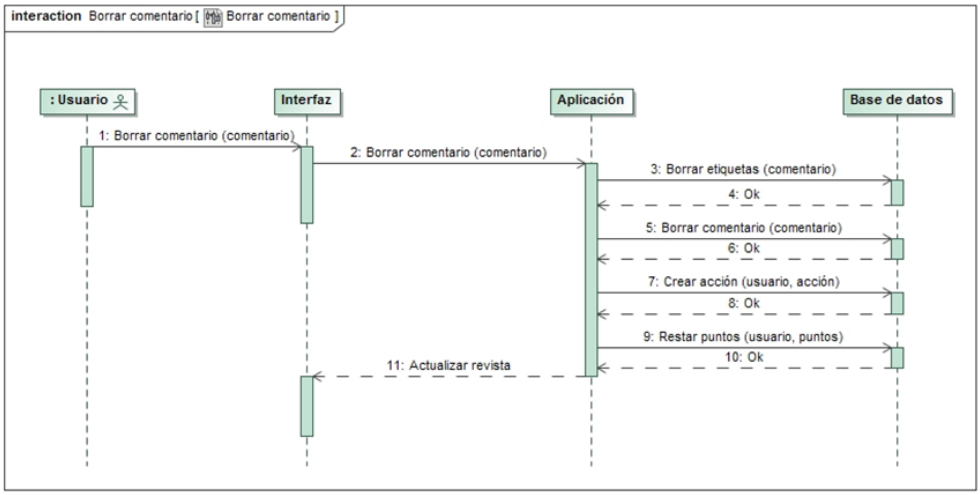


Figura 2.8: Diagrama de secuencia de "Borrar comentario"

2.3.7. "Me gusta"

Un usuario indica su opinión favorable sobre una noticia.

Se registra en la base de datos y se actualiza esa información en la revista.

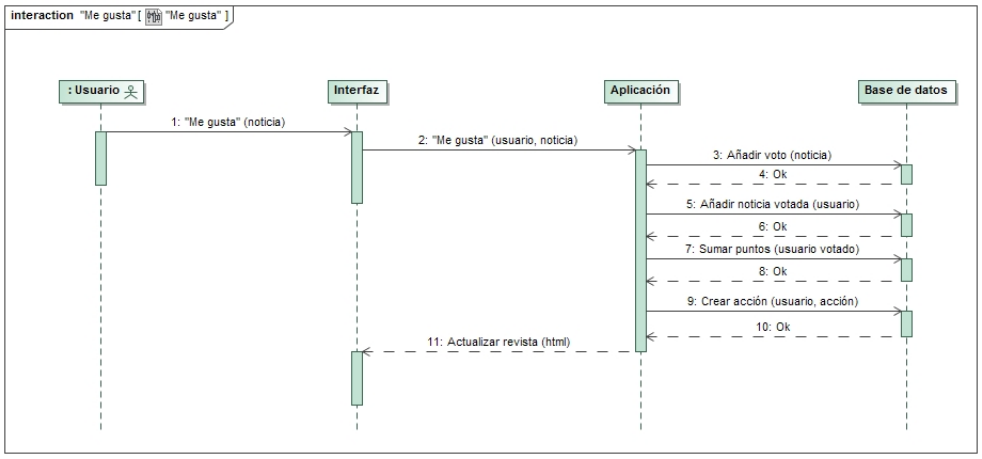


Figura 2.9: Diagrama de secuencia de "Me gusta"

2.3.8. Ir a la fuente de una noticia

Redirecciona a la página original de donde se ha obtenido la noticia.

Al pinchar en la imagen de una noticia, ésta actúa como un hiperenlace y devuelve la página de la fuente de donde se obtuvo esa noticia.

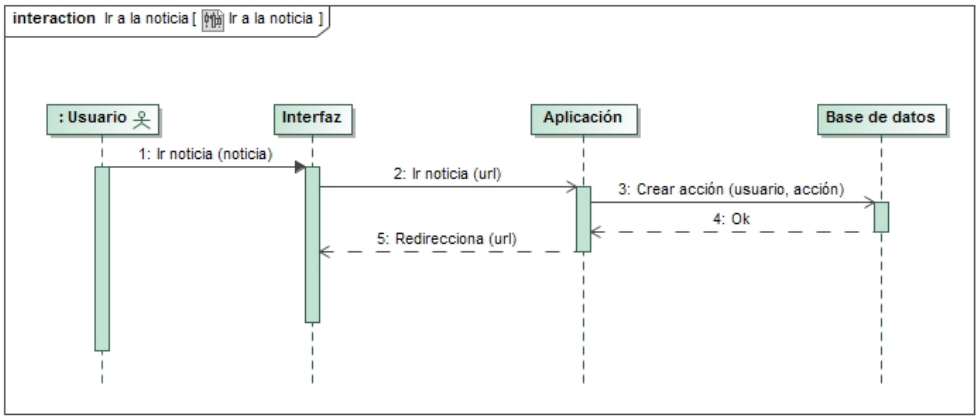


Figura 2.10: Diagrama de secuencia de "Ir a la noticia"

2.4. Aspecto general de la aplicación

A la hora de diseñar la interfaz de la aplicación se ha buscado la sencillez. Por eso, todas las vistas están divididas en tres partes:

2.4.1. Cabecera

En esta sección, se presenta en primer plano el nombre de la aplicación, y en un segundo plano una pequeña descripción y algunas opciones, como la de login/logout, poner el enlace de una noticia para publicarla o buscar noticias por alguna etiqueta. Además, el profesor verá también un enlace al listado de usuarios de la revista para acceder a sus historiales. Esta sección es independiente de la vista que se esté presentando.

2.4.2. Panel lateral

En la parte izquierda aparecen una sucesión de elementos que pueden ser útiles o interesantes para el usuario: una lista de los últimos comentarios, la puntuación del usuario, un ranking de puntuaciones y una lista de las etiquetas más utilizadas. Esta sección también es independiente de la vista.

2.4.3. Contenido

Por último, la parte fundamental de la interfaz, donde se muestra la información que corresponda a cada vista, que puede ser:

- Noticias

  Es la vista principal. Las noticias se muestran en dos columnas, y cada una de ellas contiene el comentario del autor, luego la imagen con los titulares a su derecha, y debajo otra información más los comentarios que pueda haber.

- Formulario para editar una noticia

  Es un formulario para editar y publicar una noticia cuyo enlace URL se ha indicado previamente. En este formulario aparecen dos campos

de texto, uno para el comentario personal del publicador y otro con los titulares de la notica. Además, se muestran las imágenes recogidas para que el usuario elija una.

- Listado usuarios

  Sólo visible por los profesores, muestra una tabla con el nombre y puntuación de cada uno de los usuarios registrados en la aplicación.

- Historial usuario

  También visible sólo para los profesores, muestra una lista cronológica de las acciones llevadas a cabo por el usuario que se haya elegido.

Capítulo 3

Implementación

"Nuestra recompensa se encuentra en el esfuerzo y no en el resultado. Un esfuerzo total es una victoria completa"

Mahatma Gandhi

Una vez definidos los aspectos fundamentales de la aplicación, se comienza con su programación. Para realizar este paso de manera eficiente existen varios métodos. En este caso se ha empleado una metodología similar a la conocida como "desarrollo ágil". Esto es, se ha dividido la implementación en pasos, cada uno de los cuales se ha desarrollado como un proyecto en sí mismo, de forma que sólo tras probarlo y depurarlo se ha pasado al siguiente, teniendo lo anterior correctamente implementado.

3.1. Base de datos

La base de datos utilizada es SQLite3, que es relacional. Django facilita y simplifica su uso intermediando a través del archivo models.py donde se definen las tablas de datos.

A la hora de diseñar las tablas que debían contener los datos de las etiquetas, se pensó en primer lugar una única tabla con un índice extra que indicara

el origen de la etiqueta. Sin embargo, esto requeriría una comprobación de cada una de las entradas de la tabla en la base de datos, lo que ralentizaría y en algunos casos complicaría la búsqueda. Además, conllevaba que los dos índices principales de la tabla podrían ser iguales. Esto es, que el texto de la etiqueta y el identificador de la noticia y del comentario al que correspondería cada entrada coincidiesen. Esto no generaría un error porque Django genera como clave principal un identificador único para cada tabla, pero en conclusión se vió que dos tablas serían más eficientes que una sola y otorgarían mayor simplicidad. Finalmente, el diagrama de clases de la base de datos (excluyendo las tablas auxiliares que emplea Django) se ven en la siguiente figura:

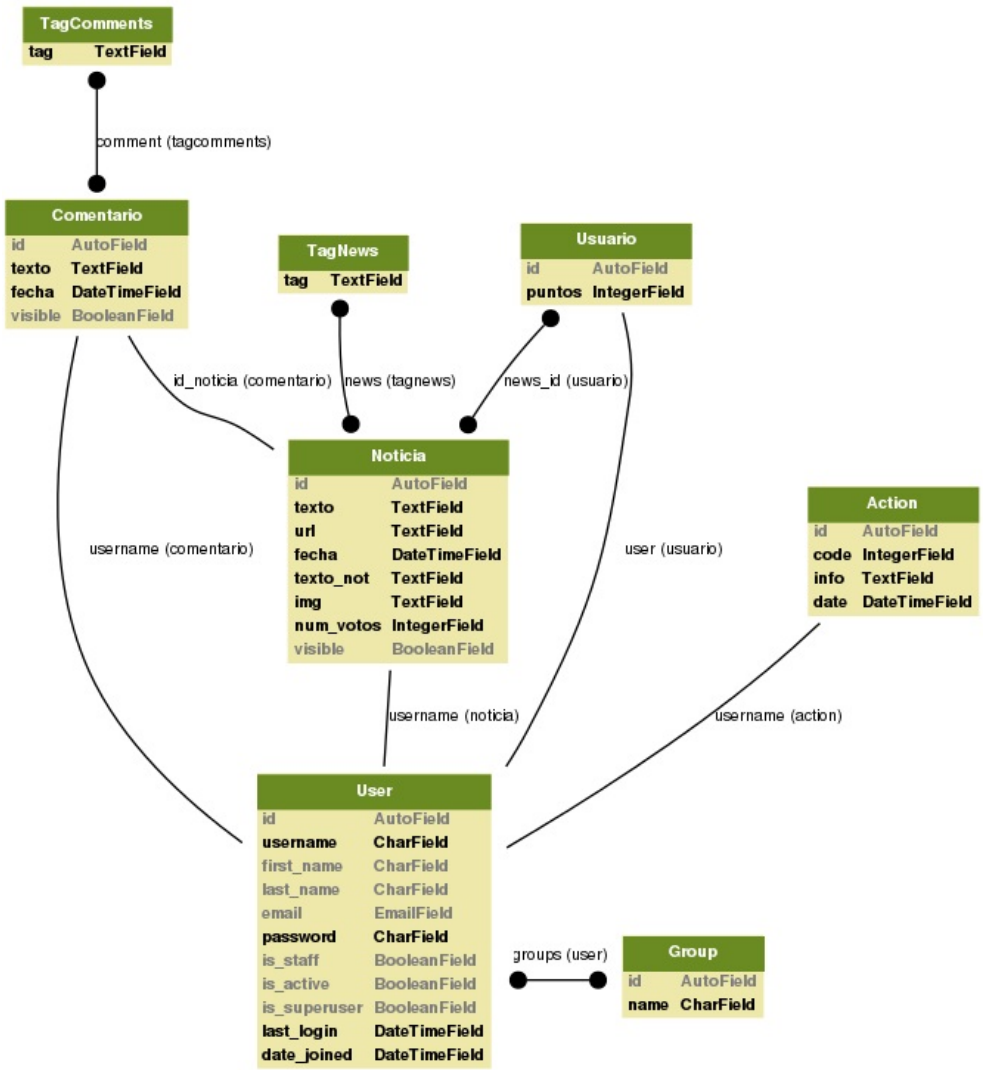


Figura 3.1: Diagrama de clases de la base de datos

Las tablas empleadas son las siguientes:

- Usuario:

  Afortunadamente, Django ofrece por defecto un tipo User muy útil y básico en esta aplicación. Sin embargo, él solo no permite llevar la cuenta de los puntos de cada usuario ni la lista de noticias que le gustan. Por eso se ha creado el tipo Usuario, directamente relacionado con User (a través de la relación "OneToOneField"), con un campo entero para los puntos y un campo "ManyToManyField" para las noticias votadas.

  Además, gracias a una herramienta de Django, se ha configurado este modelo para crear automáticamente una nueva instancia cada vez que se crea un usuario.

- Noticia:

  Esta tabla se encarga de guardar la información relacionada con la noticia: relación con el autor, comentario, URL de la fuente, fecha de publicación, titulares, enlace a la imagen, puntos recibidos y si está o no borrada.

- Comentario:

  En este caso la tabla guarda un comentario, y está relacionada con el usuario que lo ha escrito y con la noticia en la que está publicado. Además, contiene el texto del comentario, la fecha de publicación y si está o no borrado.

- TagNews:

  Contiene las etiquetas escritas en las noticias, por lo que contiene el texto de la etiqueta y la relación con las noticias en las que aparece.

- TagComments:

Igual que el anterior pero relacionado con los comentarios en lugar de las noticias.

- Action:

  Esta tabla se encarga de guardar el historial de acciones de los usuarios. Está referenciado al usuario que ha realizado la acción y contiene el código (que determina el tipo de acción realizado), el identificador de la noticia o el comentario sobre el que se ha hecho la acción y la fecha en la que se llevó a cabo.

Además de estas tablas, también hay que destacar el uso de otra propiedad por defecto que permite Django que es la agrupación de usuarios en grupos. Para este proyecto se han descrito dos grupos: Alumnos y Profesores.

Por último, destacar que los campos de las fechas están configurados para tomar su valor automáticamente al crearse la tabla, pero no para actualizarse cuando se editan. Este valor es el de la fecha y hora de ese momento, y está en función de la hora española porque así se ha configurado en settings.py.

3.2. Acciones

Como ya se ha adelantado, uno de los objetivos de la aplicación es la de facilitar al profesor la calificación y seguimiento del alumno. Para ello, cada vez que un usuario realiza alguna de las funcionalidades, se crea una entrada en la base de datos para registrar esa acción.

La tabla de las acciones está compuesta por cuatro parámetros:

Username: referencia al tipo usuario implícito en Django, y será el usuario que ha realizado la acción.

Code: de tipo entero, del 1 al 8 indica la funcionalidad que se ha llevado a cabo.

Info: aunque es de tipo texto, es el identificador de la noticia o el comentario sobre el cual se ha realizado la acción.

Date: indica la fecha y la hora en la que se realizó la acción.

3.3. Puntos

El sistema de puntuación tiene como objetivo incentivar y premiar el uso de la aplicación.

Existen tres formas de obtener puntos: al publicar (5 puntos), al comentar (2 puntos) y al recibir un "Me gusta" (1 punto) en una noticia propia. De esta forma, la aportación más importante como es la de publicar una noticia es notablemente más premiada que la de comentar. La acción de visitar la fuente de una noticia y la de dar a "Me gusta" no están premiadas por no asegurar un esfuerzo por parte del usuario.

La puntuación de cada usuario se recoge en el parámetro "puntos" de la tabla Usuario.

3.4. Interfaz gráfica

Anteriormente se ha explicado el diseño de la interfaz con la división de bloques y su estructura. Ahora, en este apartado, se explica la parte de la implementación:

3.4.1. Plantillas HTML

Para desarrollar las distintas páginas HTML necesarias se ha usado un conjunto de plantillas que heredan de una base, y todas están recogidas en el paquete de Django templates. Esto significa que se pueden definir bloques de contenido en la plantilla con un nombre, y las plantillas que hereden de ella sólo tienen que introducir el contenido que quieran en esos bloques. A continuación se detallan las características y los objetivos de cada plantilla:

base.html

Plantilla "padre" de todas las demás, se encarga de mostrar la cabecera y el panel lateral, ya que son iguales para cualquier vista, y deja definido un bloque de contenido que será el que rellenen las distintas plantillas hijas. Para mostrar la información de la cabecera y el panel necesita los siguientes parámetros:

- Lista de los últimos comentarios.
- Lista de los usuarios con más puntos.
- Booleano informando de si el usuario es profesor.
- Puntos del usuario
- Principales etiquetas: lista de las etiquetas más usadas a partir de un mínimo.

Además, establece los parámetros de la etiqueta <head> necesarios para todas las plantillas. Esto es, especifica la codificación (se trabaja en todo el proyecto con UTF-8), el logotipo y nombre que debe aparecer en la pestaña del navegador y se encarga de enlazar desde la cabecera las plantillas a los documentos externos necesarios, que son la hoja de estilo, la librería jQuery y las funciones de AJAX.

Destacar que aquí se ha podido introducir algunas de las propiedades exclusivas de HTML5. Por ejemplo, se ha especificado que el tipo de dato para introducir los enlaces de las noticias sea URL, de forma que detecta la validez del formulario sin enviarlo, y también se ha configurado ese formulario para que no tenga autocompletar y el de buscar etiquetas para que sí lo tenga.

revista.html

Esta plantilla se compone de dos bloques iterativos idénticos para mostrar las noticias de la columna derecha (noticias pares) y las de la columna izquierda (noticias impares).

Se aplica para varias vistas:

- Mostrar la revista general
- Mostrar un comentario específico. Se puede acceder desde el enlace de "últimos comentarios" o desde un enlace en el historial de acciones (un comentario borrado, creado o editado) de un usuario.
- Mostrar las noticias relacionadas con una etiqueta. Al buscar una etiqueta a través del formulario de la cabecera o a través de un enlace de las etiquetas más usadas, la aplicación muestra una revista con las noticias que tengan esa etiqueta en su comentario propio (escrito por el autor al publicar) o en cualquiera de los comentarios añadidos.

  Los parámetros que recibe se explican a continuación:

  - Dos listas de tuplas, formadas por una noticia y la lista de sus comentarios no borrados. Se han utilizado tuplas porque la lista de los comentarios debe estar relacionada con su noticia, y además las plantillas de Django permiten sacar los datos que recibe en forma de tupla.
  - Número de comentarios: indica el número máximo de comentarios que deben mostrarse por cada noticia. Si se supera, se muestra adicionalmente un enlace con AJAX para mostrarlos todos. Tiene dos posibles valores: un valor pequeño definido por una constante, o uno grande para que se muestren todos.
  - Noticias votadas: listado de noticias votadas por el usuario al que se le muestra la vista. De esta forma se muestra el icono de "Me gusta" si no es así (y no es el autor).
  - Comentarios especiales: lista de comentarios que deben ir resaltados. Esto se utiliza porque cuando se quiere ver un comentario específico (ya sea desde la lista de los más recientes en el panel o desde el historial de acciones) se muestra junto con la noticia y

los demás comentarios, y de esta forma la hoja de estilo le da un color y una fuente especial para destacar. También se emplea en la búsqueda de una etiqueta, donde se resaltan los comntarios con esa etiqueta.

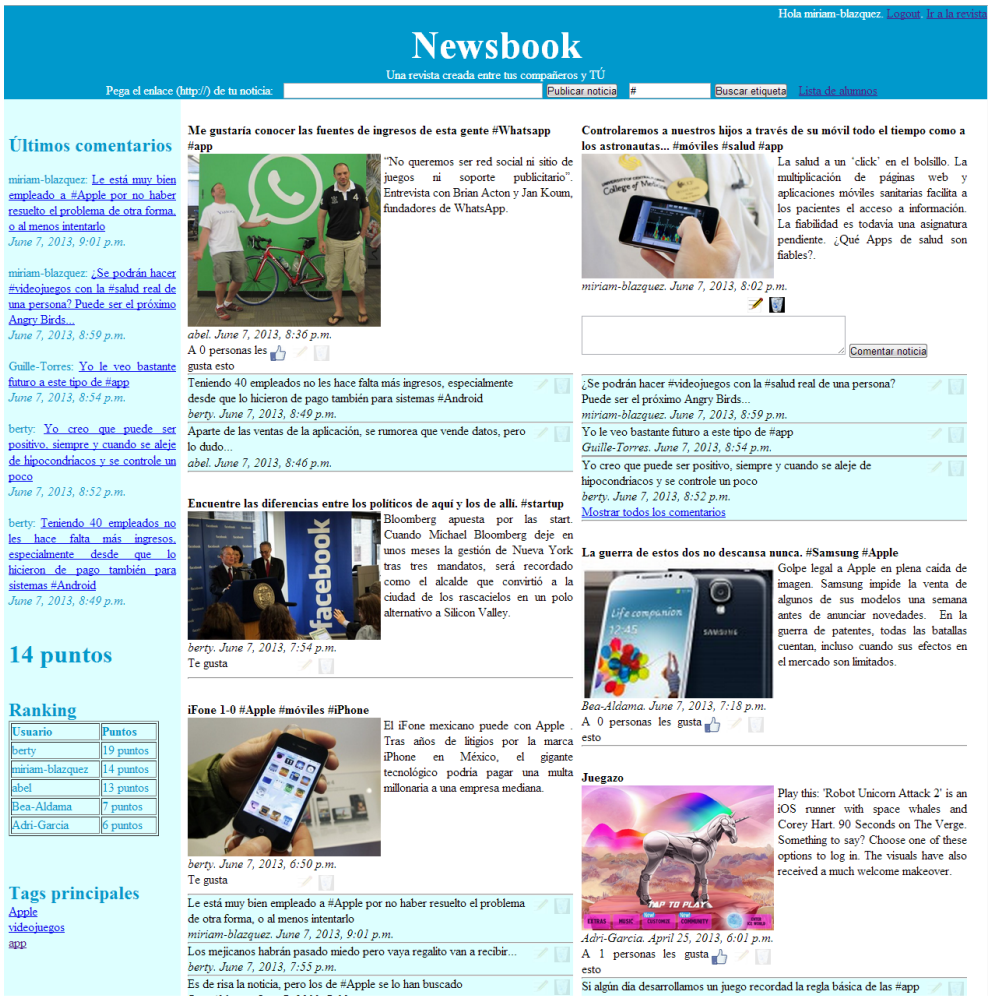


Figura 3.2: Aspecto de la revista en la página principal

publicacion.html

Esta plantilla se usa para la publicación o edición de una noticia, y lo único que aporta es un formulario como ya se ha explicado donde elegir y editar la información de la noticia. Los parámetros que recibe son:

- Identificador de la noticia: sólo necesario cuando se trata de editar una noticia, para no guardarla como nueva sino actualizarla.
- Titulares de la noticia.
- Lista de imágenes capturadas. En el formulario se añade a estas imágenes una miniatura de la página web de la noticia, ya que a veces las noticias no tienen imágenes. Estas miniaturas se obtienen mediante la web "thumbshots.com" de forma gratuita, pero con la condición de incluir un enlace a su web, que en este caso está al final del panel lateral.
- Comentario del autor.
- URL de la fuente de la noticia.

Figura 3.3: Aspecto del formulario para editar una noticia

studentslist.html

Sólo disponible para los profesores, recibe la lista de usuarios para mostrar una tabla con todos los usuarios registrados (y enlazado a sus respectivos historiales) con sus puntuaciones.

Usuarios de la revista

Pincha en un usuario para ver su historial

Usuario	Puntos
berty	19 puntos
miriam-blazquez	15 puntos
abel	13 puntos
Bea-Aldama	8 puntos
Adri-Garcia	6 puntos
Luis-Poker	5 puntos
Isra-Diaz	5 puntos
Juli-Alegre	5 puntos
Elena-Calvo	2 puntos
Fran-Moises	2 puntos
Bernard	1 puntos
Fernando-Iglesias	0 puntos
Javi-Perez	0 puntos
alumno	0 puntos
profesor	0 puntos
Guille-Torres	0 puntos
Luis-Alonso	0 puntos

Figura 3.4: Aspecto de la lista de usuarios

actions.html

También accesible únicamente para profesores desde la lista de usuarios, recibe la lista de acciones de un usuario para mostrar la fecha y una breve descripción (donde el comentario o noticia sobre el que tiene lugar la acción es un enlace para verlo aparte.)

- Me gusta en Noticia. Fecha: April 18, 2013, 3:13 p.m.
- Noticia visitada. Fecha: April 18, 2013, 8:51 p.m.
- Noticia visitada. Fecha: April 18, 2013, 8:51 p.m.
- Noticia visitada. Fecha: April 18, 2013, 8:51 p.m.
- Noticia creada. Fecha: April 18, 2013, 8:51 p.m.
- Noticia visitada. Fecha: April 21, 2013, 2:13 p.m.
- Me gusta en Noticia. Fecha: April 21, 2013, 2:13 p.m.
- Noticia visitada. Fecha: April 22, 2013, 10:08 p.m.
- Noticia visitada. Fecha: April 22, 2013, 10:08 p.m.
- Noticia visitada. Fecha: April 22, 2013, 10:38 p.m.
- Noticia visitada. Fecha: April 22, 2013, 10:38 p.m.
- Noticia visitada. Fecha: April 23, 2013, 1:02 p.m.
- Me gusta en Noticia. Fecha: April 24, 2013, 8:51 p.m.
- Noticia creada. Fecha: April 24, 2013, 8:59 p.m.
- Noticia visitada. Fecha: April 24, 2013, 8:59 p.m.
- Noticia visitada. Fecha: April 24, 2013, 9:09 p.m.
- Noticia visitada. Fecha: April 24, 2013, 9:10 p.m.
- Comentario creado. Fecha: April 24, 2013, 9:13 p.m.
- Noticia visitada. Fecha: April 24, 2013, 9:15 p.m.
- Noticia visitada. Fecha: April 24, 2013, 9:15 p.m.
- Comentario creado. Fecha: April 24, 2013, 9:16 p.m.
- Noticia creada. Fecha: June 7, 2013, 8:02 p.m.
- Comentario editado. Fecha: June 7, 2013, 8:03 p.m.
- Noticia editada. Fecha: June 7, 2013, 8:04 p.m.
- Me gusta en Noticia. Fecha: June 7, 2013, 8:04 p.m.
- Comentario creado. Fecha: June 7, 2013, 8:05 p.m.
- Me gusta en Noticia. Fecha: June 7, 2013, 8:06 p.m.
- Comentario creado. Fecha: June 7, 2013, 8:59 p.m.
- Comentario creado. Fecha: June 7, 2013, 9:01 p.m.
- Noticia editada. Fecha: June 7, 2013, 9:02 p.m.

Figura 3.5: Aspecto del historial de acciones de un usario

error.html

En ocasiones pueden producirse errores previstos en la aplicación. Para estos casos hace falta mostrar un sencillo mensaje informando de que algo no ha salido como se esperaba e informando por qué, aparte de los casos de errores externos 400 y 500. Por ello, para todos los errores controlados dentro del

programa se pasa un mensaje de error a esta plantilla para que se lo muestre al usuario.

3.4.2. Funciones de AJAX

Como se ha dicho anteriormente, la aplicación hace uso de la librería jQuery para implementar una serie de funciones AJAX con el objetivo de hacer la aplicación sencilla, cómoda y eficiente. Para poder enlazar y distinguir los elementos de la página con funciones AJAX y además reemplazar algunas partes del código, se sirve de las etiquetas presentes en el código HTML y de sus atributos "class" e "id".

En la mayoría de los casos el evento de JavaScript se produce al hacer click sobre un enlace, y lo que hace AJAX es cancelar el transcurso normal del enlace intermediando, de forma que extrae del elemento la dirección a la cual hay que realizar la petición y la hace el mismo, y cuando recibe la respuesta sustituye la parte de código que se le pida por esa respuesta.

A continuación se enumeran los diferentes elementos donde se han implementado estas funciones:

- Icono de "Me gusta"

  Al hacer click, se envía una petición GET al servidor a través del enlace. Sin embargo, en lugar de comportarse como un enlace normal, la función jQuery rechaza la redirección y acepta introducir código HTML en el lugar de una pequeña parte de la noticia que afecta al texto "A [x] personas les gusta esta noticia" y al icono de "Me gusta". De esta forma la aplicación, aparte de realizar las acciones correspondientes, devuelve un código HTML con el texto "A [x+1] personas os gusta esto".

  Si se produce algún error se devuelve un mensaje resaltado explicando el motivo.

- Enlace de "Mostrar más comentarios"

Como ya se ha explicado, el número de comentarios que aparece debajo de una noticia es limitado. Para ver los comentarios anteriores existe un enlace con el texto “Mostrar más comentarios” que devuelve todos los comentarios que tiene la noticia. Esto no es trivial, ya que surgen dos cuestiones.

La primera es que se pueden solicitar los comentarios que faltan y añadirlos o solicitar toda la lista y quitar los que ya se muestran. Lógicamente solicitar sólo los necesarios es más eficiente, pero genera importantes complicaciones si se ha eliminado un comentario. Por ello, se ha optado por solicitar todos de nuevo.

La segunda cuestión, es conseguir mostrar ese fragmento de la página igual que antes. Como se está empleando una función AJAX, el nuevo código procede del gestor de la aplicación, el fichero views.py, sin utilizar plantilla alguna. Las plantillas simplemente identifican que el tipo de dato es fecha (DateTimeField) y lo muestran de una manera por defecto. Para imitar esto, ha sido necesario identificar el formato y orden por defecto en que muestra Django cada uno de los elementos de la fecha (día, mes, hora, uso de pm/am, uso de mayúsculas, etc.) y especificarlo desde el fichero views.py. Desgraciadamente, el idioma por defecto de la librería utilizada para mostrar la fecha (strftime) es el inglés, ya que no tiene acceso al fichero settings.py, por lo que sólo queda parecido.

- Iconos de “Borrar” y “Editar”

  Al colocar el ratón sobre la noticia o sobre un comentario, los iconos correspondientes de “Borrar” y “Editar” se vuelven completamente opacos. Esto es, visibles perfectamente. Y cuando el ratón sale de la zona de esa noticia o de ese comentario, la opacidad varía sustancialmente para dejar los iconos casi transparentes. De esta forma, los iconos sólo aparecen cuando pueden ser necesarios y dejan el aspecto de la revista más limpio. Para modificar la opacidad de esos elementos se recurre a su estilo, ya que es una propiedad de CSS3.

- Icono de “Borrar”

  Al hacer click, se envía una petición como ya se ha explicado, pero en este caso no se devuelve nada, de forma que el comentario o la noticia desaparece.

  Si se produce algún error se sustituye el comentario o noticia por un mensaje resaltado explicando el motivo.

- Icono de “Editar” comentario

  Al hacer click, se envía la petición y lo que devuelve es un formulario con un campo de texto editable y con el valor del comentario.

- Formulario de comentar noticia

  Con el fin de mostrar sólo lo necesario y de obtener un contenido limpio, el formulario para comentar una noticia no se muestra a simple vista para cada una de las noticias. Pero con el fin de facilitar y simplificar las aportaciones este formulario debe ser fácilmente accesible sin enlaces a otras páginas. Por ello, la solución adoptada es que se muestre únicamente cuando el ratón se sitúa sobre la noticia, y vuelva a desaparecer cuando el ratón salga de ella.

  Destacar que en este caso no se envía una petición para recibir el formulario. El formulario existe desde que se carga la página pero no se muestra. La función AJAX sólo hace que aparezca y desaparezca.

  Las funciones se asocian a elementos en el momento en el que la página ha termiando de cargarse. Cuando se introduce contenido en la página de forma dinámica surge el problema de enlazarlo con una función. Esto sucede para los comentarios que se añaden al hacer click en “Mostrar más comentarios”. Para poder asociar estos elementos a las funciones AJAX que cambian su opacidad o ejecutan su funcionalidad se utiliza la función on() sobre un elemento superior (en principio con un identificador único para distinguirlo de otros) que exista desde que se cargó la página . A

partir de ahí se puede descender hasta el elemento nuevo ya que se le busca sólo cuando el evento se dispara.

3.4.3. Hoja de estilo CSS3

A través de la hoja de estilo se define el aspecto de la página y, al igual que las funciones AJAX, se sirven de las etiquetas presentes en el código para dar estilo a las diferentes partes por separado.

La hoja de estilo en este proyecto se describe en un fichero externo enlazado a la página HTML. De esta forma, el contenido y el estilo quedan separados, aunque existen un par de excepciones (fragmentos de código HTML enviados desde views.py en las funciones AJAX) en los que el estilo se define junto con el código HTML. Las normas de estilo más importantes son las siguientes:

Tema

En aplicaciones web importantes como redes sociales los colores suelen ser blanco y otro más, quedando bastante sencillas y amigables. Por otra parte, los diarios online apenas aportan un color a excepción del de la publicidad, y el aspecto no es amigable.

Por esta razón los colores deben ser blanco de fondo y otro más, y tras varias pruebas el color elegido es el azul. De esta manera, y gracias a las recomendaciones recibidas durante las pruebas, la cabecera es de un azul intenso con las letras en blanco, el panel azul claro con letras en azul más oscuro y el fondo de los comentarios también en azul claro. Todo lo demás queda en un sencillo negro sobre fondo blanco.

Texto especial

Existen tres pequeñas excepciones al estilo general del texto para destacarlo: el comentario de la noticia, que es en negrita; las fechas, que son en cursiva; los comentarios “resaltados” de los que ya se ha hablado, que tienen un fondo distinto y además están en negrita.

Para los dos primeros casos se podría usar sencillamente las etiquetas de strong () y emphasize (), pero inicialmente su estilo no era ese y por lo tanto en un futuro se puede desear incluir un cambio en el tipo de letra o en el color. Por eso, para facilitar una posible edición, su estilo reside en el archivo .css.

Distribución cabecera

Como es lógico, la cabecera se sitúa en la parte superior. Esto se consigue asignándole una anchura del 100 % y una altura de 120 píxeles.

La distribución de los elementos en el interior es bastante sencilla: el saludo inicial o el formulario de login (dependiendo de si ya está autenticado o no) se indica en la esquina superior flotando a la derecha; la imagen con el título y el logotipo en el centro; los formularios, desplegados en línea para no colocarse uno encima del otro y justificados justo debajo del título.

Distribución panel

Para situar el panel como una barra vertical situada en el lateral izquierdo, el CSS le adjudica una anchura del 20 % y la propiedad de flotar a la izquierda. Esto significa que el siguiente contenido intentará situarse a su derecha. El texto queda con márgenes y justificado.

Distribución contenido

El contenido en general debe ocupar el resto del espacio, por lo que se le asigna un margen superior y lateral correspondiente a la cabecera y al panel lateral.

Distribución noticias

Para la mayoría de las vistas no hay más influencia del css, pero sí la hay para la vista de la revista y es algo compleja. De fuera hacia adentro la distribución se realiza de la siguiente manera:

- Columnas

El contenido se divide en dos columnas idénticas que muestran una sucesión de noticias con sus comentarios. Esta división no se puede realizar mediante el atributo "float" usado para el panel, como se explicará más adelante, por lo que se emplea posicionamiento absoluto; se establece un margen lateral correspondiendo a su posición y una anchura del 40 %, de forma que se ocupa ya todo el ancho de la página (20 % para el panel lateral y 40 % para cada columna de noticias).

- Noticias

  Cada columna está formada por una sucesión de noticias con sus comentarios, todos ellos con un margen lateral para dejar espacio libre a los lados. Para conseguir que los titulares de las noticias aparezcan al lado de la imagen, ésta usa la propiedad "float" a la izquierda.

  Tras los titulares se usa la propiedad "clear" para que el texto quede debajo de la foto y no a su lado. Las columnas usan posicionamiento absoluto por esta razón, ya que la propiedad "clear" anularía el uso de "float" tanto para la imagen como para la columna entera.

  Además, para conseguir que los iconos de borrar, editar y "Me gusta" queden a la derecha, se vuelve a hacer uso de la propiedad float pero al lado contrario, la derecha, para que el texto se coloque a su izquierda.

- Comentarios

  De nuevo, el primer elemento que aparece (el formulario del comentario) debe tener "clear: left". Además, como se ha visto en las funciones de JavaScript, el formulario no debe aparecer a no ser que el ratón esté sobre la noticia, por lo que también contiene inicialmente "display:none".

  Puesto que el formulario a priori está oculto, los comentarios también deben tener la propiedad "clear". Y de nuevo para colocar sus iconos a la derecha se vuelve a utilizar el "float" en los iconos. Por último, se les adjudica el mismo color de fondo que al panel lateral.

Una consecuencia de la distribución explicada es la imposibilidad de tener un "pie" de página. Repasando las decisiones tomadas sobre la distribución, no se ha podido emplear posicionamiento relativo en las columnas debido a la necesidad de emplear la propiedad "clear" en estas. Consecuentemente, al utilizar posicionamiento absoluto en las columnas, no se puede seguir la posición al final de una columna desde un elemento externo, por lo que habría que colocarlo dentro. Sin embargo, hay dos columnas y cualquiera de ellas puede ser la más larga, lo cual es impredecible ya que la altura de una columna varía con la altura de las fotos y la cantidad de los comentarios principalmente.

3.5. Seguridad

Gran parte de la seguridad de la aplicación viene proporcionada por Django de forma automática. En el desarrollo de este proyecto se ha necesitado añadir elementos de seguridad de tres maneras:

3.5.1. Seguridad en formularios descritos desde views.py

Como medida de seguridad, Django exige ciertos requisitos a la hora de trabajar con formularios. Uno de ellos es la inclusión de un "csrf_token" que se puede incluir fácilmente al trabajar con sus plantillas. Sin embargo, existe una situación en el proyecto en la cual el formulario debe crearse desde views.py. Este caso es el de la función de AJAX empleada para editar comentarios, ya que la aplicación responde dinámicamente con un fragmento de código en el que se incluye el formulario para editar el comentario.

Django ofrece una complicada solución consistente en tratar con cookies, y la opción de no exigir el elemento "csrf-token". Dado que esta funcionalidad no es vital, y que siguen existiendo otros elementos de seguridad al tratar con formularios (como la autenticación del usuario), se ha optado por lo segundo. Para ello, y tal y como se especifica en la guía oficial online, se elimina la exigencia de "csrf-tokens" a través del comando "@csrf_exempt" para todas las peticiones POST (ya que todas las peticiones POST están dirigidas a la

misma función). A continuación, se separa la petición del tipo "editar comentario" del resto, y a éstas se las vuelve a exigir el elemento de seguridad a través del comando "@csrf_protect".

3.5.2. Comprobación de autenticación

Con el objetivo de fortalecer la seguridad y dar robustez, se realizan comprobaciones de autenticación frecuentemente, incluso en casos en los que a priori parecen innecesarios:

Peticiones POST

Las peticiones POST, a diferencia de las del tipo GET, deben realizarse sólo por usuarios registrados. Por lo tanto, se establece una comprobación de autenticación al recibir cualquier petición POST y se ignora si el solicitante no se ha identificado.

Peticiones GET

Algunas vistas son accesibles para cualquier usuario, ya esté registrado o no, como la vista de noticias. Sin embargo, existen otras que tienen restricciones por el tipo de usuario. Concretamente, sólo los profesores pueden ver el historial de un alumno, el listado de usuarios o un comentario o noticia ya borrado.

Por un lado, los enlaces para ver estas vistas restringidas a profesores se presentan sólo a estos, pero por otro es muy sencillo deducir la dirección de esas vistas y acceder directamente a ellas desde la barra de direcciones. Por esta razón, también se realiza una comprobación de la identidad de los usuarios al solicitar estas vistas.

Restricciones

Como se definió en la figura 2.2, algunas funcionalidades tienen restricciones. Estas funcionalidades son borrar y editar comentarios y noticias, y las restric-

ciones son, en caso de ser un usuario del tipo Alumno, el límite de tiempo y ser autor del comentario o noticia en cuestión.

Django proporciona algunas herramientas para tratar con este tipo de restricciones, de forma que se pueden dar permisos para tratar con ciertas tablas de la base de datos, acceder a la parte administrativa, etc. Sin embargo, puesto que la restricción de límite de tiempo no está entre las facilitadas por Django, se ha optado por hacerlo directamente en el código, comprobando el tipo de usuario y el tiempo transcurrido desde la publicación del comentario o noticia.

3.6. Algoritmo de parseo de noticias.

En este apartado se analiza y explica una parte crítica de la aplicación como es la obtención de una versión representativa y reducida de una noticia.

Por una parte, la web presenta numerosas maneras de presentar la información, y por otra, la publicidad invade las páginas informativas, de forma que se trata de una tarea compleja y que de ninguna manera puede ser óptima. Por ello, se presenta la información recogida al usuario donde éste debe validar la información recogida si es correcta o editarla si no lo es.

A continuación se muestran los pasos que realiza para solventar los obstáculos que se presentan.

3.6.1. Obtención del código HTML:

A partir de la URL facilitada por el usuario, debemos obtener el código HTML de la página en esa dirección para poder analizarla y extraer los datos que queramos.

Para empezar, se utiliza la librería urllib2 para establecer la conexión con la URL y extraer el código HTML. Sin embargo, a veces este código está encriptado, generalmente con gzip. Por ello, se comprueba si tiene codificación y se descodifica si es así. Por último, para recorrer el árbol y extraer más fácilmente los datos se utiliza la librería Beautifoulsoup.

3.6.2. Recopilación de titulares:

Como ya se ha explicado, no existe una manera estandarizada o común, ni siquiera entre los grandes periódicos, para presentar su información. Por lo tanto, hay que intentar recoger lo mejor posible las formas más generalizadas sin que se perjudiquen notablemente entre ellas.

Esto se consigue mediante una sucesión de filtros, cada uno de los cuales intenta extraer información útil. Aquí se explican por encima los distintos puntos en los que la aplicación intenta obtener los titulares:

1. Etiqueta <title> en <head>

   A menudo existe esta etiqueta y suele contener el titular principal de la noticia. El único problema que presenta es que a menudo va acompañado del nombre del dominio del periódico. Para intentar filtrar esa información buscamos dos caracteres que suelen separar el titular del dominio, que son "|" y "-", y si se encuentra cualquiera de ellos extrae solo la primera parte. Algunas veces también contiene la fecha de la noticia pero sin ningún caracter distintivo, por lo que el algoritmo no lo elimina.

2. Etiquetas <h1> y <h2>

   Como es lógico, los titulares casi siempre se muestran con un tamaño de fuente mayor que el texto de la noticia y en negrita. Esto se puede conseguir utilizando CSS, pero suelen utilizarse las etiquetas predefinidas para esto que son <h1>...<h6>. En este caso sólo se buscan las dos etiquetas mayores <h1> y <h2>. Además, dado que a menudo los titulares se repiten, a lo largo del código se comprueba para cada frase que ésta no haya sido capturada anteriormente.

3. Etiquetas <p>

   Si después de recoger los supuestos titulares aún queda mucho texto libre, se intenta recoger el comienzo del cuerpo de la noticia.

Este texto, casi siempre, se presenta dentro de la etiqueta <p>, aunque a menudo con interrupciones de estilo para resaltar algunas partes que complican la captura. Además, este texto puede venir dentro del parámetro "string" o dentro de "contents", por lo que se analizan primero ambas posibilidades.

4. Texto de las imágenes:

   A menudo, las imágenes de las noticias incluyen en el atributo "alt" una descripción de la noticia, por lo que conviene incluirlo en los titulares que se muestran. Por ello, a la hora de capturar imágenes también se extrae (si hay) el texto que incluye.

   El contenido de los titulares es de tamaño limitado, por lo que los filtros se colocan en orden prioritario y si se sobrepasa un límite se deja de buscar más contenido.

Un problema a tener en cuenta en todos los casos es que con este método pueden añadirse frases publicitarias. Para intentar evitarlo, y digo intentar porque sin un estudio semántico o muy complejo no se puede conocer si el texto es publicitario o no, se filtran las frases cortas. Esto es, basándome en que la publicidad rara vez contiene largos textos, se cogen las frases que superen un mínimo de palabras. Otro problema poco común son las frases medio vacías que a veces aparecen. Se componen de un nombre (del editor normalmente), de un lugar, o a veces de un titular normal, seguido de muchos espacios. Por lo tanto también se detectan y descartan estas frases que pasan el filtro de la longitud.

Por lo tanto, se obtiene texto de esas cuatro formas, pero podría haber más. Concretamente, se ha estudiado obtener también titulares que están enumerados mediante la etiqueta <ul>, ya que a veces se presentan de esa forma lo titulares secundarios. Desgraciadamente, resultó ser inviable debido a que la publicidad a menudo también lo usa y era muy difícil distinguir los titulares de la publicidad.

3.6.3. Recopilación de imágenes:

Inicialmente, como se ha descrito en la introducción, la intención era la de recopilar también vídeos si se encontraban. Sin embargo, las fuentes de los vídeos son diversas y la forma en que cada fuente los introduce son muy variados. Por lo tanto, dado que no son muy frecuentes y que supone una tarea difícilmente abarcable, se ha decidido recoger únicamente las imágenes.

Dicho esto, recopilar las imágenes presentes en una página HTML es sencillo, pues vienen definidas por la etiqueta <img>. Sin embargo, por lo general hay decenas en una sola página, y por lo general querremos sólo una, dos o ninguna de las que hay. Además, al igual que sucede con el texto, existen varias maneras de presentarlo y es difícil abarcar todas.

Filtros

El objetivo de este algoritmo es el de rechazar la mayor cantidad de imágenes no deseadas, asegurándose de no rechazar ninguna deseada. Esto lo consigue aceptando las imágenes que cumplen alguna de las siguientes propiedades:

- Tener un tamaño mínimo:

  Las fotos de la noticia son generalmente grandes, tanto en ancho como en alto; las publicitarias suelen ser pequeñas, al menos en una de las dimensiones; el resto de imágenes en verdad no lo son. Su objetivo es el de recopilar información sobre los visitantes de la página gracias a la petición que se hace a la página externa en cuestión para obtener esa supuesta imagen, por lo que suelen tener un tamaño nulo (0 píxeles de ancho y 0 de alto).

  Por ello, intenta detectar el tamaño de las imágenes mediante los atributos “height” y “width”, y si supera un mínimo se recogen.

- Incluir el atributo style:

  El filtro anterior no siempre funciona, pues no siempre se indica el tamaño de la imagen o no a partir de esos atributos. Cuando se da el

caso, a veces esa información va incluida dentro del atributo style que define el estilo. Y aunque no sea así, sólo imágenes importantes contendrían información específica del estilo en el código HTML por lo que se recogen.

- Pertenecer a tumblr.com:

  Un caso particular es el de imágenes provenientes del dominio "tumblr.com", aunque no son muy comunes. Estas imágenes escapan a los filtros anteriores por lo que también se recogen.

Extracción del enlace de la imagen:

Una vez recogidas las etiquetas <img> que parecen ser las deseadas, se debe extraer el enlace en el cual residen. Este enlace suele encontrarse en el atributo "src", pero no es obligatorio.

En HTML 5 existen los atributos "data-****" para guardar cualquier tipo de información que se crea relevante. El nombre que se use puede ser cualquiera, pero en un par de periódicos nacionales encontrados y por coherencia, el atributo que interesa se llama "data-src". En estos casos, el atributo "src" existe y contiene un enlace pero no es el deseado, por lo que detectar que es una excepción es complicado. La solución por la que se ha optado es buscar primero el atributo "data-src", y en caso de no encontrarlo recoger el enlace del atributo "src".

3.7. Problemas

Durante la implementación surgieron muchos problemas que no pudieron preverse durante le diseño. Algunos ya se han ido mencionando, y del resto aquí se exponen los más importantes.

- Distribución noticias

  La idea inicial para mostrar las noticias es agruparlas en "cajas" incluyendo sus comentarios y gracias a la propiedad "float" que se colo-

caran progresivamente de arriba a abajo. Sin embargo, la variable altura de las noticias generaba huecos en blanco y cambios de posición al expandirse el formulario de comentar. Esto se podía solucionar no mostrando comentarios, pero se consideró un requisito fundamental para motivar la participación.

Por lo tanto, se optó por crear dos columnas diferenciadas, y por lo tanto dividir las noticias previamente en dos grupos. Esto tiene el inconveniente de que, al mostrar la mitad de las noticias en una columna y la otra mitad en otra, una columna puede ser notablemente más larga que la otra. Desgraciadamente, al emplear dentro de las noticias las propiedades "float" a ambos lados y "clear", no se pudo implementar una distribución tal que la columna de la derecha flotara a la derecha y la columna del medio se quedara en el hueco entre el panel y la otra columna. Hubo que recurrir a un posicionamiento absoluto como ya se ha dicho, y renunciar a la posibilidad de tener un "pie" de página.

Además, pasar la información a las plantillas se tuvo que recurrir a una solución nada trivial: crear dos listas, una para cada columna, conteniendo una sucesión de tuplas formadas por una noticia y una lista de sus comentarios no borrados.

- Error en base de datos cuyo mensaje indicaba que estaba en views.py

  Durante la depuración de una funcionalidad en la implementación surgió un error cuyo mensaje decía así: "la función X no es visible desde views.py". Obviamente centré mi atención en los ficheros views.py y urls.py, por lo que tardé bastante en descubrir que el error residía en el fichero models.py, donde había declarado uno de los parámetros en la base de datos como "UrlField". No logré entender cuál era exactamente el error en esa declaración pero al sustituirlo por un campo de texto se arregló.

- Codificación

Por defecto, la codificación empleada por Django es ASCII, pero permite trabajar con otras codificaciones y pasar de una a otra. El problema de esta codificación es que admite un limitado número de caracteres (128), de forma que provoca errores cuando recibe caracteres poco comunes. Estos errores saltaban al guardar texto de las noticias que contenían estos caracteres, por lo que se cambiaba su codificación a UTF-8. Sin embargo, luego surgieron más problemas, como que al mostrar una página correspondiente a la fuente de una noticia, dependiendo de su codificación a veces provocaba problemas también, o que a veces se añadía un espacio al campo de texto donde se guardaba la dirección URL de una noticia. Al final, se optó por cambiar directamente la codificación empleada para todo el proyecto en settings.py, y especificar ésta en la cabecera de las plantillas.

Parte III

Pruebas y resultados

Capítulo 4

Despliegue

"La fortuna favorece a los valientes"

Publio Virgilio Marón

Una vez finalizada la implementación de la aplicación se ha procedido a realizar unas pruebas para la detección de errores y recomendaciones sobre el diseño y funcionalidad de la aplicación.

Para poder realizar estas pruebas con mayor comodidad y para dar una solución final del proyecto se ha desplegado la aplicación en un servidor. El despliegue de la aplicación se ha realizado sobre una máquina virtual en la red de la Universidad Rey Juan Carlos.

Una alternativa era usar el comando "runserver" de Django sobre un ordenador corriente tal y como se hace para la depuración en local; sin embargo, esta aproximación tiene los inconvenientes, entre otros, de no ofrecer una solución definitiva ni segura y de no ofrecer concurrencia.

Por esta razón, se ha decidido realizar el despliegue sobre un servidor. A la hora de elegirlo se ha optado por un servidor Nginx en primer plano, más un servidor Apache detrás conectado a Django a través del módulo mod_wsgi. Esta disposición se hace de acuerdo a un tutorial [33] elegido entre muchos

otros, entre ellos el oficial ofrecido por Django [32]. Inicialmente se trató de hacer la configuración siguiendo el oficial, pero muy pronto, durante la configuración de wsgi, surgieron problemas que la documentación oficial [34] no supo solucionarme. Por esa razón se estudiaron otros tutoriales hasta encontrar el más completo y con una configuración acorde a las necesidades del proyecto.

Además, ha sido necesario instalar algunos programas en la máquina virtual, a la cual no se ha tenido acceso directo.

A continuación se detalla la configuración de los cuatro elementos involucrados:

4.1. Configuración de la máquina virtual

Para llevar a cabo el despliegue, se ha enviado el proyecto a la cuenta de la máquina virtual asignada para el despliegue mediante el protocolo scp. Puesto que la máquina virtual estaba limpia de programas, y no había posibilidad de uso con interfaz gráfica, todo el despliegue se ha realizado a través de la shell de Linux mediante un acceso ssh a un usuario de la máquina virtual con permisos de root.

Para el correcto funcionamiento de la aplicación obviamente se ha tenido que instalar Django y Python, y para la edición de ficheros, sobre todo los de configuración de los servidores, se ha empleado el editor vim (que también se ha tenido que instalar).

4.2. Configuración de Django

La configuración de los ficheros para realizar el despliegue es bastante sencillo. Únicamente es necesario configurar el fichero settings.py en un par de puntos.

- Modificar las variables “DEBUG” y “TEMPLATE_DEBUG” a “False”:

  Esto significa que ya no se está trabajando con el proyecto en modo de depuración. De esta manera, aumenta la seguridad del proyecto y en

caso de error no se muestra el informe detallado como antes, sino uno más sencillo y concreto con el código del error.

- Ajustar las direcciones de los archivos:

  En este caso el proyecto se ha movido a una máquina virtual distinta, por lo que se han ajustado las direcciones de los directorios del contenido estático, de las plantillas, y se ha sustituido la dirección relativa de la base de datos por una completa.

4.3. Configuración de servidor Apache

Este servidor será el encargado de conectarse a Django y devolver las peticiones que le lleguen del servidor Nginx. Para guardar los ficheros de configuración y de log se han creado dos carpetas dentro del projecto Django: /apache y /logs.

Para llevar a cabo la configuración, lo primero ha sido instalar tanto el módulo mod_wsgi como el servidor Apache, y posteriormente cargar el módulo sobre el servidor.

A continuación, tras comprobar que funcionaba correctamente (muestra un mensaje al acceder a su dirección IP), se ha configurado el puerto que va a utilizar, que será el 8080 (el puerto 80 se lo dejaremos a Nginx como se explicará más adelante). Después, se crea un fichero de configuración del sitio donde se define, para la dirección y puerto que le hemos adjudicado, datos varios necesarios para apache como direcciones de los ficheros /logs y /apache, el protocolo a seguir (aceptar todas las peticiones), etc.

Por último, se crea un archivo .wsgi en la carpeta de apache dentro del proyecto donde se configura la comunicación con Django, principalmente las direcciones de algunos directorios como el de Python y el del proyecto.

4.4. Configuración de servidor Nginx

Una vez configurado apache se procede a configurar Nginx, que se encargará de recibir todas las peticiones, pasárselas al servidor Apache y servir los ficheros estáticos.

Para empezar, se instala el servidor y se comprueba un fichero de configuración ya creado por defecto. En principio no se tocó la configuración ni debería hacer falta, pero como se explica más adelante, tras unos problemas de concurrencia se resolvió que había que eliminar la especificación del número de procesos que tenía configurado, que era 1.

A continuación, al igual que se hizo con el servidor Apache, se crea un fichero de configuración del sitio donde se definen los datos necesarios para su funcionamiento. En este caso la dirección es la misma pero el puerto es el 80 (puerto HTML) para recibir todas las peticiones. Se define la dirección del directorio /logs creado en el proyecto para guardar los .log del servidor, se configura la redirección al servidor Apache de las peticiones "/" y se sirven los contenidos estáticos cuando las peticiones sean "/static".

Por último, se copia y pega este fichero a un directorio paralelo de Nginx y se crea otro llamado proxy.conf donde se pega una configuración por defecto con datos como timeouts o tamaños límite.

Capítulo 5

Pruebas

"Hay quienes se consideran perfectos, pero es sólo porque exigen menos de sí mismos"

Hermann Hesse

Para que estas pruebas fueran concluyentes se debían realizar sobre un grupo amplio de personas, el equivalente a una clase pequeña, y durante un periodo de al menos una semana.

Desgraciadamente, no se ha podido reunir a un grupo real de estudiantes a los que podría estar dirigida la aplicación, y la mejor alternativa ha sido reunir a un grupo formado por quince personas, de perfil mayoritariamente joven y con experiencia en las redes sociales. Muchos, además, tenían conocimientos de programación web. A cinco de ellos se les proporcionó una cuenta de profesor, y a los otros diez se les dió cuentas de alumnos, de forma que todas las posibilidades de la aplicación fueran probadas.

Además, se probó el funcionamiento de la interfaz desde varios tipos de dispositivos con distintos navegadores y sistemas operativos.

- *Smartphones* y *tablets*

Se comprobó que el funcionamiento era correcto, tanto en dispositivos Android como iOS (no se dispuso de más terminales con otros sistemas operativos móviles), pese a no haber ratón. Las funciones AJAX que se disparaban con eventos dependientes del ratón (mouseEnter y mouseLeave) pasan a activarse cuando se pulsa con el dedo sobre la superficie elegida. Inicialmente no es tan sencillo e intuitivo como con un ratón pero una vez que se conoce el funcionamiento es igual de cómodo, y además no existen alternativas mejores.

- Navegadores

  Se comprobó que el funcionamiento era correcto en los cuatro navegadores más usados 1.2: Google Chrome, Mozilla Firefox, Internet Explorer y Safari. Salvo en Internet Explorer el funcionamiento era perfecto en esos navegadores en versiones relativamente modernas.

Las pruebas se realizaron activamente durante cuatro semanas, hasta que se solucionaron todos los errores que fueron apareciendo. Los más destacables se detallan a continuación.

5.1. Errores encontrados más importantes

- AnonymousUser:

  Al intentar realizar algunas de las opciones sin estar autenticado (y que Django interpreta como un usuario AnonymousUser) saltaban errores puesto que no se había tenido en cuenta en la implementación. Para solucionarlo simplemente se añadió ese caso en las restricciones y plantillas a las que afectaba.

- Servidores bloqueados

  Al configurar y probar los servidores parecía funcionar todo bien. Sin embargo, muchos usuarios dijeron que la aplicación les mostraba un error de “timeout”, lo que significaba que la aplicación se quedaba atascada, y

además tardaba varios minutos en volver a estar operativa. Tras muchas comprobaciones, se dedujo que la mayoría de las veces que esto sucedía había más de un usuario utilizando la aplicación, por lo que se debía a un problema de concurrencia. La configuración del servidor Nginx de la máquina virtual mostraba que existía sólo un núcleo con el que trabajar. Se modificó para que intentara trabajar con dos pero no mejoró, por lo que se dedujo que sólo tenía un procesador. Finalmente, por consejos a casos similares en un foro se eliminó directamente esa línea y el comportamiento mejoró, aunque todavía presenta errores esporádicos de este tipo. Además, las únicas veces que se produce este problema es al parsear noticias externas, nunca con cualquier otra petición como mostrar la fuente de la noticia, y nunca en las pruebas en local.

- La hora no se actualizaba

  Se observó que durante horas, todo lo que se publicaba lo hacía mostrando la misma hora. Esto se debía a un detalle en la función que devolvía la fecha y la hora, ya que recogía la hora guardada en una caché en vez de hacer una petición nueva.

- Internet Explorer

  A los usuarios que utilizaban una versión antigua de internet explorer no les funcionaban las funciones de AJAX. La incompatibilidad de este navegador se solucionó mediante dos modificaciones. Para empezar, se cambió la versión de la librería jQuery a una más antigua, que soportaba las versiones antiguas de Internet Explorer. Aún así, en las últimas versiones las imágenes aparecían con bordes azules, lo que se solucionó especificando la ausencia de bordes en todas las imágenes desde la hoja de estilo.

- Puntuación del “Me gusta”

  Al hacer “click” sobre el icono de “Me gusta” de una noticia, el punto correspondiente no se sumaba al autor de la noticia. Este error se debía a

que se guardaba la instacia de la tabla User, pero los puntos se guardan en la tabla Usuario asociada a User, por lo que no se actualizaba el valor. Para guardarlo correctamente hay que usar ".get_profile()" sobre el usuario.

5.2. Mejoras y consejos por feedbacks

- Diseño demasiado cargado: algunos usuarios opinaron que el contenido de la revista estaba demasiado cargado y recomendaron más espacio entre los elementos, por lo que se aumentó el margen, se redujo el tamaño máximo de texto de comentarios y noticias y se dificultó la captura de excesivos titulares al publicar noticias.

- Inicio de sesión dentro de la página: en un principio la cabecera mostraba un enlace para autenticarse si aún no lo estabas, que te llevaba a una página de login. Esto se cambió al método actual, en el que en vez de un enlace se muestra un formulario para iniciar sesión sin salir de la página de inicio.

- Mostrar fecha de los últimos comentarios.

- Utilizar justificado de texto para los titulares y el texto del panel lateral.

- Se recomendó añadir una vista de noticias favoritas (aquellas que "gustan" al usuario) y otra vista de noticias ordenadas por popularidad, pero opino que la aplicación perdería sencillez y no favorecerían los objetivos.

- Incluir un logotipo: se diseñó un logotipo acorde a la aplicación y se incluyó en la portada.

5.3. Publicación

Una vez terminada la aplicación, y como se dijo en los objetivos 1.3, se procede a publicar la aplicación. La licencia elegido es la GPL versión 3 de GNU ya que no se busca restricción alguna en la liberación de este software.

Esta licencia, en resumen, permite a cualquier usuario utilizar, modificar y distribuir el programa (tanto modificado como original) libremente para el uso que desee. Sin embargo, como autor tengo ciertos derechos, como que deberá hacerlo siempre bajo la misma licencia y sin negar mi autoría.

Para llevarlo a cabo, se ha indicado en la página el copyright, en cada fichero modificado se ha incluido un resumen de la licencia propuesto por GNU [36] y el copyright ("Copyright 2013 Alberto Fernández Zazo) y, por último, se ha incluido un fichero COPYING donde se han incluido todos los detalles de la licencia.

Capítulo 6

Conclusiones

"La verdadera sabiduría está en reconocer la propia ignorancia"

Socrates

6.1. Evaluación de los objetivos

Una vez finalizado el proyecto, se puede concluir que la aplicación obtenida es efectivamente una herramienta docente amigable, sencilla y moderna, por lo que en general se han cumplido los objetivos.

No todas las ideas principales se han podido llevar a cabo, algunas por excesiva complejidad como la de incluir vídeos, otras por cambios de orientación del proyecto naturales durante su desarrollo, como la de incluir una vista con las noticias más populares. Desgraciadamente, el parseo de noticias no es óptimo, pero sin duda alguna se puede afirmar que tiene un comportamiento razonablemente bueno y que rara vez requiere de edición por parte del usuario. El programa ha pasado a ser software libre como se propuso, y podría ser empleado por cualquiera.

En general, las pruebas han sido un éxito. Los usuarios han estado satisfechos con las funcionalidades de la aplicación una vez se han resuelto los errores

iniciales, que han sido graves. Estas pruebas han servido para asegurar un funcionamiento general correcto; pero sólo en un escenario real, con alumnos y profesores reales, podrá comprobarse el éxito del proyecto. Esto es, si atrae a los alumnos y convence a los profesores. Opino que la aplicación está preparada para ser probada en un escenario real, aunque recomendaría una revisión adicional de su seguridad y su robustez, y además habría que mejorar el despliegue actual pues ha demostrado no ser lo suficientemente robusto en pruebas reales con varios usuarios.

6.2. Líneas futuras

Con vistas al futuro, será necesario llevar a cabo dos tareas fundamentales:

- Mejoras

  Por una parte, la mejora constante de la aplicación mediante la inclusión de herramientas y elementos gráficos que hagan la aplicación más agradable y divertida.

  Estas mejoras pueden ser, por ejemplo, medallas en el ranking de usuario, la inclusión de gráficas y estadísticas que ayuden al profesor a evaluar la aplicación y el seguimiento de los alumnos, una herramienta del profesor para proponer fuentes de noticias, etc.

- Mantenimiento

  Por otra parte, debido al continuo desarrollo de la red y de HTML 5, con el tiempo cambia rápidamente la forma de hacer webs, y cada fuente nueva de noticias que se encuentre puede ser incompatible con el algoritmo usado actualmente. Por esa razón, necesitará constante renovación para ajustarse.

6.3. Opinión personal

A título personal, ha sido un placer desarrollar este proyecto desde el primer día, planificando, diseñando y siguiendo todos y cada uno de los pasos necesarios en el desarrollo de este software. La idea de esta aplicación procede de mi tutor, pero he tenido total libertad para desarrollarla a mi gusto, y aunque fue complicado al principio, pronto me hice una idea en la cabeza y he disfrutado dándole forma y llevándola a cabo.

He aprendido mucho del desarrollo web y de recopilar la información necesaria para realizar un diseño acertado que evite problemas innecesarios en la implementación, aunque también he aprendido que durante la implementación surgen imprevistos que obligan a cambiar el diseño. Habría deseado incluir más propiedades incorporadas en HTML5, pero la mayoría no tenía mucho sentido en la aplicación o no eran soportadas por algunos navegadores.

Sólo han existido dos tareas que me han dado quebraderos de cabeza y que han requerido mucho más tiempo del que deberían. La primera es la inclusión de comentarios junto a las noticias, cuyas dificultades ya he detallado. Afortunadamente, esto nunca fue desesperante sino un reto, ya que estaba convencido de que era una parte fundamental para lograr incentivar al usuario. La segunda, sin embargo, ha sido la única parte del proyecto que me ha desesperado: el despliegue. Esto se debe a que por mala suerte y (sobre todo) por torpeza con Lunix, durante dos semanas no hice otra cosa que intentar averiguar la causa de su mal funcionamiento sin demasiado éxito.

Bibliografía

[1] Sistema operativo GNU. ¿Qué es software libre? [En línea]. Disponible en http://www.gnu.org/philosophy/free-sw.es.html

[2] Elkenstein.org. ¿Qué es REST?. [En línea]. Disponible en http://rest.elkstein.org/2008/02/what-is-rest.html

[3] Entorno de desarrollo Netbeans. Características. [En línea]. https://netbeans.org/features/index.html

[4] Entorno de desarrollo Eclipse. ¿Qué es eclipse? [En línea]. Disponible en http://www.eclipse.org/home/newcomers.php

[5] Plataforma de desarrollo Django. [En línea]. Disponible en https://www.djangoproject.com/

[6] Plataforma de desarrollo Ruby on Rails. Iniciación. [En línea]. Disponible en http://guides.rubyonrails.org/getting_started.html

[7] Ruby. Acerca de Ruby. [En línea]. Disponible en http://www.ruby-lang.org/es/about/

[8] Pyhton. Acerca de Python. [En línea]. Disponible en http://www.python.org/about/

[9] Oracle. Sobre la tecnología Java. [En línea]. Disponible en http://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html

[10] Java. Sobre Java. [En línea]. Disponible en http://www.java.com/es/about

[11] Tiobe. Índice de Tiobe. [En línea]. Disponible en http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

[12] W3schools. Introducción a HTML. [En línea]. Disponible en http://www.w3schools.com/html/html_intro.asp

[13] W3C. Wiki de HTML. [En línea]. Disponible en http://www.w3.org/community/webed/wiki/HTML#HTML

[14] W3schools. Introducción a HTML5. [En línea]. Disponible en http://www.w3schools.com/html/html5_intro.asp

[15] W3C. Guía breve de CSS. [En línea]. Disponible en http://www.w3c.es/Divulgacion/GuiasBreves/HojasEstilo

[16] W3C. Introducción a Javascript. [En línea]. Disponible en http://www.w3schools.com/js/js_intro.asp

[17] StatCounter. Estadísticas del uso de los navegadores en Europa en el último año. [En línea]. Disponible en http://gs.statcounter.com/#browser-eu-quarterly-201202-201301-bar

[18] Crummy. Documentación de Beautiful Soup. [En línea]. Disponible en http://crummy.com/software/BeautifulSoup/bs4/doc/

[19] lxml. Página inicial. [En línea]. Disponible en http://lxml.de

[20] Blog de Ian Bicking. Comportamiento de parseadores de HTML. [En línea]. Disponible en http://blog.ianbicking.org/2008/03/30/python-html-parser-performance/

[21] LibrosWeb. Introducción a AJAX. [En línea]. Disponible en http://www.librosweb.es/ajax/capitulo_1.html

[22] W3schools. Tutorial de AJAX. [En línea]. Disponible en http://www.w3schools.com/ajax/

[23] Revista hala. Inicio. [En línea]. Disponible en http://revistahala.com/

[24] Magazine Factory. Introducción a Magazine Factory. [En línea]. Disponible en http://magazinefactory.edu.fi/index.php?str=15

[25] Scoop.it. Página de inicio. [En línea]. Disponible en http://www.scoop.it/

[26] Scoop.it. Revista "All Geeks". [En línea]. Disponible en http://www.scoop.it/t/all-geeks

[27] Paper.li. Página de inicio. [En línea]. Disponible en http://paper.li/learn-more.html

[28] Paper.li. Revista "TodayInTech". [En línea]. Disponible en http://paper.li/untanglingweb/1350211456

[29] Mozilla. Complementos de Firefox. Firebug. [En línea]. Disponible en https://addons.mozilla.org/es/firefox/addon/firebug/

[30] Google developers. Chrome DevTools. [En línea]. Disponible en https://developers.google.com/chrome-developer-tools/

[31] Django. Documentación bases de datos. [En línea]. Disponible en https://docs.djangoproject.com/en/dev/ref/databases/

[32] Django. Documentación despliegue con WSGI. [En línea]. Disponible en https://docs.djangoproject.com/en/dev/howto/deployment/wsgi/

[33] Ventana azul. Guía para configurar un servidor para Django en Ubuntu. [En línea]. Disponible en http://www.ventanazul.com/tutoriales/guia-configurar-servidor-django-nginx-apache-ubuntu

[34] Google. Guía de instalación de modwsgi. [En línea]. Disponible en https://code.google.com/p/modwsgi/wiki/QuickInstallationGuide

[35] DocForge. Web application framework. [En línea]. Disponible en http://docforge.com/wiki/Web_application_framework

[36] GNU. GPL. [En línea]. Disponible en http://www.gnu.org/licenses/gpl-3.0.html#howto