UNIVERSIDAD
REY JUAN CARLOS

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE LA TELECOMUNICACIÓN

Curso Académico 2015/2016

Trabajo Fin de Grado

DESARROLLO Y DESPLIEGUE DE APLICACIONES EN EL ÁMBITO EDUCATIVO

Diseño y despliegue de un planeta de blogs

Autor : Adrián Sáez Clemente

Tutor : Gregorio Robles Martínez

Agradecimientos

Quiero agradecer a mis padres, por haber hecho posible que este largo camino haya dado sus frutos de una manera satisfactoria y exitosa. Gracias por vuestro apoyo constante. Sin vosotros, nada de esto podía haber sucedido.

"El que da, no debe volver a acordarse; pero el que recibe nunca debe olvidar"

Resumen

- `Objetivos`: En este proyecto se realiza el desarrollo de una aplicación web que se basa en la creación de un planeta de blogs donde sus principales objetivos son la formación y el aprendizaje de los usuarios.

- `Tecnologías`: El proyecto ha sido realizado en un entorno Linux, usando *Python* como lenguaje de programación y Django como framework. Destacar el uso de tecnologías modernas como *HTML5*, *CSS3*, *AJAX* y diseño responsive con *BootStrap* que han dado dinamismo, rapidez, adaptabilidad y estética a la aplicación.

- `Contexto`: El proyecto está pensado e implementado para un entorno de ámbito educativo. Concretamente para círculos con un número variable de usuarios, como aulas de universidad o de instituto, aulas online, cursos en academias, etc.

Summary

- `Objetives`: This project develops a web application that is based on creating a planet blogs where its main objectives are the formation and learning of its members.
- `Technologies`: The project was carried out in a Linux environment, using programming language *Python* and `Django` as framework. Emphasize the use of modern technologies like *HTML5*, *CSS3*, *AJAX* and responsive design (*BootStrap*) given dynamism, speed, adaptability and aesthetics to the application.
- `Context`: The project is designed and implemented for environment education. Specifically for circles with a variable number of users, such as university classrooms or school, online classes, courses in academies, etc.

Índice general

Agradecimientos **3**

Resumen **5**

Summary **7**

Lista de figuras **13**

1. Introducción **1**

1.1. Descripción del proyecto 1

1.2. Contexto tecnológico 2

2. Objetivos **3**

2.1. Objetivo general 3

2.2. Objetivos específicos 3

3. Tecnologías utilizadas **5**

3.1. Django y Python 5

3.1.1. Django 5

3.1.2. Python 6

3.2. HTML5 6

3.3. JavaScript 7

3.3.1. jQuery 7

3.3.2. jQuery-ui 8

3.3.3. AJAX 8

3.3.4. JSON 9

3.4. CSS3 9
3.5. BootStrap 10
3.6. Git 13
3.7. Pivotal tracker 14
3.8. Crontab 15

4. Diseño e implementación **19**
4.1. Arquitectura general 19
4.2. Estructura Django del proyecto 22
4.2.1. Introducción a la estructura de Django 22
4.2.2. Diseño de la base de datos 23
4.2.3. Estructura de documentos *HTML* 28
4.2.4. Diseño de las vistas y URLs 29

5. Prueba en entorno real **61**
5.1. Preparación de la prueba 61
5.1.1. Pasos iniciales 63
5.1.2. Archivo *nohup.out* 66
5.1.3. Control estadístico 66
5.2. Resultados de la prueba 67

6. Conclusiones **71**
6.1. Consecución de objetivos 71
6.2. Aplicación de lo aprendido 73
6.3. Lecciones aprendidas 73
6.4. Trabajos futuros 73

A. Instalación y uso **75**
A.1. Instalación del entorno en el servidor 75
A.2. Uso de la aplicación 77

B. Manual de usuario **79**
B.1. Página de inicio 79

B.2. Página de registro 80
B.3. Presentación de tutor 82
B.4. Presentación de alumno 86
B.5. Personalizar diseño 88
B.6. Hilo de alumno 88
B.6.1. Entradas 90
B.6.2. Logo 95
B.6.3. Entradas más valoradas 96
B.6.4. Usuarios suscritos 96
B.6.5. Paginador y flecha 96
B.7. Hilo de profesor 98
B.8. Opciones de menú en cabecera 100
B.8.1. Mis hilos 100
B.8.2. Título del hilo 100
B.8.3. Buscar 100
B.8.4. Puntuación 100
B.8.5. Ayuda 102
B.8.6. Cierre de sesión 102

C. Descarga de la aplicación **105**

Bibliografía **107**

Índice de figuras

3.1. Tamaños en diferentes dispositivos 12
3.2. Ejemplo para dispositivos medianos (clase col-md-x) y pequeños (clase col-xs-x) 13

4.1. Estructura de directorios. Utilizando la herramienta *tree*. 21
4.2. *Estructura general del sistema* `Django`. 23
4.3. *Estructura gráfica de la base de datos.* 28
4.4. *Página de inicio.* 31
4.5. *Página de registro de un alumno.* 32
4.6. *Página de registro de un tutor.* 33
4.7. *Presentación del perfil de un alumno.* 35
4.8. *Gestionar diseños personalizados.* 36
4.9. *Presentación del perfil de un tutor.* 38
4.10. *Modificar una contraseña desde el perfil de un tutor.* 39
4.11. *Página principal de un hilo limpio.* 40
4.12. *Página principal de un hilo de un alumno.* 42
4.13. *Página principal de un hilo de un tutor.* 43
4.14. *Página de búsqueda avanzada de entradas.* 44
4.15. *Página de ranking.* 45
4.16. *Página de ranking.* 46
4.17. *Página con las bases de las puntuaciones.* 47
4.18. *Página con las bases de las puntuaciones.* 48
4.19. *Eliminar hilo en una presentación de alumno.* 49
4.20. *Eliminar hilo en una presentación de tutor.* 50
4.21. *Comentario agregado en una entrada.* 51

4.22. *Comentario eliminado en una entrada.* 52
4.23. *Valoración de una entrada.* 53
4.24. *Lectura de una entrada.* 54
4.25. *Visitantes de una entrada.* 55
4.26. *Eliminación de una entrada en el hilo de un tutor.* 56
4.27. *Eliminación de un alumno en el hilo de un tutor.* 57
4.28. *Eliminación de un diseño en el hilo de un alumno.* 58
4.29. *Ejecución de una búsqueda.* 59
4.30. *Enlace de cierre de sesión.* 60

5.1. Hilo de la prueba. 65
5.2. Suscriptores de la prueba. 65
5.3. Estadísticas generales de la prueba. 69
5.4. Estadísticas individuales de dos usuarios de la prueba. 69

B.1. Página de inicio. 79
B.2. Inicio de sesión. 80
B.3. Inicio de sesión 81
B.4. Rellenar nick 81
B.5. Registro erróneo 82
B.6. Información de registro 83
B.7. Página de presentación de tutor. 84
B.8. Formulario para la creación de un hilo. 84
B.9. Sección de *Hilos disponibles* en una presentación de tutor. 85
B.10. Sección de *Mis hilos* en una presentación. 85
B.11. 86
B.12. *Página de presentación de alumno.* 86
B.13. Sección de *Hilos disponibles* en una presentación de alumno. 87
B.14. Error al introducir *RSS* en una presentación de alumno. 88
B.15. Eliminar hilo de la sección de *Mis hilos* en una presentación de alumno. 89
B.16. *Gestionar diseños personalizados.* 89
B.17. Hilo vacío de un alumno. 90

B.18. Hilo con entradas de un alumno. 91
B.19. Lectura de entradas. 92
B.20. Información emergente de un comentario. 93
B.21. *Valoraciones del tutor en entradas.* 94
B.22. Información emergente de un alumno. 94
B.23. Logo, nombre y eslogan aleatorio de la aplicación. 95
B.24. Entradas más valoradas del hilo de alumno. 96
B.25. Usuarios suscritos al hilo. 97
B.26. Paginador y flecha. 97
B.27. Hilo con entradas de un tutor. 98
B.28. *Eliminaciones en hilo de tutor.* 99
B.29. Valoración de tutor en una determinada entrada. 99
B.30. *Interacciones de usuario en la página de Ranking.* 101
B.31. Opciones de estadísticas. 102
B.32. *Visualización de opciones estadísticas.* 103
B.33. Ayuda. 104
B.34. Información del creador del hilo. 104

Capítulo 1

Introducción

1.1. Descripción del proyecto

En este proyecto se implementa una aplicación web que consiste en el desarrollo de un planeta de blogs. Está enfocado al ámbito educativo y puede ser usado en cualquier entorno que simule un aula, ya sea en universidades, institutos, colegios, academias, cursos online, etc,...

El contenido de la aplicación recoge automáticamente mensajes o entradas de un conjunto de blogs externos que comparten una temática común.

El tutor será el responsable de crear hilos donde los alumnos podrán suscribirse y formar parte de la temática de interés. Dentro de cada hilo, los alumnos podrán visualizar las entradas del conjunto de blogs que están suscritos a ese hilo.

Como novedad incluye la posibilidad de interactuar, valorar y comentar los mensajes de compañeros de una forma divertida mediante un proceso de gamificación. En este proceso los participantes de cada hilo podrán luchar por un puesto en la clasificación definitiva consiguiendo puntos (llamados planets) y subiendo de nivel según el compromiso y la capacidad de interacción de cada alumno.

Existe una sección de búsqueda para poder encontrar entradas introduciendo nick de usuario y/o identificador de entrada. Y además, dentro de la aplicación se puede encontrar dos secciones con las bases de las puntuaciones y las estadísticas generales de cada usuario.

1.2. Contexto tecnológico

Actualmente, la sociedad se encuentra en un proceso de evolución enorme en la era de la tecnología. Esta evolución llama a la mayoría de los usuarios del mundo que usan internet a aprender a manejar nuevas aplicaciones, herramientas o tecnologías para poder obtener conocimientos nuevos y compartirlos. Todo esto contrasta con un proceso de asimilación y descubrimiento por otra parte de usuarios con una cierta edad que nunca convivieron con el mundo de "los aparatos e internet".

En la actualidad tecnológica, se está produciendo una gran crecimiento y progresión en el ámbito móvil, con la aparición de los denominados smartphones o teléfonos inteligentes, aportando una herramienta para conectarse a internet desde cualquier sitio.

El principal problema es la gran diversidad de smartphones y de computadoras modernas, referido tanto a la diversidad hardware —diferentes tamaños de pantalla— como a la variedad de sistemas operativos, e incluso entre ellos, diferentes versiones. En este proyecto se ha fijado como fuente de test y desarrollo, un sistema operativo Linux Ubuntu 14.04 con un navegador Firefox, que es uno de los navegadores de internet más utilizados en la actualidad. El desarrollo de aplicaciones utilizando tecnologías libres y abiertas, en concreto *CSS*, *HTML* y *JavaScript*, tiene como ventaja la creación de un programa compatible con casi cualquier dispositivo, teniendo que preocuparse exclusivamente de pequeñas variaciones debidas a las diferencias entre navegadores de internet (ya que se cambia la idea de adaptarse a diferentes sistemas operativos por diferentes navegadores de internet) y a los diferentes tamaños de pantalla y métodos de control (pantallas táctiles en móviles y tablets o ratón y teclado en ordenadores).

Capítulo 2

Objetivos

2.1. Objetivo general

El objetivo principal del proyecto es el desarrollo de una aplicación educativa que englobe y unifique multitud de temas para que a través de ellos los usuarios puedan interactuar, formarse correctamente y optimizar su aprendizaje.

2.2. Objetivos específicos

- *Usabilidad sencilla:* Cuanta más sencillez de uso, mayor número de personas podrán manejar la aplicación. Fue implementada para personas que no están muy familiarizadas con las nuevas tecnologías, de esta forma se conseguiría abarcar mayor variedad de usuarios.

- *Integración de elementos sociales:* Estamos en un contexto en el que se muestra un gran interés en los elementos sociales. La aplicación integra funciones sociales para la atracción de usuarios. Estas funciones son muy importantes para demostrar la capacidad del usuario de interactuar dentro de un mundo digital. Con un simple comentario, un "Me gusta", un "No me gusta" o una valoración por parte del tutor, se crean vínculos digitales capaces de mover diferentes sentimientos y pensamientos en las personas como son la curiosidad o el afán de superación.

- *Inclusión de gamificación:* Otro punto importante de la aplicación es su moneda de cambio, el *planet*. A partir de la creación de esta moneda se inicia un proceso de gamificación

que alienta la participación y el compromiso, y mejora la experiencia del usuario. Se obtiene una motivación y una recompensación extra con la realización de tareas a cambio de estas monedas. Así mismo se incita a una mayor colaboración de los usuarios mediante la retroalimentación continua sobre el progreso en el ranking. Mejor posición en el ranking significaría el logro de un determinado objetivo personal.
En resumen, volvemos a una de las primeras actividades que realizamos en la vida, jugar. Una actividad que es la principal fuente de aprendizaje cuando somos niños y el principal motivo de diversión cuando somos adolescentes e, incluso, adultos.

- *Adaptabilidad:* La aplicación ha sido implementada con un diseño responsive mediante *BootStrap*. El fin de este diseño es mejorar la experiencia del usuario con una adaptabilidad de todos los elementos del contenido de una página web. Con este fin se logra reducir el uso constante del zoom sobre las webs, de modo que el ancho de pantalla se adapta al dispositivo. Además se simplifica el acceso a los menús y contenidos, y se elimina la ejecución de aplicaciones secundarias que no funcionan en dispositivos móviles.

- *Dinamismo y rapidez:* Otro de los objetivos primordiales es la obtención de una aplicación dinámica. En el momento en que un usuario interactua de alguna manera en cualquier documento *HTML*, tiene la sensación de que los cambios se producen instantáneamente. Esto se ha logrado gracias a la tecnología *AJAX*. Esta tecnología permite realizar funciones dinámicas sin necesidad de recargar la página web, por lo que proporciona más ligereza y velocidad, reduciento el tiempo implicado en cada transacción.

Capítulo 3

Tecnologías utilizadas

3.1. Django y Python

3.1.1. Django

`Django` es un framework de desarrollo web de código abierto. Está escrito en *Python*, y respeta el paradigma Modelo-Vista-Controlador. Este patrón se usa para desarrollar software y se basa en la separación de los datos, de la interfaz del usuario y de la lógica interna. En aplicaciones web se usa con mucha frecuencia, dónde la vista es la página HTML, el modelo es el Sistema de Gestión de Base de Datos y la lógica interna, y el controlador es el responsable de recibir los eventos y darles solución.

-Modelo: Es la capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos: como acceder a estos, como validarlos, cual es el comportamiento que tiene, y las relaciones entre los datos. Se utiliza como representación de la información en el sistema. Trabaja junto a la vista para mostrar la información al usuario y es accedido por el controlador para añadir, eliminar, consultar o actualizar datos.

-Vista: Es la capa de la lógica de negocios. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada. Se presenta como un puente entre los modelos y las plantillas.

-Controlador: Es el elemento más abstracto. Recibe, trata y responde los eventos enviados por el usuario o por la propia aplicación. Interactua tanto con el modelo como con la vista.

3.1.2. Python

Python es un lenguaje de programación con una sintaxis muy limpia y que favorece un código legible. Se ejecuta utilizando un programa intermedio llamado intérprete, en lugar de compilar el código a lenguaje máquina que pueda comprender y ejecutar directamente una computadora. Tiene una ejecución más lenta que los lenguajes compilados pero es más flexible y portable.

Tiene la característica de tipado dinámico que se refiere a que no es necesario declarar el tipo de dato que va a contener una determinada variable, sino que su tipo se determinará en tiempo de ejecución según el tipo del valor al que se asigne, y el tipo de esta variable puede cambiar si se le asigna un valor de otro tipo. Además, *Python* es un lenguaje fuertemente tipado que no permite tratar a una variable como si fuera de un tipo distinto al que tiene, es necesario convertir de forma explícita dicha variable al nuevo tipo previamente.

El intérprete de *Python* está disponible en multitud de plataformas (UNIX, Solaris, Linux, DOS, Windows, OS/2, Mac OS, etc.) por lo que si no utilizamos librerías específicas de cada plataforma nuestra aplicación podrá correr en todos estos sistemas sin grandes cambios.

Es un lenguaje orientado a objetos. Este concepto se define como un paradigma de programación en el que los conceptos del mundo real relevantes para nuestro problema se trasladan a clases y objetos en nuestra aplicación. La ejecución de la aplicación consiste en una serie de interacciones entre los objetos.

En este proyecto, *Python* se usa principalmente para el desarrollo de las vistas en el lado del servidor, donde mediante peticiones de los usuarios, recoge y procesa la información y devuelve los datos a la presentación correspondiente. También se usa para recoger el modelo de datos y las URLs necesarias para enlazar con las vistas.

3.2. HTML5

HTML5 (HyperText Markup Language) Es la versión 5 del lenguaje en el que se escriben las páginas webs. Un lenguaje que sólo son capaces de leer los últimos navegadores, ya que los que son más anticuados hay algunas etiquetas que no son capaces de interpretar. Sirve para integrar contenidos multimedia, flash y para dar un sentido semántico a cada parte de la página web.

HTML5 disminuye el tiempo de carga de las páginas y también ayuda a bajar el ratio de texto/*HTML* ya que es un lenguaje más simple. Es un lenguaje más semántico que dota de un significado a las diferentes partes de la página Web. Ayuda a los buscadores a entender la página web y también sirve para que los navegadores muestren de forma inteligente diferentes versiones de la página Web sin tener que usar plug-ins para dispositivos específicos.

HTML5 incluye más elementos gráficos y multimedia. Permite la inserción de etiquetas canvas, que sustituyen a las animaciones en Flash y también permite incluir de forma muy sencilla vídeos, música y otros elementos multimedia como por ejemplo los videojuegos. Además incluye la geolocalización. Es posible saber desde que lugar se esta visualizando un sitio Web, gracias a sistemas de referenciación como el GPS, la tecnología 3G de los dispositivos móviles o las conexiones tipo WiFi.

HTML5 permite el uso de las Webs offline. Aunque no tengas conexión a Internet si el programador lo desea podrás ver parte o toda la página Web, e incluso interactuar con los contenidos. Esto mejora la usabilidad de la Web y permite al usuario trabajar con aplicaciones online desconectado, algo importante en los dispositivos móviles cuando te mueves en entornos en los que no tienes una conexión de calidad.

3.3. JavaScript

3.3.1. jQuery

jQuery es una librería *JavaScript* pensada y creada para acceder fácilmente a los elementos del DOM, para automatizar tareas comunes y simplificar las labores complicadas de *JavaScript*.

Esta librería da una serie de ventajas:

- Facilidad de acceso en el documento.
- Para la localización de partes específicas en la estructura del documento *HTML*, *jQuery* tiene un mecanismo de selector sólido y eficiente con el que puede recuperar piezas exactas del DOM (Document Object Model).
- Modifica la apariencia de un sitio web. Puede llenar vacios que *CSS* no puede cubrir. Y también puede cambiar clases y estilos aplicados en *CSS* incluso después de que la página ya esté renderizada en el navegador.

- Altera contenidos del documento. *jQuery* puede modificar cualquier contenido en el documento.

- Responde a la interacción del usuario. Ofrece una forma elegante de controlar acciones y nos presenta una amplia variedad de eventos que responden a diversas acciones del usuario sin necesidad de saturar el código *HTML*.

- Simplifica tareas comunes de *JavaScript*. Esta librería ofrece mejoras en el momento de desarrollar en *JavaScript*, nos ahorra tiempo y el código será siempre más limpio y sencillo.

3.3.2. jQuery-ui

jQuery-ui es una biblioteca de componentes para la librería *jQuery* que le añaden un conjunto de plug-ins (drag, drop, sorting, resizing, etc.), widgets (accordion, date picker, dialog, slider y tabs) y efectos visuales (show, hide, toggle, color animation, class manipulation, etc.) para la creación de aplicaciones web.

El "accordion" es el widget que se usa en la pestaña de estadísticas y sirve para mostrar la información detallada de un solo usuario. Si clicas en otro nombre del acordeón, se oculta el usuario seleccionado anteriormente y se expande la información del usuario actual.

3.3.3. AJAX

AJAX (Asynchronous *JavaScript* And *XML*) se usa para la transferencia de información mediante *JavaScript*. Esta información fluye por dos canales entre el cliente y el servidor, y puede ser transferida en formato *XML*, texto simple u otros estándares como JSON.

AJAX sirve para actualizar una porción de nuestra página web sin necesidad de cargar el código, atributos gráficos e imágenes de nuevo. Esto provoca mayor rapidez en la interacción cliente-servidor. El cliente no percibe demoras puesto que las comunicaciones se producen en segundo plano y no hay interrupciones. *AJAX* tiene la capacidad de comprimir las funciones de una aplicación en un espacio *HTML* más reducido de lo que antes podía ser necesario.

Algunos de los inconvenientes de *AJAX* son el aumento de la complejidad en el desarrollo de aplicaciones, problemas con restricciones de seguridad relacionados con su uso, y aumento

de la dificultad en la indexación para los motores de búsqueda ya que el contenido deja de ser estático.

Dentro de la aplicación, la mayoría de las acciones que ocurren sin recargar la página se produce por el intercambio de datos entre cliente y servidor mediante *AJAX*, con peticiones `GET` y `POST`. Como por ejemplo, agregar una asignatura a "Mis hilos", enviar y eliminar un comentario, o clicar en los botones de "Me gusta/No me gusta".

3.3.4. JSON

JSON es un formato de fichero de texto ligero utilizado para el intercambios de datos. Básicamente JSON describe los datos con una sintaxis dedicada que se usa para identificar y gestionar los datos. Este formato es simple, directo, ligero y muy importante para el desarrollo de aplicaciones basadas en lenguajes Web y para dispositivos móviles. Cada dato es un par nombre/valor.

En las vistas del proyecto, JSON es un formato que se usa para enviar datos de determinados modelos a la página *HTML*. Las funciones del fichero *JavaScript* captan los modelos en este formato para después recopilar dicha información y manejarla según la lógica de cada función. Por ejemplo, para poder mostrar un popup con la información de un usuario al pasar el ratón por encima de su nombre, la función *JavaScript* recolectó el modelo de datos `User` para poder elegir la información del usuario de interés y manejarla de tal forma que en la página se contemple un popup con dicha información.

3.4. CSS3

CSS (Cascading Style Sheets) se refiere a la tecnología desarrollada para separar la presentación de la estructura *HTML* de un sitio web. Esta tecnología aplica reglas de estilo a los elementos *HTML*, quedando de esta manera separada de la estructura *HTML*. Poco a poco este lenguaje se ha ido haciendo más importante entre los diseñadores gracias a toda la facilidad de uso, y los resultados que son muy flexibles.

CSS3 es la versión 3 de *CSS*, donde se definen las características de este lenguaje, añadiendo muchas funciones y mejorado otras. En una hoja de estilo ó fichero .css, se definen reglas en las

cuales se van a aplicar diferentes y muy variados efectos a las etiquetas *HTML*. Las reglas se forman a partir de selectores, atributos y valores.

Los selectores se usan para definir sobre qué etiqueta *HTML* vamos a aplicar atributos y valores. Existen selectores de etiquetas *HTML*, los cuales se utilizan escribiendo el nombre de la etiqueta a la que se aplicará el estilo. También existen selectores de identificador, que son aquellos que empiezan por "#" seguido del id de la etiqueta a la que queremos darle los atributos. Y por último, selectores de clase que se escriben en el documento *CSS* comenzando por "." y va seguido del nombre de la clase de la etiqueta.

Los atributos son efectos o estilos que aplicaremos a las etiquetas *HTML*. Una vez definidos los selectores y los atributos, se les da un valor a los atributos. Se aplican sobre texto, enlaces, imágenes, listas, tablas, formularios, etc. También incluye la posibilidad de realizar animaciones con porcentajes de tiempo.

En el proyecto, cada documento *HTML* lleva consigo su fichero *CSS*. Gracias a él, cada página muestra los colores apropiados, los distintos tipos de bordes para los diferentes elementos o los distintos tipos de letras con sus efectos en cursiva, negrita, etc.

Con el uso de *CSS3*, lo más destacado son algunas de las animaciones que se encuentran por toda la aplicación. Por ejemplo, en la página de registro, mientras el usuario está escribiendo su nick de usuario en el formulario, justo debajo aparece una rueda giratoria y cuando deja de escribir enseña una línea informativa que aclara si es "nick correcto" o "nick ocupado".

3.5. BootStrap

BootStrap es un framework originalmente creado por Twitter, que permite crear interfaces web con *CSS* y *JavaScript* y cuya particularidad es la de adaptar la interfaz del sitio web al tamaño del dispositivo en que se visualice, ya sea pantalla de PC, portátil, tablet, Smartphone,.. Esta técnica de diseño y desarrollo se conoce como diseño adaptativo o 'responsive design'.

El beneficio de usar diseño adaptativo en un sitio web, es principalmente que el sitio web se adapta automáticamente al dispositivo desde donde se acceda. Aun ofreciendo todas las posibilidades que ofrece *BootStrap* a la hora de crear interfaces web, los diseños creados con *BootStrap* son simples, limpios e intuitivos, esto les da agilidad a la hora de cargar y adaptarse a otros dispositivos. El framework trae varios elementos con estilos predefinidos fáciles de

configurar como botones, menús desplegables, formularios incluyendo todos sus elementos, y *jQuery* integrado para ofrecer ventanas y tooltips dinámicos.

El uso de *BootStrap* ofrece una serie de ventajas:

- Permite tener un mismo lenguaje para la comunicación entre diseñadores y equipo de desarrollo.
- Tiene un comportamiento unificado entre diferentes navegadores.
- Es compatible con navegadores modernos, dispositivos y navegadores antiguos (hasta Internet Explorer 8)
- Soporte avanzado para Responsive web Design (se podría utilizar como NO responsive pero forzándolo).
- Comportamiento y plugins basados en *jQuery* muy útiles y usados habitualmente.
- *BootStrap* es responsive y viene con un grid de 12 columnas y cuenta con diferentes 'Media queries': 768 px (tablets), 992 px (Desktop) y 1200 px (Large Desktop) + Mobile (por defecto en *BootStrap*).
- Y por último y más importante, usa un sistema de rejillas.

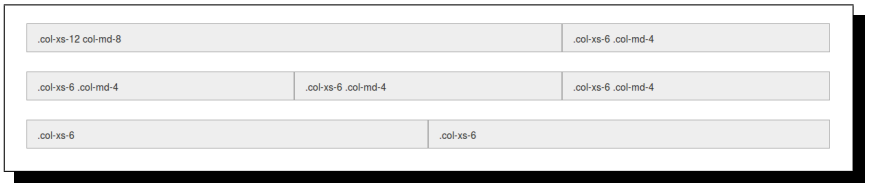
BootStrap incluye una rejilla fluída pensada para móviles. Esta rejilla crece hasta 12 columnas a medida que crece el tamaño de la pantalla del dispositivo. *BootStrap* incluye clases *CSS* para utilizar la rejilla directamente en tus diseños y también define mixins de LESS para que puedas crear diseños más semánticos.

El diseño de páginas basado en rejilla se realiza mediante filas y columnas donde se colocan los contenidos. La rejilla de *BootStrap* funciona del siguiente modo: Las filas siempre se definen dentro de un contenedor de clase .container (anchura fija) o de tipo .container-fluid (anchura variable), de esta forma las filas se alinean bien y muestran el padding correcto. Las filas se utilizan para agrupar horizontalmente a varias columnas. El contenido siempre se coloca dentro de las columnas, ya que las filas sólo deberían contener como hijos elementos de tipo columna. *BootStrap* define muchas clases *CSS* (como por ejemplo .row y .col-xs-4) para crear rejillas rápidamente. La separación entre columnas se realiza aplicando padding. Para contrarrestar sus efectos en la primera y última columnas, las filas (elementos .row) aplican márgenes negativos.

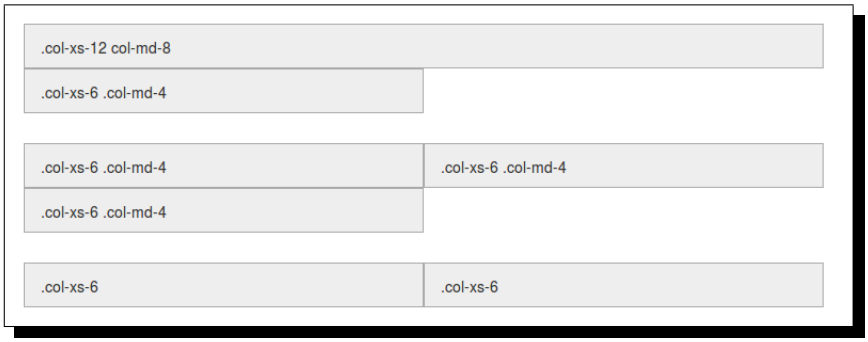
	Dispositivos muy pequeños Teléfonos (<768px)	**Dispositivos pequeños** Tablets (≥768px)	**Dispositivos medianos** Ordenadores (≥992px)	**Dispositivos grandes** Ordenadores (≥1200px)
Comportamiento	Las columnas se muestran siempre horizontalmente.	Si se estrecha el navegador, las columnas se muestran verticalmente. A medida que aumenta su anchura, la rejilla muestra su aspecto horizontal normal.		
Anchura máxima del contenedor	Ninguna (`auto`)	`728px`	`940px`	`1170px`
Prefijo de las clases CSS	`.col-xs-`	`.col-sm-`	`.col-md-`	`.col-lg-`
Número de columnas	12			
Anchura máxima de columna	`auto`	`60px`	`78px`	`95px`
Separación entre columnas	`30px` (`15px` a cada lado de la columna)			

Figura 3.1: Tamaños en diferentes dispositivos

Las columnas de la rejilla definen su anchura especificando cuántas de las 12 columnas de la fila ocupan. Si por ejemplo quieres dividir una fila en tres columnas iguales, utilizarías la clase .col-xs-4 (el 4 indica que cada columna ocupa 4 de las 12 columnas en las que se divide cada fila).



(a) Dispositivo mediano



(b) Dispositivo pequeño

Figura 3.2: Ejemplo para dispositivos medianos (clase col-md-x) y pequeños (clase col-xs-x)

Por último, hay que decir que *BootStrap* permite el anidamiento, el reordenamiento y el desplazamiento de las columnas (tanto por la izquierda como por la derecha).

En la aplicación se usa un plugin de ventanas modales de *BootStrap* que cabe mencionar. El plugin se llama *BootStrapDialog* y es una ventana de cuadro de diálogo / emergente que aparece en la parte superior de la página actual. Esta ventana es un cuadro de diálogo con cabecera, cuerpo y footer, y además tiene añadidos botones para realizar la acción que se desee.

En la página principal de un hilo en la parte del tutor, puede saltar dicha ventana si el tutor intenta eliminar una entrada o un alumno suscrito.

3.6. Git

Git es un sistema de control de versiones distribuido de código abierto relativamente nuevo que nos ofrece las mejores características en la actualidad, pero sin perder la sencillez. Es el sis-

tema de control de versiones más usado por desarrolladores en el mundo. A los programadores ayuda a ser más eficientes en su trabajo, ya que ha universalizado las herramientas de control de versiones del software que hasta entonces no estaban tan popularizadas y tan al alcance del común de los desarrolladores.

Git es multiplataforma, por lo que puedes usarlo y crear repositorios locales en todos los sistemas operativos más comunes, Windows, Linux o Mac. Existen multitud de GUIs (Graphical User Interface o Interfaz de Usuario Gráfica) para trabajar con *Git*. Una de ellas es por línea de comandos.

Dentro de este proyecto, para hacer uso de este sistema de control de versiones se ha utilizado *GitHub*. *GitHub* es un servicio para alojamiento de repositorios de software gestionados por el sistema de control de versiones *Git*. Por tanto, *Git* es algo más general que nos sirve para controlar el estado de un desarrollo a lo largo del tiempo, mientras que *GitHub* es un sitio web que usa *Git* para ofrecer a la comunidad de desarrolladores repositorios de software. En definitiva, *GitHub* es un sitio web pensado para hacer posible el compartir el código de una manera más fácil.

Una de sus características más importantes es la posibilidad de crear ramificaciones y trabajar en cada una de ellas sin afectar al resto, permitiendo estar trabajando con varios aspectos complejos de un programa de forma simultánea y completamente aislada, o dando la posibilidad de dividir un proyecto para desarrollar partes específicas para un determinado servicio o un determinado cliente, e incluso facilitar el desarrollo de proyectos en los que trabajen varias personas.

3.7. Pivotal tracker

Pivotal Tracker es un servicio para la gestión ágil de proyectos y colaboración creado por Pivotal Labs. Esta herramienta incluye soporte para el intercambio de archivos y gestión de tareas, así como para la planificación de las iteraciones. Además posee una API para extensiones y herramientas de terceros.

Las ventajas de usar Pivotal Tracker son:

- Es gratis y no tiene ningún tipo de limitación. Es decir no hay licencia “community” y otra “enterprise”, ni nada parecido.

- Esta en la nube. Con lo que no tenemos que instalar nada en nuestros ordenadores, todo lo haremos a través del navegador.
- Se centra en algo muy interesante que es la gestión de la pila de producto, y métricas como la velocidad.
- Se puede usar para trabajo con equipos descentralizados, puesto que está en internet. Además se puede crear varios perfiles de acceso por lo que podría haber usuarios que, por ejemplo, sólo tienen permiso para ver las cosas, pero no para modificarlas.
- Guarda histórico de nuestras acciones.

Al logarse en la página web se muestra un "Dashboard" que es el panel de control donde se incluyen todos los proyectos creados.

Al crearse un nuevo proyecto, todas las historias de usuario se ordenan en varias pilas que podemos ver en la zona principal del proyecto.

- **Current**: Son las historias de usuario que tenemos planificadas para el sprint en el que estamos.
- **Backlog**: Son las historias de usuario que tenemos planificadas para sprints futuros. Pivotal Tracker nos va marcando donde empiezan los sprints. Esto lo hace de forma automática en función de nuestra velocidad media.
- **Icebox**: Son historias de usuario que todavía no tenemos planificadas.
- **Done**: Esta pila por defecto no se muestra, pero si la activamos nos enseñaría los sprint que ya han pasado. Aquí podemos ver cuándo se hizo qué.

3.8. Crontab

Crontab es una herramienta especializada en la automatización de tareas y su principal pilar es el cron. En el sistema operativo Unix, cron es un administrador regular de procesos en segundo plano (demonio) que ejecuta procesos o guiones a intervalos regulares (por ejemplo, cada minuto, día, semana o mes). Los procesos que deben ejecutarse y la hora en la que deben hacerlo se especifican en el fichero *Crontab*. El demonio cron inicia de `/etc/rc.d/` o

`/etc/init.d` dependiendo de la distribución. Cron se ejecuta en el background, revisa cada minuto la tabla de tareas *Crontab* `/etc/crontab` o en `/var/spool/cron` en búsqueda de tareas que se deban cumplir. Como usuario podemos agregar comandos o scripts con tareas a cron para automatizar algunos procesos. Esto es útil por ejemplo para automatizar la actualización de un sistema, ejecutar un script que tenemos para contabilizar cualquier asunto o bien para avisarnos periódicamente del estado de algún servicio en nuestro servidor Linux. Para automatizar una tarea necesitamos un script hechos por nosotros, el cual queremos ejecutar. En este caso bastará con escribir nuestro script y luego darle permisos de ejecución:

```
#chmod +x miscript.py
```

Para añadir el cron, ejecutamos la edición del *Crontab* con

```
crontab -e
```

en algunas distros (como debian) nos da la opcion de elegir el editor de textos que deseemos, los demás nos quedamos con *vi*. El archivo *Crontab* lucirá algo así:

```
# m h dom mon dow user command
```

donde:

- **m**: Corresponde al minuto en que se va a ejecutar el script, el valor va de 0 a 59.
- **h**: La hora exacta, se maneja el formato de 24 horas, los valores van de 0 a 23, siendo 0 las 12:00 de la medianoche.
- **dom**: Hace referencia al día del mes. Por ejemplo, se puede especificar 15 si se quiere ejecutar cada día 15.
- **mon**: Hace referencia al mes. Por ejemplo, se puede especificar 11 si se quiere ejecutar cada mes de noviembre.
- **dow**: Significa el día de la semana, puede ser numérico (0 a 7, donde 0 y 7 son domingo) o las 3 primeras letras del día en inglés: mon, tue, wed, thu, fri, sat, sun.
- **user**: Define el usuario que va a ejecutar el comando, puede ser root, u otro usuario diferente siempre y cuando tenga permisos de ejecución del script.

- **command**: Refiere al comando o a la ruta absoluta del script a ejecutar. Por ejemplo: /home/usuario/scripts/actualizar.sh. Si acaso llama a un script, éste debe ser ejecutable.

Dentro del código de la aplicación, existe un script llamado `actualizar.py`. Es el encargado de guardar las entradas automáticamente en base de datos. Para su automatización se creó un cron que lo ejecuta cada minuto. Para ello, el fichero *Crontab* fue modificado añadiendo las dos últimas líneas:

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

*/1 * * * * cd /home/nombreMaquina/directorioGitProyecto/PlanetaBlogs/TFG;
python actualizar.py;
```

Capítulo 4

Diseño e implementación

4.1. Arquitectura general

En esta sección se define la estructura interna del proyecto. En la figura 4.1 se muestra una imagen detallada de los directorios y subdirectorios con sus respectivos archivos que hacen posible el funcionamiento de la aplicación. En dicha estructura cabe destacar los siguientes bloques:

- **actualizar.py**: Es un script *Python* que realiza la función de actualización automática de entradas. Cada vez que un alumno ingresa una nueva entrada en su blog, este script es el encargado de detectar esos nuevos cambios a través de los RSS guardados en base de datos. Se ejecutará cada minuto.

- **bbdd.sqlite**: Es la base de datos del proyecto donde se van a guardar todos los datos estructurados a partir de los modelos definidos en `models.py`

- **planetablogs**: Contiene la parte fundamental de la funcionalidad del proyecto. Dentro de este directorio se encuentran distintos ficheros python. `admin.py` hace la función de modificar la interfaz de administración que `Django` genera por defecto. En `formularios.py` se definen todos los formularios que serán utilizados por los usuarios dentro de la aplicación para crear nuevos elementos o actualizar elementos existentes en la base de datos. En `models.py` se encuentran los modelos que se convertirán en tablas dentro de la base de datos. Estos modelos se encargan de representar los datos dentro de la aplicación y contienen los campos necesarios y el comportamiento de los datos que

serán almacenados. Por último, `views.py` es un fichero donde se implementa toda la lógica del proyecto llevando a cabo todas las peticiones de los usuarios.

- **plantillas**: En este directorio se encuentran todas las plantillas de los documentos *HTML* que serán enviados al usuario y mostrados por el navegador para dar una interfaz gráfica a la aplicación.

- **static**: Esta carpeta contiene las librerías *BootStrap*, *jQuery* y *jQuery-ui*. También incluye un subdirectorio donde se guardan tanto las imágenes propias del proyecto como las imágenes de perfil introducidas por los usuarios en el campo de registro. Y además, se localizan todos los ficheros *.js* y *.css* que añadirán funcionalidad y definirán el diseño a su respectivo documento *HTML* o página web.

- **TFG**: Contiene ficheros básicos para el correcto funcionamiento de la aplicación. `settings.py` es un fichero de configuración que contiene todas las rutas relativas o directorios necesarios. `urls.py` define todas las URLs y las asocia a una vista en `views.py` que se encargará de procesar las peticiones de los clientes.

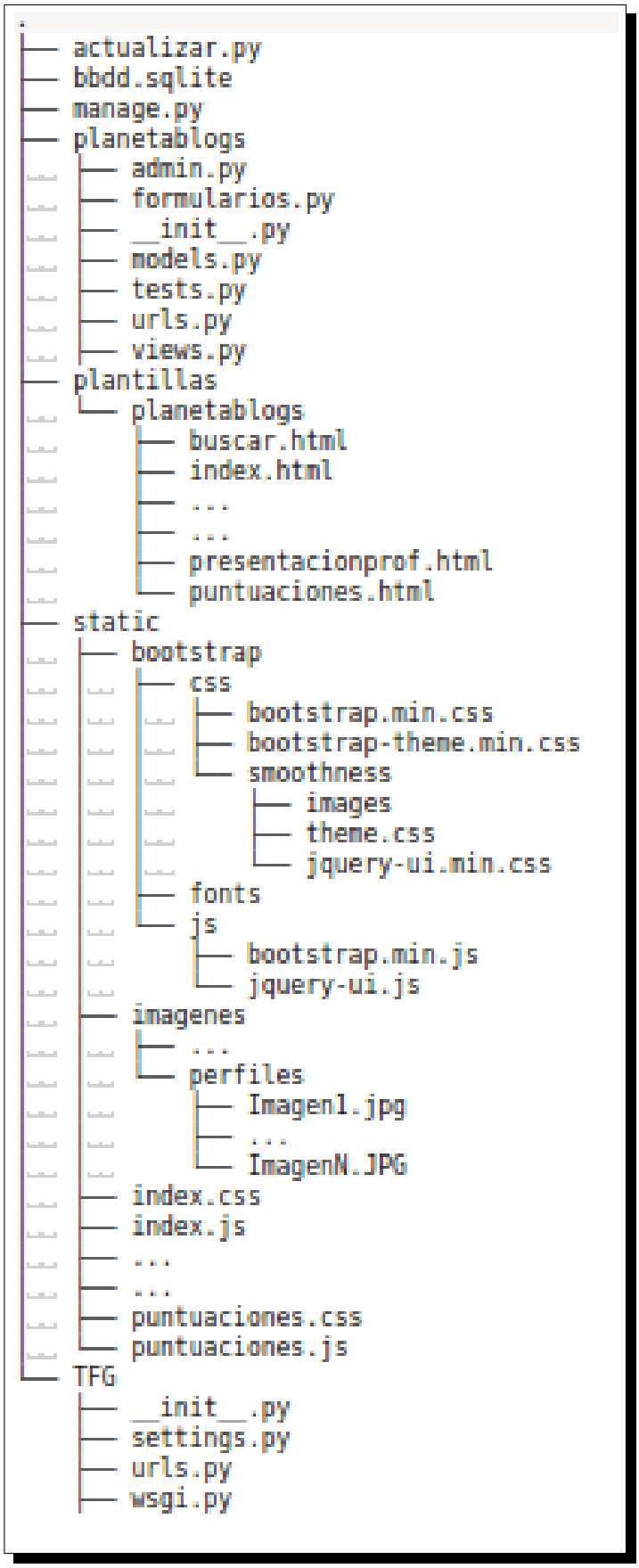


Figura 4.1: Estructura de directorios. Utilizando la herramienta *tree*.

4.2. Estructura Django del proyecto

En esta sección se expone una descripción más detallada del funcionamiento del proyecto. Se empieza con una introducción que cuenta la estructura o esqueleto del sistema `Django` y termina con una explicación más específica del diseño de las distintas partes implementadas en él.

4.2.1. Introducción a la estructura de Django

En la figura 4.2 se muestra un boceto de cómo funciona `Django` internamente. Hay dos partes diferenciadas que separan al cliente del servidor. El cliente es el navegador y el servidor `Django`. El cliente tiene las funciones de enviar peticiones al servidor y de mostrar la presentación o interfaz al usuario final. Las peticiones las lleva a cabo el cliente a través del usuario en el momento en que éste pulsa sobre un enlace o pide una URL en concreto en la barra del navegador mientras que el servidor se encarga de procesar dichas peticiones y de transferir los datos necesarios al cliente para que puedan ser visualizados por el usuario.

El proceso del sistema `Django` comienza con una solicitud por parte del usuario. Esta solicitud puede ser de tipo `GET` para conseguir información de la base de datos o de tipo `POST` para guardar o modificar información de la base de datos. La petición es enviada por el cliente o navegador mediante una URL. `Django` atrapará esa URL y comprobará en su fichero `urls.py` si la URL existe o no —en el caso de que la URL no exista `Django` devolverá al cliente una plantilla *HTML* de error—. En caso de que la petición sea correcta, `Django` mediante el *URLConf* interpretará dicha URL y se dirigirá a la vista apropiada —en el fichero `urls.py` cada URL está asociada a una vista—. Dentro del fichero `views.py` se ubicará la vista que se necesita, la cual se encargará de interactuar con el modelo, bien para obtener datos —en caso de petición `GET`— o bien para guardar o modificar algún elemento del modelo de datos —en caso de petición `POST`—. La vista se redirigirá al fichero `forms.py` en caso de que la petición se hubiera dado a través de un formulario. Por último, la función final de la vista será la de llamar a la plantilla situada en el directorio de templates, y ésta plantilla se ocupará de renderizar la respuesta a la solicitud del cliente.

`Django` se responsabiliza del envío de los diferentes documentos que se encuentran en la parte del servidor. La respuesta al cliente se traduce en forma de documento *HTML*. Este

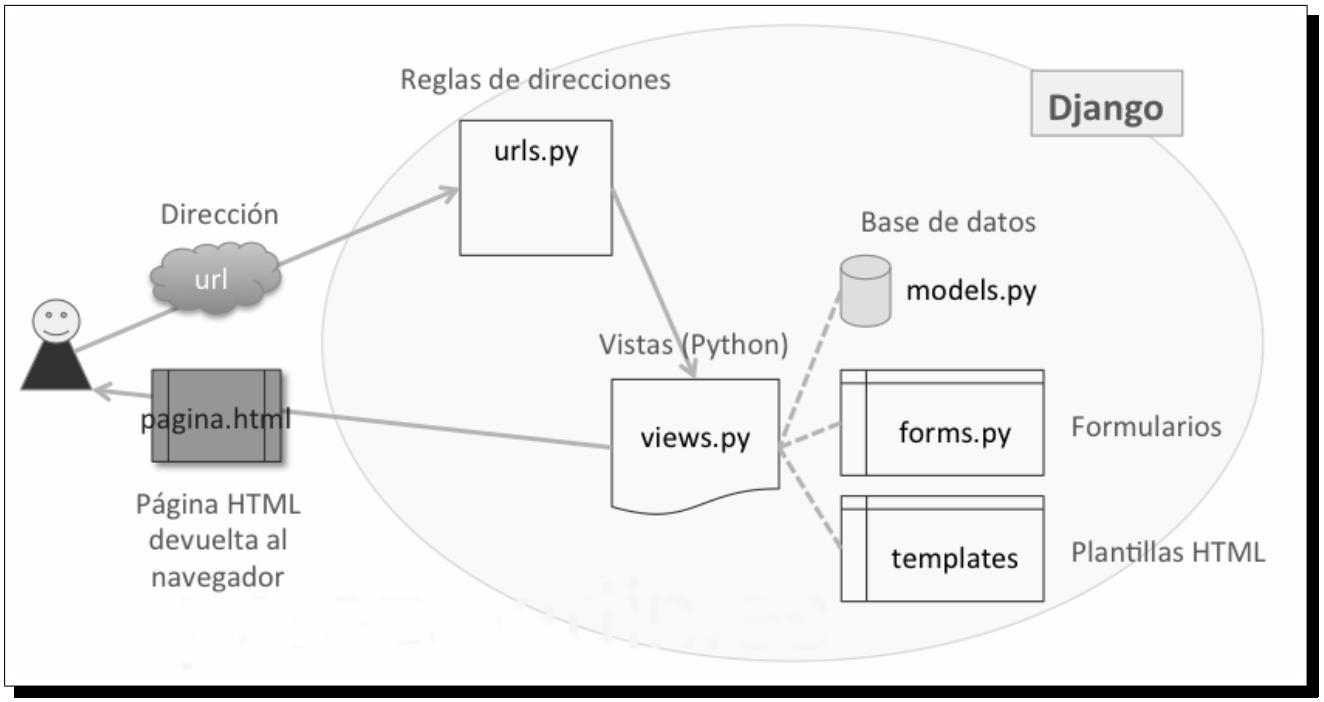


Figura 4.2: *Estructura general del sistema* `Django`.

documento contendrá las plantillas necesarias para mostrar los datos servidos al cliente por parte de `Django`. El servidor también se encarga de servir los ficheros referenciados dentro de ese documento *HTML*. El fichero *JavaScript* añadirá la funcionalidad necesaria —mediante *JavaScript* puro, *jQuery* o *jQuery-ui*— para dar distintos comportamientos a los datos servidos —datos proporcionados directamente en plantillas o serializados con formato *JSON*—

En la interacción del usuario con la interfaz recibida por el cliente cabe destacar el uso de *AJAX*. El usuario usará la aplicación de tal modo que cuando envíe una petición *AJAX* a través del fichero *JavaScript* referenciado en la página *HTML* que esté manejando, el servidor contestará enviando los datos que pide sin necesidad de recargar esa página desde la cual se realizó dicha petición *AJAX*.

4.2.2. Diseño de la base de datos

El diseño de la base de datos se resume visualizando la gráfica 4.3 donde aparecen todos los modelos definidos en el fichero `models.py` dentro del bloque “planetablogs” del sistema de directorios y se manifiestan todos los elementos de todos los modelos con las respectivas

relaciones entre ellos.

La gráfica ha sido obtenida mediante una extensión de `Django` denominada *Graphviz*.

Models

Cada modelo se convierte en una tabla en la base de datos. Los modelos definidos en este proyecto son:

- **User**: Este modelo es creado por *Django* y simboliza un usuario genérico que se registra a la aplicación introduciendo sus atributos: Nombre de usuario *Username*, nombre de pila *Firstname*, primer apellido y segundo apellido *Lastname*, contraseña *Password* y dirección de correo electrónico *Email*. Dentro del fichero `models.py` se añaden dos nuevos atributos *Imagen* y *Estilo* —se agrega con la función `addtoclass`—. El atributo *Imagen* recogerá la ruta de un archivo de imagen, que una vez introducida mediante un formulario de registro junto con el resto de campos, será guardada en la carpeta "perfiles" dentro del sistema de directorios del proyecto. El atributo *Estilo* recogerá el nombre del estilo que el usuario personalizó previamente (en el momento del registro de usuario, este atributo guarda una cadena vacía). `User` contendrá todos los usuarios de aplicación previamente registrados, y ninguno de éstos podrá coincidir con otro en nombre de usuario puesto que en la página de registro se aplica un filtro al introducir el dato *Username*.

- **Profesor**: El modelo `Profesor` hereda todas las propiedades del model `User` mediante un *ForeignKey*. Se crea un atributo *ProfesorId* que coincide con el id que ocupa en la tabla del modelo `User`.

- **Alumno**: El modelo `Alumno` hereda todas las propiedades del model `User` mediante un *ForeignKey*. Se crea un atributo *AlumnoId* que coincide con el id que ocupa en la tabla del modelo `User`.

- **Asignatura**: Este modelo representa una asignatura. Cada asignatura es creada como un hilo de aplicación con su propia temática identificativa. Los atributos de este modelo son: *Titulo* guarda un string de máximo 35 caracteres y define el título de la asignatura, *Descripcion* guarda un texto de máximo 150 caracteres y da una descripción breve y concisa sobre la temática del hilo, *Entradas* es un número entero que indica el número

total de entradas que tiene esa asignatura y *Creador* es el identificador del tutor que creó la asignatura.

- **RSS**: El modelo `RSS` representa al blog o blogs de los usuarios. Cada usuario necesita un blog nuevo para cada asignatura en la que se suscriba. Tiene como campos: *RSS* que es la URL del RSS del blog y *UltimaFecha* que guarda la fecha de la entrada de ese blog que se guardó por última vez en base de datos.

- **Entrada**: Este modelo representa una entrada de blog. Sus propiedades o comportamientos son los siguientes: *Entrada* es un campo entero que guarda el identificador de la entrada —no es el campo pk de la tabla, sino un contador que se incrementa para diferenciar entradas con mismo valor pero de distintas asignaturas—, *Titulo* guarda un string de máximo 10 caracteres y define el título de la entrada, *Descripcion* guarda un texto sin longitud máxima de caracteres y da una descripción detallada de la entrada, *Fecha* guarda la fecha en la que se creó la entrada, *Link* salva la URL donde se ubica la propia entrada dentro del blog de usuario, *UrlBlog* guarda la URL del blog de usuario, *TotalUp* guarda el número total de valoraciones positivas que recopila la entrada, *TotalDown* guarda el número total de valoraciones negativas que recopila la entrada, *Total* guarda la puntuación que lleva hasta el momento la entrada teniendo en cuenta valoraciones positivas y valoraciones negativas —esta puntuación se usa para imprimir en la sección de entradas más valoradas—, *TotalComentarios* cuenta el total de comentarios que contiene la entrada —este entero se mostrará entre paréntesis al lado de "Mostrar comentarios"—, *PuntuacionTutor* guarda la puntuación de la entrada recibida por el tutor, *Visitantes* recoge en un array de cadenas con los ids de los alumnos que han visitado una determinada entrada (se define visitar a la acción de hacer click sobre el enlace del título de cada entrada, click sobre el campo *Link*) y *Visitas* es un entero que guarda el número de visitas totales de la entrada y que corresponde con la cantidad de visitantes.

- **Comentario**: El modelo `Comentario` simula un comentario correspondiente a una entrada. Sus atributos son: *Descripción* guarda el contenido del comentario, *Fecha* se queda con la fecha en la que fue enviado dicho comentario y *Username* salva el nick del usuario que realizó el comentario.

- **Up**: El modelo `Up` representa una valoración positiva de una entrada determinada.
- **Down**: El modelo `Down` representa una valoración negativa de una entrada determinada.
- **Valoracion**: `Valoracion` representa la valoración total de cada usuario. Tiene como atributos: *Puntos* guarda el total de puntos obtenidos y *Nivel* guarda el nivel en el que se situa con respecto a los puntos obtenidos.
- **Extra**: El modelo `Extra` define, como su propio nombre indica, extras de una entrada. *Leido* recoge en un array de cadenas los ids de los alumnos que han leido una determinada entrada.
- **Diseño**: El modelo `Diseno` define los diferentes diseños de la aplicación creados por todos los usuarios. *Estilo* guarda el valor del nombre del diseño y *Imagen* recoge la cadena de la ruta donde se encuentra la imagen de fondo para la aplicación.

Relaciones

Las relaciones entre modelos que tienen lugar en el proyecto son:

- **Profesor**: El modelo `Profesor` se vincula con el modelo `User` ya que un profesor es un usuario. El atributo pk de un elemento `Profesor` puede ser diferente al atributo pk del elemento `User` que hereda.
- **Alumno**: El modelo `Alumno` se relaciona con el modelo `User` puesto que un alumno es un usuario. Al igual que en el caso del modelo `Profesor`, el atributo pk de un elemento `Alumno` puede ser diferente al atributo pk del elemento `User` que hereda. Esto se debe a que a la hora de realizar un registro en la aplicación, se puede elegir entre hacerlo como profesor o hacerlo como alumno.
- **Asignatura**: Este modelo tiene una relación *ManyToMany* con el modelo `Profesor` puesto que una asignatura puede ser enseñada por varios profesores y un profesor puede enseñar varias asignaturas. Lo mismo ocurre con el modelo `Alumno`, una asignatura puede ser aprendida por muchos alumnos y un alumno puede tener muchas asignaturas, aunque en este caso se implementa de manera diferente a la relación con `Profesor`.

Para realizar la relación `Asignatura` - `Alumno` se ha utilizado un modelo intermedio `RSS` a través del argumento *Through*. Con este argumento, a la relación *ManyToMany* se le está señalando que va a actuar un modelo intermediario. El modelo intermediario se utilizará para regular esta relación por lo que se pueden colocar campos adicionales en `RSS` para que `Asignatura` pueda acceder a más datos relevantes de `Alumno`.

- **RSS**: El modelo `RSS` tendrá dos relaciones *ForeignKey* con `Alumno` y `Asignatura` debido a que un RSS sólo puede pertenecer a un alumno y a una asignatura. Los alumnos se suscriben a las asignaturas por medio de un RSS que además de ser válido, no debe estar repetido en base de datos, sino se muestra un error de validación.

- **Entrada**: `Entrada` se relaciona con los modelos `Alumno` y `Asignatura`. Una entrada debe ser única para un alumno y en una asignatura. Mediante el proceso de actualización automática de entradas, las entradas serán guardadas en este modelo asociadas a un alumno y una asignatura o hilo en concreto.

- **Comentario**: El modelo `Comentario` tiene una correspondencia directa con los modelos `Alumno`, `Asignatura` y `Entrada`. Cada comentario será escrito por un alumno, en una entrada y en un hilo.

- **Up**: El modelo `Up` tiene una relación con los modelos `Alumno`, `Asignatura` y `Entrada`. La acción de "hacer click" sobre el botón de valoración positiva tendrá lugar en una entrada de un hilo por medio del alumno que realice dicha acción.

- **Down**: Lo mismo pasa con el modelo `Down`. La acción de "hacer click" sobre el botón de valoración negativa tendrá lugar en una entrada de un hilo por medio del alumno que realice dicha acción.

- **Valoracion**: `Valoracion` tiene un nexo con los modelos `Alumno` y `Asignatura`. La valoración de un usuario se basa en la obtención de puntos al interactuar con la aplicación mediante entradas, comentarios, valoraciones positivas o valoraciones negativas. Una valoración de un alumno tiene que pertenecer a un hilo en concreto para poder diferenciar valoraciones en el caso de que un alumno esté suscrito a varias asignaturas.

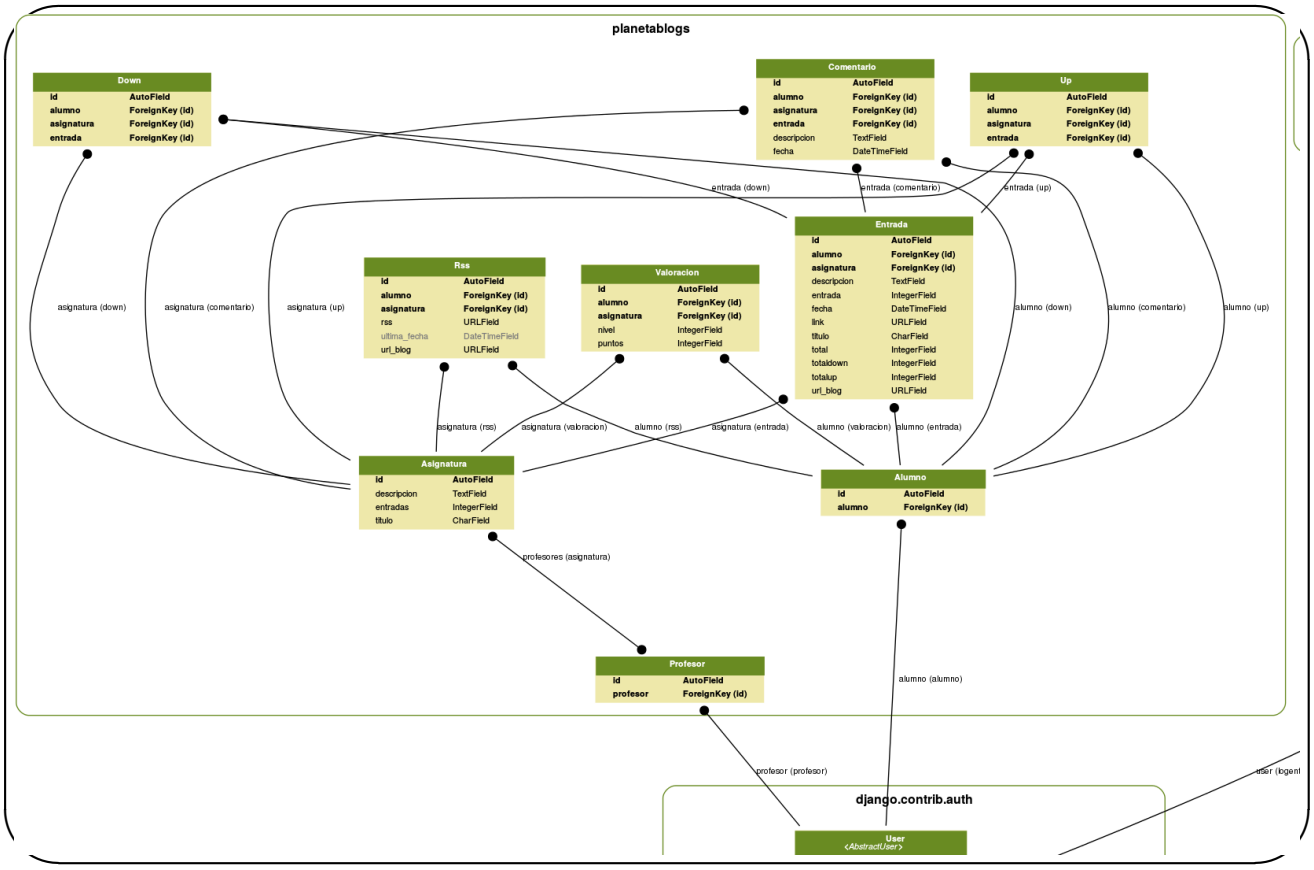


Figura 4.3: *Estructura gráfica de la base de datos.*

- **Extra**: `Extra` se relaciona con los modelos `Asignatura` y `Entrada`. Un extra debe ser único para una entrada y una asignatura. Cada registro en el modelo `Extra` esconderá el array de ids de los alumnos que hayan visitado una entrada en una asignatura.

- **Diseño**: `Diseno` se relaciona con el modelo `User`. Un registro de la tabla debe ser único para un usuario. Cada registro en el modelo `Diseno` esconderá el nombre del diseño creado por el usuario con su respectiva imagen de fondo.

4.2.3. Estructura de documentos *HTML*

Todas las páginas servidas por `Django` en este proyecto contienen en su cabecera (etiqueta `<head>`) las mismas referencias a ficheros *JavaScript* y a ficheros *CSS*. Estas referencias se vinculan con las tecnologías que serán usadas dentro de los documentos *HTML*: Las bibliotecas

JavaScript de *jQuery.js*, *jQuery-ui.js* y *BootStrap.js* son utilizadas para manejar distintas funcionalidades. Las bibliotecas *CSS* de *jQuery-ui.css* y *BootStrap.css* son utilizadas para cambios en el diseño de la interfaz. Y por último, los ficheros *JavaScript* y *CSS* propios de cada página que son utilizados para añadir y mejorar tanto funcionalidad como diseño.

Dentro de la etiqueta `<body>` se introduce un elemento `<div>` con una clase *.container* que mantiene una misma estructura en cada una de las páginas:

- **Cabecera**: Dependiendo de la plantilla que se cargue en el cliente, la cabecera puede variar desde un mensaje de bienvenida a una barra de navegación.
- **Contenedor**: En esta parte se muestra toda la información con la que el usuario final podrá interactuar.
- **Pie de página**: En todos los pies de página se muestra un correo electrónico como contacto y el nombre de la aplicación. Las únicas páginas que tienen pie de página son `index.html`, `ayuda.html` y las páginas de registro.

4.2.4. Diseño de las vistas y URLs

Por la parte del servidor de un proyecto `Django` existe una parte importante que se encarga de procesar las solicitudes del cliente. Esta parte se almacena en un fichero llamado `views.py`. Está compuesto por vistas o funciones a las que `Django` accede después de que el usuario realice una petición mediante una URL. El fichero `urls.py` define todas las URLs a las que se asociará una vista distinta. A continuación la vista procesa la solicitud y devuelve al cliente la información pedida en forma de documento *HTML* o *JSON*.

Cada una de esas vistas se explican en las siguientes subsecciones:

Página de inicio

El documento *HTML* mostrado en la figura 4.4 es la página de inicio. La página es servida tras una previa petición `GET` del cliente mediante el correspondiente recurso [1]. La vista asociada a ese recurso es `inicio`, la cual devuelve la plantilla `inicio.html` que crea el documento *HTML* de la respuesta.

[1] URL correspondiente a la página de inicio: `/planetablogs/login/`

A esta página puede entrar cualquier usuario externo que pretenda crearse un perfil en la aplicación o quiera acceder a su perfil una vez creado.

En la cabecera se visualiza un mensaje de bienvenida animando a un ingreso en la aplicación. En el contenedor existen dos partes diferenciadas:

- ***Regístrate***: Se ofrece la posibilidad de darle a dos botones coloreados en verde, *Alumno* o *Tutor*. Debajo de los botones se halla un mensaje explicativo que resumen la acción de pulsar sobre ambos botones. Si se pulsa sobre el botón *Alumno* se redirecciona a la página que se encarga de crear un perfil como alumno. Si se pulsa sobre el botón *Tutor* se redirecciona a la página que se encarga de crear un perfil como tutor.

- ***Identifícate***: En el formulario, una vez creado un usuario de aplicación, el usuario puede introducir sus datos (`usuario` y `contraseña`) para identificarse y poder entrar a su perfil de usuario dentro de la aplicación.

  Una vez se introducen los datos y el usuario pulsa en el botón *Aceptar*, el cliente realiza una petición `POST` sobre la misma URL por lo que será procesada por la misma vista `inicio`. Esta vista identifica la petición como `POST`, almacena la información enviada mediante el formulario, y la autentifica a través del proceso de autentificación de `Django` (`.authenticate`). Si el usuario está guardado en el modelo `User`, se ejecuta el proceso de login de `Django` (`.login`). A continuación se comprueba el tipo de usuario que se ha logado en la aplicación y dependiendo de si el tipo es `Alumno` o `Profesor`, la vista envía una respuesta `HTTP` redireccionando a la página de perfil del alumno o a la página de perfil del tutor, respectivamente. En el caso de que la información sea incorrecta la vista renderizará la plantilla `inicio.html` con un mensaje de error de login.

Registro de alumno

El documento *HTML* mostrado en la figura 4.5 es la página de registro realizado por parte de un alumno. La página es servida tras una previa petición `GET` del cliente mediante el correspondiente recurso [2]. La vista asociada a ese recurso es `nuevo_alumno`, la cual devuelve la plantilla `nuevoalumno.html` que crea el documento *HTML* de la respuesta.

[2] URL correspondiente a la página de registro de un alumno: `/planetablogs/nuevoalumno/`

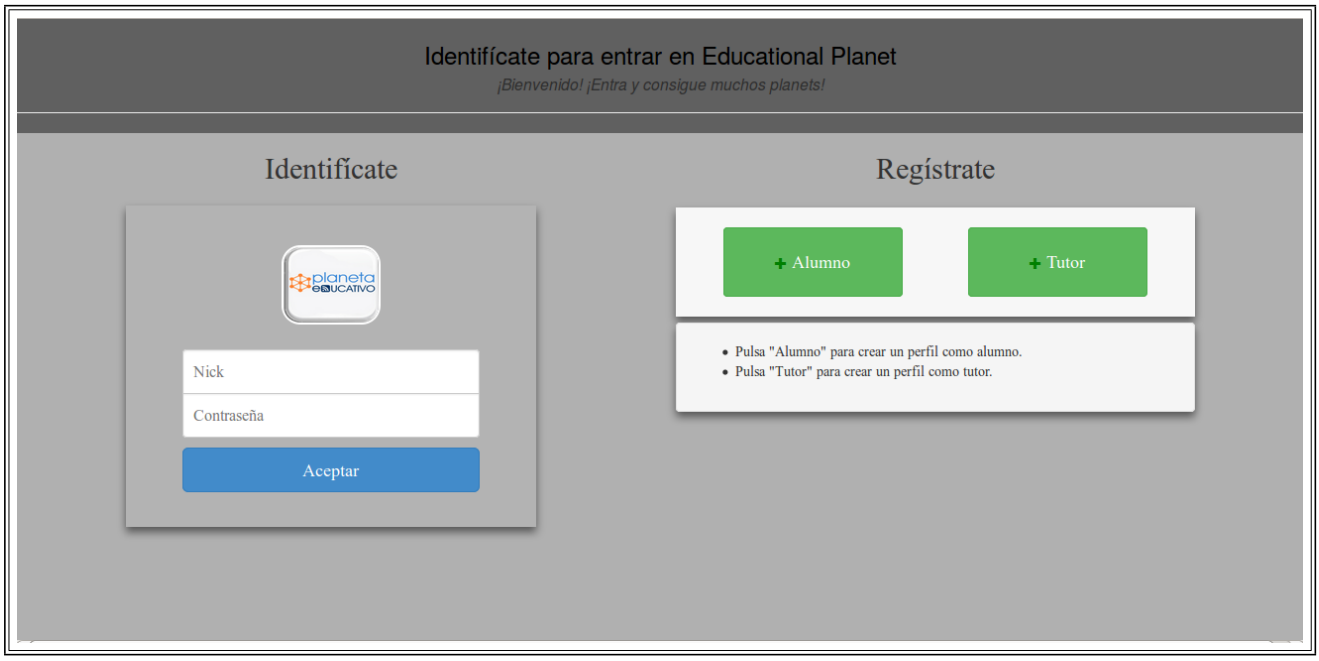


Figura 4.4: *Página de inicio.*

A esta página puede acceder cualquier usuario externo que pretenda crearse un perfil en la aplicación.

En la cabecera se visualiza un mensaje informativo aclarando que el registro del perfil se creará y guardará como alumno. En el contenedor aparece un formulario con la mayoría de los campos del modelo `User` (*Imagen*, *Nick*, *Nombre*, *Apellidos*, *Email* y *Contraseña*).

Una vez relleno, el formulario se envía pulsando sobre el botón `Guardar` coloreado en verde. En ese momento, se realiza una petición `POST` sobre la misma url por lo que será procesada por la misma vista `nuevo_alumno`. Esta vista identifica la petición como `POST`. La información es enviada mediante un formulario de registro definido en `formularios.py`, y si este formulario es válido, dicha información se almacena. La vista crea dos objetos, uno `User` y otro `Alumno`, y ambos los guarda en la base de datos. Y posteriormente, envía una respuesta `HTTP` redireccionando a la página de inicio de la aplicación. En caso de que el formulario sea inválido, la vista renderizará la plantilla `nuevoalumno.html` con un mensaje de error de registro.

En caso de error de registro, te indicará que puedes pasar el ratón por encima del icono de información situado a la izquierda del botón `Guardar`. Si esto sucede, mediante *jQuery* y el

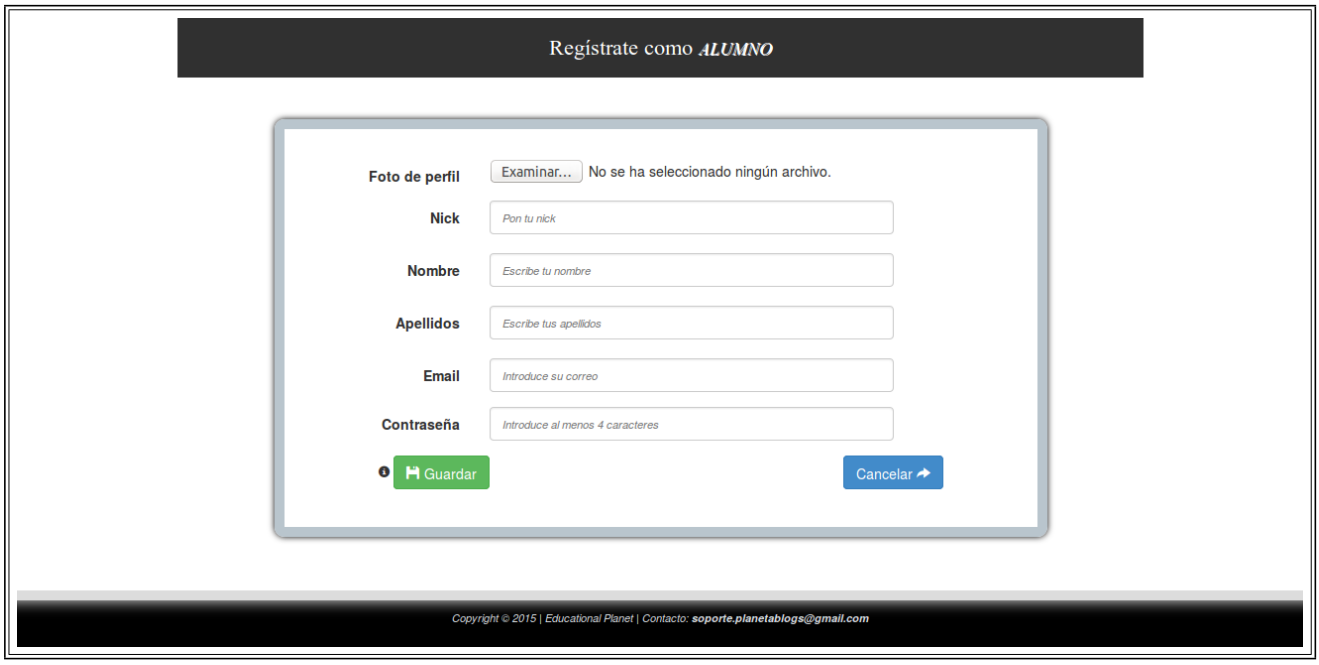


Figura 4.5: *Página de registro de un alumno.*

evento *.mouseover* en el fichero `nuevousuario.js`, se levanta un diálogo de información que se encontraba previamente oculto.

Registro de tutor

El documento *HTML* mostrado en la figura 4.6 es la página de registro realizado por parte de un tutor. La página es servida tras una previa petición `GET` del cliente mediante el correspondiente recurso [3]. La vista asociada a ese recurso es `nuevo_profesor`, la cual devuelve la plantilla `nuevoprofesor.html` que crea el documento *HTML* de la respuesta.

A esta página puede acceder cualquier usuario externo que pretenda crearse un perfil en la aplicación.

En la cabecera se visualiza un mensaje informativo aclarando que el registro del perfil se creará y guardará como tutor. En el contenedor aparece un formulario con la mayoría de los campos del modelo `User` (*Imagen*, *Nick*, *Nombre*, *Apellidos*, *Email* y *Contraseña*).

Una vez relleno, el formulario se envía pulsando sobre el botón `Guardar` coloreado en verde. En ese momento, se realiza una petición `POST` sobre la misma URL por lo que será pro-

[3] URL correspondiente a la página de registro de un alumno: `/planetablogs/nuevoprofesor/`

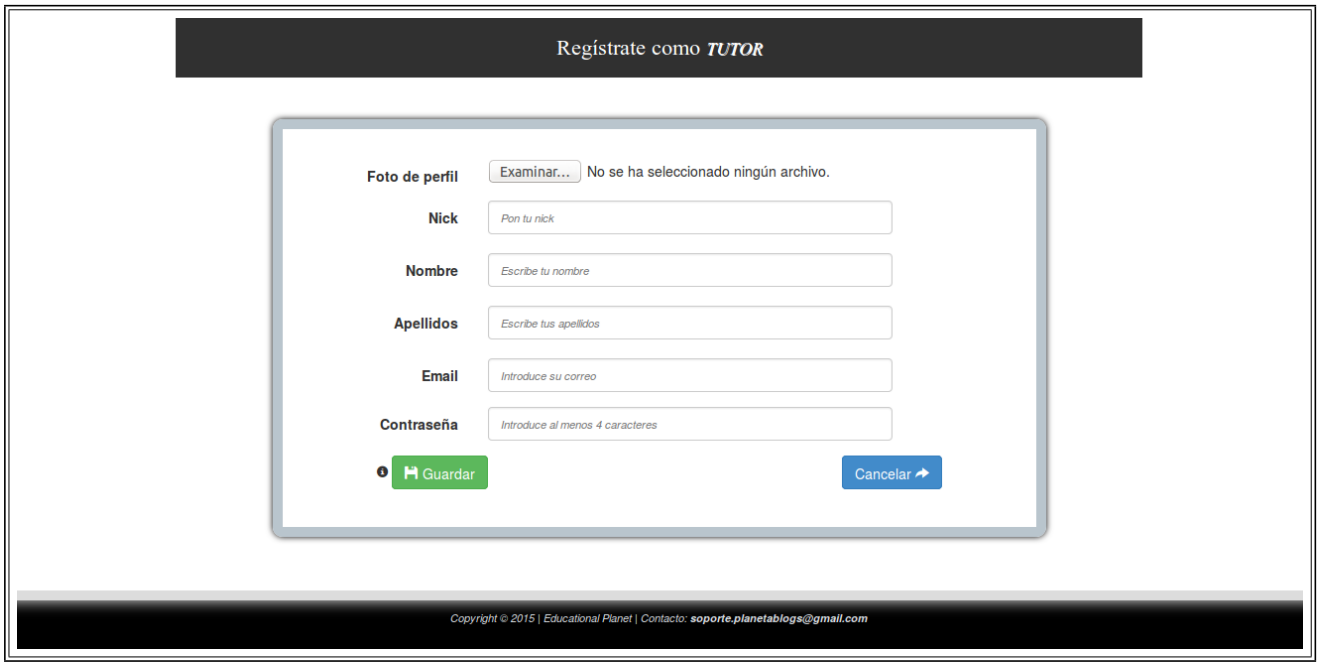


Figura 4.6: *Página de registro de un tutor.*

cesada por la misma vista `nuevo_profesor`. Esta vista identifica la petición como `POST`. La información es enviada mediante un formulario de registro definido en `formularios.py`, y si este formulario es válido, dicha información se almacena. La vista crea dos objetos, uno `User` y otro `Profesor`, y ambos los guarda en la base de datos. Y posteriormente, envía una respuesta `HTTP` redireccionando a la página de inicio de la aplicación. En caso de que el formulario sea inválido, la vista renderizará la plantilla `nuevoprofesor.html` con un mensaje de error de registro.

En caso de error de registro, te indicará que puedes pasar el ratón por encima del icono de información situado a la izquierda del botón `Guardar`. Si esto sucede, mediante *jQuery* y el evento *.mouseover* en el fichero `nuevousuario.js`, se levanta un diálogo de información que se encontraba previamente oculto.

Presentación de alumno

El documento *HTML* mostrado en la figura 4.7 es la página de presentación del perfil de un alumno. La página es servida tras una previa petición `GET` del cliente mediante el correspon-

diente recurso [4]. La vista asociada a ese recurso es `presentacionalumno`, la cual devuelve la plantilla `presentacionalum.html` que crea el documento *HTML* de la respuesta.

A esta página solamente puede acceder un usuario que se haya registrado previamente en la aplicación, y además lo haya hecho como alumno.

En la parte superior de la cabecera se encuentra la barra de navegación con dos enlaces y un menú. `Mis hilos` hace referencia a la misma URL de la página servida, por lo que la recargará. `Desconectar` para el cierre de la sesión (Sección 4.2.4). `Mis diseños` es un menú que muestra todos los estilos personalizados de cada alumno. En la última opción de este menú hay un enlace que hace referencia a la URL de gestión de diseños. En la parte inferior de la cabecera se muestra una foto de perfil y un mensaje que informa a quien pertenece el espacio de la sesión.

Dentro del contenedor hay que destacar dos partes importantes:

- **`Hilos disponibles`**: Aparecen todas las asignaturas disponibles a las que se puede suscribir el alumno. Cada formulario corresponde con una asignatura de la cual se muestra su título y su descripción. El campo del formulario se debe rellenar con un RSS válido antes de enviarlo pulsando sobre el botón `Guardar y añadir` coloreado en naranja. En ese instante, se realiza una petición `POST` sobre la misma URL por lo que será procesada por la misma vista `presentacionalumno`. Esta vista identifica la petición como `POST`. La información es enviada mediante un formulario (*FormularioAgregarRSS*) definido en `formularios.py`. La vista valida el formulario de tal forma que cumpla que la URL del RSS tiene que ser correcta y no estar ocupada por ninguna asignatura de ningún alumno registrado en la aplicación. La vista crea un objeto en la tabla del modelo `RSS` en base de datos con el dato introducido en el formulario y una fecha que marca el inicio de la actividad de ese RSS. Además, la vista crea otro objeto en el modelo `Valoracion` para inicializar la puntuaciń y el nivel de ese alumno en esa asignatura. Por último, se envía una respuesta `HTTP` redireccionando a la página de presentación del alumno. En caso de que el formulario sea inválido, la vista renderizará la plantilla `presentacionalum.html` con un mensaje de error de dato introducido.

- **`Mis hilos`**: La vista busca por el `ID` del alumno y filtra las asignaturas a las que

[4] URL correspondiente a la página de registro de un alumno: `/planetablogs/alumnos/`

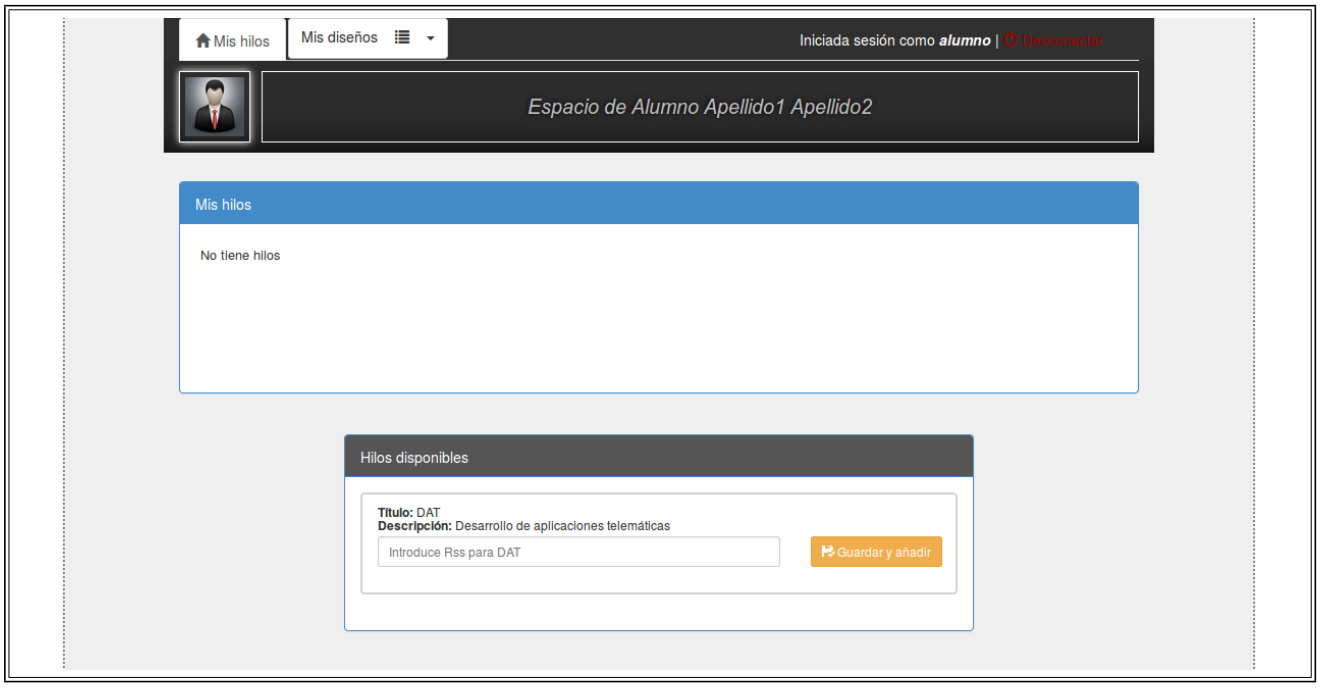


Figura 4.7: *Presentación del perfil de un alumno.*

está suscrito ese alumno para imprimirlas posteriormente a la hora de renderizar la plantilla `presentacionalum.html`.

Personalizar diseño

El documento *HTML* mostrado en la figura 4.8 es la página de gestión de diseños. La página es servida tras una previa petición `GET` del cliente mediante el correspondiente recurso [5]. La vista asociada a ese recurso es `agregardisenoalumno`, la cual devuelve la plantilla `diseno.html` que crea el documento *HTML* de la respuesta.

A esta página solamente puede acceder un usuario que se haya registrado previamente en la aplicación, y además lo haya hecho como alumno.

En la parte superior de la cabecera se encuentra la barra de navegación con dos enlaces: `Mis hilos` hace referencia a la URL del perfil del alumno y `Desconectar` para el cierre de la sesión (Sección 4.2.4).

En el cuerpo del documento hay dos partes diferenciadas:

[5] URL correspondiente a la página de gestionar diseños: `/planetablogs/alumnos/agregardisenoalumno/`

Figura 4.8: *Gestionar diseños personalizados.*

- **Agregar diseño**: Es un formulario con dos campos a introducir por el alumno. Nombre del estilo y un recurso imagen. Aparecen unas recomendaciones sobre la especificaciones del recurso a subir. Una vez relleno, el formulario se envía pulsando sobre el botón "Enviar". En ese momento, se realiza una petición `POST` sobre la misma URL por lo que será procesada por la misma vista `agregardisenoalumno`. Esta vista identifica la petición como `POST`. La información es enviada mediante un formulario de diseño definido en `formularios.py`, y si este formulario es válido, dicha información se almacena. La vista crea un objeto `Diseno` guardándolo en la base de datos. Y posteriormente, envía una respuesta `HTTP` redireccionando a la página de presentación de alumno de la aplicación. En caso de que el formulario sea inválido, la vista renderizará la plantilla `diseno.html`.

- **Eliminar diseño**: Es una lista de todos los diseños creados. Los cuales se pueden borrar pulsando su botón correspondiente.

Presentación de tutor

El documento *HTML* mostrado en la figura 4.9 es la página de presentación del perfil de un tutor. La página es servida tras una previa petición `GET` del cliente mediante el correspondiente recurso [6]. La vista asociada a ese recurso es `presentacionprofesor`, la cual devuelve la plantilla `presentacionprof.html` que crea el documento *HTML* de la respuesta.

A esta página solamente puede acceder un usuario que se haya registrado previamente en la aplicación, y además lo haya hecho como profesor.

En la parte superior de la cabecera se encuentra la barra de navegación con dos enlaces. `Mis hilos` hace referencia a la misma URL de la página servida, por lo que la recargará, `Resetear contraseña` hace referencia a la URL de cambio de contraseña y `Desconectar` para el cierre de la sesión (Sección 4.2.4). En la parte inferior de la cabecera se muestra una foto de perfil y un mensaje que informa a quien pertenece el espacio de la sesión.

Dentro del contenedor hay que destacar tres partes importantes:

- **`Crear hilo`**: Es un botón que despliega un formulario a través de un efecto `toggle` de *jQuery-ui*. Este formulario introduce dos campos, título y descripción, que definen un objeto del modelo `Asignatura`. Se envía pulsando sobre el botón `Enviar` y se realiza una petición `POST` sobre la misma URL por lo que será procesada por la misma vista `presentacionprofesor`. Esta vista identifica la petición como `POST`. La información es enviada mediante un formulario (*FormularioHilo*) definido en `formularios.py`. La vista valida el formulario y crea un objeto en la tabla del modelo `Asignatura` en base de datos con los dos datos introducidos en el formulario y un campo `entradas` que inicializa el número de entradas totales existentes en ese hilo. Se renderiza la plantilla `presentacionprof.html` en la respuesta a la petición la cual redirecciona a la página de presentación del profesor.

- **`Hilos disponibles`**: Aparecen todas las asignaturas disponibles a las que puede seguir el profesor, sin contar las asignaturas agregadas a su sección de `Mis hilos`.

- **`Mis hilos`**: La vista busca por el `ID` del profesor y filtra las asignaturas a las que

[6] URL correspondiente a la página de registro de un alumno: `/planetablogs/tutores/`

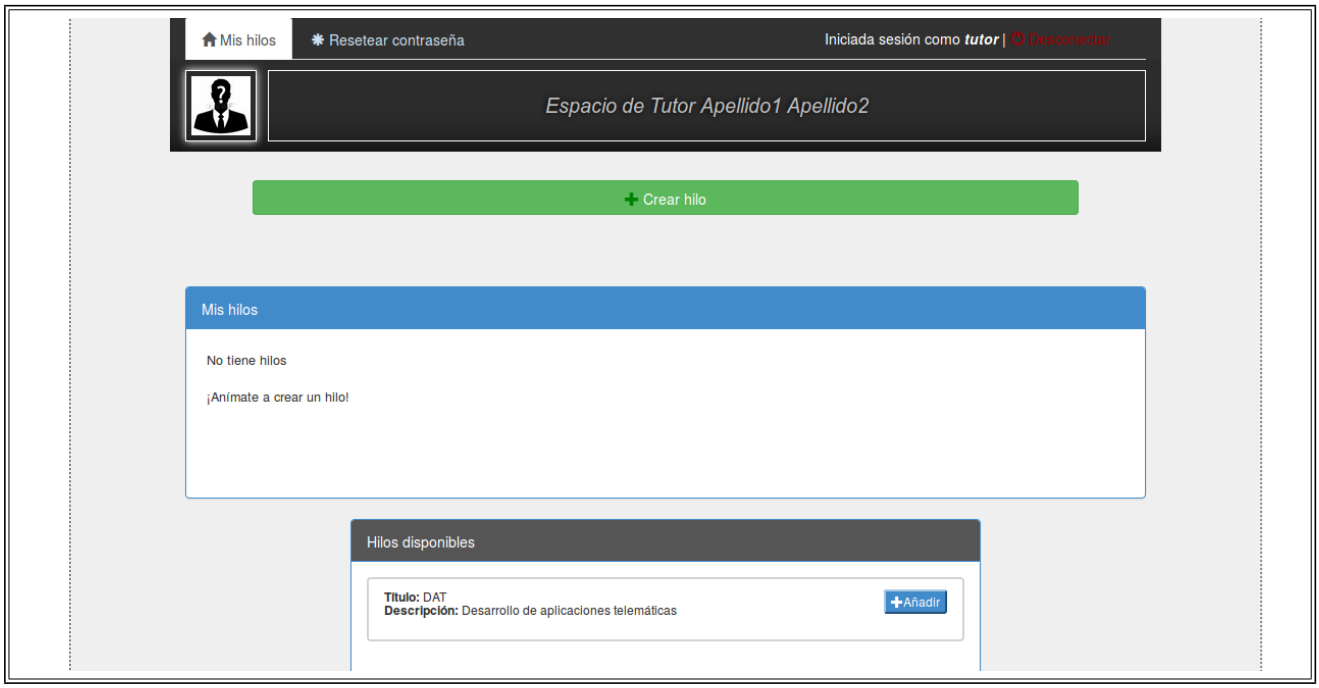


Figura 4.9: *Presentación del perfil de un tutor.*

sigue ese profesor para imprimirlas posteriormente a la hora de renderizar la plantilla `presentacionprof.html`.

Resetear contraseña

El documento *HTML* mostrado en la figura 4.10 es la página de modificación de contraseña. La página es servida tras una previa petición `GET` del cliente mediante el correspondiente recurso [7]. La vista asociada a ese recurso es `resetear_password`, la cual devuelve la plantilla `resetear_password.html` que crea el documento *HTML* de la respuesta.

A esta página solamente puede acceder un usuario que se haya registrado previamente en la aplicación, y además lo haya hecho como profesor.

En la parte superior de la cabecera se encuentra la barra de navegación con tres enlaces. `Resetear contraseña` hace referencia a la misma URL de la página servida, por lo que la recargará, `Mis hilos` hace referencia a la URL del perfil del tutor y `Desconectar` para el cierre de la sesión (Sección 4.2.4).

[7] URL correspondiente a la página de resetear contraseña: `/planetablogs/tutores/resetear_password/`

Figura 4.10: *Modificar una contraseña desde el perfil de un tutor.*

En el cuerpo del documento existe un formulario en el que el alumno que haya perdido u olvidado su contraseña, puede introducir su nick de usuario y una nueva contraseña que sustituirá a la anterior.

Hilos

En la figura 4.23 se muestran dos documentos *HTML* que recogen el contenido de un hilo. En la figura 4.23(a) se visualiza el interior de un hilo limpio, sin entradas, en la parte del alumno. En la figura 4.23(b) se visualiza el interior del mismo hilo limpio en la parte del tutor. La diferencia entre ambas imágenes se explicará en las dos próximas subsecciones.

La cabecera de ambas imágenes se pueden observar los distintos ítems de la barra de navegación. La barra de navegación será la misma dentro de todas las plantillas con la siguiente URL:

`/planetablogs/alumnos/hilo/(IdAsignatura)/[(buscar),(puntuaciones),(infopuntuaciones),(estadisticas),(ayuda)].`

- ***Mis hilos***: Es un enlace que hace referencia a la URL de la presentación del perfil de usuario (Sección 4.2.4).

- ***Asignatura***: Enlace que hace referencia a la misma URL de la página servida, por lo que la recargará (en este caso, la asignatura se llama `DAT`).

(a) Hilo limpio en alumno (b) Hilo limpio en tutor

Figura 4.11: *Página principal de un hilo limpio.*

- ***Buscar***: Es un enlace que hace referencia a la URL de búsqueda (Sección 4.2.4).
- ***Puntuaciones***: Es una lista desplegable compuesta por tres enlaces. *Ranking* hace referencia a la URL donde se muestran todas las puntuaciones de los usuarios de un determinado hilo. *Estadísticas* hace referencia a la URL donde se muestran las estadísticas de cada usuario de un determinado hilo, tanto estadísticas generales como resumen de entradas. *Información* hace referencia a la URL que recoge las bases de esas puntuaciones.
- ***Ayuda***: Es un enlace que hace referencia a la URL de ayuda (Sección 4.2.4).
- ***Desconectar***: Enlace que hace referencia a la URL servida para el cierre de la sesión (Sección 4.2.4).

También se expone, debajo de la barra de navegación, una imagen correspondiente al logo de la aplicación con su título y una frase aleatoria. Y en la parte derecha un apartado que agrupa las entradas más valoradas del hilo.

Hilo de alumno

El documento *HTML* mostrado en la figura 4.12 es el contenedor de la página de un determinado hilo al que está suscrito un alumno. La página es servida tras una previa petición `GET` del cliente mediante el correspondiente recurso [8]. La vista asociada a ese recurso es `mostrarhiloalumno`, la cual devuelve la plantilla `index.html` que crea el documento *HTML* de la respuesta.

[8] URL correspondiente al hilo al que está suscrito un alumno: `/planetablogs/alumnos/hilo/(IdAsignatura)`

A esta página solamente puede acceder un usuario que se haya registrado previamente en la aplicación como alumno y que esté suscrito a la correspondiente asignatura de interés.

La vista filtra por el usuario conectado y por el Id de la asignatura pasada como argumento en el recurso de la petición, para obtener la lista de usuarios, la lista de comentarios, la lista de entradas más valoradas, y la lista de entradas pertenecientes a esa asignatura. El contenedor de la plantilla expone la lista de entradas tal y como se muestra en la figura. En la parte derecha del contenedor se reúne la lista de usuarios suscritos a este hilo. Cabe destacar que la lista de entradas dentro del contenedor está limitada por un paginador creado por la vista. Cada página dentro del hilo mostrará en su contenedor cinco entradas como máximo. El orden de la lista de entradas está colocado de menor a mayor por antiguedad, siendo las primeras entradas visibles las más recientes.

La respuesta de la vista también contiene siete estructuras *JSON*, todos los usuarios del modelo `User` (*json_usuarios*), todas las valoraciones del modelo `Valoracion` pertenecientes a ese hilo (*json_valoracion*), todos los alumnos del modelo `Alumno` (*json_alumnos*), todas las asignaturas del modelo `Asignatura` (*json_asignaturas*), todos los profesores del modelo `Profesor` (*json_profesores*), todos los diseños del modelo `Diseno` correspondientes al usuario de sesión (*json_disenos*) y a su vez el usuario de sesión (*json_usuarios*).

Hilo de tutor

El documento *HTML* mostrado en la figura 4.13 es el contenedor de la página de un determinado hilo al que está suscrito un tutor. La página es servida tras una previa petición `GET` del cliente mediante el correspondiente recurso [9]. La vista asociada a ese recurso es `mostrarhiloprofesor`, la cual devuelve la plantilla `index_tutor.html` que crea el documento *HTML* de la respuesta.

A esta página solamente puede acceder un usuario que se haya registrado previamente en la aplicación como profesor y que esté suscrito a la correspondiente asignatura de interés.

La vista filtra por el Id de la asignatura pasada como argumento en el recurso de la petición, para obtener la lista de usuarios, la lista de comentarios, la lista de entradas más valoradas, y la lista de entradas pertenecientes a esa asignatura. El contenedor de la plantilla expone la lista de entradas tal y como se muestra en la figura. En la parte derecha del contenedor se reúne la lista

[9] URL correspondiente al hilo al que está suscrito un tutor: `/planetablogs/tutores/hilo/(IdAsignatura)`

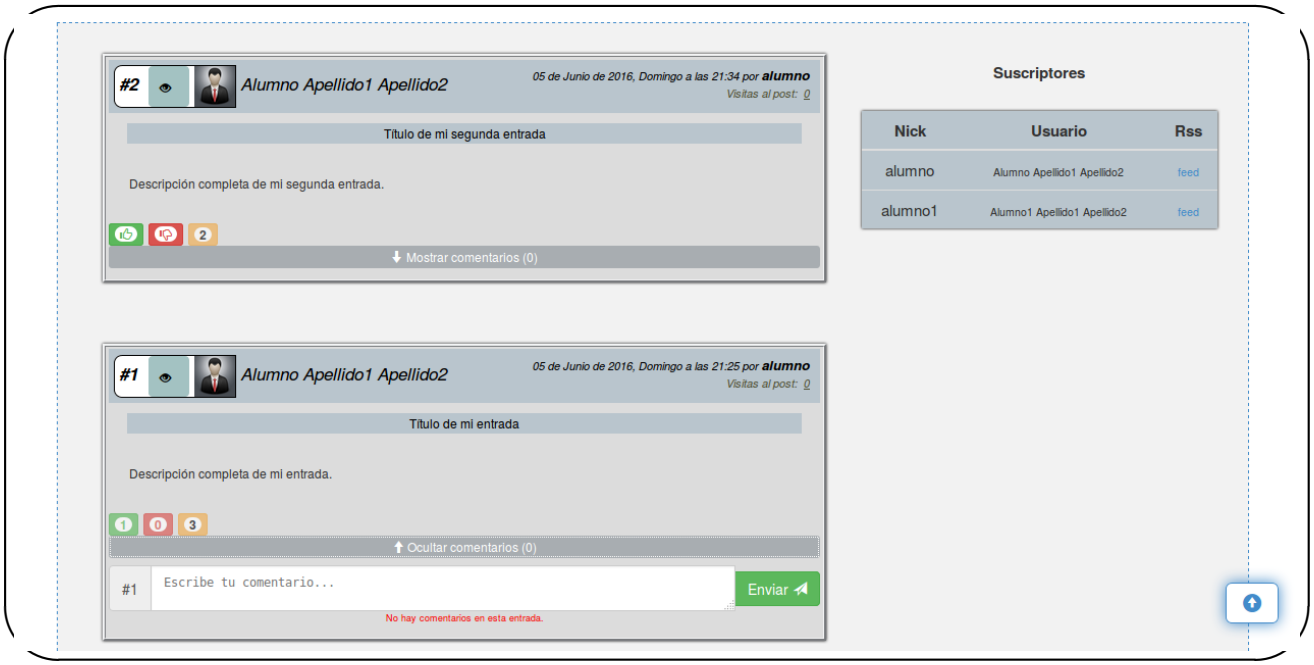


Figura 4.12: *Página principal de un hilo de un alumno.*

de usuarios suscritos a este hilo. Cabe destacar que la lista de entradas dentro del contenedor está limitada por un paginador creado por la vista. Cada página dentro del hilo mostrará en su contenedor cinco entradas como máximo. El orden de la lista de entradas está colocado de menor a mayor por antiguedad, siendo las primeras entradas visibles las más recientes.

La respuesta de la vista también contiene cinco estructuras *JSON*, todos los usuarios del modelo `User` (*json_usuarios*), todas las valoraciones del modelo `Valoracion` pertenecientes a ese hilo (*json_valoracion*), todos los alumnos del modelo `Alumno` (*json_alumnos*), todas las asignaturas del modelo `Asignatura` (*json_asignaturas*), todos los profesores del modelo `Profesor` (*json_profesores*).

La principal diferencia con las páginas de los hilos mostrados por parte de los alumnos, es que los tutores tienen el permiso de eliminar entradas de la lista de entradas o usuarios de la lista de usuarios, y además pueden puntuar de forma cuantitativa cada entrada.

Búsqueda avanzada

El documento *HTML* mostrado en la figura 4.14 pertenece la página de búsqueda avanzada de entradas. La página es servida tras una previa petición `GET` del cliente mediante el corres-

Figura 4.13: *Página principal de un hilo de un tutor.*

pondiente recurso [10]. La vista asociada a ese recurso es `buscar`, la cual devuelve la plantilla `buscar.html` que crea el documento *HTML* de la respuesta.

A esta página solamente puede acceder un usuario que se haya registrado previamente en la aplicación y que esté suscrito a la correspondiente asignatura de interés.

La cabecera reúne todos los ítems de la barra de navegación explicada en la sección 4.2.4. La parte del documento de la clase *.container* crea un formulario con una etiqueta `<select>` la cual despliega dos opciones de búsqueda a elegir, por *Nick de usuario* y por *Id de entrada*.

Ranking

El documento *HTML* mostrado en la figura 4.15 corresponde con la página de ranking de un determinado hilo. La página es servida tras una previa petición `GET` del cliente mediante el correspondiente recurso [11]. La vista asociada a ese recurso es `puntuaciones`, la cual devuelve la plantilla `puntuaciones.html` que crea el documento *HTML* de la respuesta.

A esta página solamente puede acceder un usuario que se haya registrado previamente en la

[10] URL correspondiente a la página de búsqueda: `/planetablogs/alumnos/hilo/(IdAsignatura)/buscar/`
[11] URL correspondiente a la página de ranking: `/planetablogs/alumnos/hilo/(IdAsignatura)/puntuaciones/`

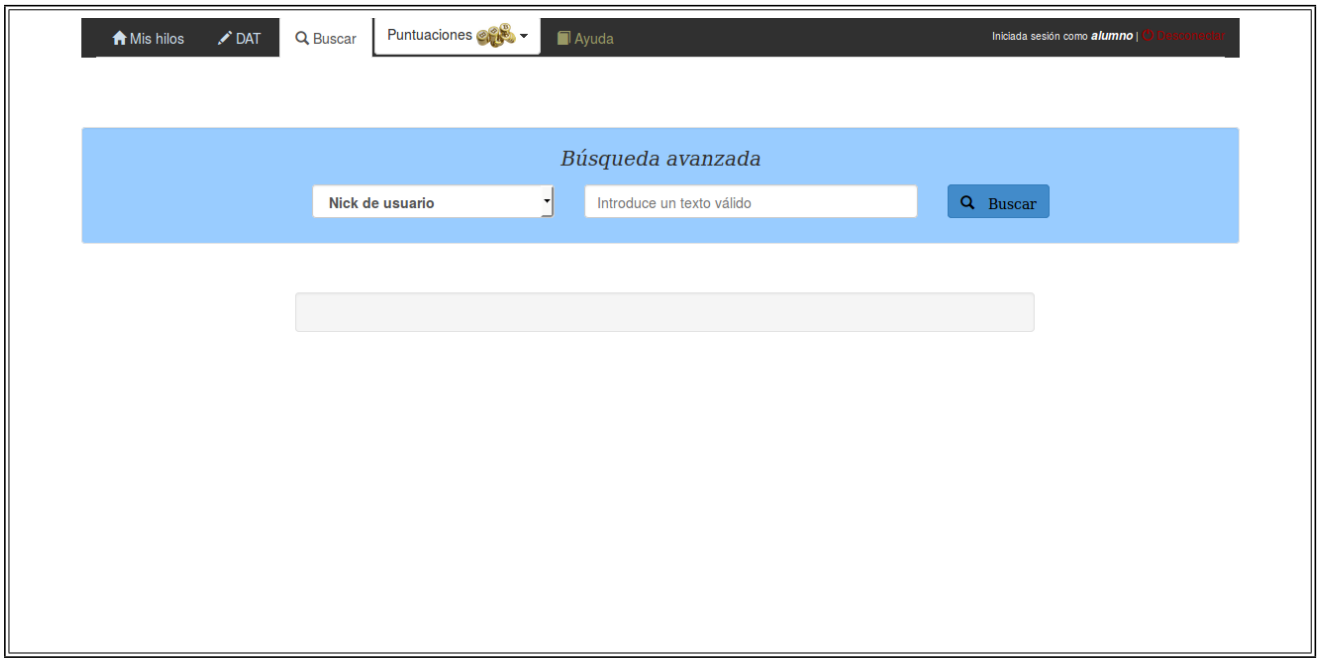


Figura 4.14: *Página de búsqueda avanzada de entradas.*

aplicación y que esté suscrito a la correspondiente asignatura de interés.

La cabecera reúne todos los ítems de la barra de navegación explicada en la sección 4.2.4.

La vista `puntuaciones` recibe como parámetro el Id del hilo, y mediante ese Id procesa la petición obteniendo la asignatura y la lista de valoraciones asociadas a esa asignatura. Serializa una lista *JSON* con todos los objetos del modelo `User` (*json_usuarios*). Toda esa información es enviada como respuesta a la solicitud en la plantilla `puntuaciones.html`.

Por lo que, la parte del documento *HTML* de la clase *.container* agrupa todos los usuarios de una asignatura en una tabla *HTML* (etiqueta `<table>`) con varios elementos de información: *Posición*, *Usuario*, *Planets* y *Nivel*.

Estadísticas

El documento *HTML* mostrado en la figura 4.16 corresponde con la página de estadísticas de un determinado hilo. La página es servida tras una previa petición `GET` del cliente mediante el correspondiente recurso [12]. La vista asociada a ese recurso es `estadisticas`, la cual devuelve la plantilla `estadisticas.html` que crea el documento *HTML* de la respuesta.

[12]URL correspondiente a la página de ranking: `/planetablogs/alumnos/hilo/(IdAsignatura)/estadisticas/`

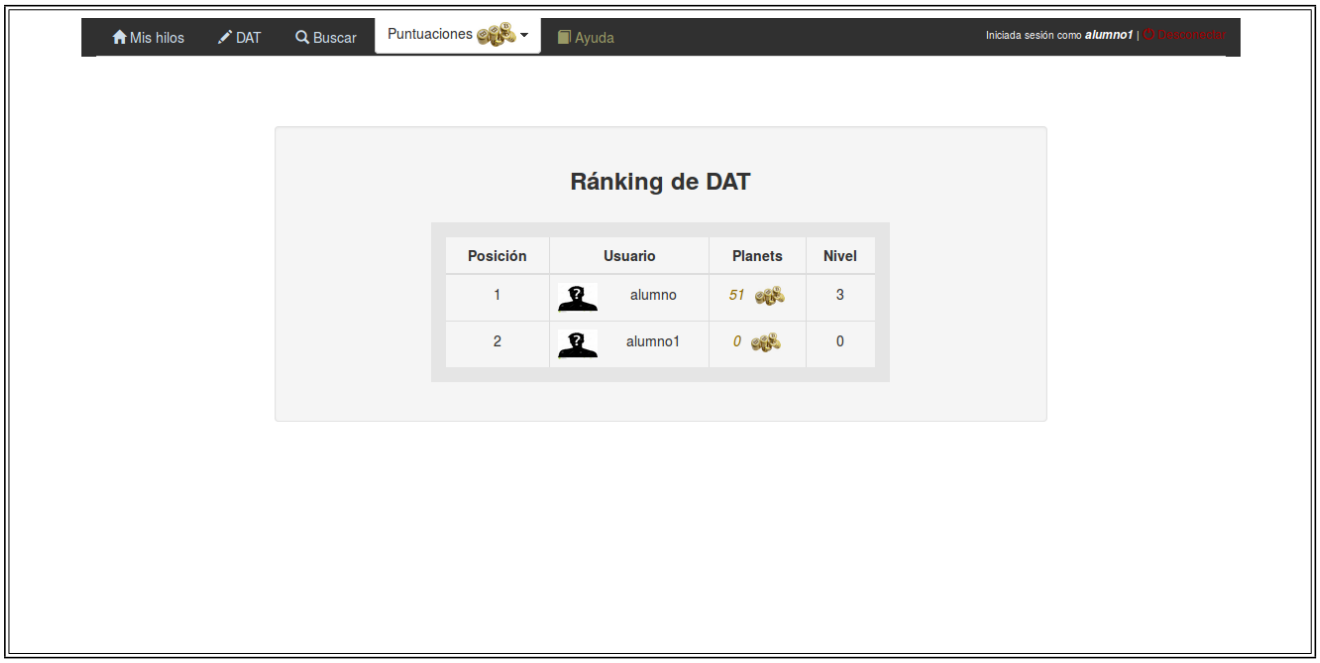


Figura 4.15: *Página de ranking.*

A esta página solamente puede acceder un usuario que se haya registrado previamente en la aplicación y que esté suscrito a la correspondiente asignatura de interés.

La cabecera reúne todos los ítems de la barra de navegación explicada en la sección 4.2.4.

La vista `estadisticas` recibe como parámetro el Id del hilo, y mediante ese Id procesa la petición obteniendo la asignatura, la lista de entradas asociadas a esa asignatura y una lista de listas que recoge información de cada alumno (datos personales, total de entradas, total de up recibidos, total de down recibidos, total up enviados, total down enviados, total comentarios recibidos y total comentarios enviados).

Toda esa información es enviada como respuesta a la solicitud en la plantilla `estadisticas.html`.

Por lo que, la parte del documento *HTML* de la clase *.container* agrupa todos los usuarioS de una asignatura en un acordeón (elemento *jQuery-ui*). Cada sección del acordeón pertenece a un usuario y muestra dos tablas (elemento `<table>`): *Estadísticas generales del alumno* y *Entradas del alumno*, con información recopilada y organizada.

En la parte superior del cuerpo de la página, mediante *radiobuttons* se da la opción de elegir lo que queremos mostrar. Al clicar sobre cada radiobutton, mediante los eventos *jQuery*, *hide()* y *show()*, escritos en el fichero estadisticas.js, se oculta o se muestra la tabla de estadísticas de

Figura 4.16: *Página de ranking.*

usuario, la tabla de resumen de entradas o ambas tablas.

Información de las puntuaciones

El documento *HTML* mostrado en la figura 4.17 muestra información sobre las condiciones o bases de las puntuaciones. La página es servida tras una previa petición `GET` del cliente mediante el correspondiente recurso [13]. La vista asociada a ese recurso es `infopuntuaciones`, la cual devuelve la plantilla `infopuntuaciones.html` que crea el documento *HTML* de la respuesta.

A esta página solamente puede acceder un usuario que se haya registrado previamente en la aplicación.

La cabecera reúne todos los ítems de la barra de navegación explicada en la sección 4.2.4.

La vista `puntuaciones` recibe como parámetro el Id del hilo, y mediante ese Id procesa la petición obteniendo el objeto completo de dicha asignatura. El contenedor de la plantilla reúne una serie de condiciones de texto plano mediante una lista numérica (etiqueta `<ol>`) y

[13] URL correspondiente a la página de bases de las puntuaciones:
`/planetablogs/alumnos/hilo/(IdAsignatura)/infopuntuaciones/`

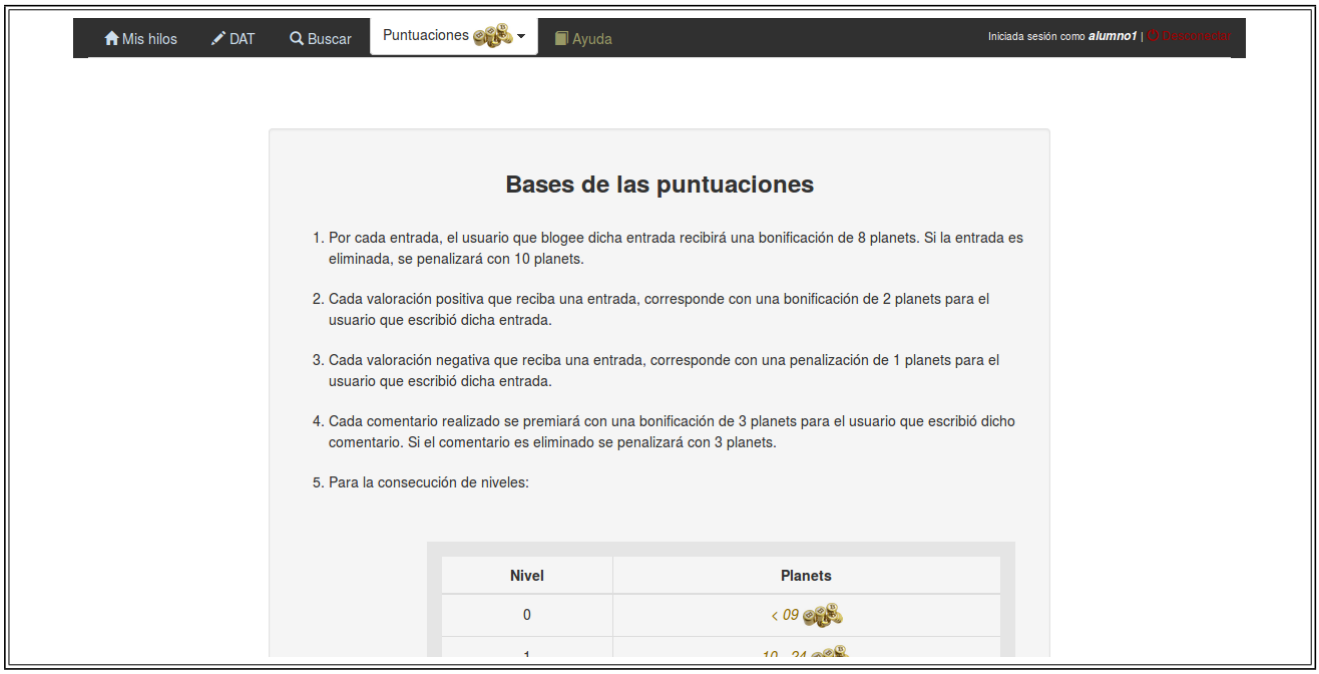


Figura 4.17: *Página con las bases de las puntuaciones.*

en la parte inferior muestra una tabla *HTML* que resume la consecución de niveles con respecto a la puntuación obtenida. Al pasar el ratón por encima de un nivel, y mediante los eventos *JavaScript*, `mouseover` y `mouseout` del fichero `infopuntuaciones.js`, se añade (*.addClass*) y se remueve (*.removeClass*) una clase *CSS* que colorea según el nivel señalado.

Ayuda

El documento *HTML* mostrado en la figura 4.18 muestra información adicional sobre el uso de la aplicación. La página es servida tras una previa petición `GET` del cliente mediante el correspondiente recurso [14]. La vista asociada a ese recurso es `ayuda`, la cual devuelve la plantilla `ayuda.html` que crea el documento *HTML* de la respuesta.

A esta página solamente puede acceder un usuario que se haya registrado previamente en la aplicación.

La cabecera reúne todos los ítems de la barra de navegación explicada en la sección 4.2.4.

La vista `ayuda` recibe como parámetro el Id del hilo, y mediante ese Id procesa la petición

[14]URL correspondiente a la página de ayuda:
`/planetablogs/alumnos/hilo/(IdAsignatura)/ayuda/`

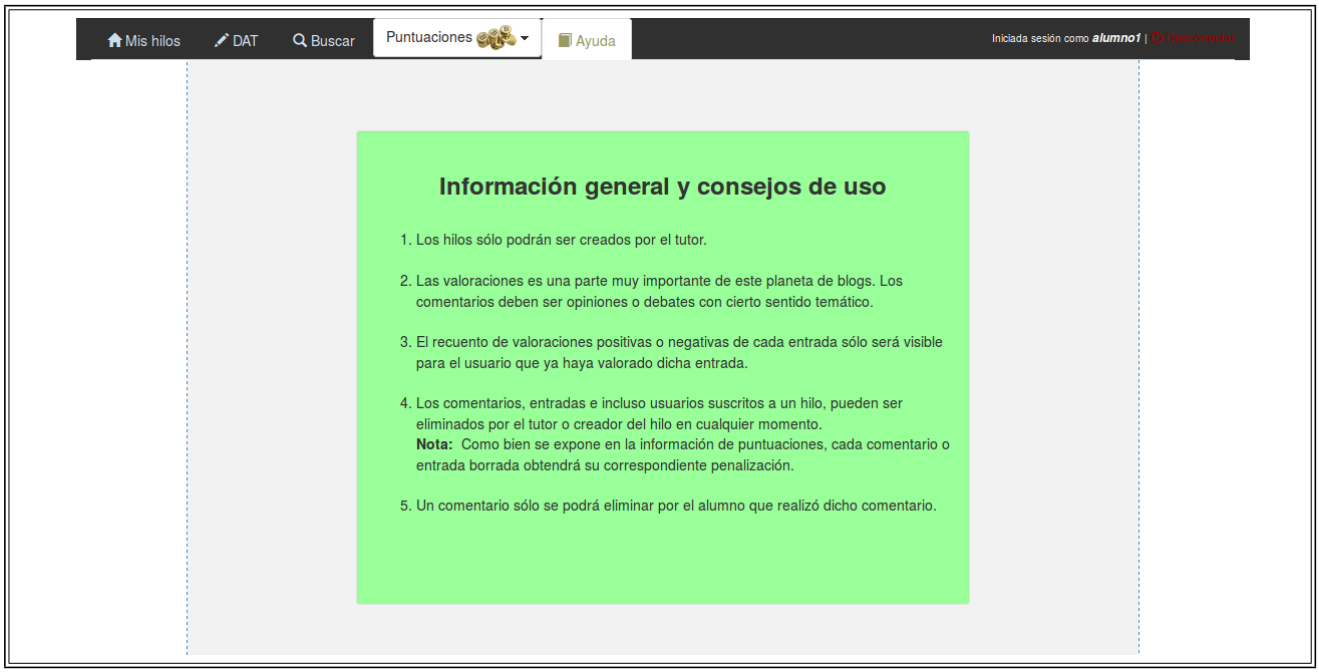


Figura 4.18: *Página con las bases de las puntuaciones.*

obteniendo el objeto completo de dicha asignatura. El contenedor de la plantilla reúne una serie de información y sugerencias de texto plano mediante dos listas numéricas (etiqueta `<ol>`).

Vistas con AJAX

Hasta ahora hemos visto todas las vistas que procesaban peticiones con necesidad de renderizar una plantilla como respuesta a esas solicitudes. Con *AJAX* se han definido numerables vistas para intercambio de datos entre cliente y servidor sin necesidad de recargar la página, simplemente enviando como respuesta el cacho de *HTML* que se necesita.

Las vistas implementadas con *AJAX* en el fichero `views.py` de este proyecto son las siguientes:

- ***Eliminar hilo en presentación de alumno***: En la sección donde se ubican *Mis hilos*, cada hilo de la lista mostrada tiene dos botones a su derecha. El botón `Eliminar` mediante el evento *onClick* llama a la función *JavaScript* `EliminarHilo` del fichero `presentacionalum.js`. Esta función lanza un diálogo definido por *BootStrap* que da la opción de eliminar el hilo o cancelar dicha acción tal y como se muestra en la figura 4.19.

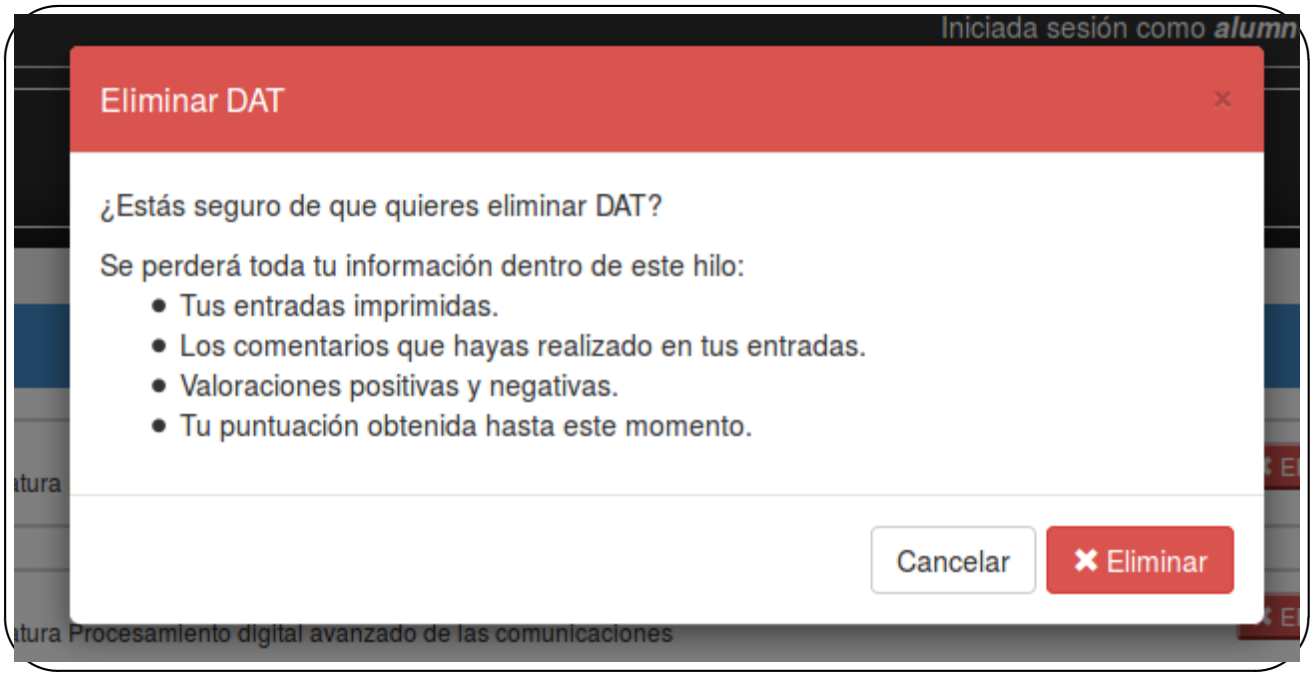


Figura 4.19: *Eliminar hilo en una presentación de alumno.*

Al hacer click sobre el botón `Eliminar` del cuadro de diálogo, se realiza una petición *AJAX* de tipo `GET` a la vista `eliminarasignaturaalumno`. La petición es enviada desde el fichero `presentacionalum.js` usando *jQuery* e incluye como dato el Id de la asignatura. Una vez llega la solicitud al servidor, la vista filtra por medio de ese Id y del Id del alumno en los modelos `RSS`, `Valoracion` y `Entrada`, para realizar un borrado (`.delete`) de todos los objetos que ofrecen alguna dependencia con el hilo. La función *AJAX* no recibe ningún dato como respuesta.

- ***Agregar hilo en presentación de tutor***: En *Hilos disponibles*, cada hilo de la lista ofrece la posibilidad de pinchar sobre un botón `Añadir` de color azul. Mediante el evento *onClick* llama a la función *JavaScript* `Agregar` del fichero `presentacionprof.js`. Esta función realiza una petición *AJAX* de tipo `GET` a la vista `agregarasignaturaprofesor`. La petición es enviada desde el fichero `presentacionprof.js` usando *jQuery* e incluye como dato el Id de la asignatura. La vista filtra por el parámetro recibido en el modelo `Asignatura`. Una vez consigue el hilo, asocia al usuario que pulsó el botón añadir a ese hilo. La función *AJAX* recibe una respuesta con código *HTML*. Mediante la función *jQuery* `.prepend` en la presentación se agrega el hilo en *Mis hilos*, mientras

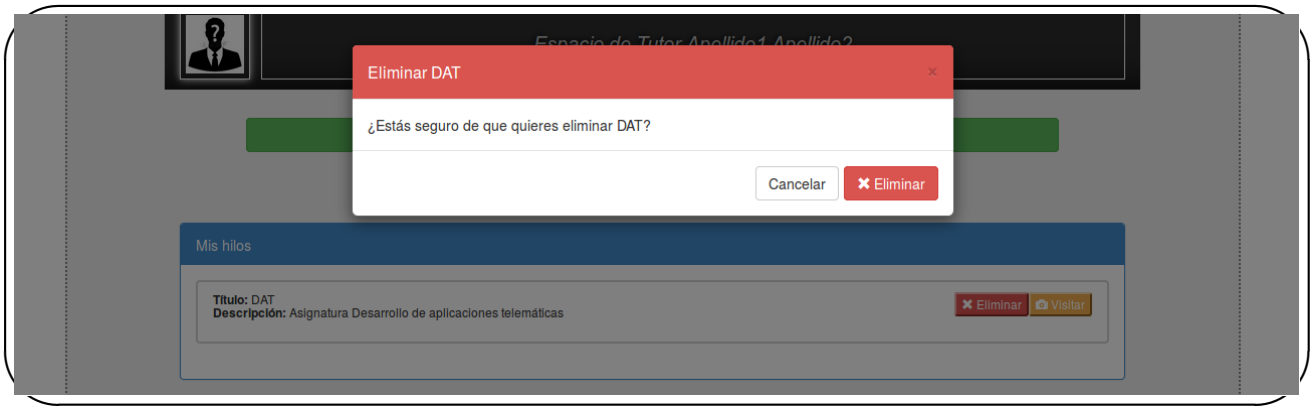


Figura 4.20: *Eliminar hilo en una presentación de tutor.*

que paralelamente con la función `.hide` de *jQuery-ui* se elimina de *Hilos disponibles*.

- ***Eliminar hilo en presentación de tutor***: En la sección donde se ubican *Mis hilos*, cada hilo de la lista mostrada tiene dos botones a su derecha. El botón `Eliminar` mediante el evento *onClick* llama a la función *JavaScript* `EliminarHilo` del fichero `presentacionprof.js`. Esta función lanza un diálogo definido por *BootStrap* que da la opción de eliminar el hilo o cancelar dicha acción tal y como se muestra en la figura 4.20.

  Al hacer click sobre el botón `Eliminar` del cuadro de diálogo, se realiza una petición *AJAX* de tipo `GET` a la vista `eliminarasignaturaprofesor`. La petición es enviada desde el fichero `presentacionprof.js` usando *jQuery* e incluye como dato el Id de la asignatura. Una vez llega la solicitud al servidor, la vista filtra por medio de ese Id y remueve el objeto con Id del tutor en el modelo `Asignatura` mediante el método *.remove*. La función *AJAX* recibe una respuesta con código *HTML*. Mediante la función *jQuery* `.prepend` en la presentación se agrega el hilo en *Hilos disponibles*, mientras que paralelamente con la función `.hide` de *jQuery-ui* se elimina de *Mis hilos*.

- ***Agregar comentario en hilo de alumno***: Las entradas de un hilo dan la posibilidad de comentar esa entrada mediante un texto introducido en una etiqueta `<textarea>` y un botón `Enviar` como se muestra en la figura 4.21.

  El botón `Enviar` mediante el evento *onClick* llama a la función *JavaScript* `AgregarComentario` del fichero `index.js`. Realiza una petición *AJAX* de tipo `GET`

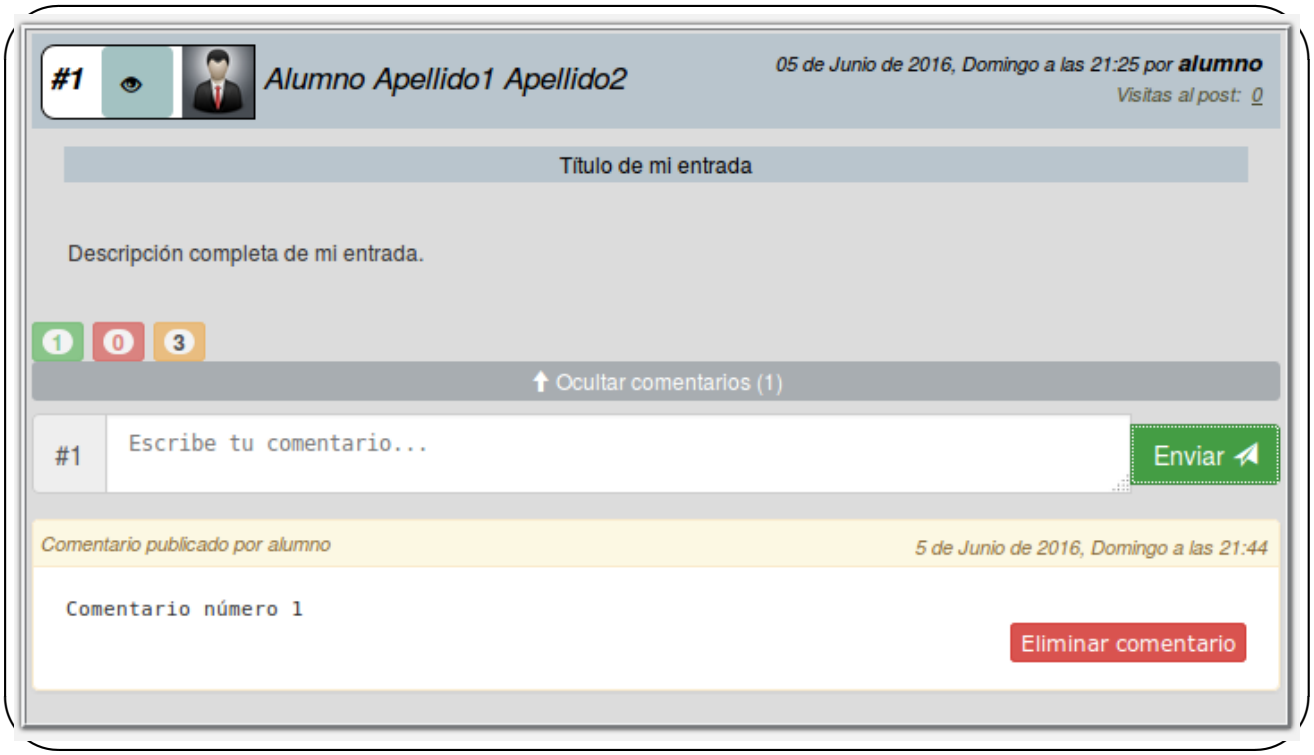


Figura 4.21: *Comentario agregado en una entrada.*

a la vista `agregarcomentario`. La petición es enviada desde el fichero `index.js` usando *jQuery* e incluye como datos el Id de la asignatura, el Id de la entrada, el Id del usuario y la descripción. Una vez llega la solicitud al servidor, si la descripción es distinto de *null* (es decir, se ha escrito algún comentario), la vista crea un objeto `Comentario` y lo guarda en base de datos. Suma la valoración del comentario al alumno de la asignatura correspondiente y actualiza su nivel. Por último, la vista declara datos a través de un elemento que incluye dos *JSON* (`json_usuario` y `json_comentario`) que se traspasarán por medio de la respuesta `HTTP` al cliente. La función *AJAX* recibe una respuesta con código *HTML* y datos *JSON*. Mediante la función *jQuery* `.append` en la plantilla se plasma el comentario en la parte inferior de la entrada.

- ***Eliminar comentario en hilo de alumno o tutor***: Las entradas de un hilo dan la posibilidad de eliminar comentarios sobre esa entrada al pulsar el botón `Eliminar comentario` que mediante el evento *onClick* llama a la función *JavaScript* `EliminarComentario` del fichero `index.js`.

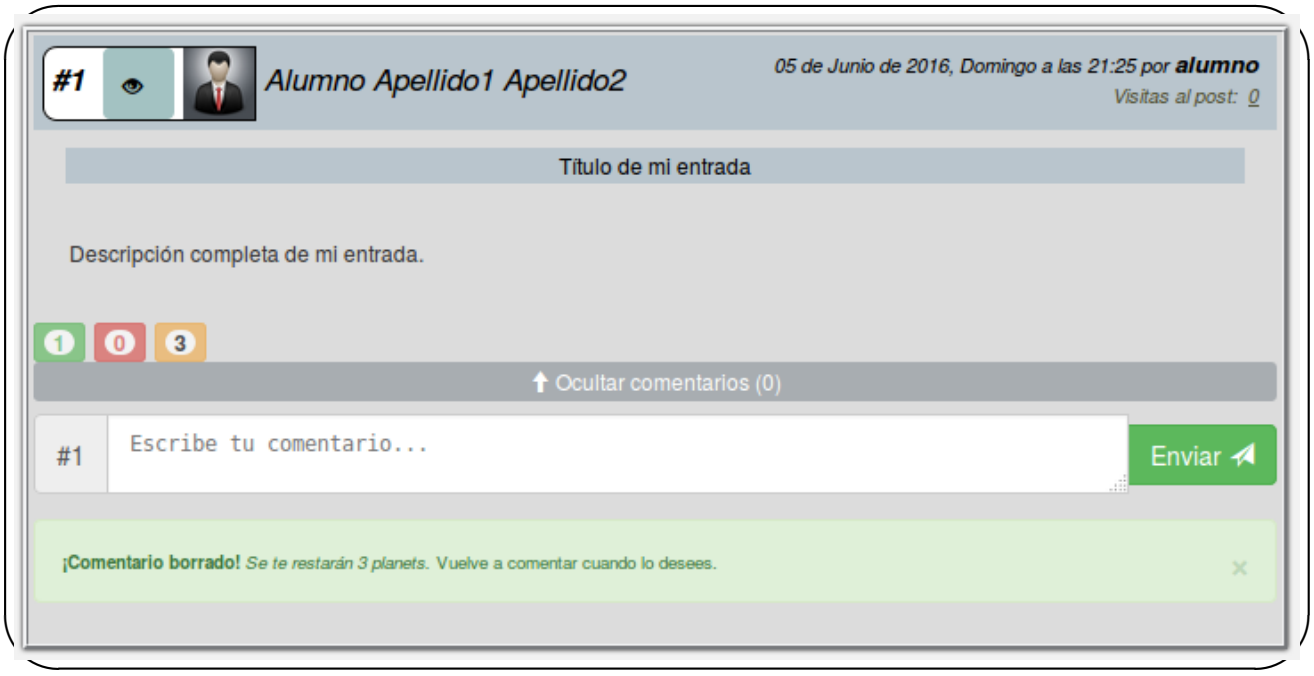


Figura 4.22: *Comentario eliminado en una entrada.*

La petición *AJAX* de tipo `GET` se vincula con la vista `eliminarcomentario`. La solicitud es enviada desde el fichero `index.js` usando *jQuery* e incluye como datos el Id de la asignatura y el Id del comentario. El servidor procesa la solicitud obteniendo el objeto `Comentario` a borrar. Cuando lo obtiene realiza un borrado con el método *.delete* y resta la puntuación del objeto `Valoracion` perteneciente al usuario que elimina el comentario y actualiza su nivel. La función *AJAX* recibe una respuesta con código *HTML*. Mediante la función *jQuery* `.remove` suprime el comentario de la plantilla y con la función `.show` muestra un mensaje de información de borrado como se ve en la figura 4.22.

- ***Valoración positiva y negativa de una entrada***: Dentro de cada entrada se agrupan dos botones, de color verde para valoraciones positivas y de color rojo para valoraciones negativas (figura 4.23(a)). Son enlaces *hyperlink* que mediante el evento *onClick* llaman a las funciones *JavaScript* `Up` o `Down` respectivamente. Estas funciones se definen en el fichero `index.js`.

  Ambas funciones envían una petición *AJAX* de tipo `GET` a su correspondiente vista mediante su recurso o URL. La petición es enviada desde el fichero `index.js` usando

(a) Botones sin valoración (b) Botones con valoración

Figura 4.23: *Valoración de una entrada.*

jQuery e incluye como datos el Id de la asignatura, el Id del alumno, el Id de la entrada y el entero Up o Down (valor del botón antes de enviar la petición).

La vista `up` procesará una valoración positiva. La vista crea un objeto en el modelo `Up` con los parámetros pasados en la solicitud. Guarda el objeto y modifica la entrada del modelo `Entrada` en la que se efectuó el click del botón positivo. En el objeto `Entrada` se cambia el campo *total* (suma dos puntos que es la puntuación que se bonifica al hacer una valoración positiva) y el campo *totalup* (indica el total de valoraciones positivas que ha recibido esa entrada). Por último, la vista modifica el objeto `Valoracion` sumando esos dos puntos y actualizando el nivel al usuario.

La vista `down` procesará una valoración negativa. La vista crea un objeto en el modelo `Down` con los parámetros pasados en la solicitud. Guarda el objeto y modifica la entrada del modelo `Entrada` en la que se efectuó el click del botón negativo. En el objeto `Entrada` se cambia el campo *total* (resta un punto que es la puntuación que se penaliza al hacer una valoración negativa) y el campo *totaldown* (indica el total de valoraciones negativas que ha recibido esa entrada). Por último, la vista modifica el objeto `Valoracion` restando ese punto y actualizando el nivel al usuario.

La función *AJAX* recibe una respuesta con código *HTML*. Mediante la función *jQuery* `.html` en la entrada del documento *HTML* se crean dos botones deshabilitados con la cantidad total de valoraciones positivas y negativas que ha recibido (figura 4.23(b))

- ***Pulsar en botón de entrada leída o en link de título de la entrada***: A la derecha de cada id de entrada, hay un botón con un icono que simboliza un ojo tachado, tal y como se puede ver en la figura 4.24. Tanto si se presiona ese botón o si pulsa el link del título de la entrada, se llamará a la función *JavaScript* `EntradaLeida` mediante el evento *onClick*.

(a) Entrada leída. (b) Entrada no leída.

Figura 4.24: *Lectura de una entrada.*

Esta función se define en el fichero `index.js`.

Dicha función envía una petición *AJAX* de tipo `GET` a su correspondiente vista mediante su recurso o URL. La petición es enviada desde el fichero `index.js` usando *jQuery* e incluye como datos el Id de la asignatura, el Id del usuario, el Id de la entrada y un booleano (indica si la entrada ha sido visitada o no).

La vista `entradaleida` creará un objeto en el modelo `Extra` con los parámetros pasados en la solicitud. Guarda el objeto y modifica la entrada del modelo `Entrada` en la que se efectuó el click del botón "LeÃdo". En el objeto `Entrada` se cambia el campo *visitantes* (en caso de que el booleano sea true, es decir, se haya visitado el post al acceder mediante el título de la entrada) y el campo *visitas* (indica el total de visitas que ha recibido esa entrada). Por último, la vista modifica el objeto `Extra` buscando el registro correspondiente a esa entrada y esa asignatura, comprueba si el usuario se encuentra en la lista de ids del campo `Leido` y si no se encuentra, se añade el id del usuario.

La función *AJAX* recibe una respuesta con código *HTML*. Mediante la función *jQuery* `.html` en la entrada del documento *HTML* se crea un botón deshabilitado y se modifica la cantidad total de visitas que ha recibido.

- ***Mostrar visitantes***: Al pulsar sobre el número subrayado de visitas totales de una entrada (figura 4.25), se llamará a la función *JavaScript* `MostrarVisitasUsuarios` mediante el evento *click* de *jQuery*. Esta función se define en el fichero `index.js`.

  Dicha función envía una petición *AJAX* de tipo `GET` a su correspondiente vista mediante su recurso o URL. La petición es enviada desde el fichero `index.js` usando *jQuery* e incluye como dato el Id de la entrada.

  La vista `mostrarvisitasusuarios` envía como respuesta a la función *AJAX* una

(a) Lista de visitantes no vacía. (b) Lista de visitantes vacía.

Figura 4.25: *Visitantes de una entrada.*

lista de los nombres de los visitantes almacenados en el campo `visitantes` de la entrada. Mediante la función *jQuery* `.html` en la entrada del documento *HTML* se crea una tabla con los nombres de dichos visitantes y se visualizarán en un popup emergente. En caso de que la lista esté vacía se mostrará un mensaje "Esta entrada no tiene visitantes".

Si se vuelve a pulsar sobre el número subrayado de visitas totales de dicha entrada, el popup se cerrará.

- ***Eliminar entrada en hilo de tutor***: Un tutor puede eliminar las entradas de un hilo escritas por un alumno a través del botón `Eliminar entrada`, tanto entradas ya valoradas (figura 4.26(a)) como entradas no valoradas (figura 4.26(b)). Al pulsar sobre el botón el evento *onClick* llama a la función *JavaScript* `EliminarEntradaPopUp` del fichero `index_tutor.js`. Esta función lanza un diálogo definido por *BootStrap* que da la opción de eliminar la entrada o cancelar dicha acción tal y como se muestra en la figura 4.26(c).

  Al hacer click sobre el botón `Eliminar` del cuadro de diálogo, se realiza una petición *AJAX* de tipo `GET` a la vista `eliminarentrada`. La petición es enviada desde el fichero `index_tutor.js` usando *jQuery* e incluye como datos el Id de la asignatura y el Id de la entrada. La solicitud es aceptada en el servidor. La vista remueve el objeto `Entrada` mediante el método *.remove* y elimina las valoraciones negativas o positivas de los modelos `Up` y `Down`. Además realiza un recálculo del objeto `Valoracion` al suprimir estas valoraciones y actualiza el nivel del alumno.

  La función *AJAX* no recibe ningún dato como respuesta.

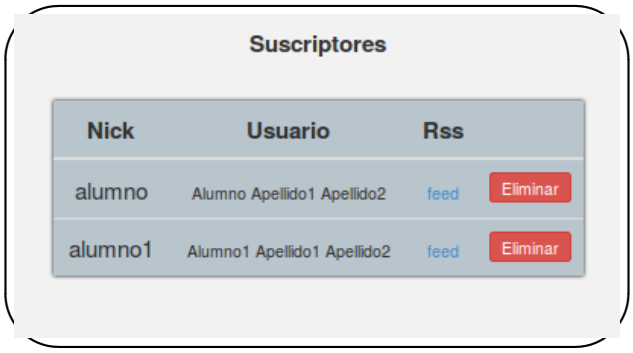
- ***Eliminar alumno en hilo de tutor***: Un tutor puede eliminar los alumnos suscritos a un hilo a través del botón `Eliminar` (figura 4.27(a)). Al pulsar sobre el botón el evento *onClick* llama a la función *JavaScript* `EliminarAlumnoPopUp` del fichero `index_tutor.js`.

(a) Entrada valorada en hilo de tutor.

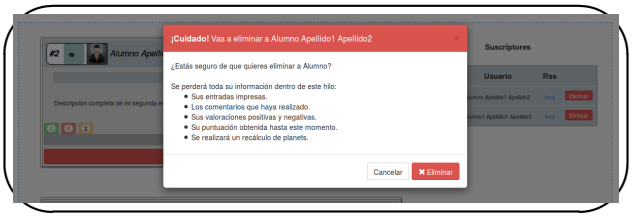
(b) Entrada sin valorar en hilo de tutor.

(c) Cuadro de diálogo de borrado.

Figura 4.26: *Eliminación de una entrada en el hilo de un tutor.*



(a) Tabla de suscriptores en hilo de tutor.



(b) Cuadro de diálogo de borrado.

Figura 4.27: *Eliminación de un alumno en el hilo de un tutor.*

Esta función lanza un diálogo definido por *BootStrap* que da la opción de eliminar el alumno o cancelar dicha acción tal y como se muestra en la figura 4.27(b).

Al hacer click sobre el botón `Eliminar` del cuadro de diálogo, se realiza una petición *AJAX* de tipo `GET` a la vista `eliminaralumno`. La petición es enviada desde el fichero `index_tutor.js` usando *jQuery* e incluye como datos el Id de la asignatura y el Id del alumno. La solicitud es aceptada en el servidor. La vista remueve el objeto `RSS` mediante el método *.delete* y procede a la eliminación de todas las entradas del alumno en ese hilo. Además se elimina su objeto `Valoracion`, por lo que se habrá realizado un borrado completo de ese alumno en ese hilo.

La función *AJAX* no recibe ningún dato como respuesta.

- ***Eliminar diseño en hilo de alumno***: Un alumno puede eliminar los diseños personalizados previamente creados a través del botón "`Eliminar (nombre estilo)`" (figura 4.28). Al pulsar sobre el botón el evento *onClick* llama a la función *JavaScript* `EliminarDiseno` del fichero `diseno.js`. Esta función elimina un diseño determinado.

  Al hacer click sobre el botón `Eliminar (nombre estilo)` se realiza una petición

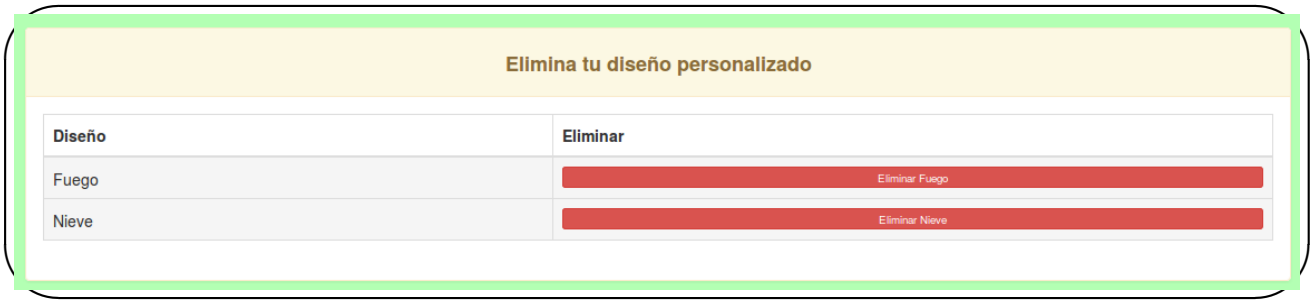


Figura 4.28: *Eliminación de un diseño en el hilo de un alumno.*

AJAX de tipo `GET` a la vista `eliminardiseno`. La petición es enviada desde el fichero `diseno.js` usando *jQuery* e incluye como dato el Id del diseño. La solicitud es aceptada en el servidor. La vista remueve el objeto `Diseno` mediante el método *.delete*. Además si el objeto eliminado se encontraba definido como estilo en el registro del usuario en el model `User`, este campo se inicializará a una cadena vacía.

La función *AJAX* no recibe ningún dato como respuesta.

- ***Búsqueda avanzada***: Un alumno o un tutor tienen la opción de buscar entradas por *Nick de usuario* o *Id de entrada*. Cada opción se selecciona con una etiqueta `<select>`. Al pulsar sobre el botón `Buscar` el evento *onClick* llama a la función *JavaScript* `Buscar` del fichero `buscar.js`.

  Ambas acciones ejecutan una petición *AJAX* de tipo `GET`. La petición es enviada desde el fichero `buscar.js` usando *jQuery* e incluye como datos el Id de la opción y el texto escrito.

  En la figura 4.29(a) se procede a buscar por *Nick de usuario*. La solicitud *AJAX* vincula su recurso *URL* a la vista `buscarNickUsuario`. Comprueba que el texto enviado en la petición no está vacío y si existe un `username` que coincide con lo introducido. Una vez validado correctamente se crean dos listas *JSON* de usuario (`list_usuario`) y de entradas (`list_entradas`) filtradas por el Id del alumno. Estos datos *JSON* son incluídas como respuesta `HTTP` al cliente. En caso de que la validación sea incorrecta se enviará como respuesta `HTTP` un *JSON* vacío que saltará en la plantilla renderizada como un mensaje de aviso.

(a) Buscar entradas por Nick de usuario.

(b) Buscar entrada por Id de entrada.

Figura 4.29: *Ejecución de una búsqueda.*

En la figura 4.29(b) se procede a buscar por *Id de entrada*. La solicitud *AJAX* vincula su recurso *URL* a la vista `buscarIdEntrada`. Comprueba que el texto enviado en la petición no está vacío y si existe un Id de entrada que coincide con lo introducido. Una vez validado correctamente se crean dos listas *JSON* de usuario (`list_usuario`) y de entrada (`list_entrada`). Estos datos *JSON* son incluídas como respuesta `HTTP` al cliente. En caso de que la validación sea incorrecta se enviará como respuesta `HTTP` un *JSON* vacío que saltará en la plantilla renderizada como un mensaje de aviso.

La función *AJAX* recibe una respuesta con código *HTML*. Mediante la función *jQuery* `.append` en el contenedor del documento *HTML* se agrega las entradas enviadas en la respuesta `HTTP`.

Cierre de sesión

El enlace mostrado en la figura 4.30 en color rojo (*Desconectar*) corresponde con el cierre de sesión por parte de alumno o tutor. Al pulsar sobre el enlace se produce una petición `GET`

Figura 4.30: *Enlace de cierre de sesión.*

del cliente. La vista asociada al recurso interno del enlace es `salir`, la cual redirecciona a la plantilla `inicio.html` que crea el documento *HTML* de la respuesta.

Este enlace se sitúa en la parte derecha de la barra de navegación de cualquier plantilla dentro de la aplicación que se cargue después de un previo registro.

Capítulo 5

Prueba en entorno real

En las secciones de este capítulo se va a detallar todo lo involucrado en la prueba de la aplicación web que se realizó en un entorno real. Este entorno se situa en una clase de 2° de Ciclos Formativos de Grado Superior de Administración de Sistemas Informáticos y Redes. Los usuarios comprometidos son el tutor, Don José Ignacio Huertas, y los doce alumnos que conformaban la clase.

5.1. Preparación de la prueba

Antes de emprender con la prueba, se organizó un proceso para asignar dicha prueba a un tutor que estuviera dispuesto a promoverla.

A través del profesor, Don Gregorio Robles Martínez y la colaboración de Don Jesús Moreno, se llegó a un acuerdo con Don José Ignacio Huertas, profesor de 2° de Ciclos Formativos de Grado Superior de Administración de Sistemas Informáticos y Redes:

From: Jesús Moreno

To: José Ignacio Huertas, Gregorio Robles, Adrián Sáez

Fecha: 19 de octubre de 2015

Hola, Adrián,

He hablado con José Ignacio Huertas, que está en copia, sobre la nueva implementación del planeta de blogs que has realizado y la ha encantado la idea.

José Ignacio da clases en un ciclo superior de informática y promueve el uso de los blogs entre

sus alumnos. Además ya conocía la antigua versión del planeta, FAVS, que ya usó con sus estudiantes hace unos años, así que está con ganas de probar esta nueva implementación de la herramienta.

Si os parece, os dejo para que podáis acordar cómo comenzar con las pruebas.

Saludos!

From: Gregorio Robles

To: José Ignacio Huertas, Jesús Moreno, Adrián Sáez

Fecha: 19 de octubre de 2015

Gracias, Jesús. ¡Y estupendo poder contar contigo José Ignacio!

Adrián, ¿cuándo crees que estaría listo todo para probarlo?

saludos, Gregorio

From: Adrián Sáez

To: José Ignacio Huertas, Jesús Moreno, Gregorio Robles

Fecha: 20 de octubre de 2015

Buenos días,

antes de nada, muchśimas gracias a todos por promover el nuevo planeta de blogs y que se pueda probar en un entorno real antes de su defensa.

Respecto a la fecha que estaría listo, ahora estoy trabajando duro para arreglar unas cosillas y calculo que a mediados de la semana que viene estará listo para ultimar detalles y poder probarlo, ¿Qué os parece?

Un saludo, Adrián.

From: José Ignacio Huertas

To: Gregorio Robles, Jesús Moreno, Adrián Sáez

Fecha: 21 de octubre de 2015

Hola Adrián,

un placer conocerte y tener la oportunidad de usar la nueva versión del planeta de blogs.

Espero tus noticias.

Un saludo,

JI

5.1.1. Pasos iniciales

`Primer paso`: Se facilitó un tutorial para tutor y alumnos, en él se indicaban unos pasos iniciales a seguir para el correcto uso de la aplicación. Estos pasos fueron los siguientes:

- Pasos iniciales del tutor:

  1. *URL*: Para comenzar a usar la aplicación, pegar en el navegador: http://tfg-asaez.libresoft.es:8080/planetablogs/login/
  2. *Crear perfil de tutor*: Pulsar sobre el botón verde TUTOR, rellenar todos los campos del formulario (incluída foto), guardar e identificarse en la página de inicio.
  3. *Presentación de tutor*: Es el perfil de cada usuario. Sólo el usuario registrado como tutor podrá crear hilos. Una vez creado, hay que añadirlo a "Mis hilos" (solamente puede agregar el hilo a "Mis hilos" el tutor que lo crea).
  4. *Dentro del hilo*: La primera vez que se visite el hilo estará vacío. Cuando los alumnos se registren y se animen a escribir entradas en el blog asociado al hilo, se mostrarán dichas entradas. El tutor hará un papel de observador aunque tendrá el poder de eliminar entradas, comentarios y alumnos. Con las correspondientes penalizaciones que sufrirán los alumnos y que están escritas en la pestaña Puntuaciones/Información.
  5. *Ayuda*: En la pestaña Ayuda en el interior de cada hilo, se mostrará tanto información general como consejos de uso, y una sección de sugerencias.
  6. *Buscar*: En la pestaña Buscar se podrá realizar una búsqueda avanzada por nick de usuario o por Id de entrada. En las entradas que se muestren, el tutor podrá borrar los comentarios que desee pero no eliminar entradas.

- Pasos iniciales del alumno:

    1. *URL*: Para comenzar a usar la aplicación, pegar en el navegador: http://tfg-asaez.libresoft.es:8080/planetablogs/login/
    2. *Crear perfil de alumno*: Pulsar sobre el botón verde ALUMNO, rellenar todos los campos del formulario (incluida foto), guardar e identificarse en la página de inicio.
    3. *Presentación de alumno*: Es el perfil de cada usuario. El alumno se encontrará con los hilos previamente creados por el tutor. Para que un usuario se suscriba a un hilo, debe introducir en su correspondiente formulario el RSS de su blog.
    4. *Dentro del hilo*: La primera vez que se visite el hilo estará vacío. Cuando los alumnos se registren y se animen a escribir entradas en el blog asociado al hilo, se mostrarán dichas entradas. El alumno tendrá la opción de interactuar comentando y valorando entradas de cualquiera de los alumnos suscritos al hilo. El alumno podrá comentar las veces que desee pero sólo podrá borrar comentarios escritos por el mismo. Y por último, sólo podrá valorar una vez por entrada (Un positivo o un negativo).
    5. *Ayuda*: En la pestaña Ayuda en el interior de cada hilo, se mostrará tanto información general como consejos de uso, y una sección de sugerencias.
    6. *Buscar*: En la pestaña Buscar se podrá realizar una búsqueda avanzada por nick de usuario o por Id de entrada. En las entradas que se muestren, el alumno podrá escribir los comentarios que desee pero sólo podrá borrarlos si no recarga la página una vez hecha la búsqueda. No tiene la opción de valorar entradas.

`Segundo paso`: Fue la creación del hilo por parte del tutor. En este caso, Don José Ignacio Huertas asoció el nombre de "2° ASIR" a su hilo (Figura 5.1).
Una vez fundado el hilo, se crearon doce perfiles correspondientes a los doce alumnos de la clase, además de un perfil de administrador que fue manejado por Adrián Sáez. Los doce alumnos se suscribieron al hilo introduciendo el RSS de su blog personal de blogdiario.com (Figura 5.2).

`Tercer paso`: Comienzo de uso del planeta. El administrador escribió una entrada en su blog con un mensaje de bienvenida para informar a los usuarios sobre la función de administrador, cooperador en la resolución de dudas y problemas, y encargado de dar soporte.

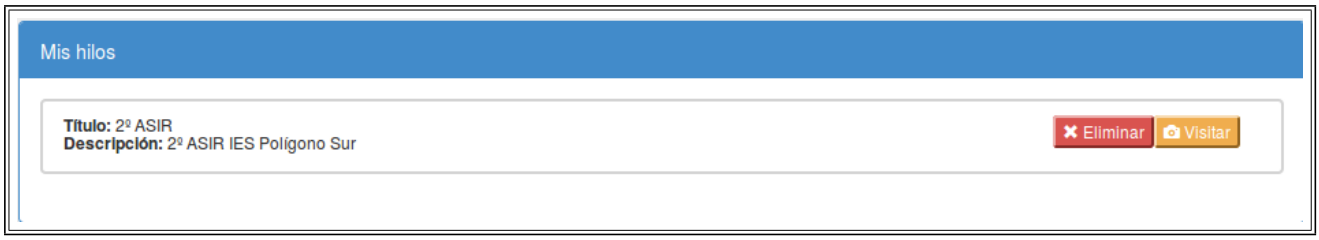


Figura 5.1: Hilo de la prueba.

Figura 5.2: Suscriptores de la prueba.

Por último, los alumnos comenzaron a escribir sus post en sus blogs y una vez plasmados en el planeta, interactuaron con los demás usuarios mediante las funciones que dispone la aplicación. La temática hablada en el hilo fue de seguridad informática y administración de servidores.

5.1.2. Archivo *nohup.out*

El servidor Django de la prueba se lanzó con el comando "nohup". Este comando permite mantener en ejecución un comando, pese a salir del terminal, ya que hace que se ejecute de forma independiente a la sesión, en segundo plano.

La salida por defecto que saldría en el terminal, será procesada a un fichero llamado `nohup.out` gracias a la ejecución de dicho comando. Este fichero aparecerá en la ruta donde nos encontremos ejecutando el programa.

5.1.3. Control estadístico

Dentro del código del proyecto, en el fichero `views.py`, cada vista se encarga de gestionar y realizar una acción pero además incluye una línea informativa que saca por la salida estándar (fichero `nohup.out`).

Las líneas informativas de todas las vistas siguen el siguiente protocolo:

```
[**INICIALESACCION**] Descripci\'on de la funci\'on de la vista.
```

donde INICIALESACCION son letras que identifican la función de dicha vista, por ejemplo:

```
[**AI**] Alumno Identificado en la aplicaci\'on como Pepe.
```

En base al contenido del fichero `nohup.out` se implementó un script para llevar un control estadístico de las acciones que realizaron los alumnos durante el periodo que duró la prueba. Este script se encarga de parsear el archivo `nohup.out`. Este fichero se compone de multitud de líneas con la nomenclatura indicada anteriormente. Cada una de estas líneas se parsea, sacando la información necesaria (acción, usuario y/o entrada) y se guarda en una lista. A través de esta

lista se crean contadores para cada acción, tanto para las estadísticas individuales (se necesitaba la acción y el usuario) como para las estadísticas generales (se necesita la acción únicamente).

Gracias a este script se pudo contabilizar el número de veces que cada usuario accedía a una pestaña o ejecutaba una operación de la aplicación.

5.2. Resultados de la prueba

A continuación se hace una valoración de los resultados obtenidos en la prueba realizada con el fin de estudiar y comprobar el funcionamiento de la aplicación en un entorno real.

- *Conclusiones de la clase*: Durante el uso de la prueba surgieron una serie de problemas que se solucionaron sobre la marcha, haciendo efectiva la función del administrador de proporcionar apoyo y soporte. El tutor facilitó la siguiente lista y el administrador propuso e implementó una solución para cada punto:

  1. Una alumna se olvidó de la contraseña y no ha podido recuperarla (no hemos visto la opción). La opción restablecer contraseña sería necesaria.
     `SOLUCIÓN`: Se ha creado una pestaña en el perfil del tutor. El alumno que haya olvidado la contraseña puede introducir su nick de usuario y su nueva contraseña.

  2. Podría existir algún mecanismo que controle a aquellos que penalizan al resto y no reciben votos positivos de nadie. Quizás se podría limitar el número de "Me gusta" y "No me gusta" de alguna forma. Por ejemplo, para poder marcar alguna entrada con un "no me gusta" debes haber recibido un número de "me gusta" antes. O incluso limitarlo a un número máximo semanal. En cualquier caso, sacar una estadística de esto y que se pueda visualizar por todos sería bueno: Participante – Entradas blog – N° me gusta (recibidos y dados) – N° no me gusta (recibidos y dados) – Comentarios (recibidos y dados)
     `SOLUCIÓN`: Dentro de cada hilo, en la sección de puntuaciones, se ha añadido otra pestaña de estadísticas en la que se podrá ver estadísticas generales con lo recomendado [Participante – Entradas blog – N° me gusta (recibidos y dados) – N° no me gusta (recibidos y dados) – Comentarios (recibidos y dados)]. También se muestra

un resumen de las entradas de cada uno de los alumnos. Respecto al mecanismo de penalización o limitación de "Me gusta", se pensará para futuras implementaciones.

3. Veo que el tutor solo puede eliminar mensajes o entradas (y penaliza al hacerlo, quitando puntos). En algunas ocasiones podría ser bueno que el tutor también pueda intervenir, de forma que un "me gusta" pueda dar puntos (no tengo claro si puntuando igual que si fuera un alumno o algo más).
   SOLUCIÓN: En la parte del tutor, en cada entrada se ha agregado un puntuador con distintas opciones/valores de puntuación para evaluar a los alumnos en cada entrada [-2, -1, 1, 2, 3 ó 4 planets]. Son puntuaciones con más poder que los simples "Me gusta/No me gusta" de los alumnos, ya que puede restar más que un "No me gusta" y sumar más que un "Me gusta".

4. En los post que se muestran no se ven vídeos o imágenes. Esto ha supuesto para mis alumnos un inconveniente grave, dado que eran necesarias para entender algunos posts. Al final se han dado cuenta de que al dar en el nombre se abre el post en el blog correspondiente.
   SOLUCIÓN: Se identificaron las etiquetas de imagen y vídeo para el parseo correcto de la descripción de las entradas. Ya se muestran imágenes y vídeos en las descripciones plasmadas en el planeta.

5. El cuadro de "Entradas más valoradas" me parece muy interesante, aunque no me sale enlazada para que pueda abrirlas directamente al hacer clic en ellas.
   SOLUCIÓN: La sección de "Entradas más valoradas" ha sido modificada con links para que se pueda clicar sobre cada entrada y te redirija al post del blog personal.

6. Quizás sería interesante, dentro de un hilo, poder crear distintas actividades. Por ejemplo, en mi hilo de 2° ASIR se me ocurre que podría existir una actividad en la que todos tuvieran que escribir una única entrada en el blog referida a algún tema concreto y tener un ranking solo de esa actividad.
   SOLUCIÓN: No se planteó ninguna solución. No obstante, se tuvo en cuenta para futuros trabajos.

- *Resultados del control estadístico*: El script extrajo los siguientes resultados estadísticos:

1. En la figura 5.3 se muestra los resultados de estadísticas generales sobre las acciones totales que se efectuaron.

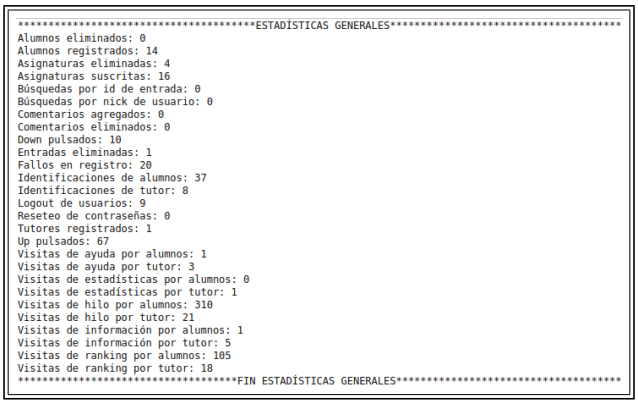


Figura 5.3: Estadísticas generales de la prueba.

2. En la figura 5.4 se puede visualizar los resultados de estadísticas individuales de dos de los doce usuarios suscritos al hilo (por simplicidad).

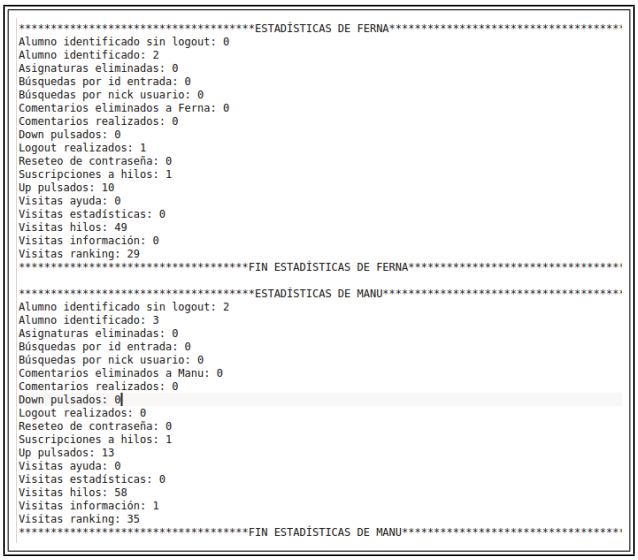


Figura 5.4: Estadísticas individuales de dos usuarios de la prueba.

Capítulo 6

Conclusiones

Este capítulo recogerá las conclusiones más destacadas, la consecución de los distintos objetivos marcados previamente a la realización del proyecto, las tecnologías y conocimientos adquiridos durante la carrera que han significado un aporte en la realización de la aplicación web, la lección de lo aprendido en su implementación y una lista de puntos a mejorar o añadir al resultado final del trabajo.

6.1. Consecución de objetivos

- *Usabilidad sencilla:* La aplicación cuenta con botones con una etiqueta totalmente descriptiva, al igual que los enlaces muestran títulos emergentes descriptivos, todo ello para facilitar su usabilidad. Además se cuenta con dos secciones de ayuda e información de uso. El resultado es una aplicación bastante sencilla de usar. El único punto más complicado se puede dar a la hora de ingresar el *RSS* en un determinado hilo disponible.

- *Integración de elementos sociales:* Objetivo logrado gracias a la sección de escribir comentarios en cada una de las entradas, la puntuación del tutor, los botones "Me gusta" y "No me gusta", el elemento popup con la información de cada usuario (foto, nombre y apellidos, correo, puntos y nivel),..

- *Inclusión de gamificación:* En la pestaña "Información de puntuaciones" se muestran todas las bases para la obtención de premio o penalización de *planets*, en resumen:
  -El usuario que blogee una entrada recibirá una bonificación de 8 *planets*. Si la entrada es

eliminada, se penalizará con 10 *planets*.

-Cada valoración positiva que reciba una entrada corresponde con una bonificación de 2 *planets* para el usuario que escribió dicha entrada. Cada valoración negativa corresponde con una penalización de 1 *planets*.

-Cada comentario realizado se premiará con una bonificación de 3 *planets* para el usuario que escribió dicho comentario. Si el comentario es eliminado se penalizará con 3 *planets*.

-La puntuación del tutor premiará o penalizará al usuario de una entrada con la misma cantidad que valore el tutor en dicha entrada.

-Tabla para la consecución de niveles.

- *Adaptabilidad:* Este objetivo ha sido logrado en cualquier navegador y a cualquier anchura de pantalla en portátiles u ordenadores de mesa. La adaptabilidad en dispositivos móviles tiene problemas con solapamiento de distintos elementos, como botones, enlaces o el menú.

- *Dinamismo y rapidez: AJAX*, *jQuery* y *jQuery-ui* han logrado este objetivo. Sin necesidad de recargar la página y con la ayuda de *jQuery*, la aplicación es capaz de:

  -Eliminar hilos, entradas y alumnos.

  -Agregar hilos disponibles a “Mis hilos”.

  -Mostrar popup con información de usuario y popup con información del creador del hilo.

  -Mostrar comentarios y ocultarlos.

  -Pulsar “Me gusta” y “No me gusta”

  -Escribir comentarios y saltar popup con información de ganancia de 3 *planets*.

  -Zoom en fotos de usuarios.

  -Búsqueda de entradas por id de entrada o por nick de usuario.

  Con la ayuda de *jQuery-ui* se ha creado un acordeón que ha hecho de la sección de estadísticas, una sección dinámica y con mucha información recogida y ordenada.

6.2. Aplicación de lo aprendido

A lo largo de la carrera he cursado asignaturas de las cuales he aprendido diferentes prácticas, conocimientos y habilidades que he aplicado en este proyecto:

1. *Servicios y aplicaciones telemáticas:*
   - Intercambio de datos mediante *XML* y JSON.
   - Relación con la interfaz de usuario: Gestión de *HTML* en el navegador, motor DOM y tecnologías *AJAX*.
   - Marcos de desarrollo `Django`: Esquema típico de la aplicación, bibliotecas y módulos habituales.

2. *Desarrollo de aplicaciones telemáticas:*
   - *HTML5*: Nuevos elementos, canvas, LocalStorage, navegación off-line, etc.
   - *CSS3*: Conceptos básicos, estándares, reglas, *CSS3*, dinamicidad e interactividad.
   - *JavaScript*: Conceptos básicos, interacción con el árbol DOM, *AJAX*.
   - *jQuery*: Interacción con el árbol DOM, funcionalidades añadidas con *jQuery-ui*.
   - *GitHub*: Uso del cliente de *Git*. Creación y manejo de repositorio (clone, merge, push, pull,...).
   - Pivotal tracker: Gestión y administración de tareas.

6.3. Lecciones aprendidas

Enumero los puntos aprendidos en este Trabajo de Fin de Grado:

1. Crear una aplicación web completa desde cero usando `Django` y *Python*.

2. Instalación y puesta a punto de servidor de aplicación para prueba en un entorno real.

6.4. Trabajos futuros

Distintas ideas y funcionalidades a implementar para mejorar considerablemente la aplicación:

1. Adaptabilidad completa en dispositivos móviles: Mejorar la adaptabilidad en diferentes elementos que se salen de sus elementos padres y producen solapamiento en dispositivos móviles.

2. Pestaña de mensajes privados: Chats para comunicar o contactar con otros usuarios.

3. Notificaciones: Comentarios recibidos, "Me gusta/No me gusta" recibidos y mensajes privados recibidos.

4. Contraseña olvidada en la parte del alumno. Formulario con correo y nueva contraseña para poder reestablecerla.

5. Iconos "Leído" y "Favorito": En cada post, cada alumno podrá visualizar si ya lo ha leído, e incluso marcarlo como favorito.

6. Eliminaciones: Posibilidad de eliminar hilos y eliminar cuentas de usuario.

7. Paginador: Mejorar el paginador para obtener mejor accesibilidad a cualquier post.

8. Actividades: Creación de distintas actividades dentro de un hilo.

9. Buscador: Búsqueda de entradas por calendario, indicando intervalo, mes o día.

10. Listas: Pestaña o sección que de opción de crear listas personalizadas para guardar entradas.

11. Optimización de código: Eliminar repetición de código. Usar plantilla.

12. Diseños de aplicación: Dar la opción a los usuarios de personalizar el diseño de su perfil de una forma más completa.

13. Editar perfil: Dar la opción a los usuarios de editar su propio perfil.

Apéndice A

Instalación y uso

A.1. Instalación del entorno en el servidor

Para el uso de la aplicación web es necesario montar un entorno desde cero en una máquina *Linux* que hará de servidor. Y además, tener instalado una serie de herramientas antes de su ejecución desde un terminal *Linux*.

1. Instalación *Pip*: Es necesario este instalador y administrador de paquetes escritos en *Python*.

```
sudo apt-get install python-pip
```

2. Instalación `Django`: Para el desarrollo de la aplicación cabe destacar que se utilizó la versión `Django 1.6.6`.

```
sudo pip install Django==1.6.6
sudo pip install django-extensions #Extensiones de django
```

3. Instalación de utilidades de *Python*: Para el desarrollo de la aplicación cabe destacar que se utilizó la versión *Python 2.7.6*.

```
sudo pip install python-dateutil
```

4. Instalación *Git*: Permitirá descargar el proyecto ubicado en el cliente *GitHub* a la máquina servidor.

```
sudo apt-get install git
```

Antes de clonar el repositorio de *GitHub*, hay que crear un directorio donde se guardará el código del proyecto e inicializará el repositorio.

```
mkdir directorioGitProyecto
cd directorioGitProyecto
git init
git clone https://github.com/AdrianSaezClemente/PlanetaBlogs.git
```

Una vez clonado el repositorio donde se ubica el proyecto, hay que modificar unas rutas en el fichero `settings.py`: La ruta de las plantillas "TEMPLATE_DIRS", la ruta del path del fichero de configuración "SETTINGS_PATH", la ruta con los ficheros estáticos "STATICFILES_DIRS" y la ruta de los ficheros multimedia "MEDIA_ROOT". Cambiar la raíz por la ruta:

```
/home/nombreMaquina/directorioGitProyecto/PlanetaBlogs/TFG/(.....)
```

5. Instalación *Crontab*:

```
sudo apt-get install cron
```

Una vez instalado *Crontab*, editar una tarea desde el propio terminal con el comando:

```
cd /home/nombreMaquina/directorioGitProyecto/PlanetaBlogs/TFG
chmod a+x actualizar.py
crontab -e  #Para editar cron
```

Se edita el cron con los dos comandos siguientes para lanzar la actualización de entradas cada minuto y se guarda:

```
*/1 * * * * cd /home/nombreMaquina/directorioGitProyecto/PlanetaBlogs/TFG;
python actualizar.py;
```

Posteriormente, se puede comprobar si se ha incluído la tarea correctamente:

```
crontab -l  #Para ver cron
```

6. Instalación *PIL*: Esta librería de *Python* agrega soporte para abrir, manipular y guardar imágenes de gran variedad de formatos.

```
sudo apt-get install python-pil
```

7. Instalación *Feedparser*: Esta librería permite parsear etiquetas *XML*.

```
sudo pip install feedparser
```

Por último, hay que poner en marcha el servidor con *nohup*:

```
nohup python manage.py runserver 0.0.0.0:8080
```

Nohup mantendrá la ejecución del comando en segundo plano, por lo que seguirá dando servicio de la aplicación aún cerrando el terminal. Con la dirección 0.0.0.0 se puede acceder desde cualquier navegador con la IP pública o el dominio de la máquina servidor.

A.2. Uso de la aplicación

Para comenzar su uso, abrir un navegador con una de las dos URLs:

```
http://IPpublica:8080/planetablogs/login/
http://Dominio:8080/planetablogs/login/
```

Apéndice B

Manual de usuario

B.1. Página de inicio

Para entrar a la aplicación es necesario visitar la página de inicio B.1. Cuando accedes a dicha página tienes dos posibles acciones a realizar:

Figura B.1: Página de inicio.

- **Identifícate**: Iniciar sesión con un formulario que aparece para introducir nombre de usuario y contraseña B.2(a). En caso de que los datos introducidos sean erróneos saldrá un

mensaje indicando que se deben introducir datos válidos B.2(b).

- **Regístrate**: Antes de iniciar sesión como usuario dado de alta en la aplicación es obligatorio crearse un perfil pulsando sobre uno de los dos botones que aparecen en verde. Con etiqueta "Alumno" para registrarse como alumno, y con etiqueta "Tutor" para registrarse como tutor.

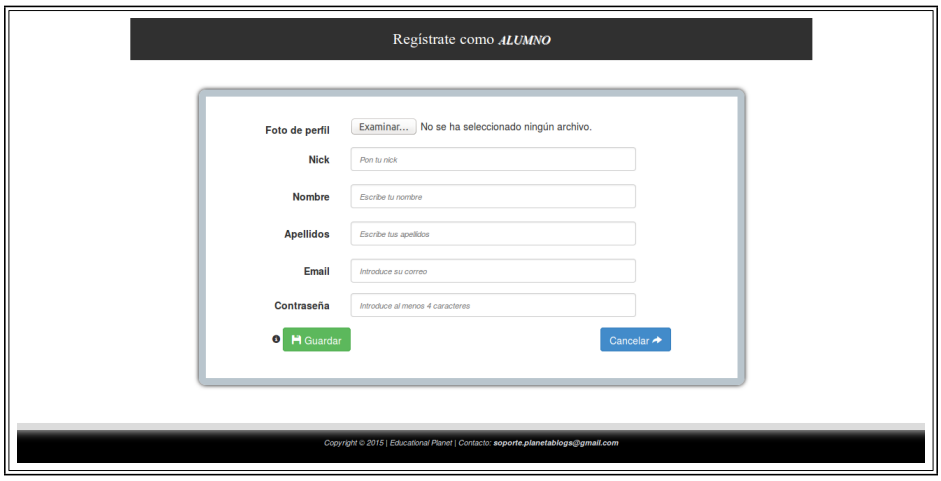
(a) Formulario de inicio de sesión. (b) Inicio de sesión errónea.

Figura B.2: Inicio de sesión.

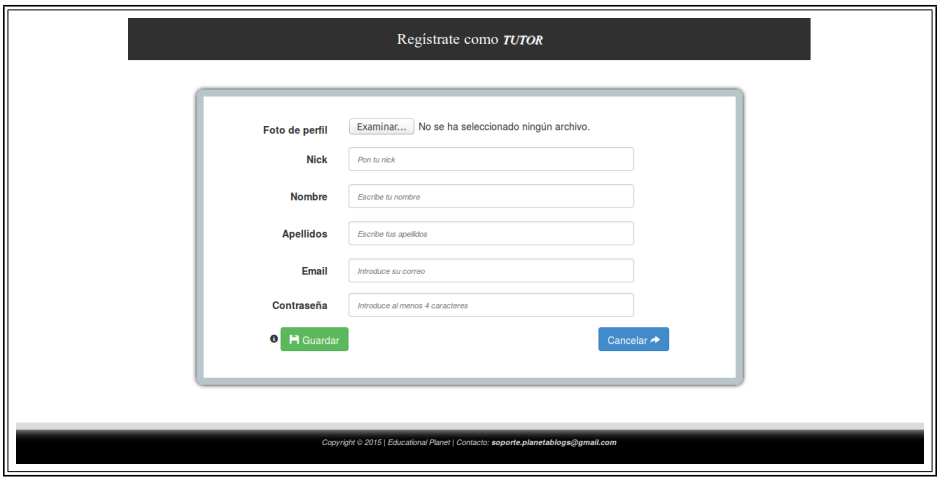
B.2. Página de registro

La creación de un perfil de usuario pasa por rellenar los campos del formulario que se observan en ambas figuras. Tanto en el registro de alumnos B.3(a), como en el registro de tutores B.3(b) se rellenan los mismos campos: Una imagen, un nick de usuario, un nombre y sus apellidos, un email y una contraseña.

El nick de usuario debe ser único y no estar ocupado por ningún integrante de la aplicación. Según se teclea el nick se mostrará si el nick está apropiado por otro usuario en rojo o por el contrario se encuentra libre en verde (figura B.4).



(a) Registro como alumno.



(b) Registro como tutor.

Figura B.3: Inicio de sesión

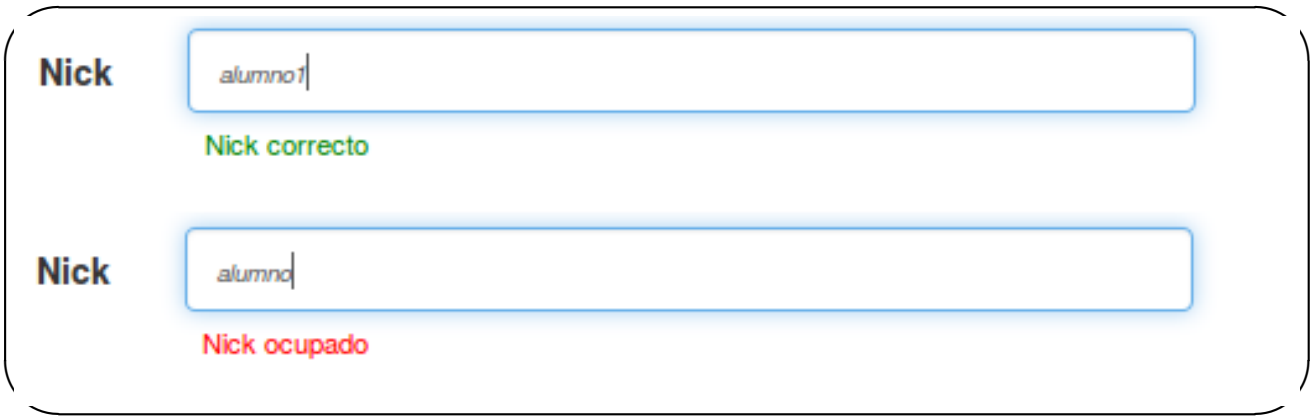


Figura B.4: Rellenar nick

Una vez relleno todos los campos del formulario se debe pulsar el botón “Guardar”. Si los datos son correctos se redirigirá a la página de inicio de sesión para que se proceda con la identificación. Si los datos son incorrectos se mostrará una alerta o panel rojo debajo del formulario con un mensaje de información errónea que anime a introducirlos otra vez. Se puede dar al icono “X” para cerrar el panel (figura B.5).

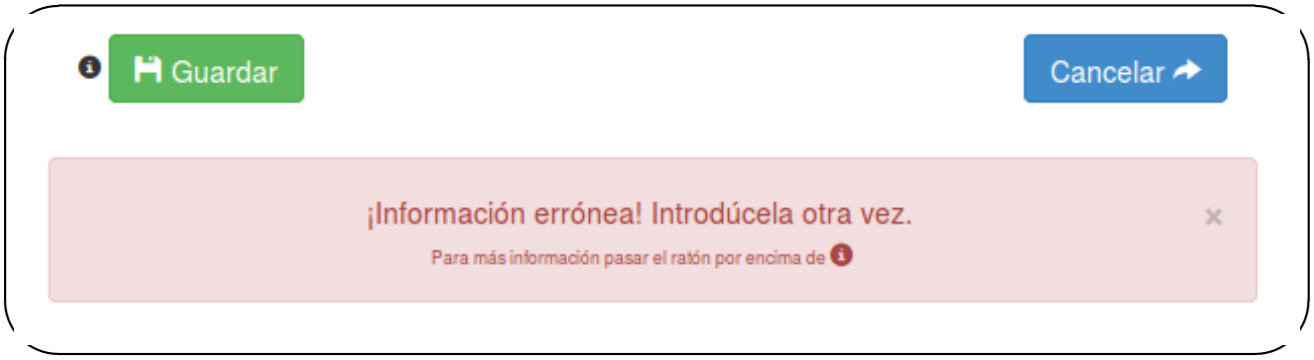


Figura B.5: Registro erróneo

Según se ve en el mensaje de información errónea existe la posibilidad de pasar el ratón por encima del icono de información, a la izquierda del botón “Guardar”. Al realizar esta acción se abre una alerta azul con un mensaje recordatorio de como deben ser los datos introducidos para un correcto registro de perfil de usuario. Son cinco condiciones que se enumeran en la figura B.6.

Otra acción a destacar dentro de la página de registro es la opción de cancelar el registro con el botón azul “Cancelar”. Si se pulsa sobre él se redirigirá a la página de inicio de sesión.

B.3. Presentación de tutor

Después de haber realizado el registro de usuario como tutor y de introducir nombre de usuario y contraseña en la página de inicio, la sesión comienza con la presentación de tutor o perfil de tutor (figura B.7). En la cabecera se ofrece la posibilidad de desconectar la sesión (a la derecha, pulsando el enlace “Desconectar” en rojo). Aparece una pestaña “Mis hilos” la cual tiene la función de actualizar la página de perfil de tutor y también otra pestaña “Resetear contraseña” la cual se encargará de redirigir a una página formulario con la posibilidad de modificar la contraseña de cualquier usuario. Además se exhibe una foto del usuario y un título

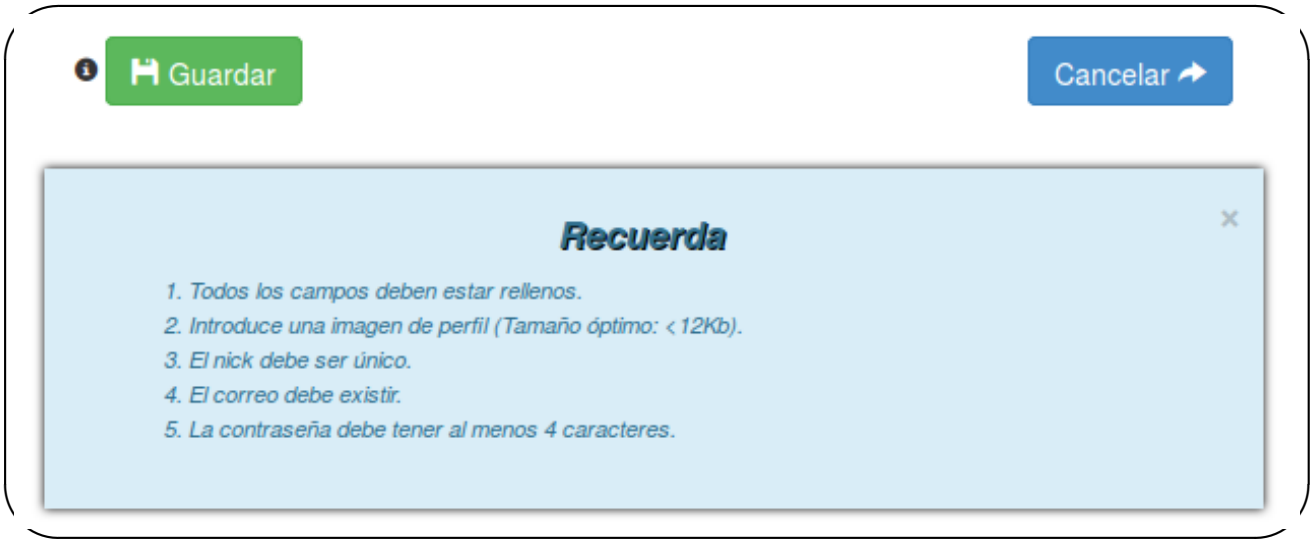


Figura B.6: Información de registro

con la expresión *Espacio de NOMBRE APELLIDO1 APELLIDO2* indicando de quien es el perfil.

Las secciones que se incluyen en la presentación de un tutor son:

- **Crear hilo**: Al presionar el botón verde y alargado "Crear hilo", se extiende un formulario como el de la figura B.8 por el cual un tutor puede fundar un hilo asignando un título y una descripción breve sobre la temática del hilo. Al enviar el formulario, el hilo pasa directamente a la sección de "Hilos disponibles".

- **Hilos disponibles**: Se presentan todos los hilos creados anteriormente a los que el tutor no está suscrito (ya sean creados o no por el tutor del propio perfil). Sin embargo, como se muestra en la figura B.9, al lado de cada hilo se expondrá un botón "Añadir" solamente si ha sido creado por dicho tutor. Si se presiona pasará a la sección de "Mis hilos".

- **Mis hilos**: Se presentan todos los hilos a los que el tutor está suscrito (figura B.10). Para cada hilo se hallan las alternativas de apretar el botón "Eliminar" para suprimir el hilo de esta sección y enviarlo a "Hilos disponibles", o de presionar el botón "Visitar" para acceder a la página propia de la asignatura pertinente. Al pulsar sobre el botón "Eliminar" se abre un cuadro de diálogo en el centro de la pantalla para asegurarse de que es la acción que realmente se quiere realizar.

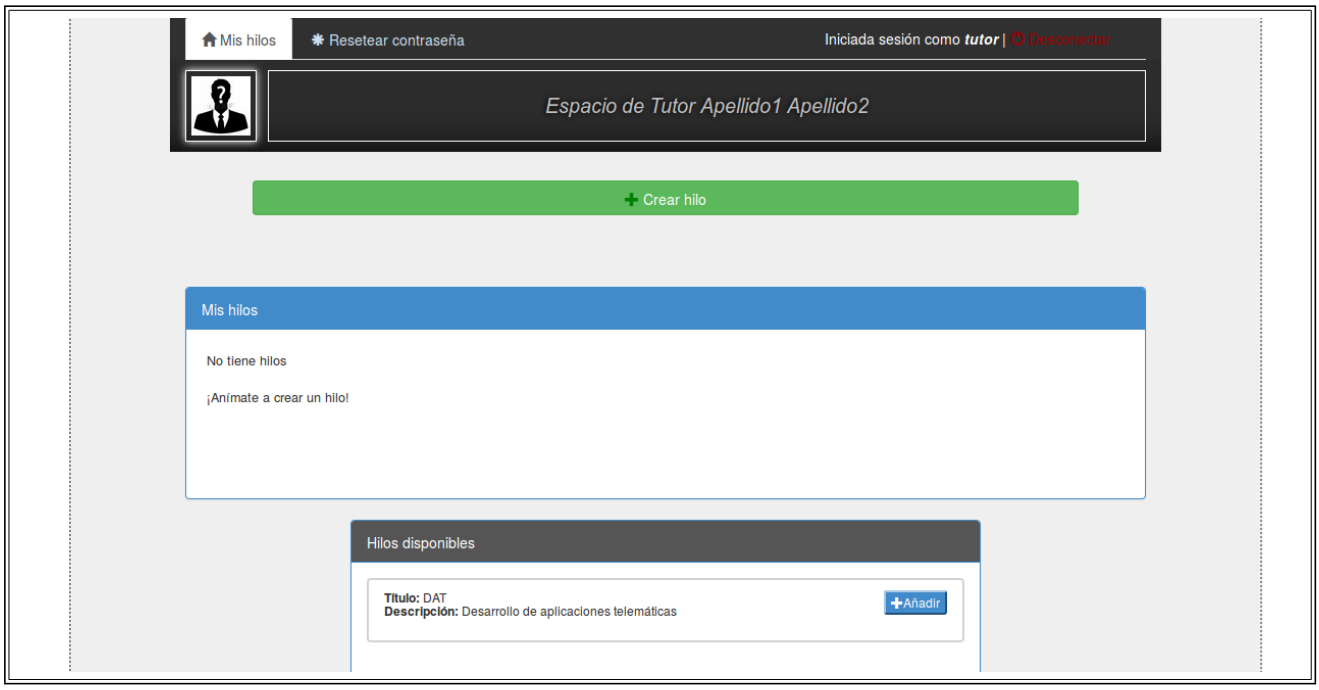


Figura B.7: Página de presentación de tutor.

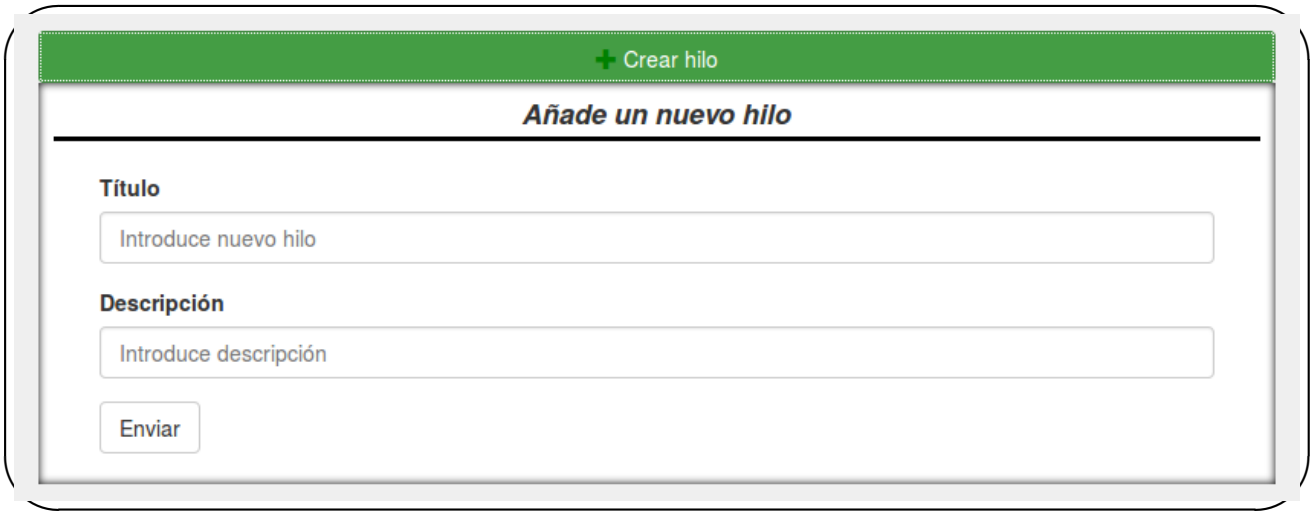


Figura B.8: Formulario para la creación de un hilo.

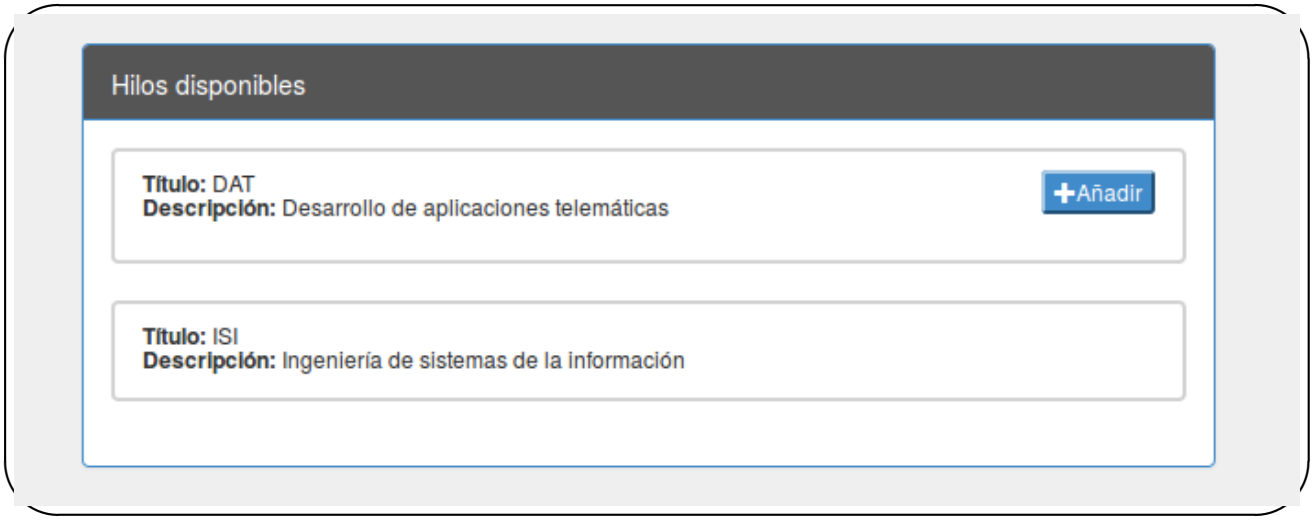


Figura B.9: Sección de *Hilos disponibles* en una presentación de tutor.

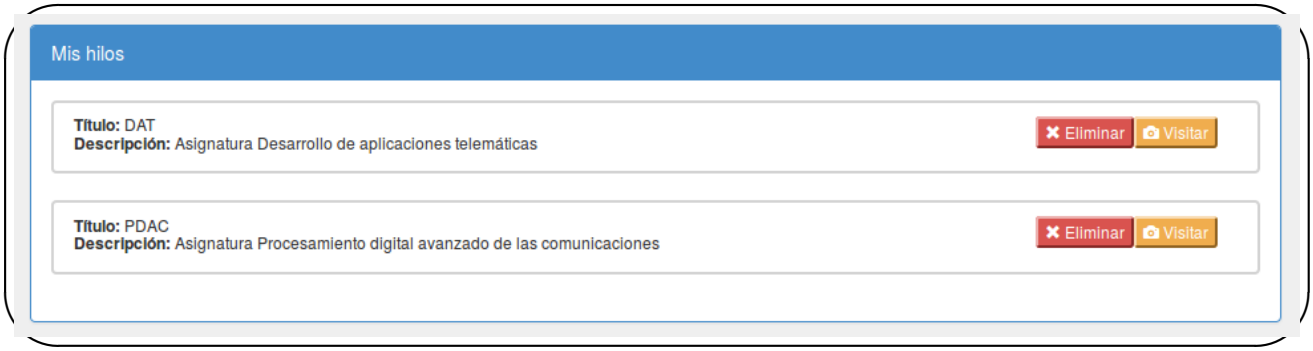


Figura B.10: Sección de *Mis hilos* en una presentación.

B.4. Presentación de alumno

Después de haber realizado el registro de usuario como alumno y de introducir nombre de usuario y contraseña en la página de inicio, la sesión comienza con la presentación de alumno o perfil de alumno (figura B.12). En la cabecera se ofrece la posibilidad de desconectar la sesión (a la derecha, pulsando el enlace “Desconectar” en rojo). Sólo aparece una pestaña “Mis hilos”, a la izquierda, la cual tiene la función de actualizar la página de perfil de alumno.

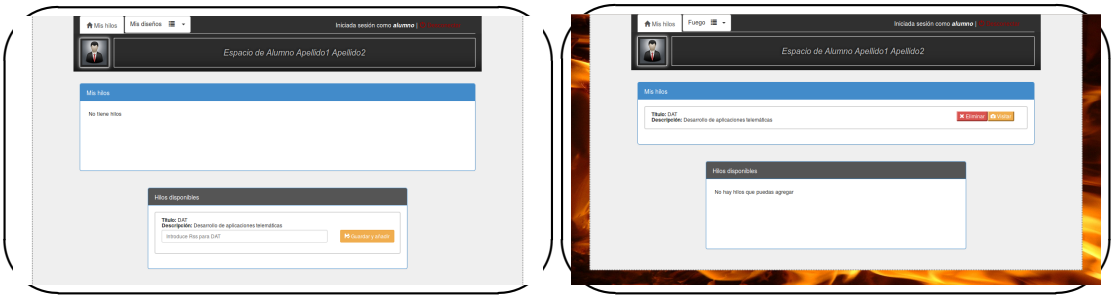
Aparece una pestaña “Mis hilos”, a la izquierda, la cual tiene la función de actualizar la página de perfil de alumno. Y un menú con todos los diseños creados por el usuario (figura B.11), al seleccionar uno de ellos se cambiará la imagen de fondo de las páginas ’Presentación de alumno’, ’Página principal del hilo’y ’Ránking’.

Figura B.11:

También se exhibe una foto del usuario y un título con la expresión *Espacio de NOMBRE APELLIDO1 APELLIDO2* indicando de quien es el perfil.

Las secciones que se incluyen en la presentación de un alumno son:

- **Hilos disponibles**: Se presentan todos los hilos creados anteriormente a los que el alumno no está suscrito. Como se ve en la figura B.13 cada hilo expone un botón “Guardar y



(a) Sin diseño. (b) Con diseño.

Figura B.12: *Página de presentación de alumno.*

añadir" para que cuando sea presionado, el hilo correspondiente pase a la sección de "Mis hilos". Pero antes se debe haber introducido un enlace *RSS* válido en el formulario proporcionado para ello. En ese *RSS* (blog) será donde el alumno escribirá sus entradas para que se publiquen automáticamente dentro de la aplicación, concretamente en el hilo al que pertenecen.

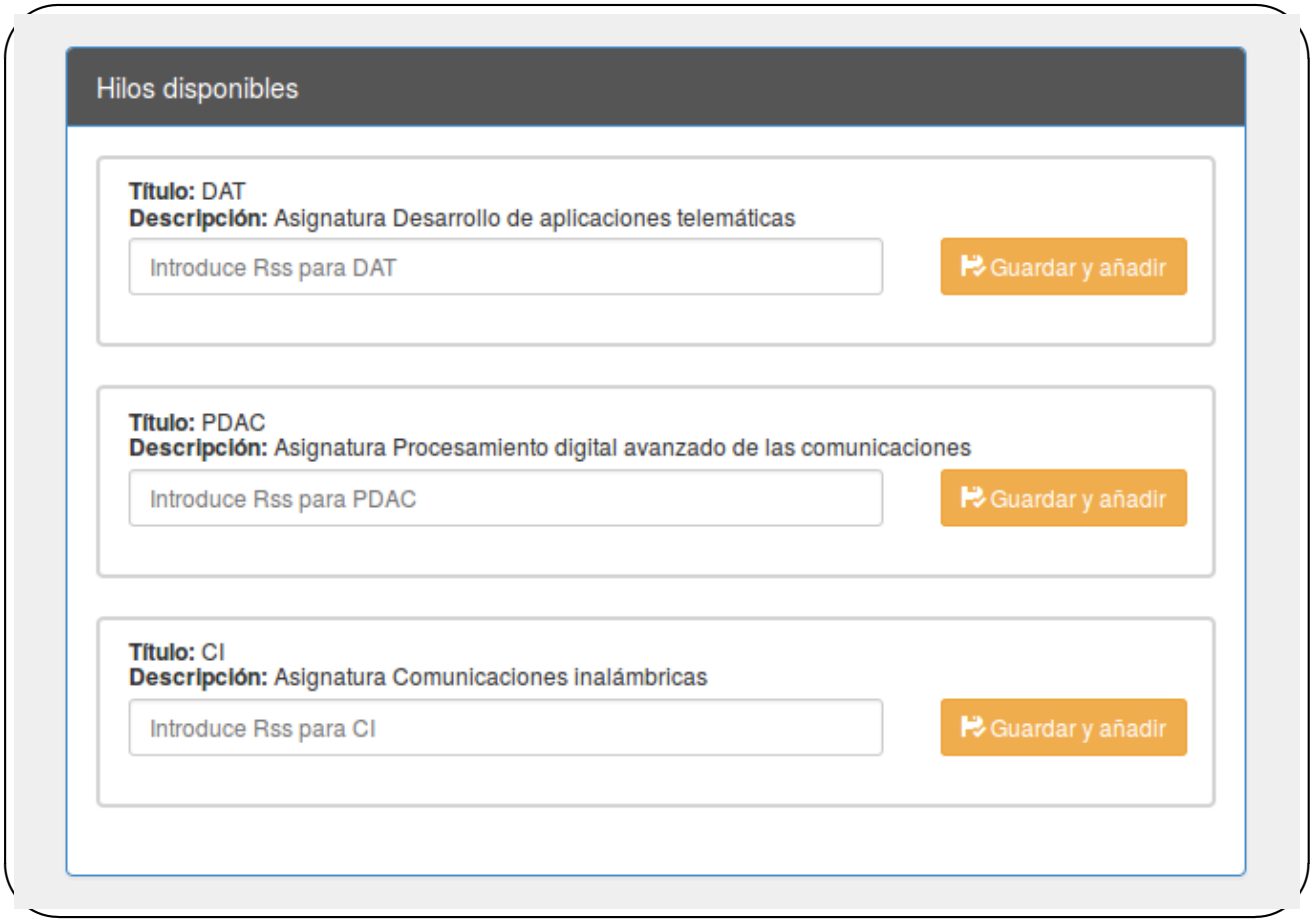


Figura B.13: Sección de *Hilos disponibles* en una presentación de alumno.

En caso de que el *RSS* no sea válido se recargará la página de presentación de alumno indicando un error en un panel de alerta marrón (figura B.14).

- **Mis hilos**: Se presentan todos los hilos a los que el alumno está suscrito (sección idéntica a la que expone la presentación de tutor (figura B.10). Para cada hilo se hallan las alternativas de apretar el botón "Eliminar" para suprimir el hilo de esta sección y enviarlo a "Hilos disponibles", o de presionar el botón "Visitar" para acceder a la página propia de

Figura B.14: Error al introducir *RSS* en una presentación de alumno.

la asignatura pertinente. Al pulsar sobre el botón "Eliminar" se abre un cuadro de diálogo en el centro de la pantalla para asegurarse de que es la acción que realmente se quiere realizar, avisando de los riesgos que ello conlleva (figura B.15).

B.5. Personalizar diseño

A la página para gestionar diseños se accede después de dar al enlace "Agregar/Eliminar diseño", al final de la lista de "Mis diseños" en la presentación del alumno (figura B.11).

Una vez dentro de la página (figura B.16(a)), se podrá crear un nuevo diseño con la inserción de un nombre de estilo y una imagen de fondo con unas características específicas que asegurarán una buena calidad en la personalización de la aplicación. Los diseños creados se unen a la tabla de "Mis diseños", desde donde se podrá pinchar sobre el botón rojo que se quiera para eliminarlos (figura B.16(b)).

B.6. Hilo de alumno

La página de una asignatura concreta por parte de un alumno se visualiza después de dar al botón "Visitar" en su presentación. En la figura B.17 se muestra un hilo vacío en sus inicios, sin entradas.

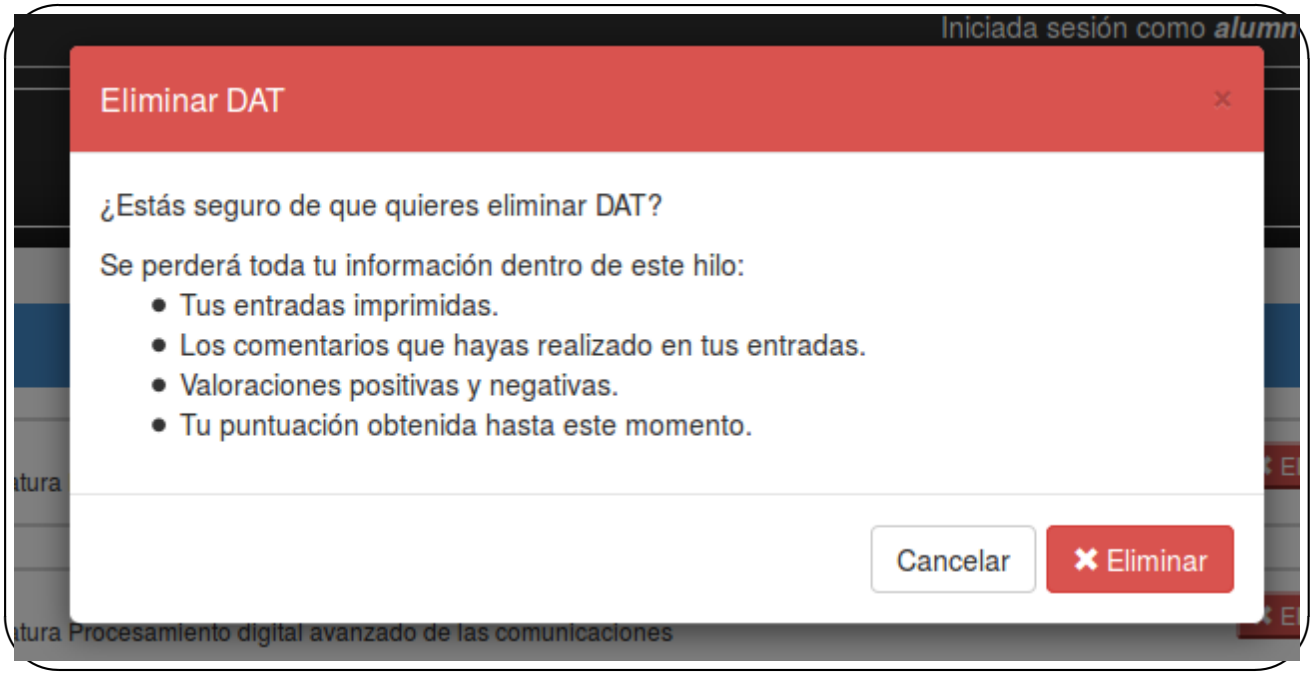


Figura B.15: Eliminar hilo de la sección de *Mis hilos* en una presentación de alumno.

(a) Página completa de personalización. (b) Sección para eliminar diseños.

Figura B.16: *Gestionar diseños personalizados.*

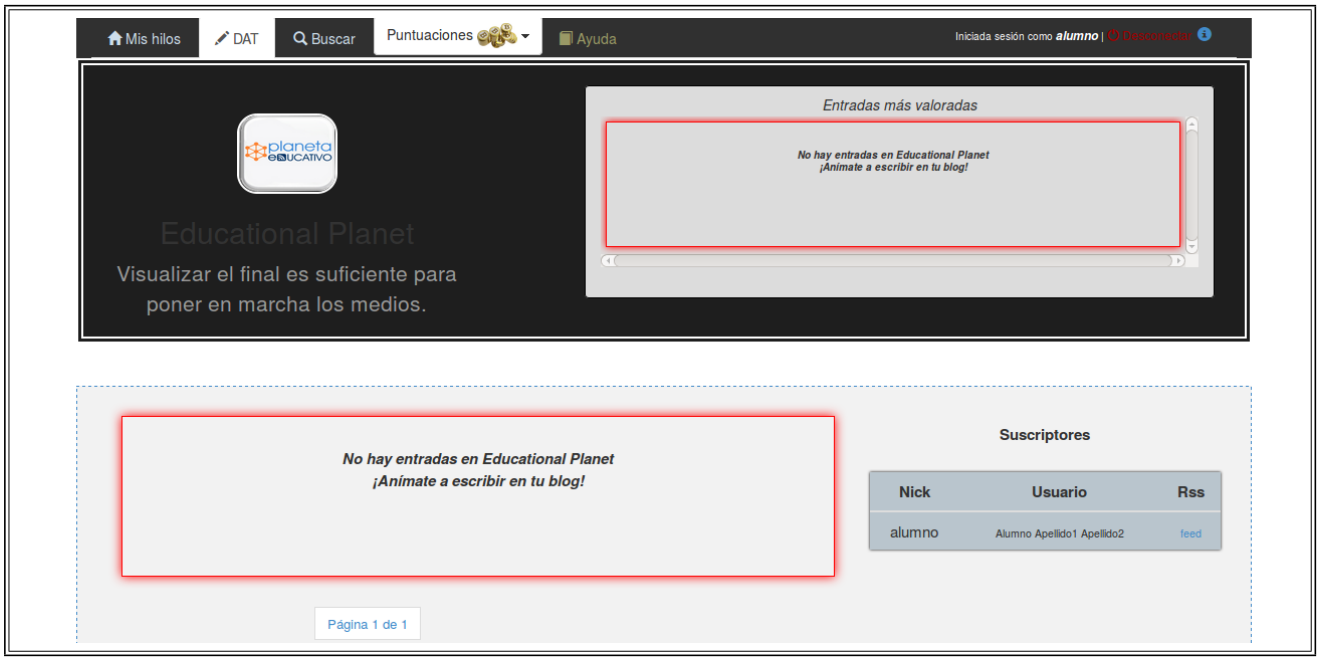


Figura B.17: Hilo vacío de un alumno.

Para ver tus propias entradas es necesario escribir previamente en el blog asociado a esa asignatura. En la aplicación se ejecuta un script automático cada sesenta segundos que se encarga de guardar y actualizar las entradas de todos los blogs de todos los alumnos que estén suscritos al hilo. Cada vez que un usuario suscrito escriba un mensaje en su blog, este mensaje se plasmará en el contenedor de la página. Para recargar la página se puede pulsar sobre la pestaña que está activa en cabecera, y que expone el título del hilo.

B.6.1. Entradas

El usuario puede interactuar con una entrada (B.18) de diversas maneras:

- **Mostrar visitantes**: Debajo de la fecha de cada entrada, hay un número subrayado que muestra el total de visitas al post. Si se pulsa, se mostrará una lista de los usuarios que previamente han pulsado sobre el enlace del título para acceder al post del blog externo. Y si se vuelve a pulsar, el popup desaparecerá.

- **Entrada leída**: El botón que se encuentra a la derecha del id de la entrada con un icono de un ojo tachado indicará si la entrada ya ha sido leída o no. Si se hace click sobre

Figura B.18: Hilo con entradas de un alumno.

dicho botón, este se deshabilitará (desapareciendo el color azul grisáceo del botón). Ver la figura B.19.

- **Valoración cuantitativa**: Los botones verde y rojo aportan una valoración cuantitativa. Si se decide por pulsar la valoración positiva (botón verde), el alumno que creó la entrada recibirá una bonificación de dos *planets* (el *planet* es la moneda de la aplicación). Si se decide por pulsar la valoración negativa (botón rojo), el alumno que creó la entrada recibirá una penalización de un *planet*. Si un usuario presiona uno de los dos botones sobre una entrada, no podrá valorar más veces en ella, ya que ambos botones emergerán deshabilitados con su contador total. En este momento, el usuario podrá visualizar el total de valoraciones positivas y negativas que posee esa entrada.

- **Valoración cualitativa**: El área facilitado para escribir un comentario aporta una valoración cualitativa. Para mostrar dicho área hay que desplegar el botón “Mostrar comentarios”, al lado del cual se encuentra un contador total de comentarios escritos sobre esa entrada. Si el usuario decide proponer una opinión, este usuario recibirá una bonificación de tres *planets* al presionar el botón “Enviar”, y simultáneamente se mostrará un popup

Figura B.19: Lectura de entradas.

informando de que acabas de recibir tres *planets* (ver figura B.20)

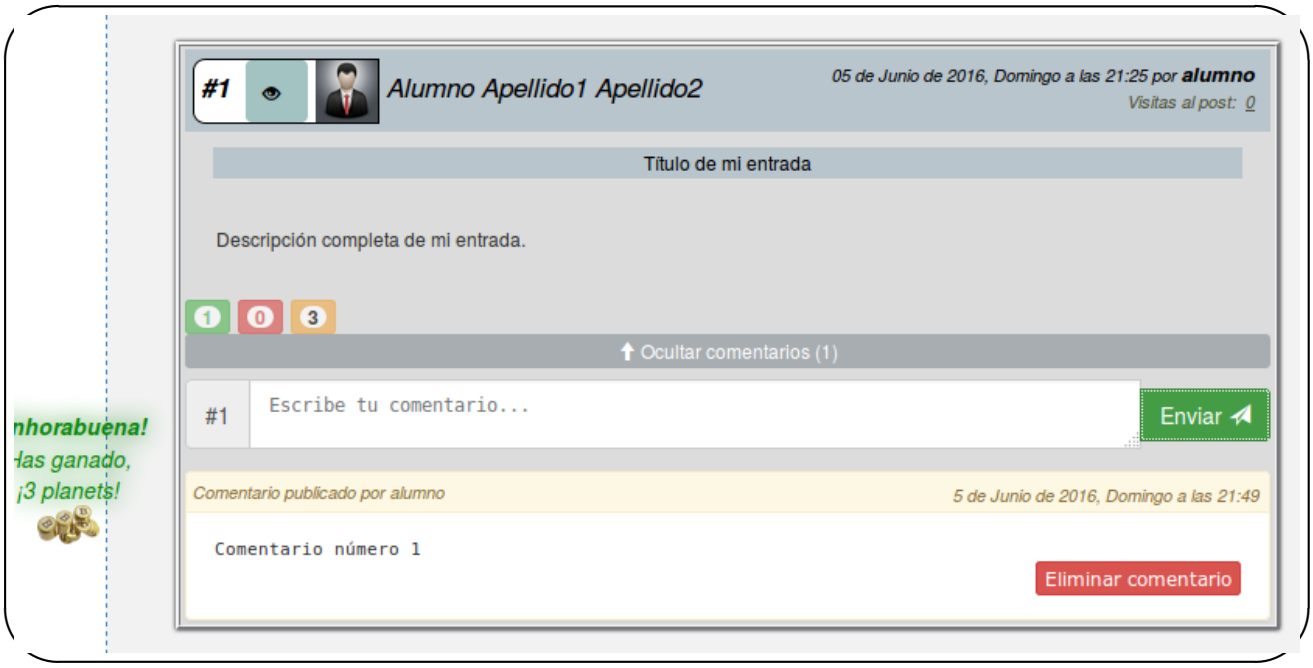


Figura B.20: Información emergente de un comentario.

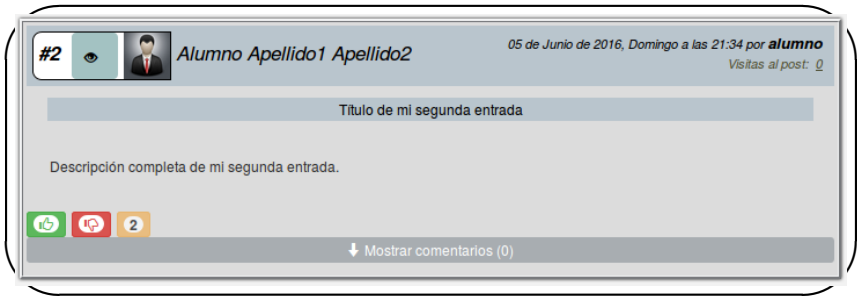
Si ese comentario es eliminado, el usuario obtendrá una penalización de tres *planets*.

- **Puntuación del tutor**: La valoración del tutor se muestra a la derecha de los botones verde y rojo. En caso de que el tutor haya calificado la entrada se podrá ver la puntuación dada en naranja (B.21(a)). En caso contrario, se obtendrá una calificación de "Sin Calificar" en azul (B.21(b)).

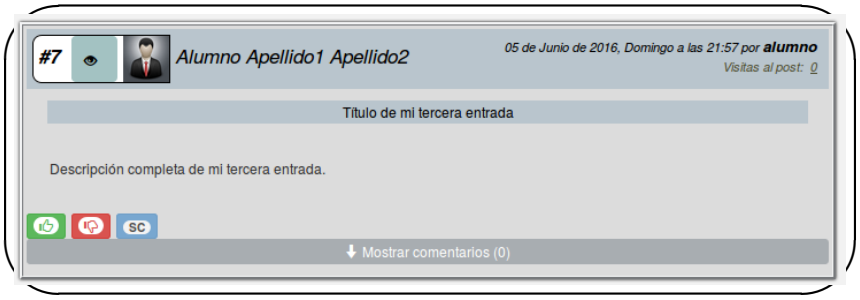
- **Ojear blog**: En una entrada existen dos enlaces que redirigirán directamente al blog del alumno donde se ubican todas sus entradas fuera de la aplicación. Los dos enlaces aparecen en forma de "Nombre y apellidos" y "Nick" (al lado de la fecha, en negrita).

  Si se presiona sobre el enlace del título, se abrirá otra pestaña con la ubicación de la propia entrada dentro del blog. Y además, el botón de entrada leída se deshabilitará (si no ha sido previamente pulsado).

- **Visualizar información del usuario**: Al pasar el ratón por encima del enlace "Nombre y apellidos" brotará un mensaje informativo con datos del usuario: Foto de perfil, nombre, apellidos, puntuación total y nivel (B.22).



(a) Entrada valorada por el tutor.



(b) Entrada no valorada por el tutor.

Figura B.21: *Valoraciones del tutor en entradas.*

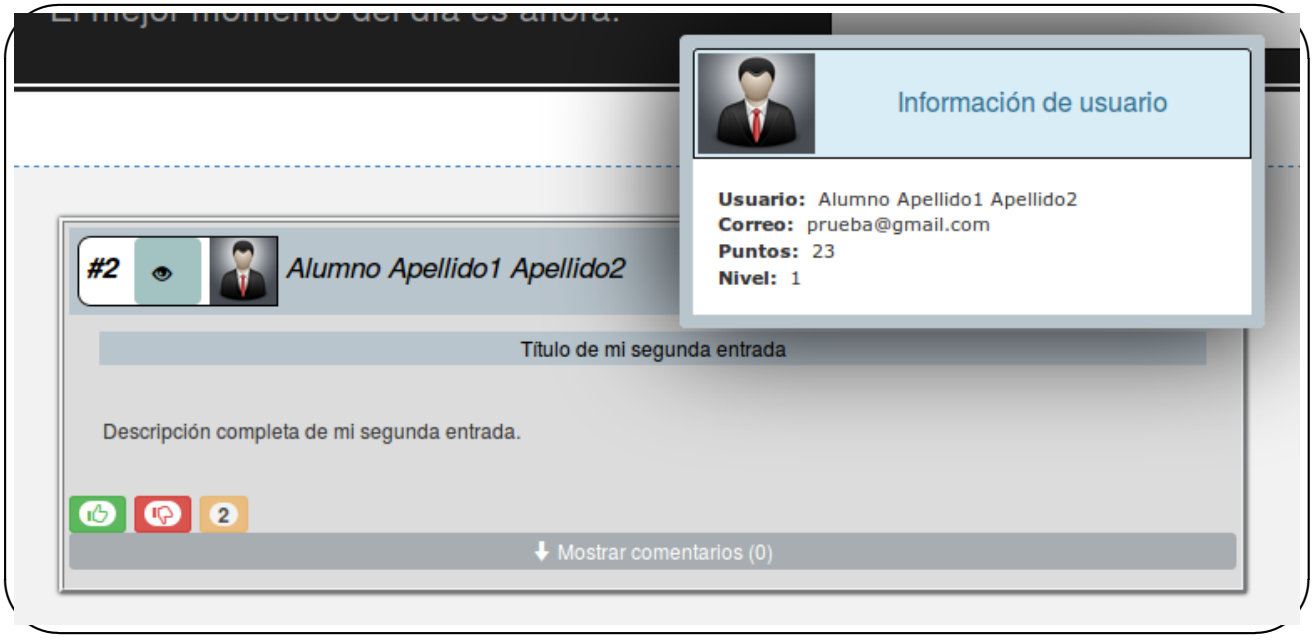


Figura B.22: Información emergente de un alumno.

- **Información adicional**: Además, la entrada muestra el número de identificación seguido de "#" y una foto. Para cada hilo no puede existir dos entradas con un mismo número identificativo.

B.6.2. Logo

Debajo de la cabecera, en la parte izquierda se puede observar el logo y el nombre de la aplicación, junto con una frase motivadora que hace de eslogan. Cada vez que se recarga la página o se presiona sobre la frase, ésta se modifica automática y aleatoriamente de entre un conjunto de 21 oraciones (B.23)

Figura B.23: Logo, nombre y eslogan aleatorio de la aplicación.

B.6.3. Entradas más valoradas

Otra parte importante se sitúa en la parte superior de la página debajo de la cabecera, donde se agrupan las entradas con más valoración cuantitativa de la asignatura. Se aplica una fórmula, la cual suma dos *planets* por cada valoración positiva y resta un *planet* por cada valoración negativa. Se calcula el total de la suma y se plasman las diez mejores entradas.

En la figura B.24 se muestra la sección donde encajan las diez entradas seleccionadas. Cada una se rellena con el número de posición que ocupa, su identificador, su título, la persona que lo escribió, su fecha, las valoraciones positivas y negativas (en color verde y rojo, respectivamente), y por último su número total de *planets*. Son enlaces que al pinchar encima de cada entrada te lleva a la página del post correspondiente.

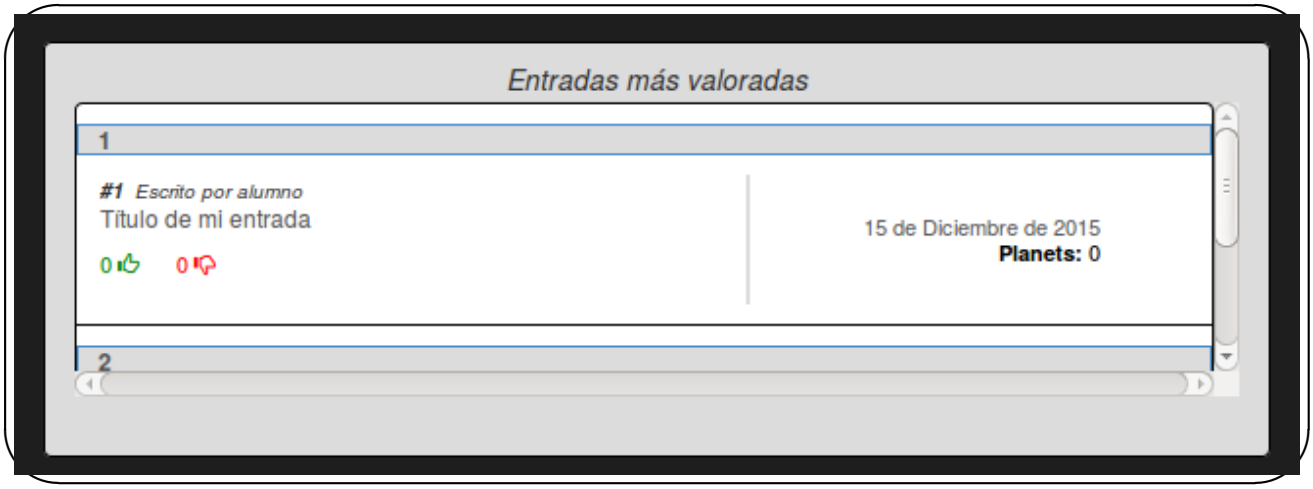


Figura B.24: Entradas más valoradas del hilo de alumno.

B.6.4. Usuarios suscritos

En la columna situada a la derecha se reúnen en una tabla todos los alumnos suscritos en el hilo. Esta tabla se compone de tres columnas correspondientes al nick de usuario, nombre y apellidos de usuario, y un enlace que redirecciona al RSS del blog del alumno. (B.25).

B.6.5. Paginador y flecha

Estas dos funciones se pueden ver en la figura B.26.

Figura B.25: Usuarios suscritos al hilo.

El paginador se encuentra en la parte inferior de la página. Cada página del hilo mostrará las cinco entradas más recientes, las posteriores cinco las pondrá en la siguiente página y así sucesivamente.

Otra función añadida es la flecha de subida situada en la parte inferior derecha de la pantalla. Ésta aparece cuando desciende el scroll una determinada altura. Si se pulsa sobre ella, el scroll asciende automáticamente hasta la parte superior de la página.

Figura B.26: Paginador y flecha.

B.7. Hilo de profesor

Se conservan las mismas funciones que en la sección B.6 (hilo de alumno) con algunos cambios en “Entradas” y en “Usuarios suscritos”. En la figura B.27 se observa como estos cambios se reducen simplemente a que un tutor tiene el permiso de eliminar cualquier entrada, eliminar cualquier alumno de un determinado hilo y valorar cualquier entrada de forma cuantitativa. Al

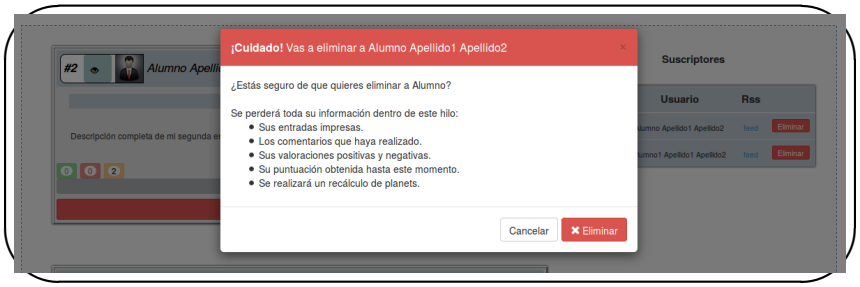
Figura B.27: Hilo con entradas de un tutor.

pulsar el botón “Eliminar entrada” se abrirá una alerta de aviso con unos puntos que se podrán leer para confirmar si realmente se quiere esa eliminación o no. Ver figura B.28(a).

Al pulsar el botón “Eliminar” se abrirá otra alerta de aviso con unos puntos que se podrán leer para confirmar si realmente se quiere eliminar ese usuario de ese hilo o no. Ver figura B.28(b).

La valoración del tutor se realiza a través de la parte de la entrada vista en la figura B.29. Los posibles valores están dentro de un rango con un límite máximo que es superior al valor de un “UP” y con un límite mínimo que es inferior al valor de un “DOWN”. El intervalo de *planets* es [-2, -1, 1, 2, 3, 4]. Una vez seleccionado el valor, se envía al pulsar sobre el botón “Enviar” azul.

(a) Eliminar entrada.



(b) Eliminar alumno.

Figura B.28: *Eliminaciones en hilo de tutor.*

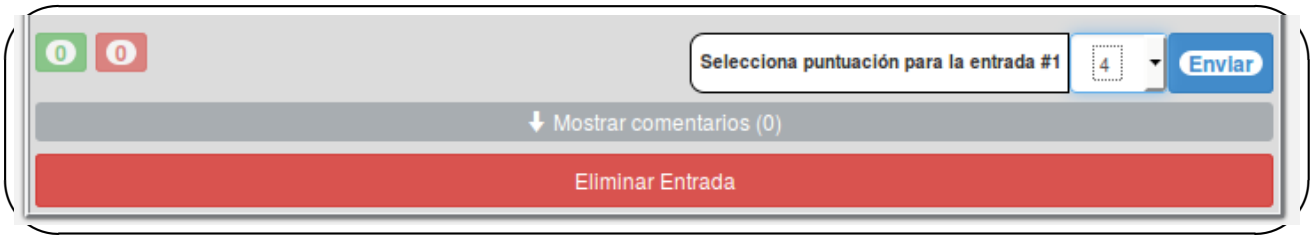


Figura B.29: Valoración de tutor en una determinada entrada.

B.8. Opciones de menú en cabecera

Dentro de la cabecera del hilo de un alumno o de un tutor se encuentran varias opciones. Cada pestaña esconde un enlace que redirigirá a su correspondiente página. Una vez el usuario se encuentre en la página del enlace pulsado, la pestaña se mostrará activa en color blanco mientras que las demás pestañas permanecerán inactivas en color negro.

B.8.1. Mis hilos

La pestaña de "Mis hilos" devolverá al usuario a su perfil o presentación descrita anteriormente en la sección B.5 (si el usuario es un alumno) o en la sección B.3 (si el usuario es un tutor).

B.8.2. Título del hilo

Esta opción es la página principal de cada hilo. Hay que tener en cuenta que por debajo de la aplicación se ejecuta un script automáticamente cada sesenta segundos para actualizar y guardar entradas nuevas de los distintos blogs, por lo que si se pulsa sobre esta pestaña, ésta hará la función de recargar la página para plasmar esas actualizaciones periódicas en el contenedor de entradas.

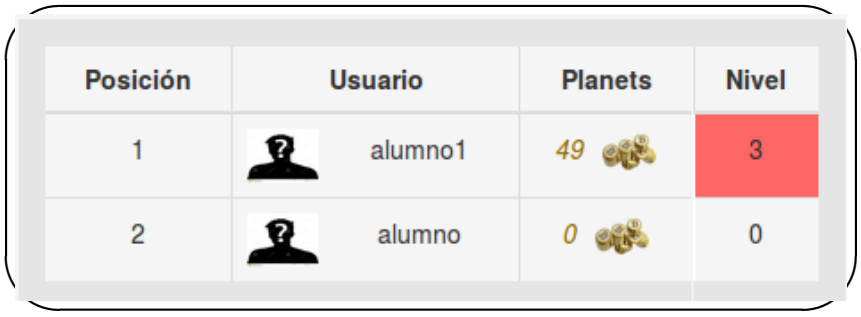
B.8.3. Buscar

La página asociada a la pestaña "Buscar" se muestra en la figura 4.14 mostrada en la sección 4.2.4 que explica la vista "Búsqueda avanzada".

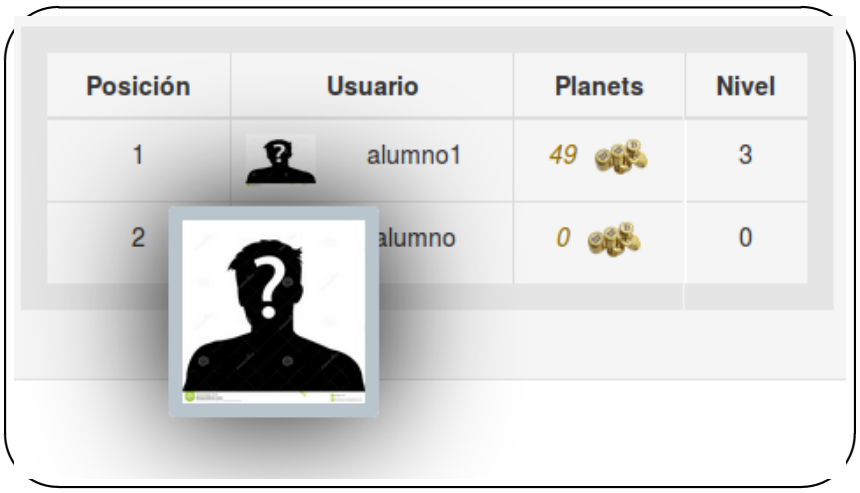
La búsqueda se puede realizar por identificador de entrada o por nick de usuario. Al elegir una opción, se debe escribir un identificador de entrada o un nick de usuario válidos. En caso de éxito se visualizará la entrada por introducción del identificador o las entradas por introducción del nick (figura 4.29). En caso de no introducir un texto válido se notificará con un mensaje en rojo en el contenedor.

B.8.4. Puntuación

Si se despliega el menú "Puntuación" emergen dos opciones:



(a) Color asociado al nivel.



(b) Foto de perfil ampliada.

Figura B.30: *Interacciones de usuario en la página de Ranking.*

- **Ranking**: La figura 4.15 de la sección 4.2.4 muestra la página al pulsar sobre el enlace de la pestaña "Ranking". La página es informativa en su mayoría. Muestra el ranking del hilo en una tabla con cuatro columnas: Posición, nick de usuario con su foto, puntuación total y nivel.

  El usuario puede interactuar con esta página poniendo el ratón encima de la foto para ampliarla, y encima del nivel para visualizar el color al que está asociado (B.30).

- **Estadísticas**: La figura 4.16 de la sección 4.2.4 muestra la página al pulsar sobre el enlace de la pestaña "Estadísticas". La página es informativa. Da la opción al usuario de elegir las estadísticas que quiere visualizar (B.31).

  Cada pestaña del acordeón refleja la información de un usuario. La información que se visualice depende de las opciones que el usuario haya elegido. En la figura B.32 se puede observar las tres opciones posibles: Sólo estadísticas generales (a), sólo resumen de

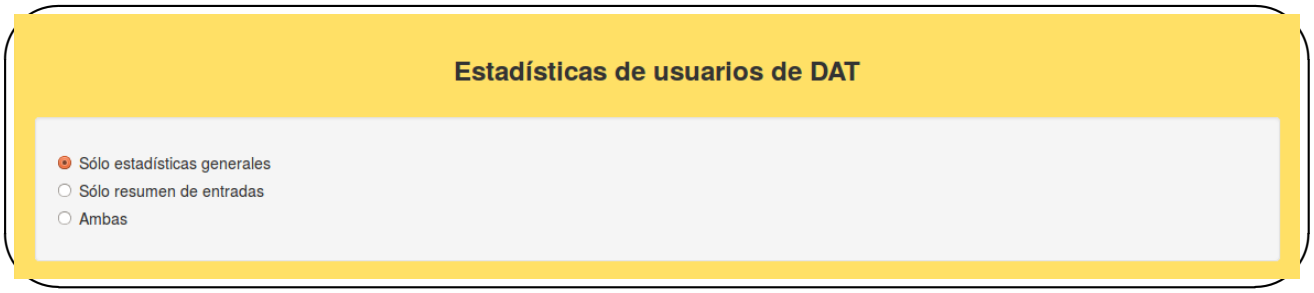


Figura B.31: Opciones de estadísticas.

entradas (b) o ambas (c).

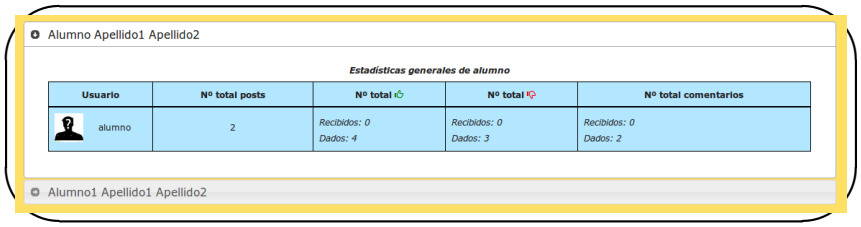
- **Información**: La figura 4.17 de la sección 4.2.4 muestra la página al presionar el enlace de la pestaña "Información". La página es plenamente informativa. Crea unas bases de juego muy importantes para que el usuario tenga noción del sistema de puntuaciones. También se muestra una tabla con el rango de puntuaciones que pertenece a cada nivel. Si se pasa el ratón por encima del nivel, éste se iluminará con su color asociado. El nivel 0 es el rojo y el nivel máximo es el verde.

B.8.5. Ayuda

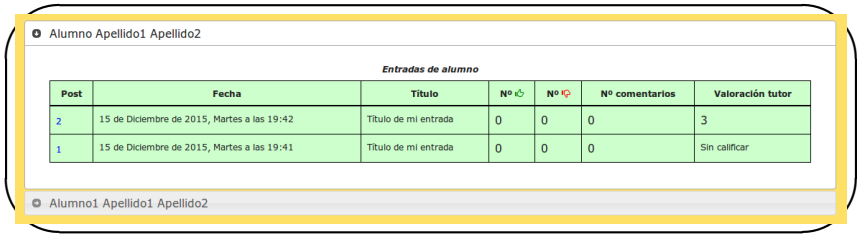
La pestaña "Ayuda" muestra una serie de información y consejos de uso meramente informativos sobre la aplicación. Además, se aconseja enviar a un correo electrónico específico con sugerencias para futuras mejoras o con planteamientos de diversos problemas que se encuentren los usuarios (figura B.33).

B.8.6. Cierre de sesión

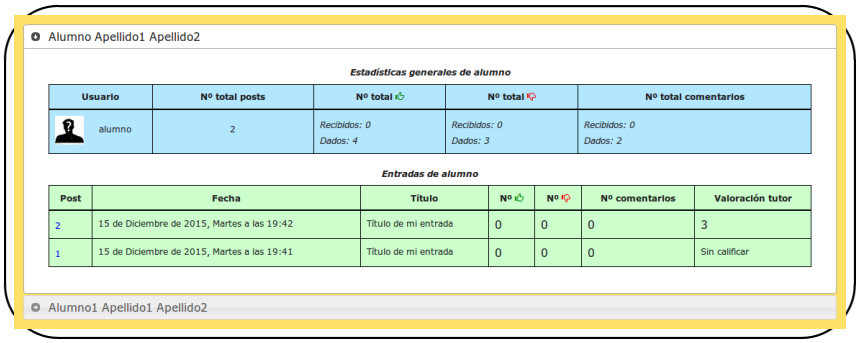
El enlace en rojo "Desconectar" expuesto en la parte derecha de la cabecera se usa para cerrar la sesión de usuario. Al presionarlo se saldrá de la aplicación y volverá a la página de inicio para logarse otra vez si se desea.
Si el usuario superpone el ratón encima del icono azul (a la derecha del enlace de desconectar), se despliega un popup con información del creador del hilo (nombre y apellidos, y correo B.34)



(a) Estadísticas generales.



(b) Resumen de entradas.



(c) Ambas opciones anteriores.

Figura B.32: *Visualización de opciones estadísticas.*

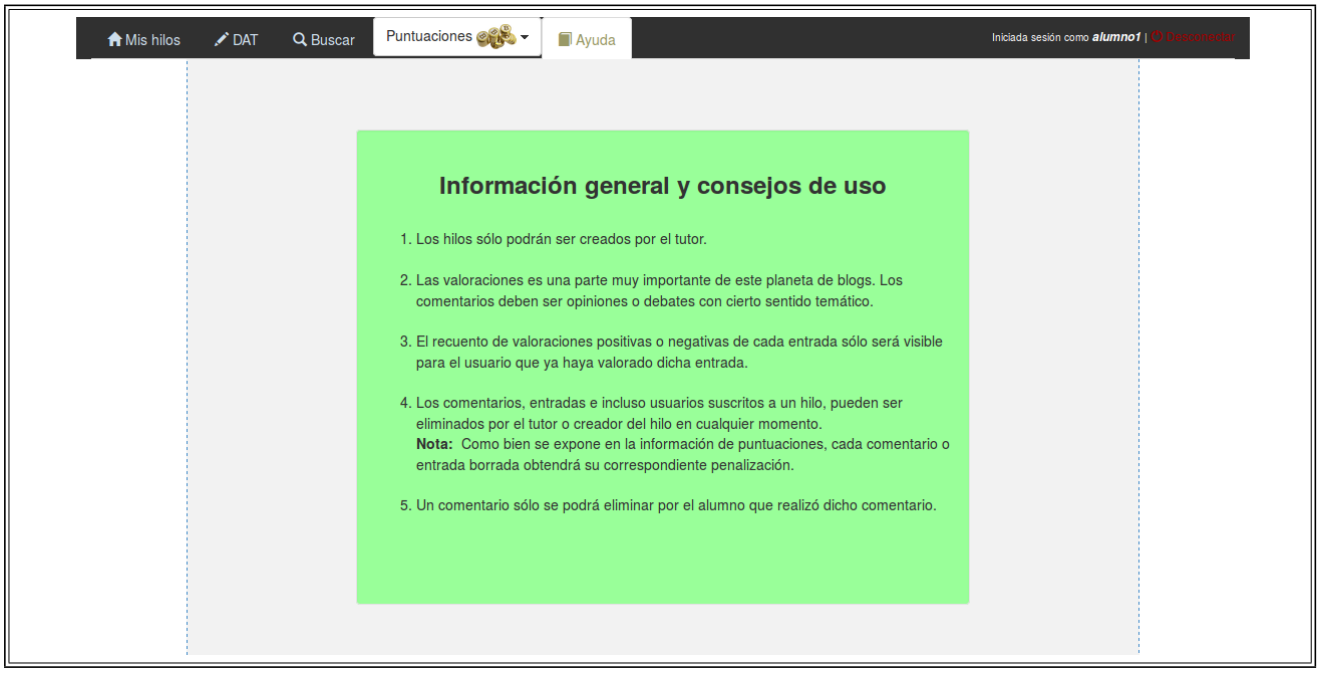


Figura B.33: Ayuda.

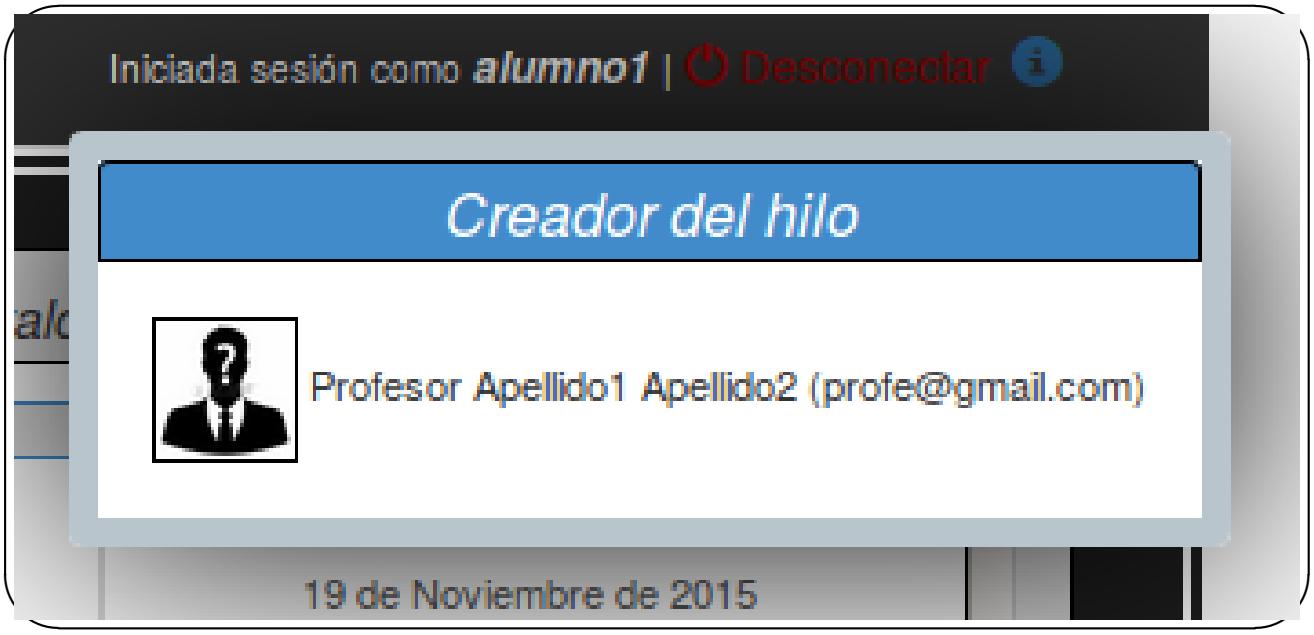


Figura B.34: Información del creador del hilo.

Apéndice C

Descarga de la aplicación

Recurso para la descarga del proyecto desde *GitHub*:
https://github.com/AdrianSaezClemente/PlanetaBlogs

Bibliografía

[1] Página oficial de *jQuery.*
http://jquery.com/

[2] Página oficial de *JSON.*
http://www.json.org/

[3] Página oficial de *BootStrap.*
http://getbootstrap.com/

[4] Página oficial de *Python.*
https://www.python.org/

[5] Página oficial de *Django.*
https://docs.djangoproject.com/

[6] Página oficial de *jQuery-ui.*
https://jqueryui.com/

[7] Recurso de descarga de la aplicación desde *GitHub.*
https://github.com/AdrianSaezClemente/PlanetaBlogs

[8] Página oficial de *Pivotal Tracker.*
https://www.pivotaltracker.com/

[9] Página oficial de *Git.*
http://git-scm.com/

[10] Tutorial de *CronTab.*
¿Cómo usar cron?

[11] Tutorial de *CSS*.

Manual web no oficial de CSS

[12] Plugin *BootStrapDialog*.

Descarga y manual de plugin BootStrapDialog