Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Herramienta para anotaciones de recursos Web

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Joan David Castelló Morales

Tutor: Félix Buendía García

Curso 2015-2016

Agradecimientos

En primer lugar, quería agradecer a mis padres todo el apoyo, tanto moral como económico, ya que sin ellos no habría sido posible estudiar esta carrera.

En segundo lugar, a mi tutor Félix Buendía por ser mi guía en el trabajo final de grado. Gracias a él y sus consejos he mantenido la motivación desde el principio hasta el final de esta etapa.

Por último, a todos los profesores y compañeros de carrera con los que he trabajado durante estos años, sin sus enseñanzas no habría llegado hasta aquí.

Resumen

El presente trabajo final de grado trata sobre el diseño e implementación de W&A (*Write and Annotate*), una aplicación web destinada a la creación de anotaciones sobre documentos en línea. Su principal objetivo es mejorar la interacción entre los profesores y sus alumnos, dándoles la capacidad de, por ejemplo, añadir comentarios explicativos en los textos académicos, o permitir que los estudiantes presenten y compartan sus dudas sobre algún fragmento en concreto.

Entre las funcionalidades implementadas destacan: la gestión y almacenamiento en bases de datos; un sistema de registro y autenticación de usuarios; un diseño sencillo, agradable y adaptable a cualquier dispositivo; y, por supuesto, la herramienta que permite realizar anotaciones en cualquier texto.

El potencial de esta aplicación reside en tres aspectos importantes. En primer lugar, se trata de un proyecto innovador, ya que el campo de las anotaciones en línea todavía es un terreno casi sin explotar. En segundo lugar, es una aplicación de futuro preparada para que la utilicen usuarios reales. Por último, cabe destacar la utilidad que aporta al ámbito académico a la hora de compartir y consultar información.

Palabras clave: aplicación web, anotaciones, documentos en línea, textos académicos.

Resum

El present treball de fi de grau tracta sobre el disseny i implementació de W&A (*Write and Annotate*), una aplicació web destinada a la creació d'anotacions sobre documents en línia. El seu principal objectiu és millorar la interacció entre els professors i els seus alumnes, donant-los la capacitat de, per exemple, afegir comentaris explicatius en els textos acadèmics, o permetre que els estudiants presenten i compartixquen els seus dubtes sobre algun fragment en concret.

Entre les funcionalitats implementades destaquen: la gestió i emmagatzemament en bases de dades; un sistema de registre i autenticació d'usuaris; un disseny senzill, agradable i adaptable a qualsevol dispositiu; i, per descomptat, la ferramenta que permet realitzar anotacions en qualsevol text.

El potencial d'aquesta aplicació resideix en tres aspectes importants. En primer lloc, es tracta d'un projecte innovador, ja que el camp de les anotacions en línia encara és un terreny quasi sense explotar. En segon lloc, es una aplicació de futur preparada perquè la

utilitzen usuaris reals. Per últim, cal destacar la utilitat que aporta a l'àmbit acadèmic a l'hora de compartir i consultar informació.

Paraules clau: aplicació web, anotacions, documents en línia, textos acadèmics.

Abstract

This final degree work treats the design and implementation of W&A (Write and Annotate), a web application destined to annotate online documents. Its main objective is to improve the interaction between professors and their pupils, giving them the ability, for example, to add explanatory comments into the academic texts, or let the students submit and share their doubts about some piece of text.

Among the implemented functions are included: management and storage in data bases; a registry and user authentication system; a simple, enjoyable and adaptable to any device design; and, of course, the tool which allows you to annotate any text.

The potential of this applications remains on three important aspects. Firstly, it's an innovative project, because the online annotations field is almost an unexploited terrain. Secondly, it is a future application prepared to be used by real users. Finally, is important to highlight the utility that provides to the academic scope on the spot of sharing and consulting information.

Keywords: web application, annotations, online documents, academic texts.

Tabla de contenidos

Índice de ilustraciones

1. Introducción

Este documento describe el trabajo realizado en el TFG titulado “herramienta para anotaciones de recursos web”. Su principal objetivo es mostrar al lector todo el desarrollo del proyecto, desde las fases iniciales hasta el final del mismo.

En este capítulo introductorio, se muestra una visión general sobre el estado actual de los mecanismos utilizados para anotar documentos, así como unos cuantos ejemplos que han servido de referencia para la implementar la aplicación protagonista de este proyecto. En adición, se explican los principales objetivos del trabajo final de grado y cómo se va a estructurar el resto de la memoria.

1.1. Contexto

En la actualidad, gracias a internet, compartir información se ha convertido en un hábito masivo para la humanidad. Foros, blogs, correo electrónico, sistemas de compartición de archivos… son solo unos pocos ejemplos de aplicaciones que permiten la comunicación entre usuarios utilizando imágenes, vídeo, audio y, sobre todo, texto.

En el ámbito de la lectura, la capacidad de compartir información textual en línea ha supuesto un avance tan enorme, que ya no es descabellado pensar que los documentos impresos acabarán siendo sustituidos por las publicaciones electrónicas.

Es cierto que muchísima gente sigue prefiriendo el tacto y la “cercanía” que ofrecen las páginas de papel frente a la lectura artificial tras una pantalla de ordenador. En cambio, no cabe duda que, con el paso del tiempo, los textos digitales han ido ganando cada vez más adeptos, gracias a las ventajas que ofrecen.

Un punto obvio es la ocupación de espacio. Mientras que en la versión de papel se necesita tener físicamente el documento, en un dispositivo electrónico se puede almacenar una biblioteca completa a la disposición del usuario.

Otro aspecto importante es la capacidad para mejorar la lectura de los textos digitales, gracias a multitud de aplicaciones o funciones software. Un ejemplo común es cambiar el tamaño de letra o ajustar el brillo de la pantalla según las necesidades del usuario.

También cabe destacar los mecanismos que permiten a los lectores de los textos electrónicos, añadir información que complemente la lectura, como son las anotaciones sobre fragmentos importantes del texto.

Las anotaciones son utilizadas habitualmente con objetivos didácticos. Por norma general, el lector escribe un comentario asociado a un fragmento de texto que le ha resultado interesante. Esta nota le puede servir en el futuro como recordatorio o simplemente como información adicional al texto.

En el papel, escribir notas a veces puede "ensuciar" el contenido, especialmente si el espacio donde anotar es limitado (suele ser así). Por otro lado, escribir comentarios o apuntes en textos digitales conlleva el uso interno de herramientas que evitan entorpecer la lectura del usuario. Por ejemplo, ocultando las anotaciones existentes en un lado de la pantalla o en elementos emergentes, resaltando ligeramente con colores los fragmentos de texto comentados…

Existen numerosas aplicaciones que dan soporte a las anotaciones de textos. A continuación se muestran algunas de ellas.

Microsoft Word

Esta aplicación creada por Microsoft es el procesador de textos más famoso y utilizado en la actualidad. Viene integrada en el paquete ofimático Microsoft Office y posee muchas opciones para manipular documentos. Entre la gran cantidad de funcionalidades, se encuentra la posibilidad de crear anotaciones.

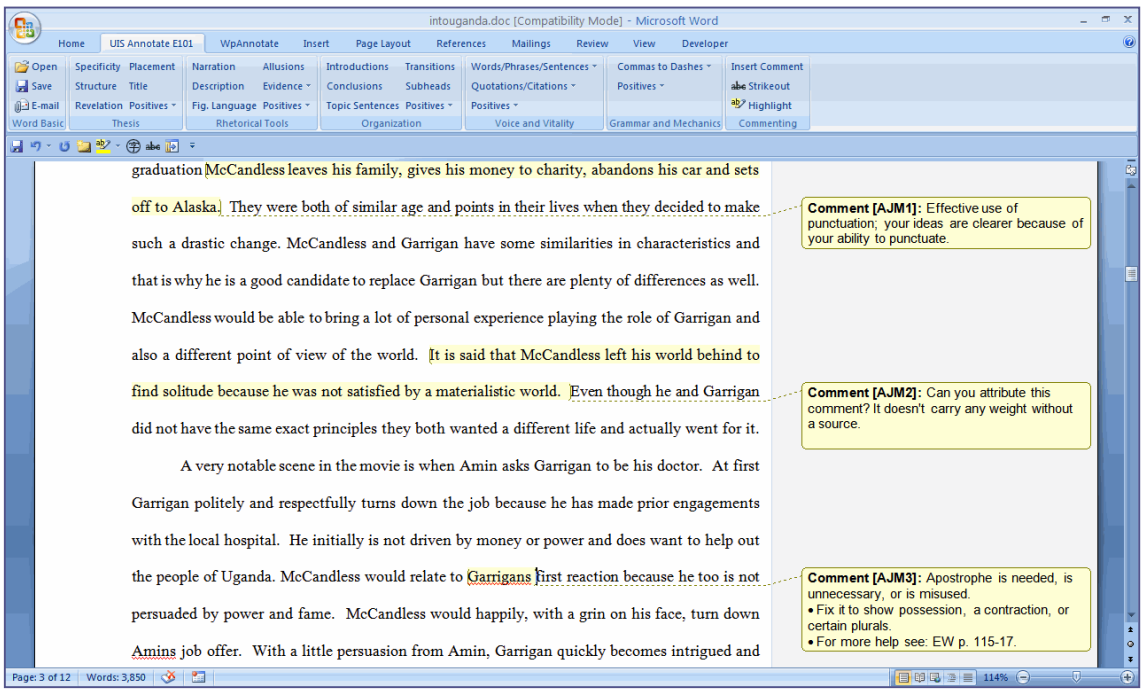

Ilustración 1 - Microsoft Word

Adobe Acrobat

Se trata de una familia de programas o aplicaciones informáticas desarrolladas por Adobe Systems, diseñados para visualizar, crear y modificar archivos con formato PDF (*Portable Document Format*). Aunque no es su función principal, Adobe Acrobat contiene herramientas sofisticadas que permiten la inserción de anotaciones sobre los documentos PDF.

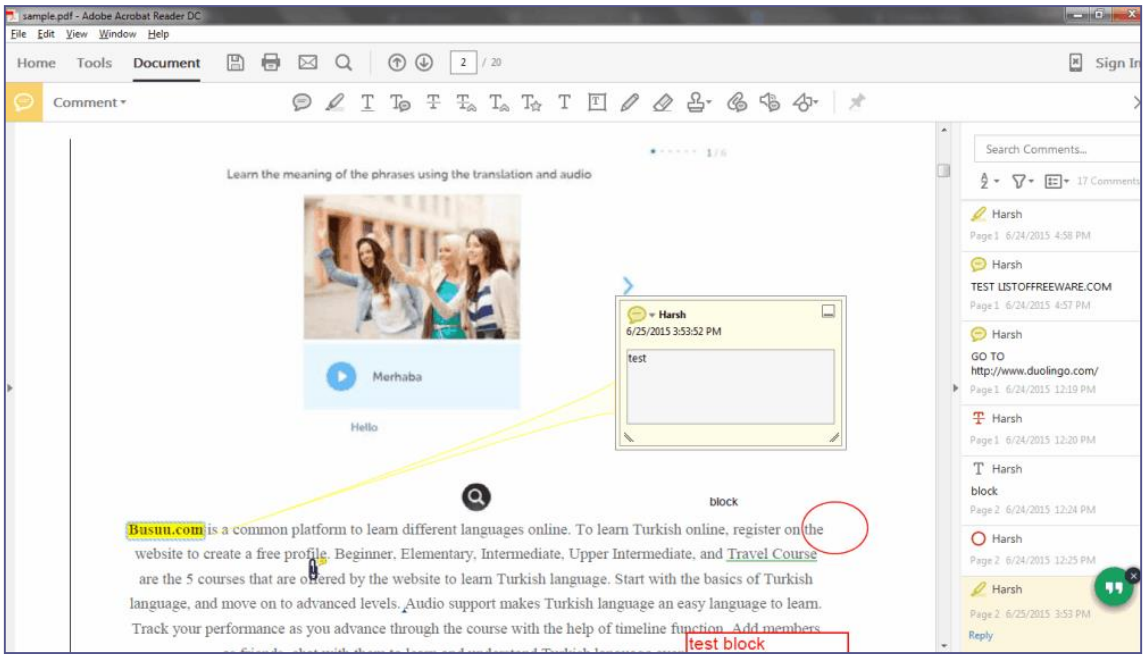

Ilustración 2 - Adobe Acrobat

Google Drive

Google Drive es un servicio de alojamiento de archivos creado por la empresa Google. Permite almacenar archivos en la nube, compartir ficheros y editar documentos de forma colaborativa con otros usuarios. Esta herramienta también permite crear documentos de texto, hojas de cálculo y presentaciones (entre otros) con el formato de Google. En los archivos de texto, se presenta la posibilidad de realizar anotaciones sobre los mismos, que pueden ser visualizadas por todos los usuarios que tengan permiso para colaborar en la edición del documento.

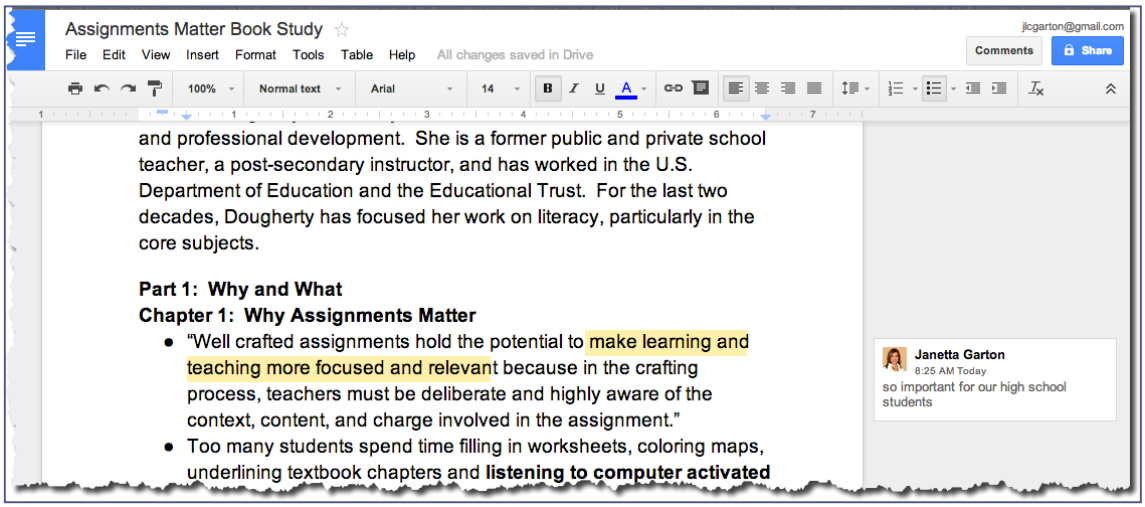

Ilustración 3 - Google Drive

Hypothes.is y Annotation Studio

Estas dos aplicaciones web son muy similares a la herramienta que se ha desarrollado en el trabajo final de grado. Ambas se basan en proyectos colaborativos para la recogida y manejo de anotaciones sobre documentos de texto en línea. Internamente están implementadas utilizando una librería javascript (Annotator.js) que proporciona la funcionalidad necesaria para anotar documentos.

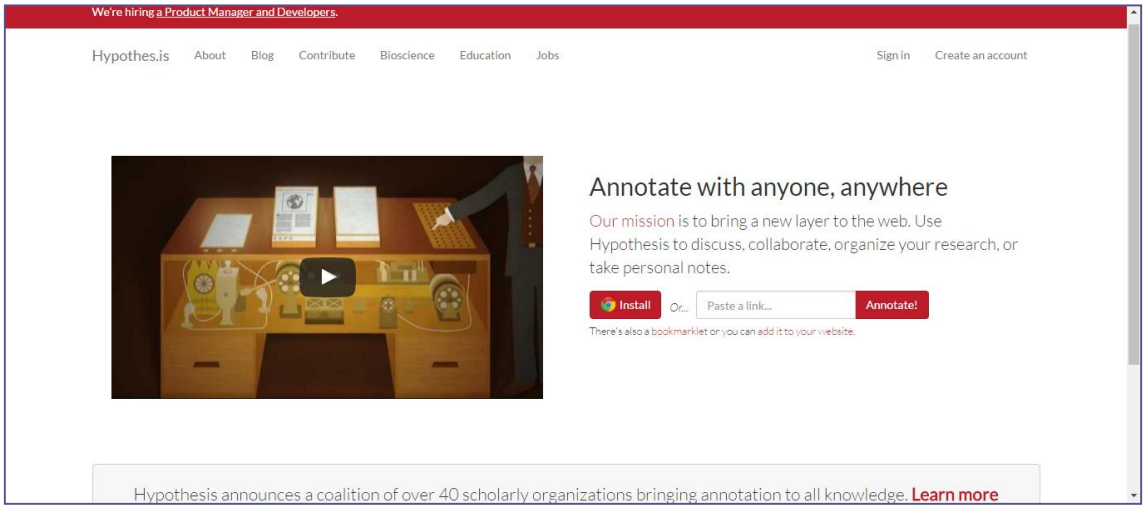

Ilustración 4 - Hypothes.is

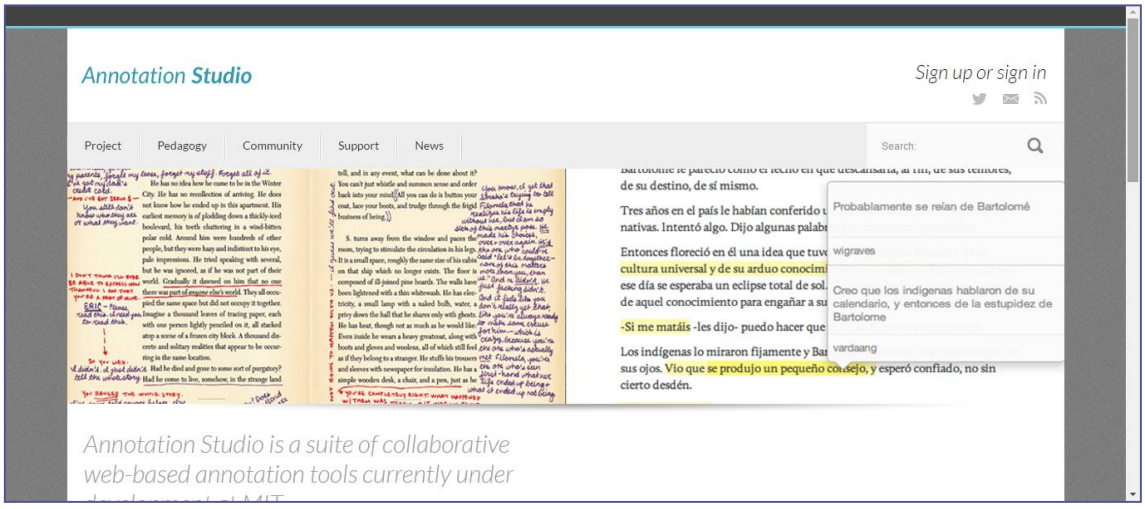

Ilustración 5 - Annotation Studio

Este tipo de herramientas colaborativas destinadas a crear anotaciones sobre textos ha despertado gran interés en el ámbito académico, especialmente entre los educadores, que han detectado un enorme potencial en ellas para la mejora y complementación de su trabajo educativo.

Según estudios realizados para comprobar los beneficios de utilizar este tipo de herramientas en el ámbito académico [4], donde se escogieron una gran cantidad de estudiantes para probar las aplicaciones, se confirmó que el uso de anotaciones proporciona varias mejoras en el proceso educacional de los alumnos.

En primer lugar, se comprobó que leer texto resaltado con colores despierta mayor interés entre los alumnos, en comparación con hacerlo de la manera habitual. Esto fomenta un mayor grado en la calidad del aprendizaje, especialmente si se utilizan las anotaciones para resaltar los puntos más importantes de un fragmento de texto.

En segundo lugar, se pudo observar una mayor comprensión lectora de los textos anotados respecto a aquellos que no poseían anotaciones. Utilizar comentarios explicativos en los fragmentos de texto más relevantes ayudaba a los lectores a asimilar mejor la información que tenían delante.

Otro punto a favor es el fomento de la búsqueda de información. Entre las anotaciones, se presentó a los estudiantes una serie de enlaces a otras páginas web, donde se podía encontrar información adicional sobre el tema del documento. El uso de comentarios a modo de referencias resulta de gran ayuda para aquellos alumnos que desean aprender más sobre la materia tratada.

En resumen, el uso colaborativo de anotaciones para el tratado de documentos en línea es un gran avance para el sistema educativo, ya que ofrece grandes ventajas tanto a profesores como a los alumnos. Entre ellas destacan las mejoras en: la calidad de aprendizaje, la motivación, la interacción alumno-educador y la facilidad para compartir información.

1.2. Objetivos

El principal objetivo de este trabajo final de grado es crear una aplicación destinada a escribir y compartir anotaciones sobre documentos de texto en línea. Se pretende que al final del proyecto esté disponible para ser utilizada por usuarios reales en alguna institución académica, donde los profesores, gracias a esta aplicación, puedan proporcionar información adicional sobre la materia impartida a sus alumnos.

Los usuarios pueden registrarse para acceder a la aplicación, donde se les da la oportunidad de crear documentos, para posteriormente comentar sobre ellos aquel contenido que les resulte interesante.

Entre las características principales que debe tener la aplicación destacan: facilidad de uso, gracias a una interfaz sencilla e intuitiva; utilidad, proporcionada por una buena cantidad de herramientas software y las funciones para manipular documentos y anotaciones; y usabilidad, para garantizar la satisfacción en el mayor número de usuarios posible.

La segunda meta del trabajo es aprender y afrontar el desarrollo de un proyecto completo desde el inicio al final, pasando por todas las etapas que lo componen: especificación de requisitos, análisis, diseño, implementación y evaluación. Además, desarrollar un trabajo de esta magnitud sirve para adquirir experiencia y competencias de cara a proyectos futuros.

1.3. Estructura de la memoria

En este documento se detalla el desarrollo completo del proyecto, empezando por la especificación de requisitos, donde se presenta una primera visión de los objetivos que debe cumplir la aplicación implementada. Este apartado sirve de punto de inicio al desarrollador para asentar las ideas y clarificar qué debe incluir la aplicación y como hacerlo.

En segundo lugar, se explica la fase de análisis, donde se detallan los casos de uso de la aplicación, identificando los actores y las acciones que puede realizar cada uno de ellos. También se muestran gráficamente las relaciones entre las clases y el flujo de mensajes dentro de la aplicación, mediante los diagramas de clases y de secuencia respectivamente.

Tras esto, se incide en la fase de diseño, en la cual se habla del tipo de arquitectura que sigue la aplicación y la interacción entre los distintos componentes que la forman, mostrando también algunos ejemplos de las funciones más relevantes.

En cuarto lugar, se da paso a la implementación. Esta etapa constituye el grueso de la memoria y del tiempo dedicado al proyecto. Se habla de las diferentes tecnologías y herramientas que se han utilizado para crear la aplicación. Además, se explican en detalle todas las operaciones que han permitido la implementación de la interfaz de usuario, la funcionalidad interna de la aplicación y la interacción con la bases de datos.

A continuación se explican los distintos métodos utilizados para evaluar el estado y el funcionamiento de la aplicación. Para ello se presentan una serie de pruebas típicas de la fase de evaluación de cualquier proyecto web, como son la comprobación de enlaces rotos o la validación de las hojas de estilo.

Por último, se expresan las conclusiones, tanto personales como técnicas, acerca de la aplicación y la etapa del trabajo final de grado. El documento se cierra con el listado de referencias bibliográficas consultadas en el transcurso del proyecto.

2. Especificación de requisitos

2.1. Introducción

Este apartado corresponde a la especificación de requisitos software (ERS) de la aplicación web destinada a realizar anotaciones. La presente ERS cumple las directrices establecidas por el estándar IEEE 830, aunque se han suprimido algunos puntos considerados como innecesarios.

2.1.1. Propósito

La ERS tiene como objetivo explicar las diferentes funcionalidades y requisitos de la aplicación, que deben estar presentes para satisfacer las necesidades de los usuarios. La audiencia de este escrito se compone principalmente del desarrollador software, el tutor del trabajo final de grado, el tribunal y los usuarios finales del sistema, es decir, los profesores y alumnos que utilicen la aplicación.

2.1.2. Ámbito

La aplicación de anotaciones web, que a partir de ahora llamaremos W&A (*Write & Annotate*), tiene fines didácticos y no lucrativos. Su principal objetivo es facilitar la interacción alumno-profesor, mediante la posibilidad de realizar comentarios sobre el texto de documentos académicos en línea.

En cuanto a los distintos tipos de usuarios que componen el sistema, destacan los usuarios no registrados o visitantes, los usuarios registrados y los administradores. A continuación se describen las acciones que pueden realizar cada uno de ellos:

- Usuarios no registrados o visitantes: pueden realizar la operación de registro y convertirse en usuarios registrados. También son capaces de visualizar la página de inicio de la aplicación. En cambio, no tienen acceso a las posibles operaciones sobre anotaciones y documentos (ver, crear, borrar y modificar).

- Usuarios registrados: pueden ser profesores o alumnos. Ambos tipos pueden visualizar todas las anotaciones y documentos generados. Respecto a los alumnos, únicamente se les permite trabajar con anotaciones propias, esto incluye las operaciones de creación, borrado y edición. Los profesores, en cambio, pueden eliminar anotaciones de cualquier usuario, y también tienen permiso para crear nuevos documentos y borrarlos.

- Administradores: tienen permiso para realizar cualquier acción sobre las anotaciones, los documentos y los usuarios registrados. Todas las operaciones las realizan directamente desde la base de datos, donde se guarda información.

2.1.3. Definiciones, siglas y abreviaturas

- Administrador: persona encargada de la gestión y mantenimiento de la aplicación web y las bases de datos.

- AJAX: *Asynchronous Javascript And XML*. Técnica de desarrollo web para crear aplicaciones dinámicas e interactivas.

- *Annotator*: librería javascript utilizada en la aplicación para añadir la funcionalidad de anotar textos.

- Anotación: apunte o comentario creado por los usuarios que se asocia a un fragmento de texto determinado.

- API: *Application Programming Interface*. Conjunto de funciones y procedimientos que ofrece cierta librería para ser utilizado por otro *software* como capa de abstracción.

- Aplicación web: herramienta que puede ser utilizada por los usuarios accediendo a un servidor web a través de internet.

- Base de datos: sistema encargado de la persistencia y almacenamiento de la información relativa a la aplicación.

- Bootstrap: conjunto de herramientas de código abierto destinado al diseño de páginas y aplicaciones web.

- CSS: *Cascade Style Sheets*. Lenguaje utilizado para definir y crear la presentación de un documento HTML.

- Documento: archivo de texto utilizado por los usuarios para añadir sus anotaciones.

- *Framework*: estructura conceptual y tecnológica de soporte definido utilizada para la organización y desarrollo de *software*.

- HTML: *HyperText Mark Language*. Lenguaje de marcas de texto que sirve de referencia para la elaboración de páginas web.

- HTTP: *HyperText Transfer Protocol*. Protocolo de comunicación que permite la transferencia de información en la web.

- IEEE: *Institute of Electrical and Electronic Engineers*. Asociación mundial de ingenieros dedicada a la estandarización y desarrollo en áreas técnicas.

- Interfaz: conjunto de elementos visibles para el usuario con los que puede interactuar con el objetivo de efectuar acciones en el sitio web.

- Javascript: lenguaje de programación interpretado.

- JQuery: librería javascript que permite simplificar la manera de interactuar con los documentos HTML.

- *Login*: acción realizada por el usuario para identificarse dentro de la aplicación web, proporcionando para ello sus credenciales (nombre de usuario y contraseña).

- Método: conjunto de líneas de código encargado de ofrecer una función determinada.
- Mongo DB: sistema de bases de datos orientado a documentos, desarrollado bajo el concepto de código abierto.
- PHP: lenguaje de programación utilizado en el lado del servidor y diseñado para el desarrollo web de contenido dinámico.
- *Plugin*: complemento que se relaciona con otro programa para añadirle una funcionalidad específica.
- Registro: acción realizada por un usuario para almacenar sus datos y formar parte de la aplicación web.
- *Responsive*: propiedad asignada a una página web cuando ésta es capaz de adaptarse al tamaño de cualquier pantalla o dispositivo sin que esto afecte negativamente al contenido de la web.
- *Script*: programa (código) almacenado en un archivo de texto.
- Silex: *framework* escrito en PHP para la creación de aplicaciones web. Se basa en un sistema de rutas que facilita el manejo de controladores y funciones para cada método HTTP.
- URL: *Uniform Resource Locator*. Dirección utilizada por los navegadores para encontrar recursos web.
- Usuario: cualquier persona que hace uso de la aplicación.
- W&A: *Write and Annotate*, es el nombre de la aplicación web implementada en este trabajo final de grado.

2.1.4. Visión general

En los siguientes apartados de la ERS se da, en primer lugar, una descripción general de la aplicación, donde se explica la perspectiva de la misma, las diferentes funciones que se pueden realizar, el perfil de los usuarios a los que va destinado el producto y las principales limitaciones que se imponen en la etapa de desarrollo. En segundo lugar se describe el conjunto de requisitos específicos que deben satisfacerse en la implementación del sistema. Estos requisitos hacen referencia a la interfaz, las funciones, el rendimiento y los atributos del sistema.

2.2. Descripción general

2.2.1. Perspectiva del producto

W&A interacciona con las tecnologías software listadas a continuación:

- *Annotator*: librería escrita en javascript que constituye el eje central de la aplicación. Con ella se implementa la funcionalidad para trabajar con anotaciones.

- Base de datos: fundamental en el almacenamiento de la información referente a los usuarios, los documentos y las anotaciones creadas. Se utiliza MongoDB para cumplir con este objetivo.

- Servidor web: alojamiento de la aplicación con la finalidad de permitir el acceso de los usuarios.

- Lenguajes de programación: se utiliza HTML, CSS, javascript y PHP para el desarrollo del producto.

- Eclipse: plataforma utilizada en la fase de desarrollo de la aplicación para la escritura y organización de código.

2.2.2. Funciones del producto

- Visualización e interacción con la página de inicio de la aplicación: cuando un usuario accede a la web de W&A se le muestra una página inicial en la que puede navegar por sus menús y obtener información relativa al origen de la aplicación, así como una pequeña guía de uso del *annotator*.

- Registro: un usuario no registrado puede pasar a formar parte del sistema. Para ello debe introducir sus datos personales y unas credenciales que le permitan acceder a la aplicación en futuras ocasiones.

- Identificación o *login*: un usuario registrado puede acceder a la aplicación mediante esta opción, proporcionando su nombre de usuario (correo electrónico) y su contraseña.

- Cerrar sesión o *logout*: un usuario identificado puede cerrar su sesión de usuario con esta opción, volviendo así a la página de inicio de la aplicación.

- Visualización y edición del perfil de usuario: cada usuario identificado puede acceder a su perfil personal para comprobar sus datos personales y modificarlos.

- Listado de documentos/anotaciones: los usuarios identificados pueden ver un listado completo de los documentos y anotaciones existentes.

- Crear documentos: un usuario identificado como profesor puede generar un nuevo documento anotable, subiéndolo a la web directamente desde su ordenador personal.

- Compartir documentos: un usuario identificado puede elegir con que usuarios compartir

sus propios documentos e indicar los permisos de interacción con ellos.

- Borrar documentos: los usuarios identificados como profesores pueden eliminar aquellos documentos que tengan permiso para hacerlo.

- Comparar documentos: los usuarios identificados conservan un historial de versiones de los documentos para poder comparar como han ido evolucionando sus cambios en el tiempo.

- Realizar anotaciones: los usuarios identificados pueden crear anotaciones sobre un documento determinado, seleccionando para ello un fragmento del texto que quieran comentar.

- Editar anotaciones: los usuarios identificados pueden modificar aquellas anotaciones en las cuales tienen permiso para hacerlo.

- Borrar anotaciones: los usuarios identificados pueden eliminar aquellas anotaciones en las cuales tienen permiso para hacerlo.

- Compartir anotaciones: los usuarios identificados y autores de las anotaciones tienen la capacidad de elegir el grado de privacidad de las mismas, pudiendo así compartirlas con otros usuarios y dándoles permisos de edición y/o borrado.

Se explican con más detalle las diferentes funciones de la aplicación en la sección 2.3.2. de este documento.

2.2.3. Características de los usuarios

Los usuarios principales de la aplicación son los alumnos y profesores de alguna institución académica (en este caso podría ser un centro de idiomas). El perfil medio de los alumnos debería ser constituido por jóvenes entre 18 y 30 años, aunque siempre hay excepciones de edad. También cabe resaltar que deberían ser novatos en el idioma que pretenden aprender. En cambio los profesores, por norma general deberían ser personas de edad más avanzada y con extensos conocimientos en el idioma que se imparte en los cursos. Los alumnos y profesores pueden realizar las acciones indicadas en el punto 2.2.1. de este mismo documento.

El resto de usuarios son personas no registradas en el sistema (o no identificadas), y su perfil abarca todas las variantes: desde niños que entran en la web por curiosidad hasta ancianos que se interesan por lo que están estudiando sus nietos, etc. Estos usuarios no tienen permisos para trabajar con los documentos y/o las anotaciones, como se ha visto anteriormente en el punto 2.2.1.

El principal propósito de la aplicación es **facilitar** la interacción del alumno y el profesor, por tanto, es de vital importancia el desarrollo de una interfaz intuitiva, sencilla de manejar, fácil de aprender y en la que se utilice un lenguaje formal, pero no demasiado técnico.

2.2.4. Restricciones

- La disponibilidad de W&A debe ser casi absoluta, excepto por tareas de mantenimiento o por cortes en la red de telecomunicaciones.

- Utiliza el protocolo de comunicaciones TCP/IP.

- La aplicación utiliza un protocolo robusto de autenticación para garantizar la seguridad de los usuarios. Todos ellos disponen de un nombre de usuario (su correo electrónico) y una contraseña de acceso.

- Toda la información privada de los usuarios sólo puede ser vista por ellos mismos. Es muy importante mantener la privacidad para cumplir con la Ley Oficial de Protección de Datos (LOPD).

- No hay ningún problema para acceder a la web con cualquier ordenador o dispositivo que disponga de una conexión a internet y tenga instalados alguno de los navegadores más conocidos como son *Internet Explorer*, *Mozilla Firefox* o *Google Chrome*.

2.3. Requisitos específicos

En este apartado se describen con un mayor nivel de detalle todos los aspectos referentes a los requisitos que debe cumplir la aplicación, incluyendo las características y bocetos de la interfaz; la lista completa de funcionalidades con sus elementos de entrada, salida y restricciones; los factores que afectan al rendimiento de W&A; y los atributos del sistema, como la seguridad y el mantenimiento.

2.3.1. Interfaces externas

Como se ha indicado anteriormente, la interfaz de usuario debe ser sencilla de aprender y manejar. También es importante no incluir elementos que despisten al usuario de su tarea principal en la aplicación, que es el tratado de documentos y anotaciones. Esto implica no sobrecargar la interfaz con imágenes y/o elementos móviles, utilizar fondos y colores sencillos, definir claramente la posición de los elementos de la interfaz, separar los bloques de contenido con un espaciado, colocar la información más relevante a la vista del usuario en todo momento y utilizar texto legible (formato, tamaño, color, contraste…) y sin errores ortográficos.

También es interesante que la interfaz pueda adaptarse, en cuestión de tamaño y estilo, al dispositivo desde el cuál un usuario accede a la aplicación. Por lo tanto se tienen en cuenta estas características de adaptabilidad para cumplir con la filosofía del *Responsive Web Design* (RWD).

Para los primeros bocetos de la página web se ha utilizado Pencil, una herramienta de prototipado muy sencilla e intuitiva de utilizar. Se muestran a continuación.

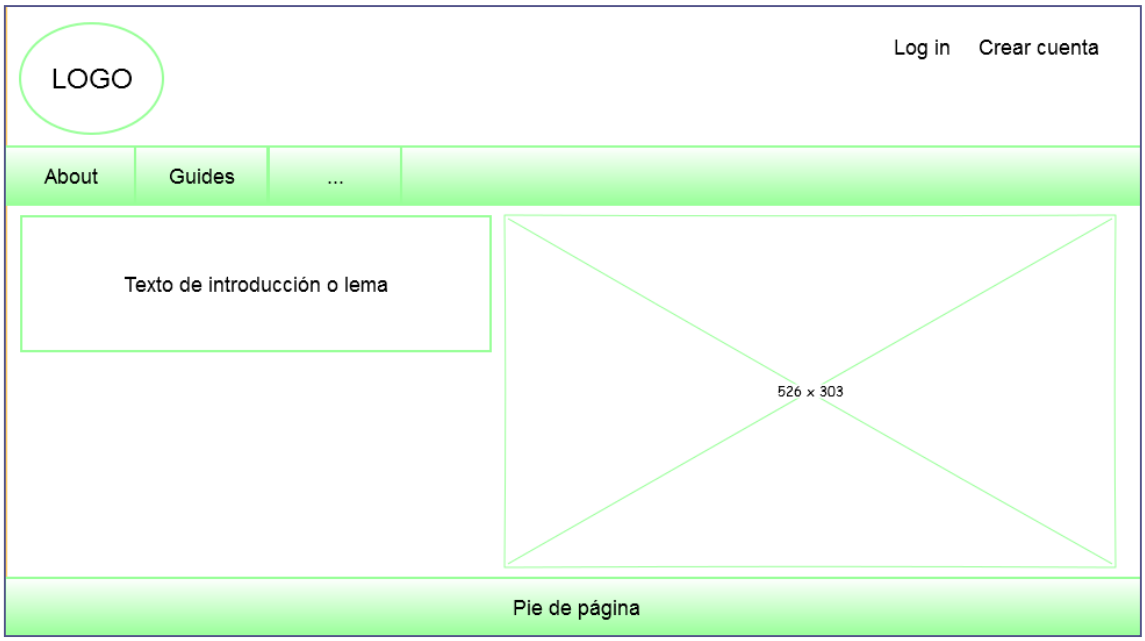

Ilustración 6 - Boceto de la página de inicio

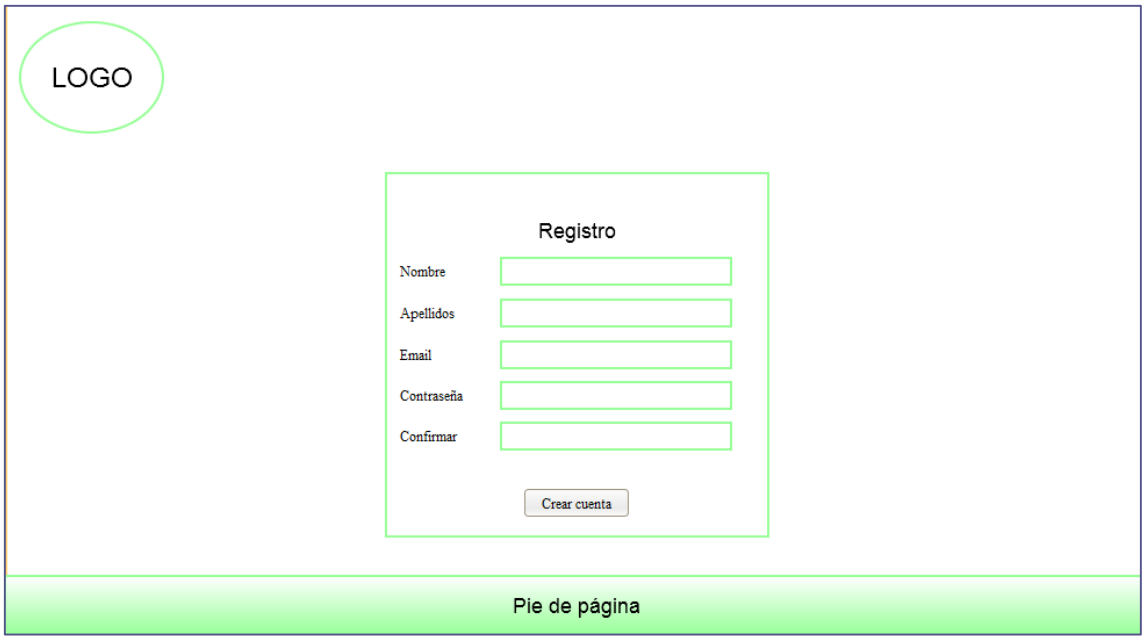

Ilustración 7 - Boceto de la página de registro

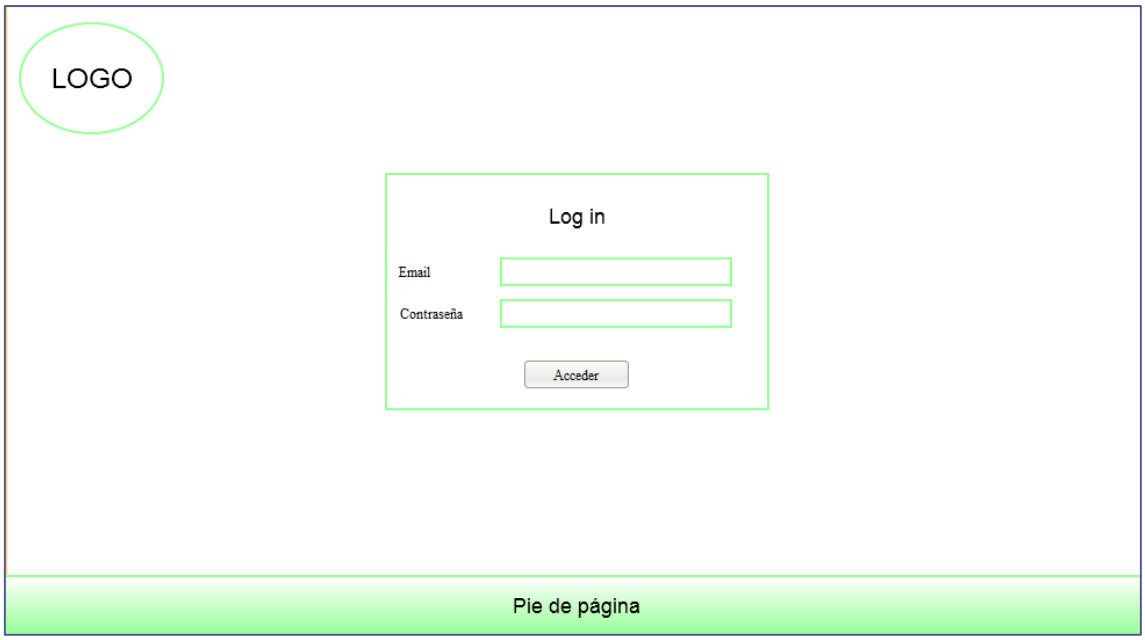

Ilustración 8 - Boceto de la página de *login*

Ilustración 9 - Boceto de la página de listados

Ilustración 10 - Boceto de la página del documento

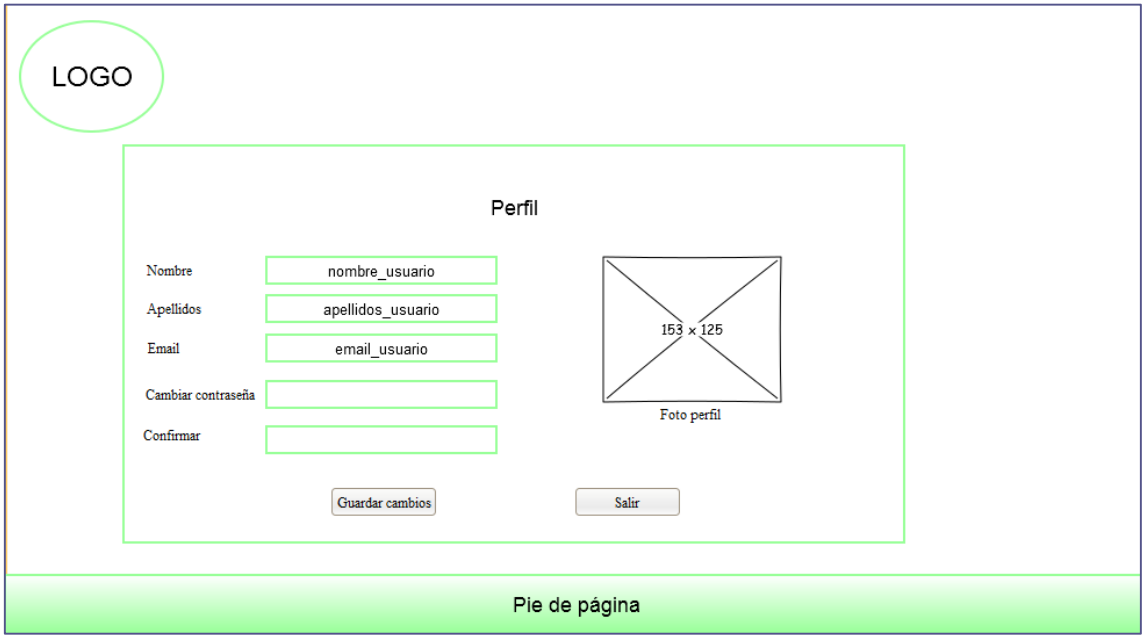

Ilustración 11 - Boceto de la pagina del perfil

2.3.2. Funciones

En esta sección se listan en su totalidad las funciones que se pueden realizar con W&A, profundizando un poco más en la información que se ha proporcionado en el apartado 2.2.2. de la ERS. Para diferenciar las funciones, se separan dependiendo del tipo de usuario de la aplicación.

Usuario no registrado o visitante

Función 1	**Visualización de la página de inicio**
Entrada	URL de la página web W&A.
Proceso	El usuario accede a la página web desde su navegador.
Salida	Se muestra la página de inicio de W&A.
Restricciones	Sin restricciones.

Función 2	**Interacción con las opciones del menú de inicio**
Entrada	Clic en los enlaces del menú de inicio.
Proceso	El usuario accede a otra página donde se muestra información relativa a la opción seleccionada.
Salida	Se muestra la página con la respectiva información de la pestaña seleccionada del menú.
Restricciones	Sin restricciones.

Función 3	**Registro**
Entrada	Campos del formulario de registro.
Proceso	Tras acceder al formulario de registro desde la página inicial, el usuario rellena todos los campos y envía la solicitud de registro al servidor, clicando en un botón diseñado para ello.
Salida	Se informa al usuario sobre el éxito o fallo en el registro. En caso de éxito, se inserta el usuario en la base de datos.
Restricciones	Los campos del formulario son obligatorios. No puede usarse un correo que no sea válido o que ya esté registrado en el sistema. La contraseña debe tener cierta longitud y ser igual que su respectiva confirmación.

Usuario registrado (alumno)

Función 4	**Identificación o *login***
Entrada	Campos del formulario de *login*.
Proceso	Tras acceder al formulario de identificación desde la página inicial, el usuario indica su correo electrónico y contraseña, y envía la solicitud de *login* al servidor.
Salida	Si la identificación es correcta, se muestra la página con los listados de documentos y anotaciones. En cambio si la identificación no es correcta, se informa con un mensaje de error.
Restricciones	Los campos del formulario son obligatorios. El correo proporcionado debe estar registrado en el sistema y la contraseña debe ser correcta.

Función 5	Cerrar sesión o *logout*
Entrada	Clic y confirmación de la opción *logout*.
Proceso	El usuario usa el enlace de "cerrar sesión" y se le pregunta si está seguro de realizar esa acción. Si la respuesta es afirmativa, se desconecta la sesión y vuelve a la página inicial. Si la respuesta es negativa, permanece en la misma página.
Salida	Cierre de sesión y vuelta a la página de inicio de la aplicación
Restricciones	El usuario debe tener iniciada la sesión.

Función 6	Visualización del perfil de usuario
Entrada	Clic en el botón asociado al perfil.
Proceso	El usuario accede a sus datos personales tras clicar la opción "perfil".
Salida	Se muestra la página de perfil.
Restricciones	El usuario debe tener iniciada la sesión.

Función 7	Edición del perfil de usuario
Entrada	Nuevos datos en los campos relativos al perfil de usuario.
Proceso	El usuario modifica la información de su perfil ingresando sus nuevos datos y guardando los cambios.
Salida	Se muestra una confirmación de la información guardada y se modifica la fila correspondiente en la base de datos.
Restricciones	El usuario debe tener iniciada la sesión.

Función 8	Listado de documentos
Entrada	Identificarse (función 4).
Proceso	Tras identificarse, se muestra la página interna de la aplicación. A la izquierda de la misma se sitúa el listado completo de documentos.
Salida	Listado de documentos extraídos de la base de datos.
Restricciones	El usuario debe tener iniciada la sesión.

Función 9	Listado de anotaciones
Entrada	Identificarse (función 4).
Proceso	Tras identificarse, se muestra la página interna de la aplicación. A la derecha de la misma se sitúa el listado completo de anotaciones.
Salida	Listado de anotaciones extraídas de la base de datos.
Restricciones	El usuario debe tener iniciada la sesión.

Función 10	Búsqueda de documentos
Entrada	Cadena de texto en el cuadro de búsqueda de documentos.
Proceso	El usuario escribe texto en el cuadro de búsqueda situado en la parte superior de la tabla de documentos y envía el formulario al servidor pulsando el botón de búsqueda. Los documentos se filtran por título o por autor.
Salida	Listado de documentos filtrados, extraídos de la base de datos.
Restricciones	El usuario debe tener iniciada la sesión.

Función 11	Búsqueda de anotaciones
Entrada	Cadena de texto en el cuadro de búsqueda de anotaciones.
Proceso	El usuario escribe texto en el cuadro de búsqueda situado en la parte superior de la tabla de anotaciones y envía el formulario al servidor pulsando el botón de búsqueda. Las anotaciones se filtran por contenido o por autor.
Salida	Listado de anotaciones filtradas, extraídas de la base de datos.
Restricciones	El usuario debe tener iniciada la sesión.

Función 12	Visualizar documentos
Entrada	Clic en el enlace asociado a un documento.
Proceso	El usuario selecciona uno de los documentos de la lista y es redirigido a una nueva página donde se muestra el contenido de dicho documento y las anotaciones correspondientes.
Salida	Se abre una nueva página con el contenido del documento.
Restricciones	El usuario debe tener iniciada la sesión.

Función 13	Añadir anotaciones
Entrada	Comentario sobre un fragmento de texto.
Proceso	El usuario selecciona con el ratón una parte del texto, entonces se abre una interfaz donde puede escribir cualquier comentario y guardarlo.
Salida	Se incluye la nueva anotación en la lista y en la base de datos. Además se cambia el color del texto anotado.
Restricciones	El usuario debe tener iniciada la sesión.

Función 14	Editar anotaciones propias
Entrada	Nuevo comentario sobre una anotación existente.
Proceso	El usuario selecciona con el ratón una anotación escrita por él mismo y se muestra de nuevo la interfaz para anotar. Tras esto modifica el contenido del comentario y guarda los cambios.
Salida	Se cambia el contenido de la anotación en el texto, la lista de anotaciones y se edita la fila correspondiente en la base de datos.
Restricciones	El usuario debe tener iniciada la sesión y ser propietario de la anotación.

Función 15	Borrar anotaciones propias desde el texto
Entrada	Selección y clic en la opción de borrado de una anotación existente.
Proceso	El usuario selecciona con el ratón una anotación escrita por él mismo y se muestra de nuevo la interfaz para anotar. Tras esto pulsa el botón de borrado.
Salida	Se elimina la anotación en el texto (volviendo al color original), en la lista de anotaciones y en la base de datos.
Restricciones	El usuario debe tener iniciada la sesión y ser propietario de la anotación.

Función 16	Borrar anotaciones propias desde la lista
Entrada	Clic y confirmación de la opción de borrado asociada a una anotación en la lista.
Proceso	El usuario clica en el botón de borrado situado a la derecha de cada anotación de la lista. Entonces se abre un cuadro de confirmación que puede ser aceptado o rechazado.
Salida	Se elimina la anotación en el texto (volviendo al color original), la lista de anotaciones y en la base de datos.
Restricciones	El usuario debe tener iniciada la sesión y ser propietario de la anotación.

Usuario registrado (profesor)

Función 17	Crear documentos
Entrada	Clic en el botón para "subir" documentos
Proceso	El usuario clica en el botón de subida situado en la parte superior de la tabla de documentos y elige un archivo de su espacio personal para añadirlo a la aplicación.
Salida	Nuevo documento insertado en la lista y en la base de datos.
Restricciones	El usuario debe tener iniciada la sesión. El documento debe tener formato de texto.

Función 18	Borrar cualquier documento
Entrada	Clic y confirmación de la opción de borrado asociada a un documento.
Proceso	El usuario clica en el botón de borrado situado a la derecha de cada documento en la lista. Entonces se abre un cuadro de confirmación que puede ser aceptado o rechazado.
Salida	El documento se borra de la lista y de la base de datos.
Restricciones	El usuario debe tener iniciada la sesión.

Función 19	Borrar cualquier anotación desde la lista
Entrada	Clic y confirmación de la opción de borrado asociada a una anotación en la lista.
Proceso	El usuario clica en el botón de borrado situado a la derecha de cada anotación de la lista. Entonces se abre un cuadro de confirmación que puede ser aceptado o rechazado.
Salida	Se elimina la anotación en el texto (volviendo al color original), en la lista de anotaciones y en la base de datos.
Restricciones	El usuario debe tener iniciada la sesión.

Además puede realizar todas las funciones del usuario alumno.

Administrador

Función 20	Consulta a la base de datos
Entrada	Escritura de los comandos de consulta en un terminal con acceso a la base de datos.
Proceso	El administrador escribe en el terminal la operación de consulta sobre las colecciones de usuarios, documentos o anotaciones y se visualiza el resultado.
Salida	Listado de elementos (filas o colecciones) extraídos de la base de datos.
Restricciones	Sin restricciones.

Función 21	Inserción en la base de datos
Entrada	Escritura de los comandos de inserción en un terminal con acceso a la base de datos.
Proceso	El administrador escribe en el terminal la operación de inserción sobre las colecciones de usuarios, documentos o anotaciones y se añaden nuevas filas.
Salida	Se inserta una nueva fila en la colección correspondiente de la base de datos.
Restricciones	El identificador del elemento insertado debe ser nuevo.

Función 22	Edición en la base de datos
Entrada	Escritura de los comandos de edición en un terminal con acceso a la base de datos.
Proceso	El administrador escribe en el terminal la operación de edición sobre las colecciones de usuarios, documentos o anotaciones y se modifican las filas correspondientes.
Salida	Se modifica la fila deseada en la base de datos con la nueva información indicada.
Restricciones	El elemento a editar debe existir en la base de datos.

Función 23	Borrado en la base de datos
Entrada	Escritura de los comandos de borrado en un terminal con acceso a la base de datos.
Proceso	El administrador escribe en el terminal la operación de borrado sobre las colecciones de usuarios, documentos o anotaciones y se eliminan las filas correspondientes.
Salida	Se elimina la fila deseada en la colección de la base de datos.
Restricciones	El elemento a borrar debe existir en la base de datos.

2.3.3. Requisitos de rendimiento

En el peor de los casos, W&A tiene que soportar la carga de todos los alumnos y profesores de los cursos trabajando con ella simultáneamente, un caso altamente improbable. Como estimación, la aplicación soportaría la mayoría del tiempo a menos de quince personas trabajando a la vez. Para este escenario mucho más realista, la aplicación

debe ofrecer unos tiempos de respuesta bastante aceptables (menor a diez segundos).

Respecto a la base de datos, es donde se guarda la información de todos los usuarios, documentos y anotaciones, por tanto debe tener una capacidad de almacenamiento elevada (del orden de miles de registros), ya que se prevé el uso frecuente de la misma.

2.3.4. Atributos del sistema

Para terminar con la redacción de la ERS, es recomendable mostrar algún aspecto importante sobre el sistema, como la seguridad. En el caso de esta aplicación, la protección de los datos de usuario se implementa mediante un mecanismo de credenciales. El usuario se identifica utilizando su nombre único (correo electrónico) y su contraseña. Solamente los usuarios registrados e identificados tienen acceso a las funciones relacionadas con el tratamiento de documentos y anotaciones (visualización, creación, edición y borrado).

En cuanto a la seguridad de la base de datos, el administrador es el único que tiene acceso a la misma y a las operaciones permitidas sobre ella, por tanto debe ser la persona responsable de su mantenimiento.

3. Análisis

3.1. Introducción

Este capítulo tiene como principales objetivos analizar los requisitos presentados en la ERS y mostrar de manera gráfica como se relacionan entre ellos. Para tal fin se han construido un conjunto de modelos que ayudan a entender el funcionamiento de la aplicación. Dichos modelos utilizan UML (*Unified Modeling Language*), un lenguaje gráfico, creado para visualizar, especificar, construir y documentar un sistema software. Es el más conocido y utilizado en la actualidad para tareas de modelado.

En los siguientes puntos se muestran los diferentes diagramas creados para esta fase de análisis, entre los que destacan el diagrama de casos de uso, el de clases y el de secuencia. Los dos primeros modelos han sido implementados con UMLet, una herramienta gratuita dedicada a tal propósito. Por otro lado, los diagramas de secuencia han sido desarrollados haciendo uso de WebSequenceDiagrams, una aplicación en línea de libre acceso que se puede encontrar siguiendo este enlace: https://www.websequencediagrams.com.

3.2. Casos de uso

Los casos de uso describen el conjunto de pasos o actividades que debe realizar un actor para llevar a cabo una acción determinada dentro del sistema. Un actor es un usuario que interactúa con el sistema. En este caso, los actores son: usuarios no registrados (o visitantes), alumnos, profesores y administradores.

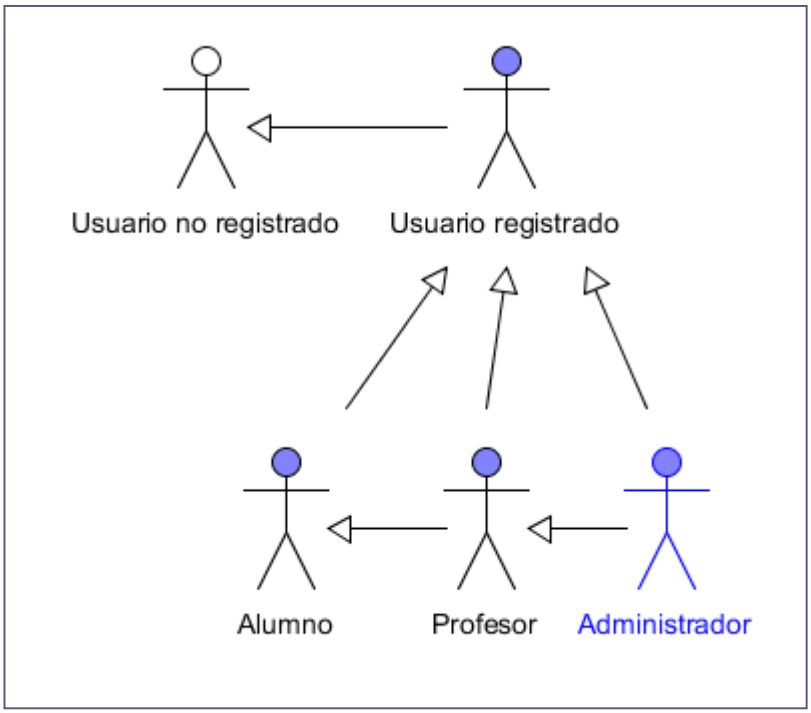

Ilustración 12 - Relación de actores UML

En la ilustración 12 se puede apreciar cómo funciona la herencia entre los distintos actores. Un usuario registrado puede realizar diversas acciones, extendiendo las capacidades que posee un usuario visitante. Existen tres tipos de usuarios registrados: alumnos, profesores y administradores. Cada uno de ellos tiene un mayor grado de actuación respecto al anterior, como puede observarse en la figura.

Siguiendo con el modelado, ahora se muestra como los actores interaccionan con las funcionalidades explicadas en el apartado 2.3.2 de este documento. Para ello se ha dibujado un diagrama para cada tipo de actor.

Usuario no registrado (o visitante)

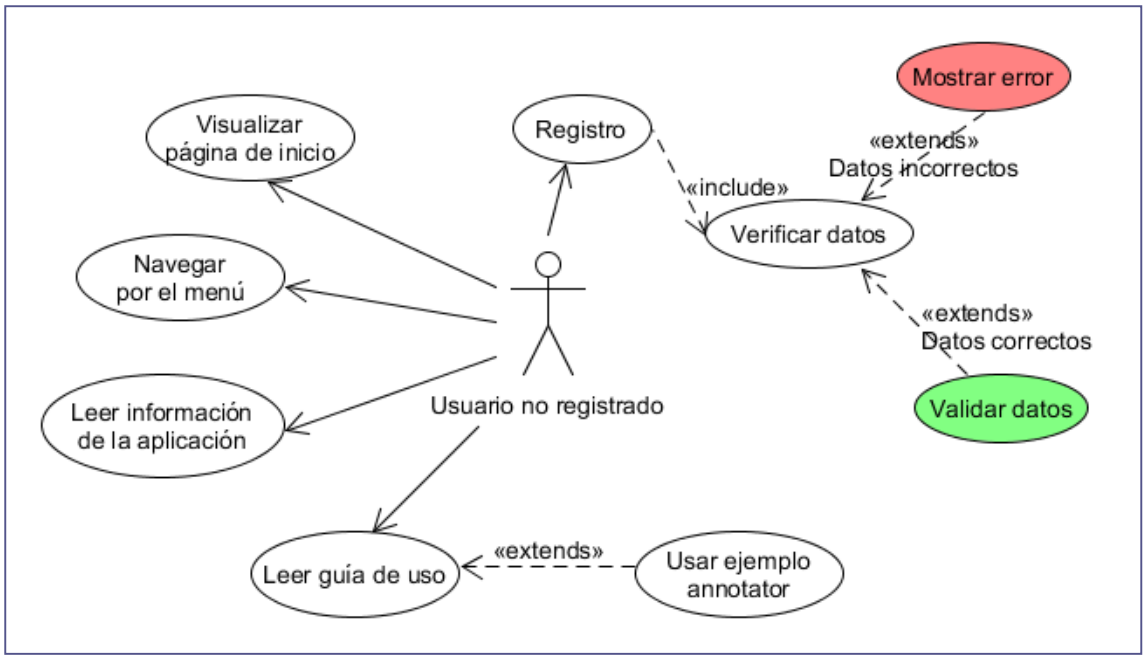

Ilustración 13 - Diagrama de casos de uso (Usuario no registrado)

Los casos de uso “leer información de la aplicación” y “leer guía de uso” forman parte de las opciones del menú de inicio.

Usuario alumno

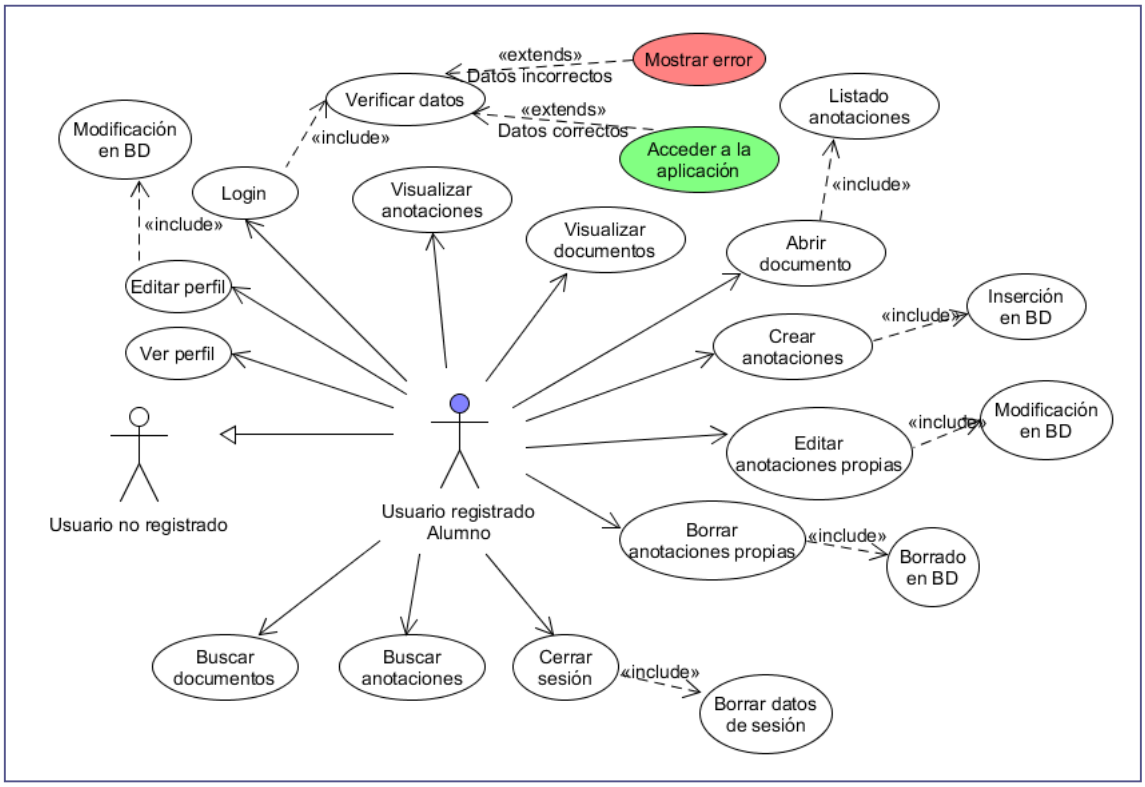

Ilustración 14 - Diagrama de casos de uso (Alumno)

Usuario profesor

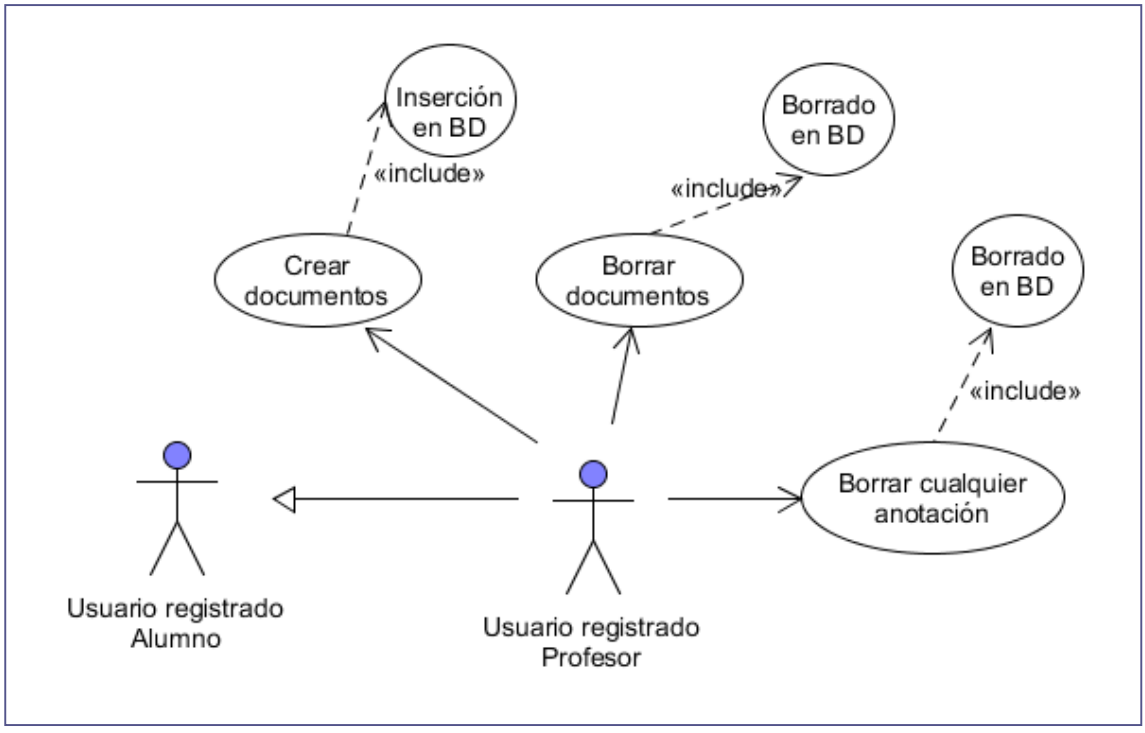

Ilustración 15 - Diagrama de casos de uso (Profesor)

Administrador

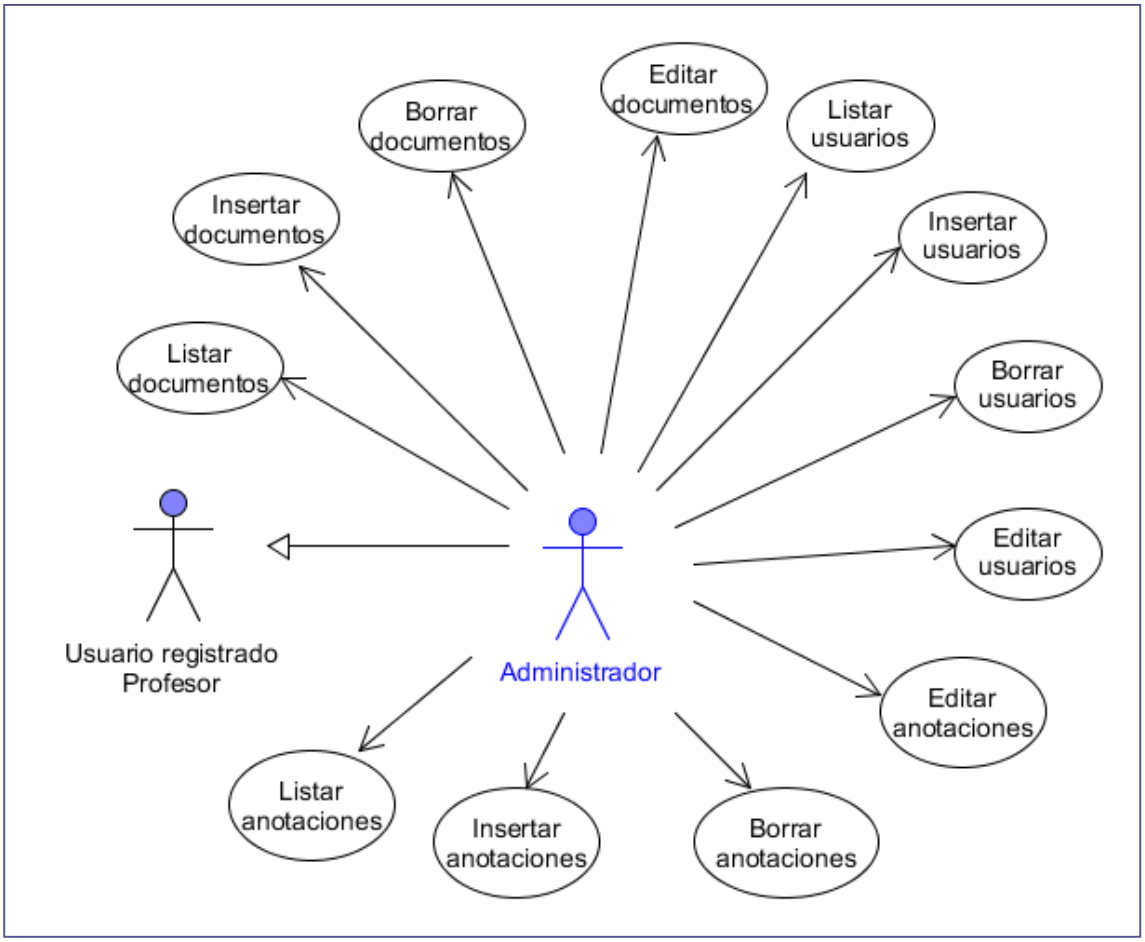

Ilustración 16 - Diagrama de casos de uso (Administrador)

Los casos de uso mostrados en la ilustración 16 reflejan acciones que el administrador efectúa directamente sobre la base de datos.

3.3. Diagrama de clases

Un diagrama de clases es un esquema de la estructura estática del sistema. En él se muestran las clases con sus atributos y operaciones, además de las relaciones entre ellas. El objetivo principal de este tipo de diagramas es enseñar de manera gráfica como van a relacionarse los elementos (clases) principales de la aplicación.

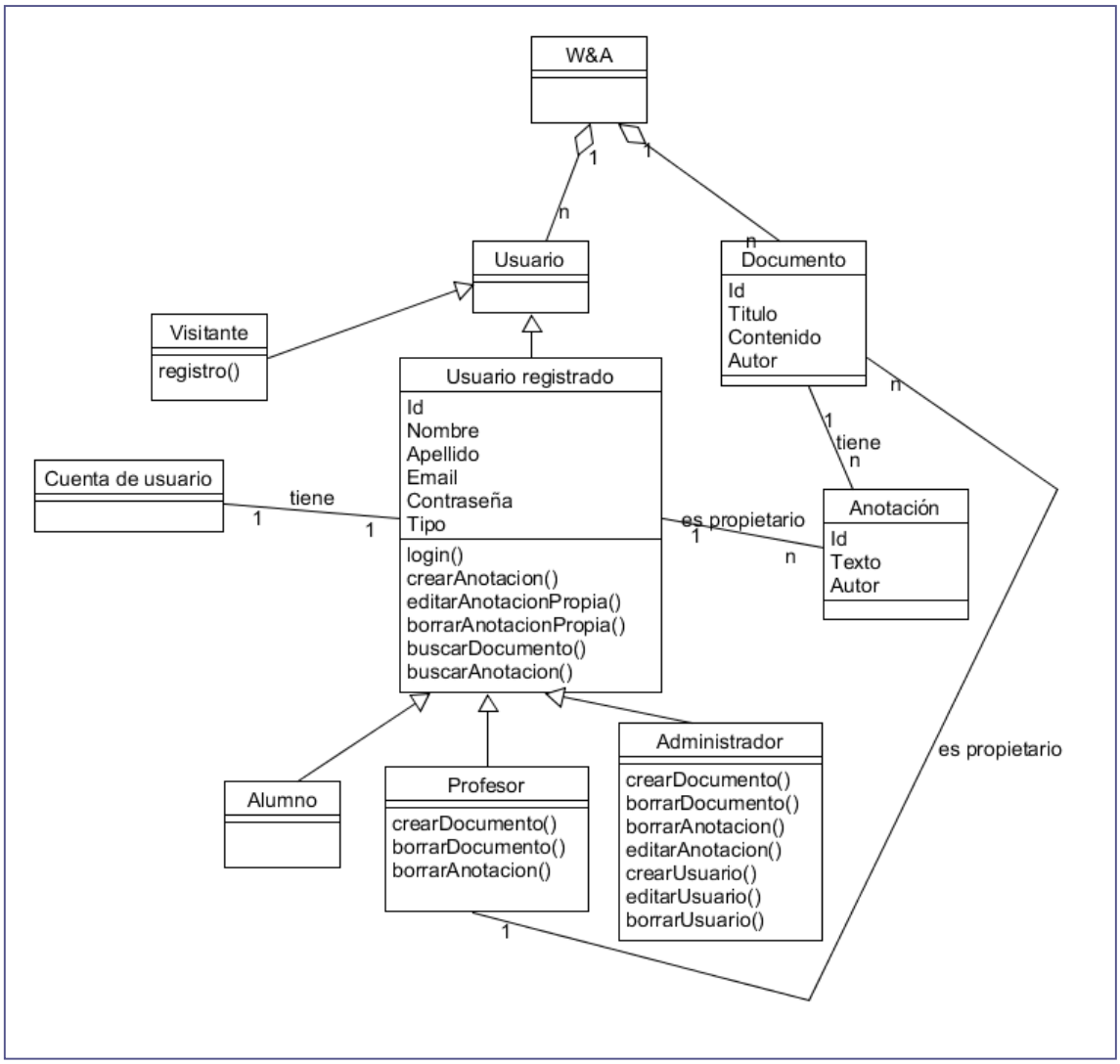

Ilustración 17 - Diagrama de clases

Cabe destacar que las funciones del administrador se realizan directamente desde la base de datos del sistema, como ya se ha indicado anteriormente.

3.4. Diagramas de secuencia

Para terminar la fase de análisis de la aplicación, se incluyen los diagramas de secuencia relativos a cada función. Un diagrama de secuencia es un esquema que representa la comunicación interna del sistema, es decir, muestra los mensajes necesarios que deben enviarse entre dos (o más) entidades para efectuar una operación determinada. A continuación se listan los diferentes diagramas de secuencia para cada caso de uso.

Registro

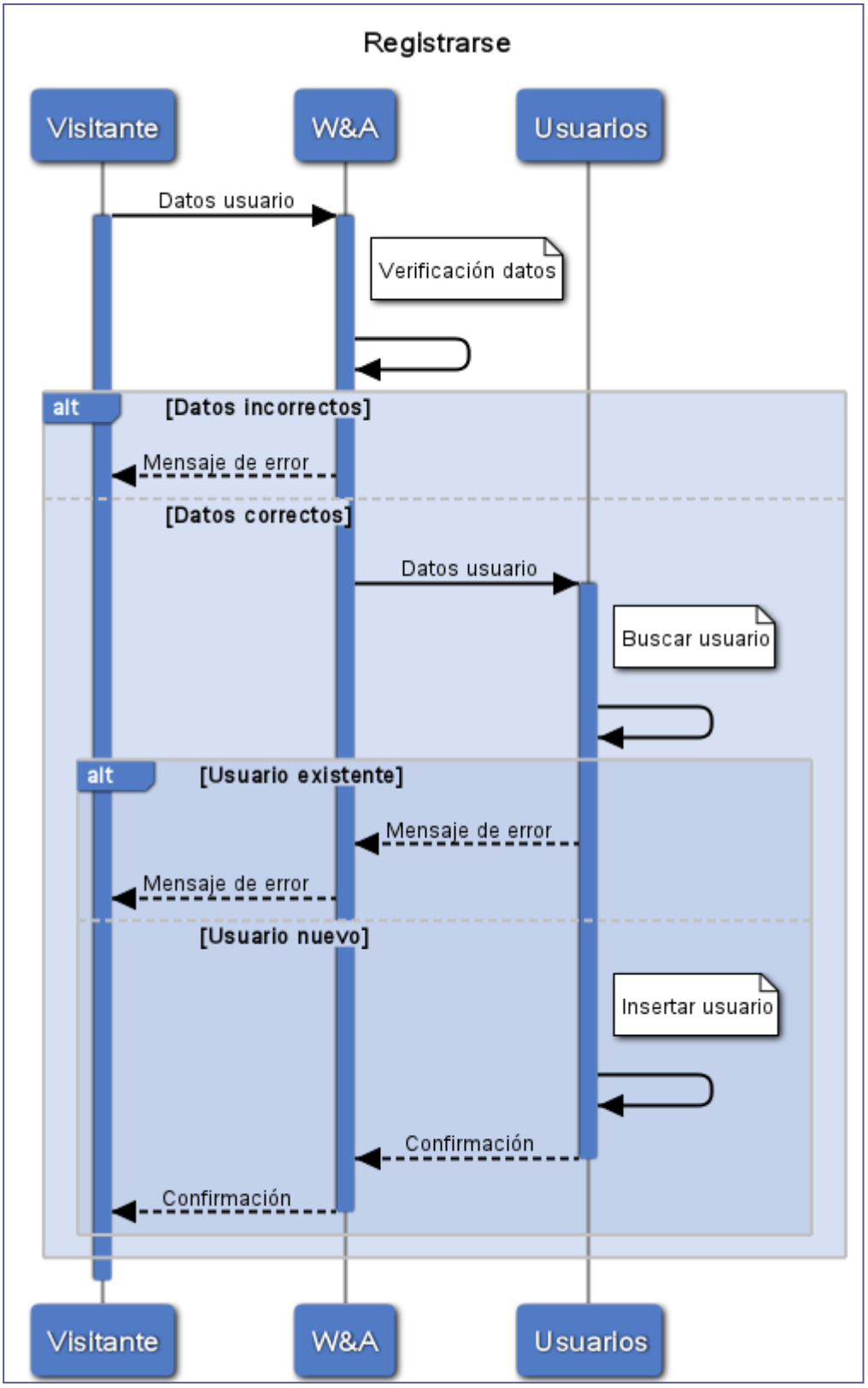

Ilustración 18 - Diagrama de secuencia (Registro)

La operación de registro la inicia un usuario no registrado (o visitante) que desea empezar a formar parte del sistema. En primer lugar envía sus datos a la aplicación haciendo uso de un formulario web. Si la información es correcta, se busca el nombre del usuario (correo electrónico) en la base de datos y, si no existe, se crea una nueva cuenta. En cambio, si el usuario ya existía previamente o si alguno de los campos del formulario es incorrecto, se muestra un mensaje de error adecuado a la situación.

Login

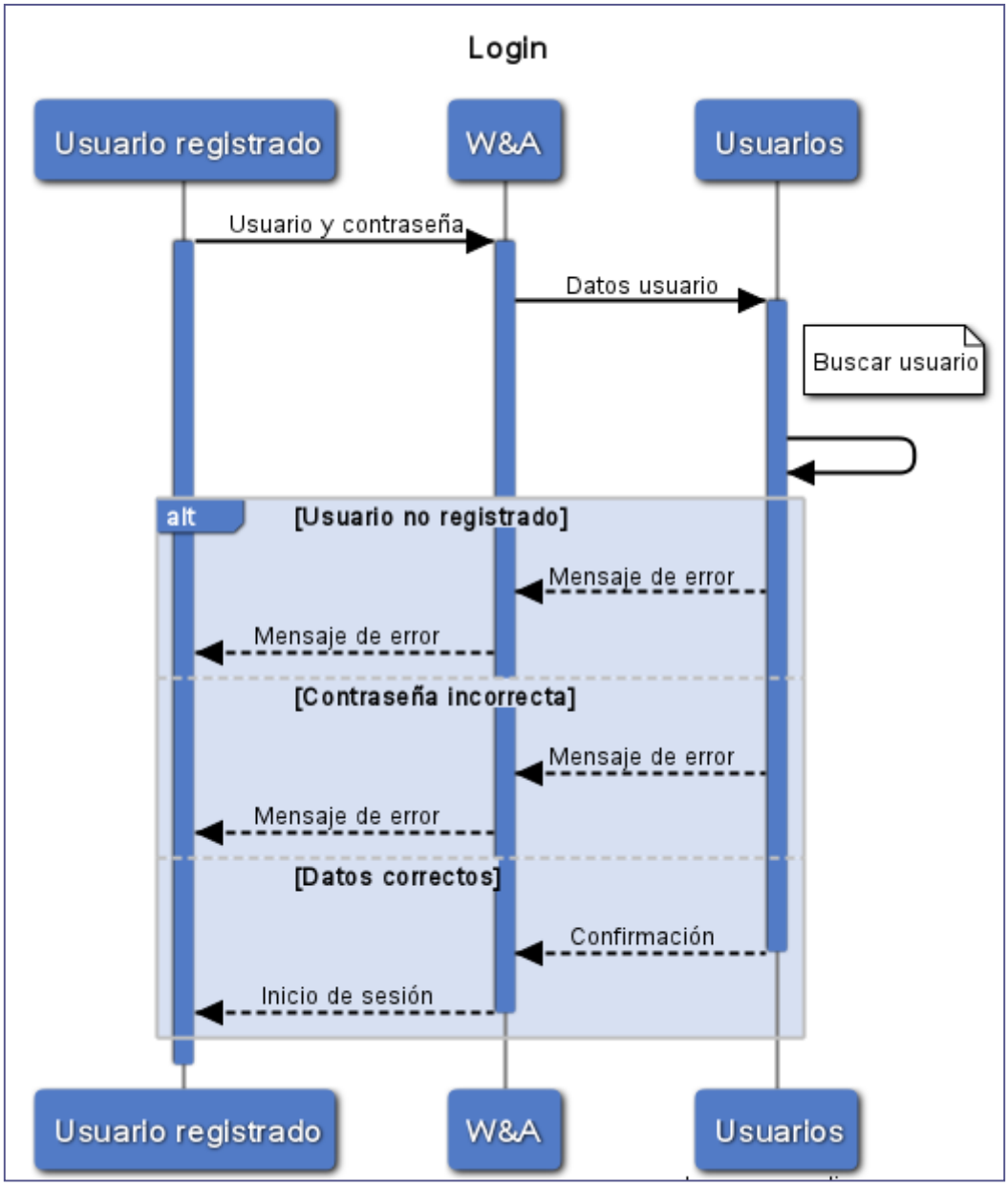

Ilustración 19 - Diagrama de secuencia (*Login*)

El intercambio de mensajes para la función de *login* es muy similar al de registro. De nuevo, el usuario envía sus datos a la aplicación (en este caso el usuario y la contraseña). Si

la información proporcionada concuerda con la de la base de datos, se inicia la sesión de usuario, si no, se muestra un mensaje de error.

Crear documento

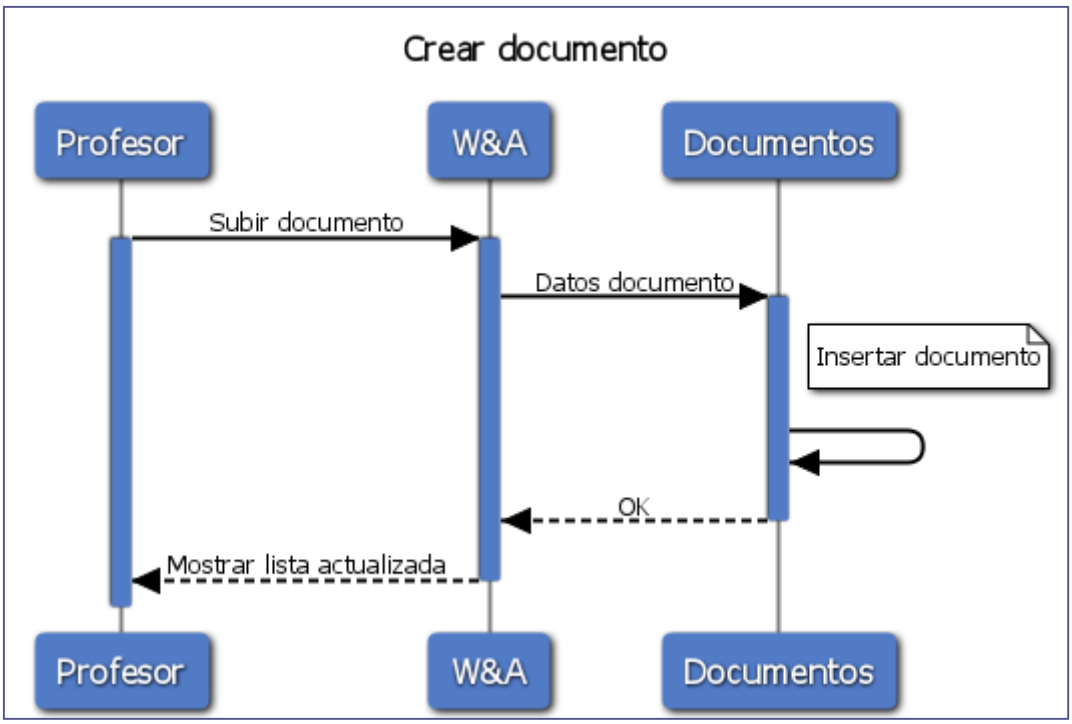

Ilustración 20 - Diagrama de secuencia (Crear documento)

Cualquier usuario registrado como profesor tiene disponible en su interfaz la opción para subir documentos. Una vez elegido localmente el archivo, la información relativa al mismo es insertada en la base de datos y en la lista de documentos.

Borrar documento

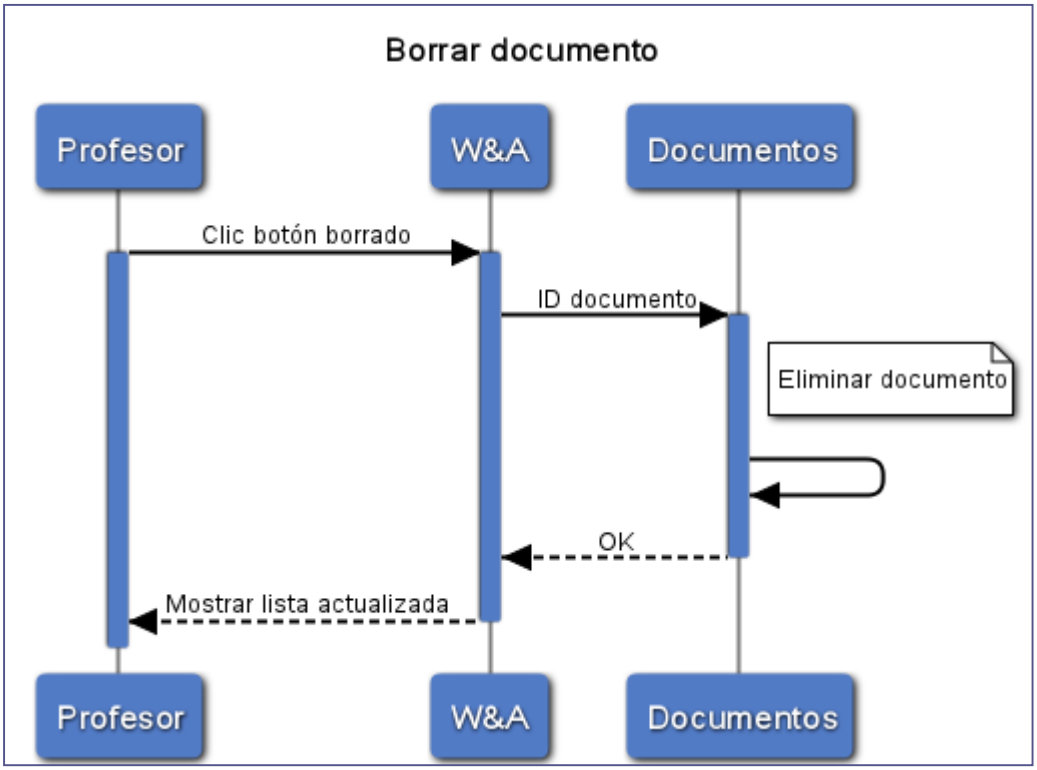

Ilustración 21 - Diagrama de secuencia (Borrar documento)

Un profesor también puede borrar cualquier documento del sistema, para ello debe pulsar el botón de borrado asociado a un documento concreto. Entonces se busca en la base de datos por su identificador único y se elimina.

Crear anotación

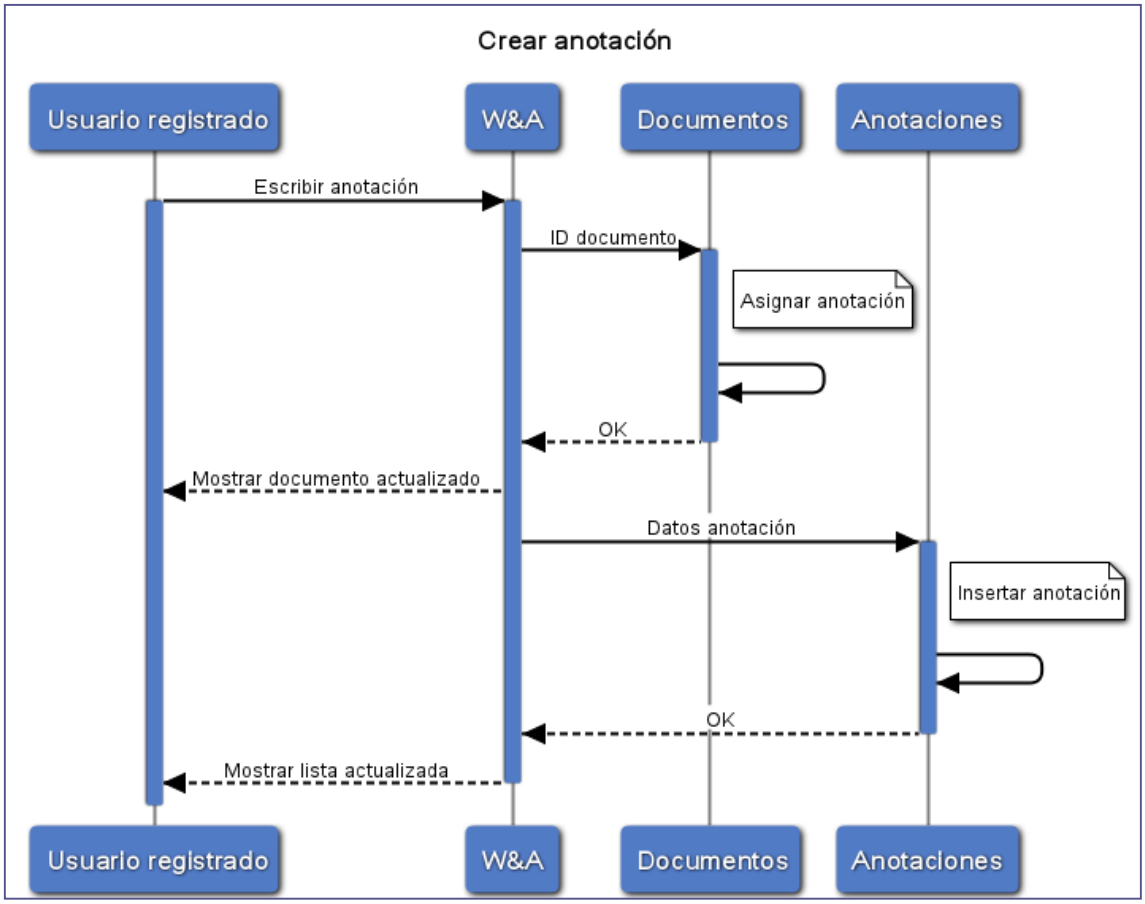

Ilustración 22 - Diagrama de secuencia (Crear anotación)

Cualquier usuario registrado puede crear anotaciones sobre los documentos. Para ello simplemente deben seleccionar el fragmento de texto que deseen y escribir un comentario. La anotación se envía a la base de datos y se guarda, añadiéndola también a la lista de anotaciones y sobre el propio documento. El texto anotado queda resaltado de color amarillo.

Editar anotación

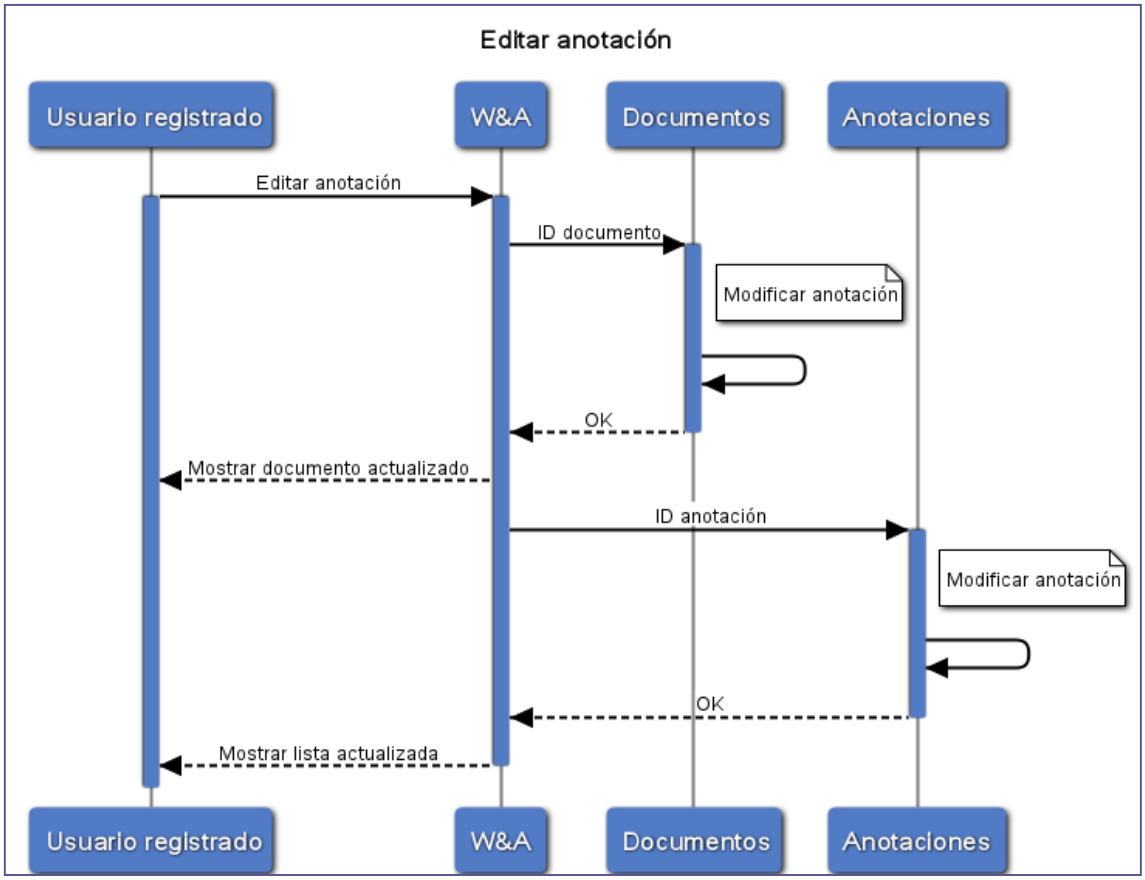

Ilustración 23 - Diagrama de secuencia (Editar anotación)

Todo usuario registrado puede editar únicamente sus propias anotaciones (excepto el administrador, que puede modificar cualquiera desde la base de datos). Para dicho propósito, tiene que seleccionar el fragmento de texto anotado y escribir un nuevo comentario. Con esto, utilizando el identificador de la anotación, se busca en la base de datos y se modifica la línea correspondiente. También se muestran los cambios sobre el listado de anotaciones y sobre el propio texto del documento.

Borrar anotación

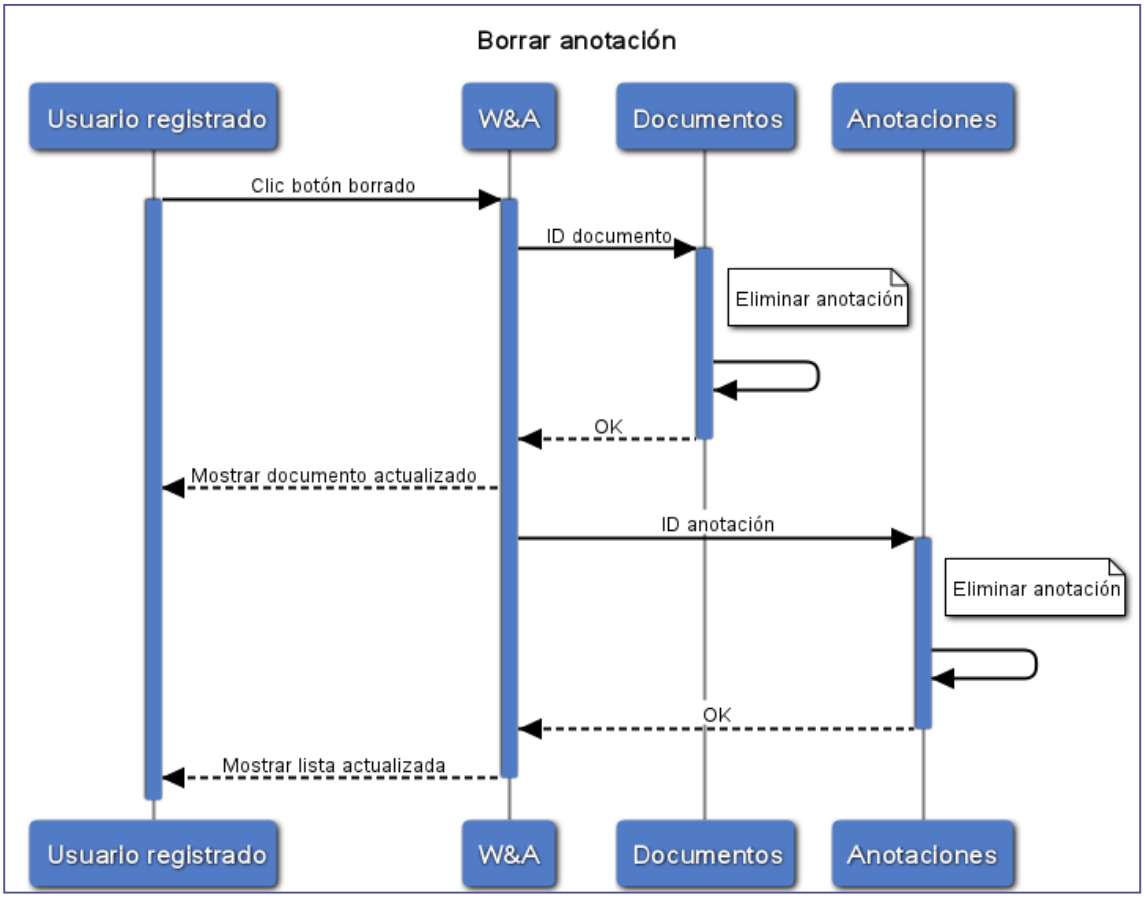

Ilustración 24 - Diagrama de secuencia (Borrar anotación)

Respecto al borrado, los profesores, los administradores y los propietarios pueden borrar una anotación concreta, por consiguiente, los usuarios registrados como alumnos no pueden eliminar las anotaciones de los demás. El proceso de borrado es el mismo tanto con las anotaciones propias como con las ajenas, así que se ha utilizado un solo gráfico para representar el escenario. Si se quiere eliminar una anotación, se selecciona y se pulsa el botón de borrado asociado a la misma (ya sea sobre el texto o en el listado de anotaciones). Una vez hecho esto, se usa el identificador para buscar la línea correspondiente en la base de datos y eliminarla definitivamente, reflejándolo así en el sistema.

Buscar documentos/anotaciones

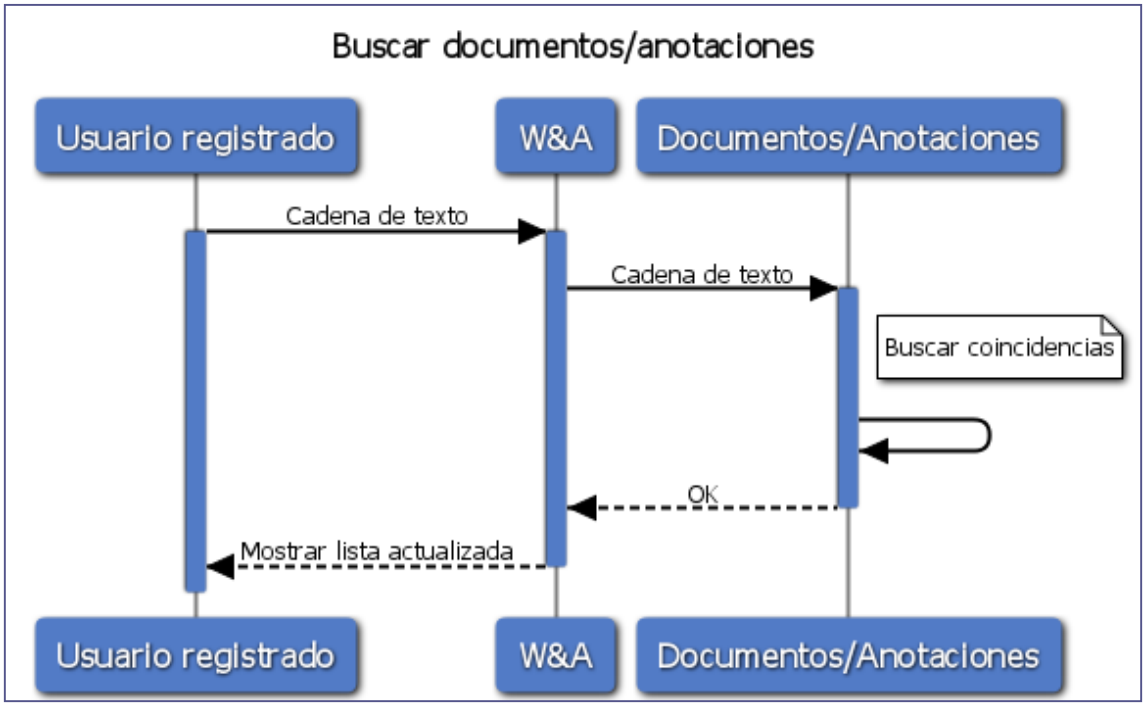

Ilustración 25 - Diagrama de secuencia (Búsqueda)

Los usuarios disponen de un cuadro de búsqueda encima de las listas de anotaciones y documentos para que puedan filtrarlos según un criterio (con una cadena de texto). Una vez enviado el formulario de búsqueda al servidor, se hace una consulta en la base de datos y se obtienen las filas que cumplan con la condición especificada. Para mostrarlo al usuario, se actualiza la lista correspondiente con los documentos/anotaciones filtrados.

Operaciones sobre la base de datos

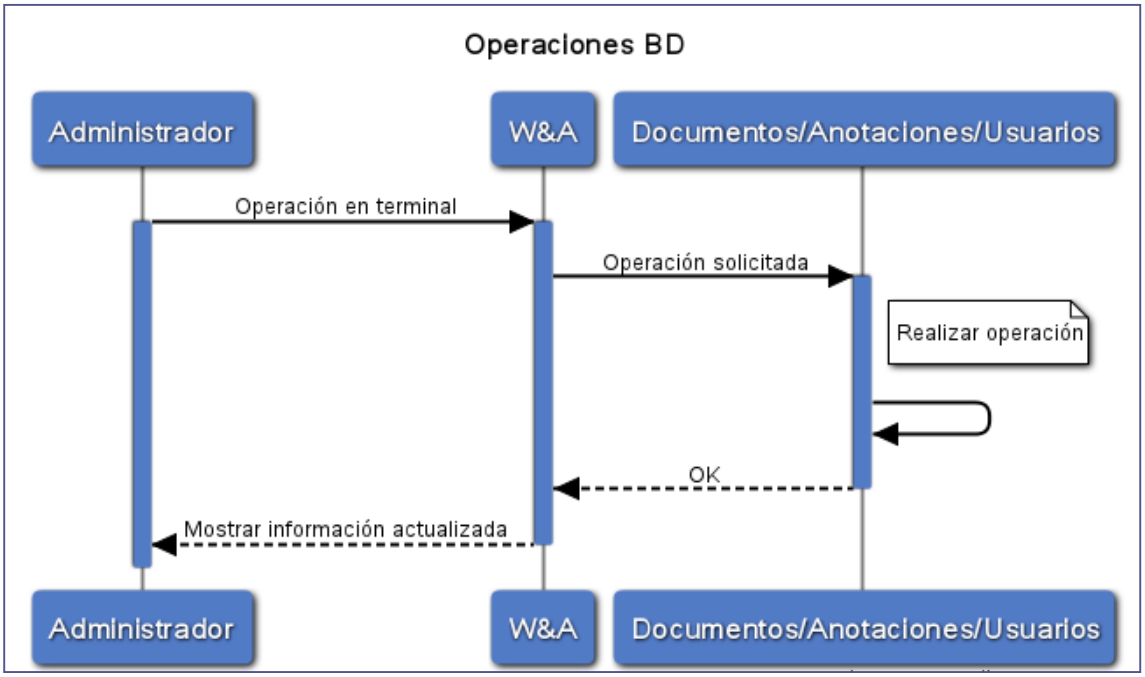

Ilustración 26 - Diagrama de secuencia (Operaciones BD)

El administrador es el único usuario capaz de interactuar directamente con la base de datos. En la ilustración 26 se han resumido todas las funciones que puede realizar: borrar, editar y crear sobre las colecciones de usuarios, documentos y anotaciones.

4. Diseño

4.1. Introducción

Una vez terminada la fase de análisis se da paso a la etapa de diseño, cuyo objetivo es mostrar cómo está estructurada internamente la aplicación y como se relacionan entre ellos los mecanismos que la forman.

Es muy importante definir la arquitectura adecuada a la hora de desarrollar una aplicación. Por ejemplo, en ciertas ocasiones conviene separar la funcionalidad de los datos y de la interfaz de usuario, formando así un sistema de varias capas independientes, tal como se hace en la famosa arquitectura multicapa. En cambio, en algunos escenarios tal vez no sea de vital importancia repartir claramente las funciones entre los elementos del sistema, como ocurre en la arquitectura cliente-servidor.

En el caso de W&A se ha optado por una arquitectura MVC (Modelo-Vista-Controlador). Se detalla en el siguiente apartado.

4.2. Arquitectura MVC

MVC es un patrón que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Este modelo se basa también en la reutilización del código, un aspecto importante para las labores de programación.

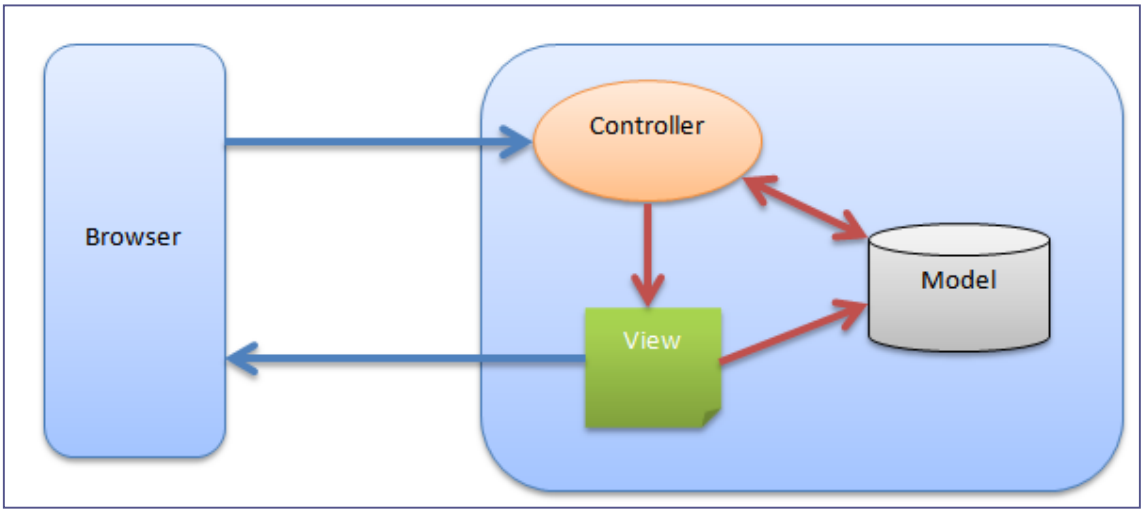

Ilustración 27 - Arquitectura MVC

En el siguiente punto se describen los tres componentes que forman la arquitectura MVC, así como las interacciones entre ellos.

4.2.1. Componentes

- Modelo**:** Constituye la lógica de negocio de la aplicación y los mecanismos necesarios para el acceso y manipulación de datos. El modelo está formado por una serie de componentes independientes de la vista y el controlador, permitiendo así la reutilización de código y el desacoplamiento entre las capas.

- Vista**:** Como su propio nombre indica, contiene el código que produce la visualización de la interfaz de usuario. Se encarga de presentar la información en un formato adecuado para interactuar con ella.

- Controlador**:** Es el cerebro de la aplicación. Responde a eventos (generalmente iniciados por el usuario) e invoca peticiones al modelo cuando se realiza una solicitud sobre la información. También es el encargado de proporcionar los datos necesarios para modificar la visualización de la interfaz. Por tanto, se podría decir que el controlador es un intermediario entre la vista y el modelo.

Por lo que respecta a W&A, el modelo correspondería, por un lado, a todos los archivos y métodos javascript que ofrecen funcionalidad a la aplicación, y por otro lado, a la base de datos y las funciones que interactúan con su información. En segundo lugar, la vista estaría compuesta por el código HTML, CSS y PHP que proporciona la información de salida a la interfaz de usuario. Por último, el controlador haría referencia a aquellos métodos que responden a los eventos "disparados" sobre la aplicación, como clicar un botón o escribir una nueva anotación sobre un documento.

4.2.2. Interacción

El comportamiento típico dentro de la arquitectura MVC suele seguir los siguientes pasos:

1. El usuario realiza una solicitud al sitio web, utilizando para ello los elementos de la interfaz (vista).
2. La vista indica al controlador que evento se ha "disparado", y éste solicita al modelo la ejecución del método asociado a tal evento.
3. El modelo obtiene los datos necesarios de la base de datos y se los envía al controlador.
4. El controlador reenvía la información obtenida del modelo a la vista para que ésta muestre el resultado al usuario.
5. El proceso se repite cuando el usuario realiza una nueva acción sobre la interfaz.

Aunque la implementación interna de cada función se explica con detalle e ilustraciones en el punto 5 del documento, a continuación se muestra un ejemplo del proceso de comunicación entre los componentes MVC.

Registro

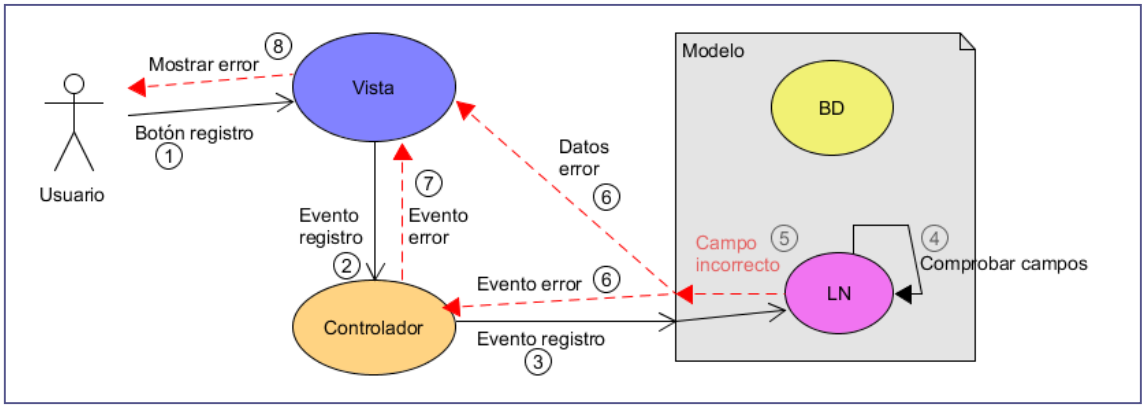

Ilustración 28 - MVC Error registro (Campos incorrectos)

En la ilustración 28 se detalla cómo se produce un intento fallido de registro, debido a la inserción de campos incorrectos en el formulario enviado al servidor (formato de email no válido, contraseña corta, campos en blanco…).

1. El usuario rellena los campos del formulario y hace uso del botón asociado a la petición de registro, es decir, interactúa con la interfaz gráfica del sistema (vista).
2. La vista informa del evento "disparado" por una acción del usuario al controlador.
3. El controlador indica al modelo que se ha ejecutado el evento correspondiente al registro de un usuario.
4. El modelo ejecuta el código asociado a la validación de campos del formulario de registro.
5. El modelo detecta un error en la información recibida.
6. El modelo le envía al controlador una señal indicando el error en la solicitud de registro. También le envía a la vista los datos necesarios para generar la salida correspondiente.
7. El controlador indica a la vista qué debe mostrar al usuario.
8. La vista, haciendo uso de los datos recibidos, muestra al usuario un mensaje de error en el registro.

Cabe destacar que en este caso, el modelo no hace uso de la base de datos, solamente actúa la lógica de negocio (LN).

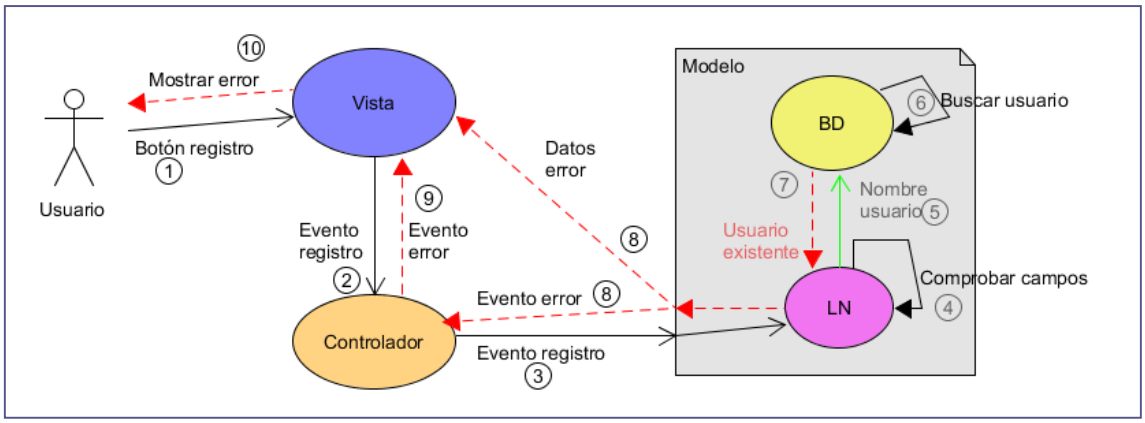

Ilustración 29 - MVC Error registro (Usuario existente)

En este segundo caso, se da una situación similar a la anterior. Aquí los campos del formulario son validados correctamente, pero el usuario indicado ya existe, y por tanto se muestra otro mensaje de error. Se puede apreciar como en esta ocasión el modelo accede a la base de datos para buscar el nombre de usuario correspondiente.

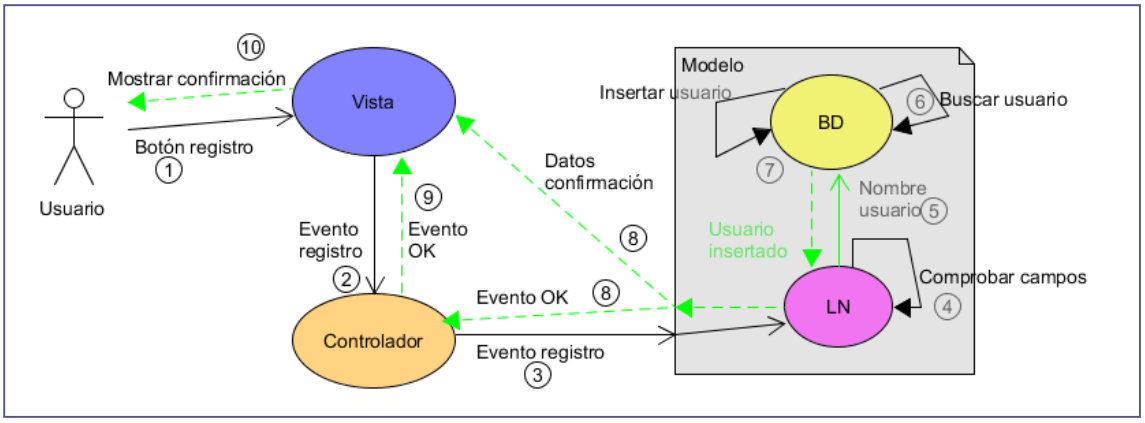

Ilustración 30 - MVC Registro correcto

El tercer gráfico representa una solicitud de registro exitosa. Como se puede apreciar, el procedimiento es muy parecido a los dos anteriores (y así ocurre con todas las funcionalidades). La peculiaridad de este proceso de comunicación es que el modelo realiza una operación de inserción de la información del usuario en la base de datos, pasando así a formar parte del sistema.

Como ya se ha comentado, se explica con más detalle este ejemplo y todas las funciones del sistema en el apartado de implementación (5).

5. Implementación

5.1. Introducción

En la sección de implementación se profundiza en el funcionamiento interno de W&A. En primer lugar, se describen todas las tecnologías y herramientas utilizadas durante el transcurso del trabajo. En segundo lugar, se muestran los aspectos relativos a la interfaz de usuario. Por último, se explican los métodos que dan funcionalidad a la aplicación y la interacción con la base de datos.

5.2. Tecnologías

5.2.1. HTML

HTML (*HyperText Markup Language*) es un lenguaje de marcado utilizado para la elaboración de páginas web. Se trata de un estándar que define la estructura básica y el código para la creación de contenido web.

Tiene algunos inconvenientes: depende de otros lenguajes de programación (como javascript) para poder implementar aplicaciones con una funcionalidad compleja, como es el caso de W&A; además, su mantenimiento es costoso, especialmente si el número de páginas es elevado. A pesar de ello, se ha elegido HTML para el desarrollo de la estructura de las distintas páginas que componen W&A, ya que se trata de un estándar muy extendido y sencillo de utilizar.

5.2.2. CSS

CSS (*Cascading Style Sheets*) es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML o XML. Surgió con la idea principal de separar el contenido web de su presentación, ya sea dentro de un mismo documento o utilizando ficheros diferentes. Esta separación contribuye positivamente a la legibilidad de los documentos y el mantenimiento de los mismos. También permite reutilizar el código fácilmente cuando se utilizan varias páginas con el mismo estilo.

Las principales desventajas de CSS son, en primer lugar, la necesidad de escribir el código en un orden determinado, ya que las órdenes se ejecutan en cascada. Una ordenación incorrecta podría suponer el efecto contrario al deseado. En segundo lugar, varias operaciones CSS pueden no ser compatibles con algunos navegadores y sus versiones antiguas. Este hecho implica que el navegador utilice su estilo por defecto, pudiendo no coincidir con la visualización que el programador estaba buscando.

En este proyecto se han utilizado varios archivos escritos en CSS para definir la presentación de las páginas que componen la aplicación. El motivo de la elección de este lenguaje es el mismo que en el caso de HTML, su sencillez y uso extendido para la creación de estilos web.

5.2.3. Javascript

Se trata de un lenguaje de programación interpretado. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Principalmente se utiliza para la programación en el lado del cliente, implementado como parte de un navegador web, permitiendo mejoras en la interfaz de usuario y páginas web dinámicas. También se puede utilizar en la parte del servidor (Server-Side JavaScript o SSJS), aunque para el caso de W&A se ha utilizado PHP, que se explica más adelante en el punto 5.2.6. de este capítulo.

Javascript puede incrustarse directamente en las páginas HTML para añadirles funcionalidad, ya sea importando un fichero externo o incluyendo el código dentro del propio documento. Lo recomendado es separar la funcionalidad del contenido, pero en ocasiones conviene incluir algunos métodos javascript dentro del documento HTML. Por ejemplo, cuando una función se utiliza exclusivamente en una página (y no en varias), esta técnica es más eficiente que la de importar un fichero externo.

Uno de los puntos fuertes de javascript es su capacidad para aligerar la carga del servidor, realizando acciones en el cliente. Un ejemplo práctico que lo demuestra es la validación de formularios. Si se escribe una rutina javascript que compruebe los datos insertados por el usuario, el cliente se encargará de evitar el envío del formulario al servidor hasta que todos los campos sean correctos, evitando así una carga innecesaria sobre el lado del servidor. Otra gran ventaja que ofrece este lenguaje es la capacidad de crear contenido web dinámico, algo muy común hoy en día que ayuda a mejorar considerablemente la experiencia del usuario.

Por otro lado, el principal problema de javascript es la seguridad. Los fragmentos de código, una vez añadidos a las páginas web en los servidores, son descargados y ejecutados en el navegador del cliente, permitiendo así que cierto código malicioso pueda ser ejecutado en la máquina cliente con el objetivo de explotar alguna vulnerabilidad de seguridad conocida en alguna de las aplicaciones, navegadores o el mismo sistema operativo.

W&A hace uso de este lenguaje para implementar la funcionalidad en el lado del cliente. Esto incluye la validación de formularios, respuestas a eventos, creación y carga de contenido dinámico… Además, para alcanzar los principales objetivos de la aplicación, se han utilizado las librerías JQuery y Annotator.js. Se explican a continuación.

JQuery

Es una biblioteca javascript de código abierto, que permite simplificar la manera de interactuar con los documentos HTML, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web, la cual se explica en el apartado 5.2.4.

Su uso en W&A ha sido crucial para manejar los distintos eventos que se pueden "disparar" con las acciones del usuario, por ejemplo al pulsar un botón, usar un enlace o enviar un formulario. También cabe destacar la facilidad que JQuery ha aportado al manejo de efectos dinámicos como respuesta a eventos. Algunos relevantes son: el efecto de desvanecimiento cuando se abren y cierran cuadros de diálogo, el procesado y actualización de las tablas mediante peticiones AJAX, la modificación de estilos CSS…

Annotator.js

Annotator es el núcleo principal sobre el cual gira la aplicación W&A. Se trata de otra librería javascript de código abierto, cuyo propósito es otorgar la capacidad de realizar anotaciones sobre cualquier texto. Dispone de una gran cantidad de *plugins* que pueden ser añadidos a una página web. Entre las funcionalidades que ofrecen estos complementos, destacan: el almacenamiento de anotaciones en bases de datos; la posibilidad de añadir etiquetas a los comentarios; compartir permisos de visualización, edición y borrado; filtrar las anotaciones por contenido o autor; un sistema de autenticado…

Pese a la dificultad que supone añadir mayor funcionalidad o modificar la existente en una librería creada por otros programadores, se ha conseguido adaptar Annotator a las necesidades del proyecto. Gracias a ello, los usuarios podrán disfrutar de una buena experiencia con el tratado de anotaciones y documentos.

5.2.4. AJAX

AJAX (*Asynchronous Javascript And XML*) es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma, es posible realizar cambios sobre las páginas sin necesidad de recargarlas o saltar a otras, mejorando la interactividad, velocidad y usabilidad de las aplicaciones. Todo esto conlleva inevitablemente a una mejor experiencia del usuario.

En cambio, AJAX también tiene algunas desventajas. El hecho de utilizar una sola página para mostrar dinámicamente todo el contenido, dificulta las labores de programación y depuración de errores. Otro punto a tener en cuenta es la "desaparición" de algunas opciones tradicionales de los navegadores, como los botones de retroceso y avance. Estas acciones quedan inservibles, ya que el usuario no cambia de página y, por tanto, clicar en "retroceder" o "avanzar" no le devolverá al estado anterior de la misma.

Esta técnica es bastante importante en W&A, ya que ha evitado que se creen más páginas de las necesarias. Sin AJAX, el envío de cualquier formulario desencadenaría en un refresco de la misma página, algo que puede ser fastidioso para el usuario, especialmente si

existen errores en el proceso de validación de campos. Por último, cabe resaltar la actualización dinámica del contenido de la aplicación. Por ejemplo, un usuario puede crear o borrar un documento y ver perfectamente como dicho documento aparece o desaparece de la lista correspondiente, sin necesidad de saltar a una nueva página o recargar la actual.

5.2.5. Bootstrap

Se trata de un conjunto de herramientas (*framework*) de código libre para el diseño de sitios y aplicaciones web. Contiene plantillas con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS. También dispone de extensiones javascript adicionales.

Entre las principales ventajas de Bootstrap destacan: el uso del atributo *class* de cada elemento HTML para crear el estilo, un mecanismo bastante sencillo; el ahorro de tiempo invertido en la programación de la interfaz de usuario es considerable, ya que unos pocos atributos *class* sobre un mismo elemento HTML (comúnmente un <*div*>) equivalen a docenas de líneas de código CSS; y la posibilidad de añadir el llamado RWD (*Responsive Web Design)*, una filosofía de diseño cuyo objetivo es adaptar la apariencia de las páginas web al dispositivo que se esté utilizando para visualizarlas. Esto se consigue gracias al sistema de rejilla utilizado en Bootstrap, que por defecto dispone de doce columnas que distribuyen y redimensionan el contenido web según el tamaño del dispositivo.

Por otro lado, no es un *framework* ligero, y lo más probable es que nunca se utilicen todas sus capacidades en una página web determinada. Adicionalmente, Bootstrap "ensucia" muchísimo el documento HTML, ya que, como se ve en la ilustración 31, para añadir un estilo se requieren numerosas clases (atributo *class)*, lo cual dificulta considerablemente la lectura del código. Por último, un detalle negativo importante y común en la mayoría de *frameworks*, es la necesidad de ceñirse a los métodos programados. Si se desea añadir una funcionalidad, o en este caso, un nuevo estilo que no esté implementado, el trabajo puede complicarse bastante, e incluso llegar a ser frustrante.

```
98  <!-- Lastname Field -->
99  <div id="userLastnameInput" class="form-group">
100     <label class="control-label" for="userLastname">Last Name</label>
101     <input type="text" name="userLastname" id="userLastname" class="form-control" placeholder="Enter yo
102     <!-- Success and Error Icons -->
103     <span id="userLastnameSuccess" class="glyphicon glyphicon-ok form-control-feedback hide"></span>
104     <span id="userLastnameError" class="glyphicon glyphicon-remove form-control-feedback hide"></span>
105     <!-- Error Message (Empty field) -->
106     <div id="userLastnameErrorMessageBlank" class="errorMessage hide" role="error-message-blank">This f
107 </div>
```

Ilustración 31 - Ejemplo de clases Bootstrap

Pese a tener bastantes inconvenientes, Bootstrap ha sido clave para darle a W&A una apariencia agradable, sin tener que escribir "a mano" todo el código de la interfaz, ya que hubiese requerido mucho del tiempo dedicado a otros aspectos del proyecto. Además, siempre es interesante aprender y utilizar *frameworks* para darle un mayor grado de calidad al trabajo.

5.2.6. PHP

PHP (*Hypertext Preprocessor*) es un lenguaje de programación de uso general de

código del lado del servidor. Se orienta principalmente al desarrollo web de contenido dinámico con acceso a información almacenada en una base de datos.

Este lenguaje es fácil de aprender y está muy bien documentado debido a su popularidad. Adicionalmente, su buena integración con la mayoría de conectores a bases de datos actuales, permite al desarrollador elegir la mejor opción de almacenamiento para su trabajo.

Algunas de sus desventajas son la necesidad de instalar un servidor web (obviamente) y la carga que supone sobre el mismo. También afecta negativamente a la legibilidad del código cuando se incluyen funciones PHP dentro de las páginas HTML.

Este lenguaje es de vital importancia en W&A para la interacción con la base de datos (operaciones sobre las colecciones de usuarios, documentos y anotaciones), la gestión de sesiones de usuario y el manejo de las peticiones HTTP realizadas por el cliente. Esto último es facilitado por Silex, un *micro-framework* que se explica a continuación.

Silex

Se trata de un *framework* PHP destinado a la creación de aplicaciones simples en un solo archivo. Silex pretende ser conciso, extensible y testeable. Su funcionamiento se basa en la definición de controladores asociados a una ruta, que se ejecutan cuando ésta coincide con una petición del usuario. Dicha ruta consta de un patrón que define la dirección que apunta a un recurso determinado, y un método HTTP (*GET*, *POST*, *PUT*, *DELETE*, *PATCH* u *OPTIONS*) que describe la interacción con el recurso. Para aclarar esto se muestra un ejemplo ilustrado:

```php
require_once __DIR__.'/../vendor/autoload.php';

$app = new Silex\Application();

$app->get('/hello/{name}', function($name) use($app) {
    return 'Hello '.$app->escape($name);
});

$app->run();
```

Ilustración 32 - Ejemplo Silex

En este método, cuando la dirección */hello/{name}* reciba una petición *GET*, se ejecutará el código asociado al controlador (que mostrará por pantalla "*Hello ”* + *$name*).

W&A emplea este *framework* para el manejo de las peticiones HTTP realizadas por el usuario a la hora de actuar sobre las anotaciones y documentos (listar, crear, modificar y borrar). Esto ha sido posible con la programación de una pequeña API (*Application Programming Interface*) formada por varios controladores asociados a cada acción, mediante el sistema de rutas explicado anteriormente.

5.2.7. MongoDB

MongoDB es un sistema de bases de datos NoSQL orientado a documentos y desarrollado bajo el concepto de código abierto. Guarda estructuras de datos en documentos BSON (*Binary JSON* (*JavaScript Object Notation*)) con un esquema dinámico, haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

Este sistema se caracteriza por ser: escalable, ya que posee mecanismos internos que permiten ir aumentando el poder de procesamiento a medida que los requerimientos de datos suben; y flexible, puesto que prescinde de un esquema inicial definido.

Los inconvenientes de MongoDB residen en la inexistencia de algunos mecanismos útiles de las bases de datos relacionales. No están presentes las transacciones, solo se garantizan operaciones atómicas a nivel de documento. Tampoco existen los *joins*. Para consultar datos relacionados en dos o más colecciones, se debe hacer más de una consulta.

¿Por qué usar MongoDB en vez de MySQL?

- Durante toda la carrera se ha trabajado con MySQL exclusivamente para el manejo de bases de datos, esta era una buena oportunidad para aprender y experimentar una alternativa.
- Al no depender de esquemas fijos, se ha podido ir modificando la estructura de los objetos (añadir o quitar atributos) sin necesidad de realizar configuraciones en la base de datos.
- Es mucho más rápido con las operaciones de lectura y escritura.
- Es más escalable, algo conveniente en una aplicación como W&A, ya que el ritmo y la cantidad de inserciones pueden volverse bastante elevados.
- El formato JSON utilizado por MongoDB es muy sencillo de tratar en el entorno de programación.
- Las operaciones *join* no son necesarias en W&A, ya que la información se extrae de las colecciones de manera individual.

5.2.8. JSON

JSON (*JavaScript Object Notation*) es un formato de texto ligero para el intercambio de datos. Básicamente, describe los datos con una sintaxis dedicada que se usa para la identificación y gestión de los mismos. Una de sus mayores ventajas es que puede ser leído por cualquier lenguaje de programación, por lo tanto, se puede utilizar para el intercambio de datos entre diversas tecnologías.

En W&A se utiliza este formato para la comunicación entre el cliente y el servidor a la hora de efectuar operaciones sobre los documentos y las anotaciones (leer, crear, modificar y borrar). Además, la base de datos MongoDB hace uso de este lenguaje, permitiendo un acceso sencillo a la información almacenada.

5.3. Herramientas utilizadas

5.3.1. Eclipse

Eclipse es una plataforma software compuesta por un conjunto de herramientas de programación de código libre. La versión utilizada para este proyecto es Eclipse PHP Mars.

Con esta herramienta se ha programado la aplicación entera, desde las páginas HTML, hasta los documentos PHP y javascript, pasando por las hojas de estilo CSS. Esto ha sido posible gracias a su editor de texto integrado con analizador sintáctico y marcación por colores.

Eclipse posee un número incontable de extensiones que le aportan diferentes funcionalidades. Para el trabajo con W&A se han utilizado dos de ellas:

- *Toad Extension*: añade la posibilidad de crear conexiones a bases de datos. Además proporciona un intérprete de órdenes para interaccionar con las colecciones almacenadas. Este *plugin* ha sido de mucha utilidad a la hora de realizar labores de administración sobre la base de datos de W&A. En la ilustración 33 se pueden observar los componentes principales de la extensión. En la parte izquierda se encuentra el listado de conexiones (arriba) y las bases de datos disponibles (abajo). El centro de la fotografía está formado en orden descendente por: el intérprete de órdenes, un cuadro que muestra el resultado de las operaciones y la consola con el historial de acciones efectuadas.

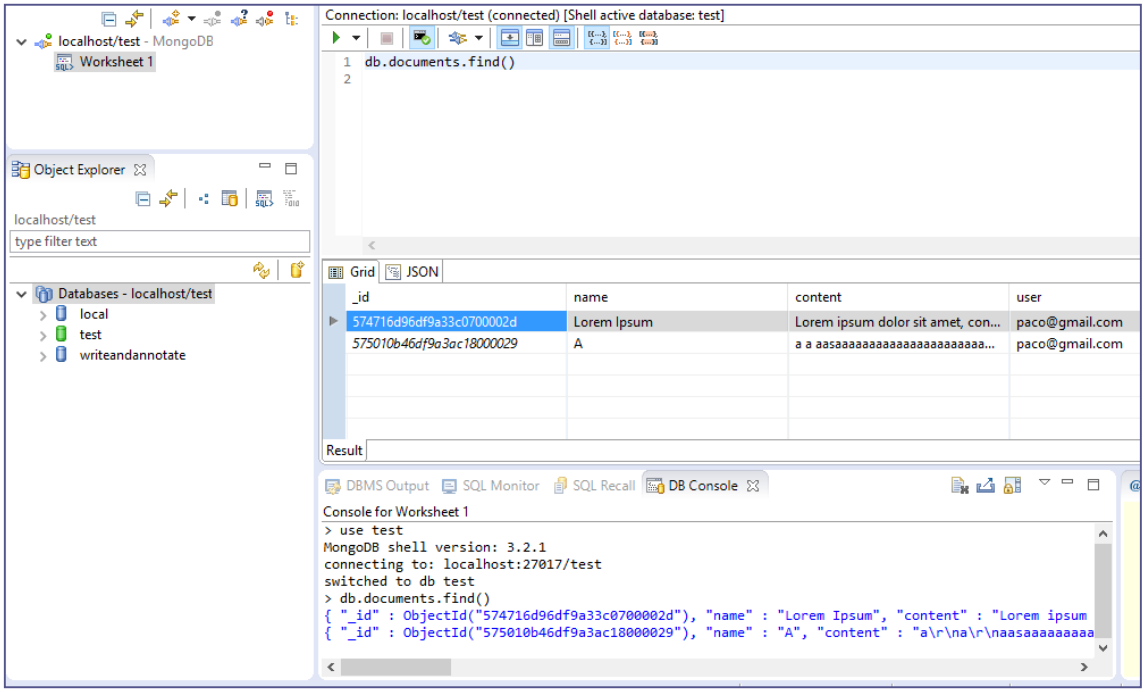

Ilustración 33 - Extensión *Toad* para Eclipse

- *Remote System Explorer:* permite al usuario conectarse a equipos remotos. Con este *plugin*, se han podido depositar los ficheros con los avances realizados del proyecto en la máquina remota asignada por la universidad. En la ilustración 34 se puede observar su interfaz. En el centro se muestra el cuadro donde se solicitan las credenciales del usuario para efectuar la conexión SSH. A la izquierda se listan tanto los directorios del equipo local como los del remoto.

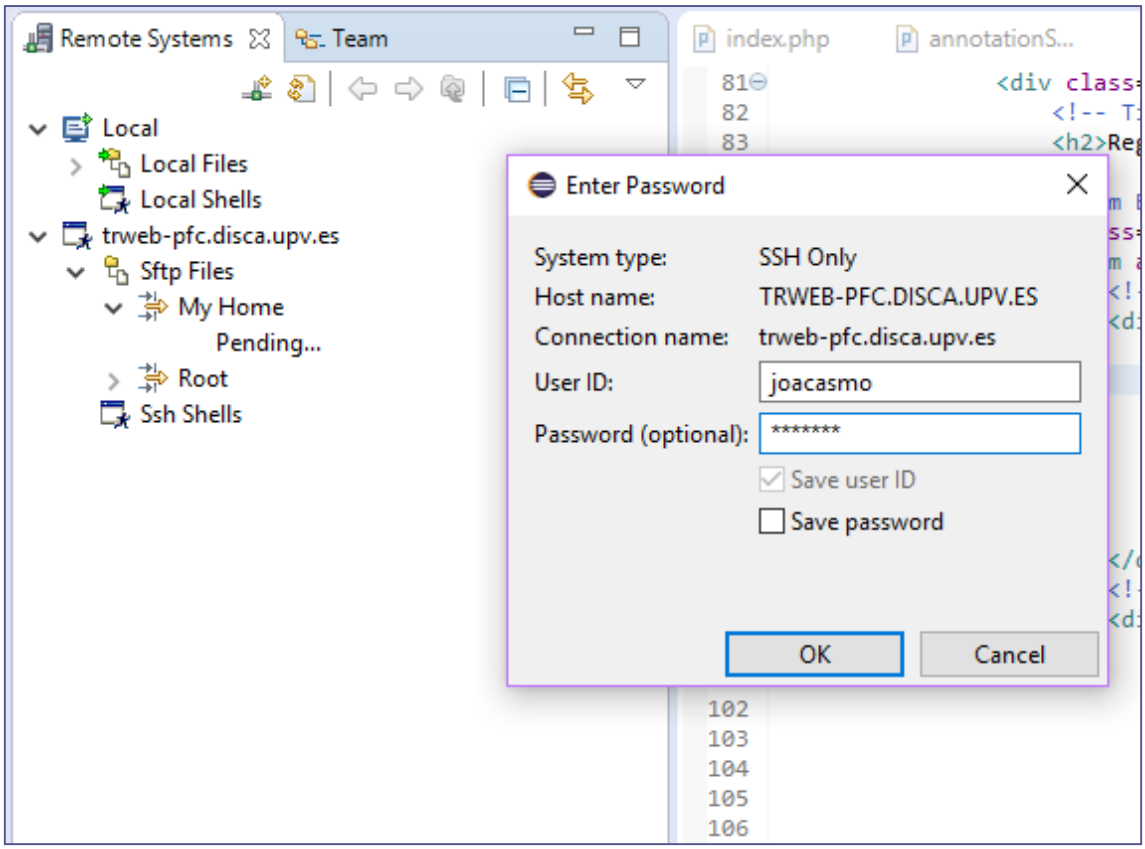

Ilustración 34 - Extensión *Remote System Explorer* para Eclipse

En definitiva, gracias a las extensiones, el editor de texto y el panel de navegación con todos los archivos, Eclipse permite juntar todo el entorno de trabajo en una sola ventana, facilitando así el desarrollo de cualquier proyecto.

5.3.2. XAMPP

XAMPP es una plataforma de software libre que consiste principalmente en un servidor web Apache, un sistema de gestión de bases de datos MySQL e intérpretes para los lenguajes de programación PHP y Perl. Se ha hecho uso de esta herramienta exclusivamente para levantar el servidor y alojar en él la aplicación, ya que no había necesidad de utilizar el resto de complementos. En la ilustración 35 se muestra el aspecto del panel de control de XAMPP.

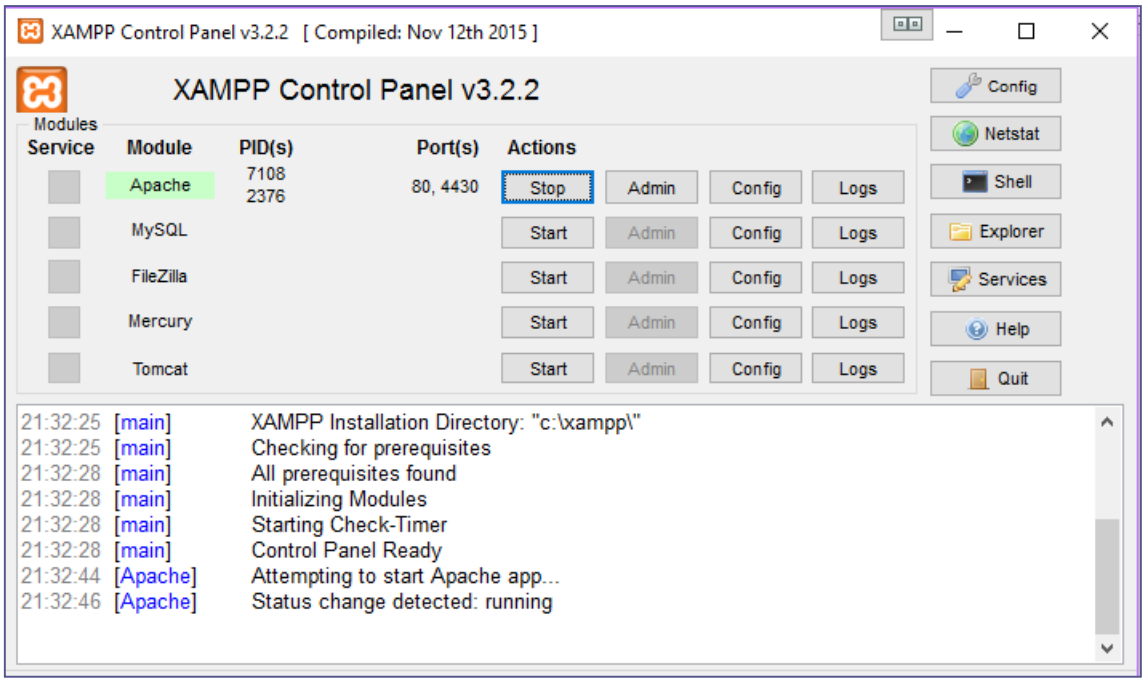

Ilustración 35 - Panel de control XAMPP

5.3.3. GIMP

GIMP (*GNU Image Manipulation Program)* es un programa de edición de imágenes digitales en forma de mapa de bits, tanto dibujos como fotografías. Posee herramientas para el retoque de imágenes, dibujo de formas, redimensionamiento, recorte, conversión de formatos y otras tareas más especializadas. Con este programa se ha creado el logo de W&A a partir de una imagen existente.

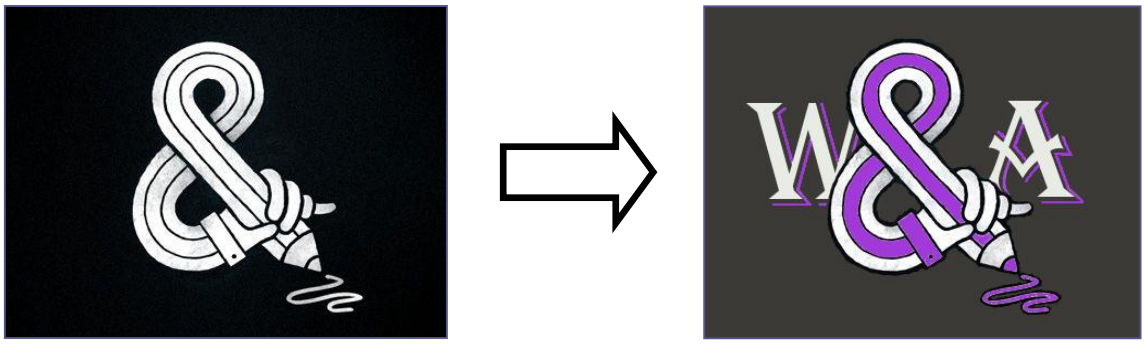

Ilustración 36 – Logo W&A

5.3.4. Otras

- Google Drive. Servicio de alojamiento de archivos utilizado para mostrar y compartir los avances de desarrollo con el tutor del proyecto.

- Símbolo del sistema. Intérprete de comandos de Windows. Se ha usado principalmente para conectar con MongoDB.

- Notepad. Editor de textos con soporte a muchos lenguajes de programación. Utilizado para modificar y visualizar algunos ejemplos de código.

- Pencil. Herramienta de prototipado. Con ella se han creado los diferentes bocetos de la interfaz de usuario.

- UMLet. Programa de modelado. Utilizado para dibujar los diagramas de casos de uso y de clases vistos en el capítulo 3 de este documento.

- WebSequenceDiagrams. Herramienta de modelado en línea. Usada para dibujar los diagramas de secuencia mostrados en el apartado de análisis.

- Microsoft Word. Procesador de textos. Con él se ha redactado y se le ha dado formato a la memoria del trabajo final de grado.

- Bloc de notas. Editor de texto básico de Windows. Utilizado para realizar apuntes recordatorios o guardar notas relativas a aspectos específicos del proyecto.

- Navegadores. Google Chrome, Mozilla Firefox, Internet Explorer, Safari y Microsoft Edge son los navegadores utilizados para probar la versión final de la aplicación. Google Chrome ha sido el protagonista para testear todas las modificaciones que se han realizado a lo largo del proyecto.

5.4. Interfaz de usuario

5.4.1. Introducción

Al tratarse de una aplicación destinada a usuarios reales, el aspecto de cada una de sus páginas debe ser agradable para la vista. Un sitio web que posea una funcionalidad maravillosa, tiende a quedar en desuso si su interfaz gráfica no es atractiva e intuitiva, ya que este hecho suele desencadenar una experiencia negativa en los usuarios del sitio. Por tanto, en W&A se ha cuidado mucho el aspecto. Mediante el uso de Bootstrap y CSS para determinar el estilo de las páginas, y jQuery para crear efectos dinámicos sobre las mismas, se ha conseguido crear una interfaz vistosa y sencilla de utilizar que garantiza una retroalimentación positiva en el usuario.

Para conseguir este objetivo, el primer paso ha sido incluir las librerías y hojas de estilo necesarias en el <*head*> de cada una de las páginas HTML que forman la aplicación, como se ve en la ilustración 37.

```
<head>
<!-- Stylesheets -->
<link rel="stylesheet" href="../bootstrap-3.3.6-dist/css/bootstrap.min.css"/>
<link rel="stylesheet" href="../annotator-1.2.10/css/annotator.css">
<link rel="stylesheet" href="../css/welcomePage.css"/>

<!-- Scripts -->
<script src="../annotator-1.2.10/lib/vendor/jquery.js"></script>
<script src="../bootstrap-3.3.6-dist/js/bootstrap.min.js"></script>
</head>
```

Ilustración 37 - Código de importación de librerías

Tras esto, las funciones y estilos programados en dichas librerías serán cargados sobre la página HTML, modificando así la visualización de la interfaz. En los siguientes puntos se muestran una por una, las diferentes pantallas de la aplicación, que son la evolución de los bocetos iniciales mostrados en el capítulo 2.3.1 de este documento. También se habla de algunos aspectos importantes como el *responsive design* y la usabilidad. Por último se mencionan algunos de los obstáculos encontrados a lo largo del desarrollo de la interfaz.

5.4.2. Página de bienvenida

Se trata de la primera página a la que accede el usuario cuando utiliza W&A. En la ilustración 38 se muestra su aspecto.

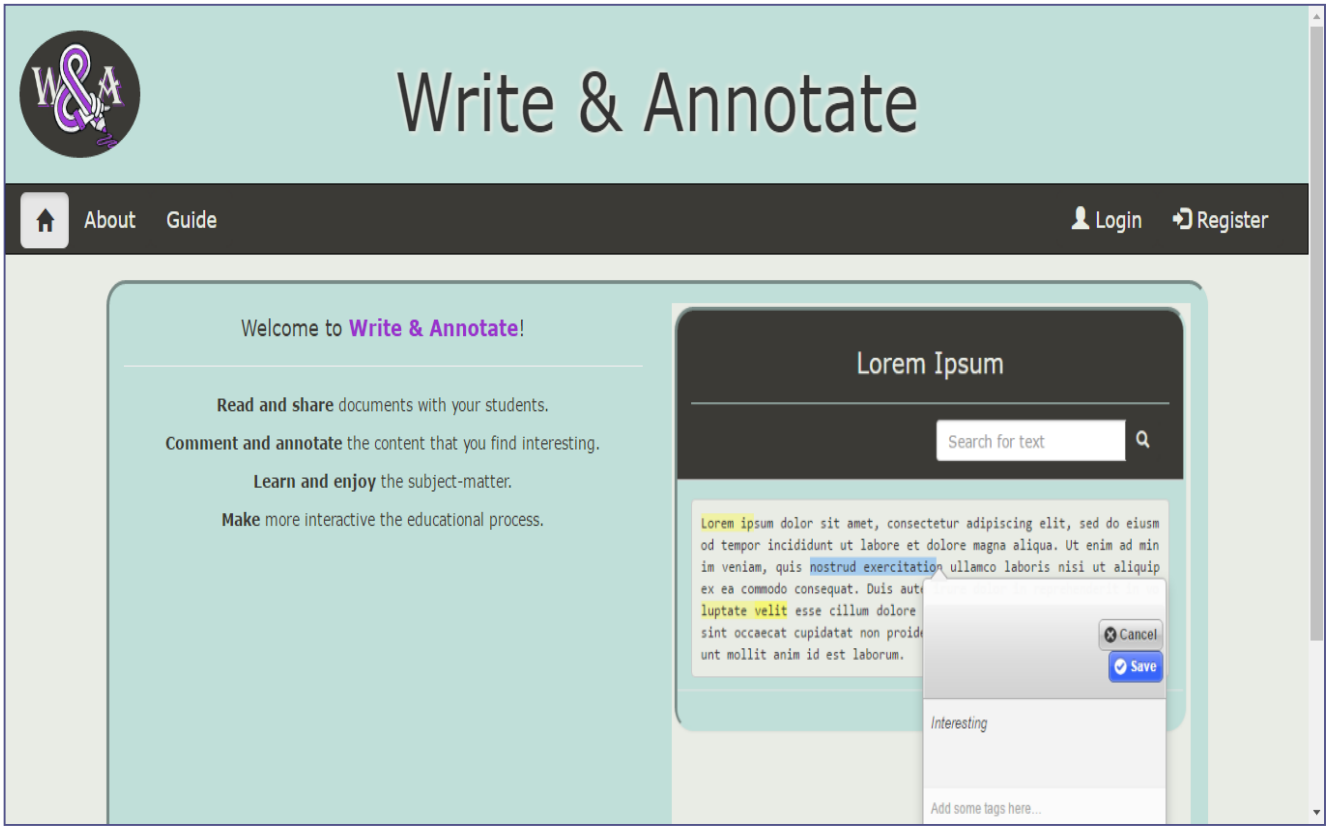

Ilustración 38 - Página de bienvenida

En primer lugar, en la parte superior de la imagen se puede observar la cabecera de la página, que está formada por el logo y el nombre de la aplicación. En segundo lugar se encuentra el menú de navegación, formado por un conjunto de botones que al interactuar con ellos cambian el contenido de la página. En tercer lugar, en el centro de la pantalla reside un panel mostrando el contenido de la web. Por último, aunque no está visible en la ilustración 38 (en las siguientes sí), al final de la página se encuentra el *footer*, un elemento común a todas las pantallas que forman W&A.

Volviendo al menú de navegación, los tres primeros botones (*home*, *about* y *guide*), pueden ser clicados para modificar la información de la página. El *home* muestra lo mismo que la ilustración anterior, un mensaje de bienvenida y una imagen relativa a un ejemplo de la funcionalidad ofrecida por la librería *annotator*. El segundo de ellos informa al usuario sobre el origen y otros aspectos de la aplicación. El tercer botón enseña un ejemplo interactivo de cómo realizar anotaciones sobre un texto. Las ilustraciones 39 y 40 corresponden a lo explicado en este párrafo.

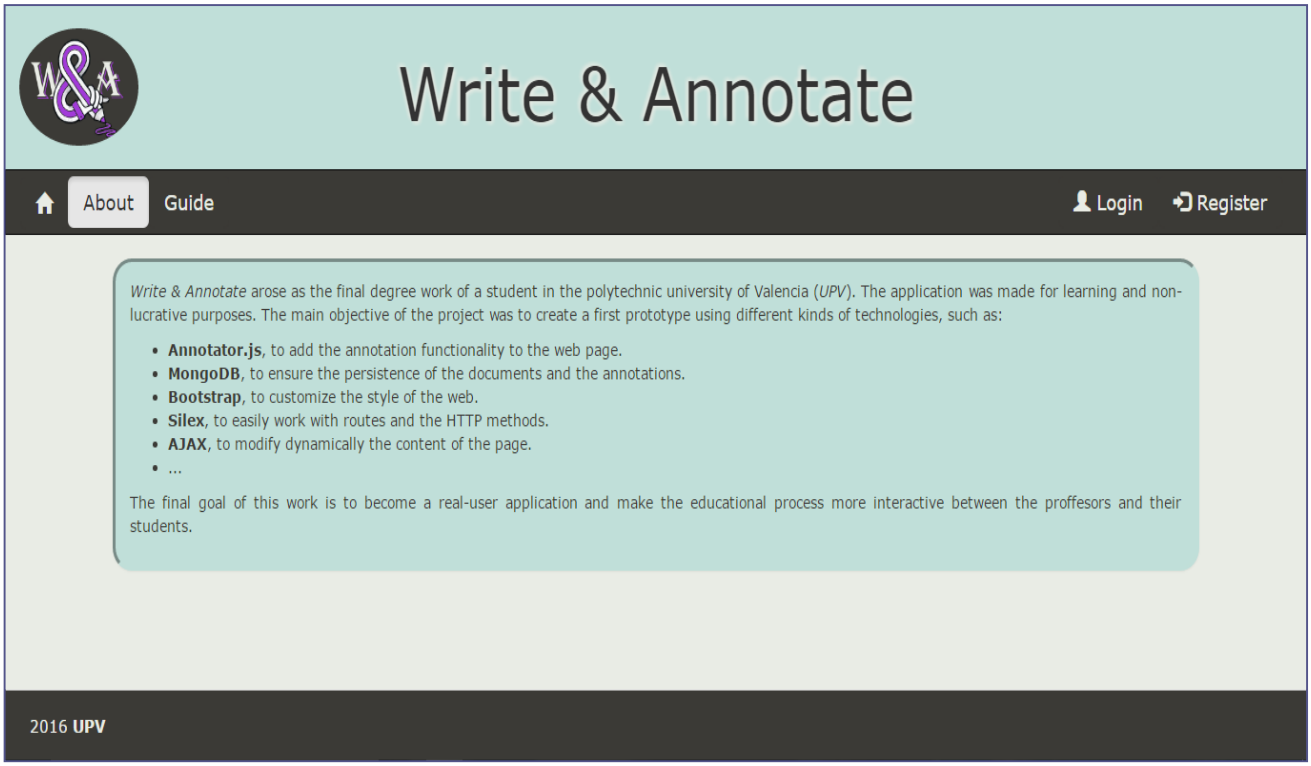

Ilustración 39 - Página de bienvenida (*about*)

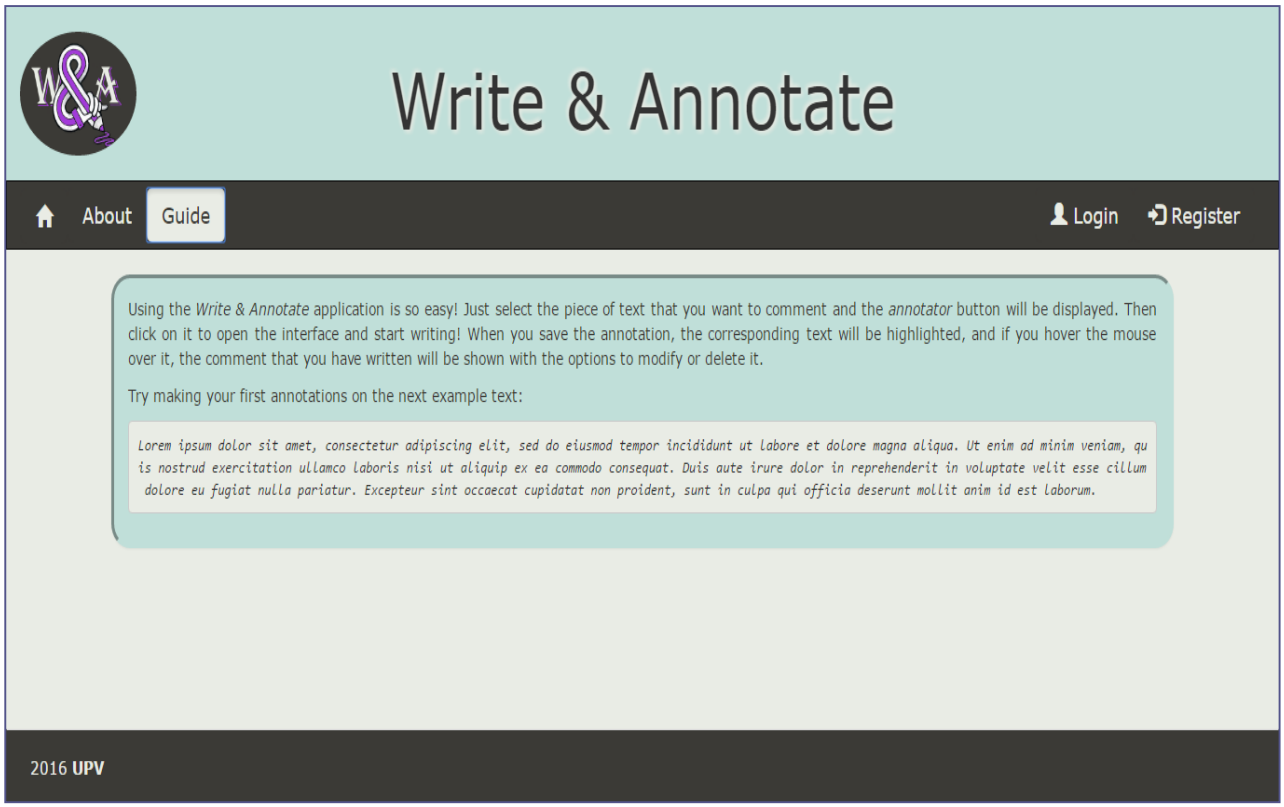

Ilustración 40 - Página de bienvenida (*guide*)

Algo a destacar sobre las opciones del menú es que modifican dinámicamente el contenido sin necesidad de recargar la página. Este hecho se consigue gracias al siguiente *script* escrito en jQuery.

```
<script>
$("[id*='menuButton']").click(function(){
    var buttonId = $(this).attr("id");
    var menuButtonNumber = buttonId.charAt(buttonId.length-1);
    var contentId = "menuContent"+menuButtonNumber;

    if($(this).hasClass("active")){}

    else{
        $("[id*='menuButton']").removeClass("active");
        $(this).addClass("active");
        $("[id*='menuContent']").addClass("hide");
        $("#"+contentId).hide();
        $("#"+contentId).removeClass("hide");
        $("#"+contentId).fadeIn(1000);
        $('html, body').animate({
            scrollTop: $("#"+contentId).offset().top + 'px'
        }, 'fast');
    }
});
</script>
```

Ilustración 41 - Código de modificación del contenido de la página

La función utiliza el número del botón correspondiente (variable *menuButtonNumber*) y el atributo *id* de cada uno de los contenidos de los paneles (variable *contentId*) para modificar la información mostrada. Se juega con los atributos *class* de cada elemento HTML para ocultar primero el contenido actual, y seguidamente mostrar los datos asociados al botón pulsado. Además, se realiza un *scroll* en la página hasta el contenido correspondiente con el objetivo de facilitar la visualización a aquellos usuarios que utilicen dispositivos pequeños, en los que el encabezado de la página y el menú puedan ocupar más de media pantalla.

Los otros dos botones del menú son el de *login* y el de *register*. Al clicar el primero de ellos se muestra un panel emergente (modal) que se explica en el apartado 5.4.6. del documento. El segundo redirecciona al usuario a la página de registro, donde puede insertar sus datos para empezar a formar parte de la aplicación.

5.4.3. Página de registro

Tras pulsar el botón *register* de la página de bienvenida, al usuario se le muestra la interfaz de la ilustración 42.

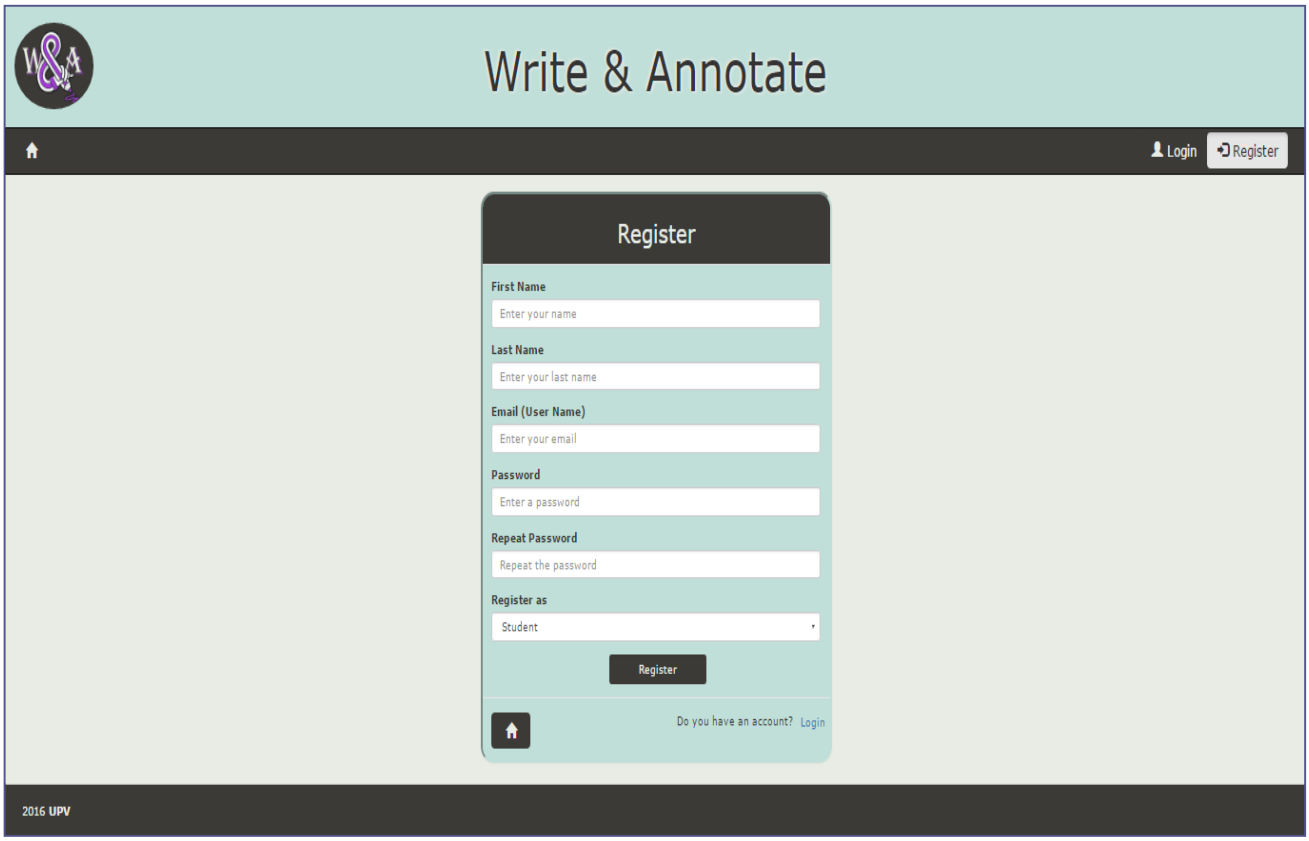

Ilustración 42 - Página de registro

En ella se pueden observar de nuevo la cabecera, el pie de página y el menú de navegación. En este caso, los botones que forman el menú corresponden al *home* (que

redirecciona al usuario a la página de bienvenida), al *login* (con el mismo efecto que el de la página de bienvenida) y al *register*.

En el panel central se muestra el formulario correspondiente a la operación de registro, donde el usuario puede rellenar los diferentes campos con su información personal y, si los datos son válidos, enviar posteriormente la solicitud de registro al servidor. En el pie del panel se observa de nuevo el botón *home* y un enlace que abre el modal del *login*, cuyo propósito en esta ocasión, es mejorar la accesibilidad a dichas opciones, ya que el usuario no debería tener a la vista el menú de navegación una vez ha llegado al final de la página. De tal modo, evitaría volver a "subir" al inicio de la misma.

Aunque en esta sección no se habla de cómo se ha implementado la validación de los campos del formulario, hay algunos aspectos visuales interesantes que merece la pena comentar. El usuario necesita conocer si la interacción con los elementos web se desenvuelve correctamente o no. Asimismo, cuando pulsa el botón de registro debería ser informado de si todo ha ido bien, o si en cambio se ha encontrado algún inconveniente. En W&A se ha programado un mecanismo para tal propósito, que muestra al usuario los siguientes resultados:

- Campo erróneo. El usuario deja incompleto algún campo o introduce un valor incorrecto, como por ejemplo el de la ilustración 43, donde el formato del email no es válido.

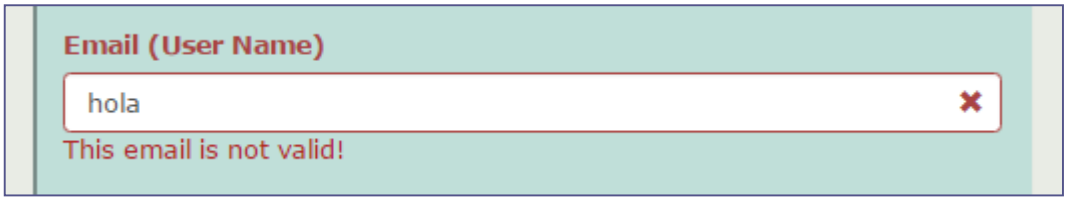

Ilustración 43 - Campo del formulario incorrecto

- Campo válido. El usuario inserta correctamente sus datos en el campo correspondiente. En este caso, la imagen muestra el resultado contrario al anterior.

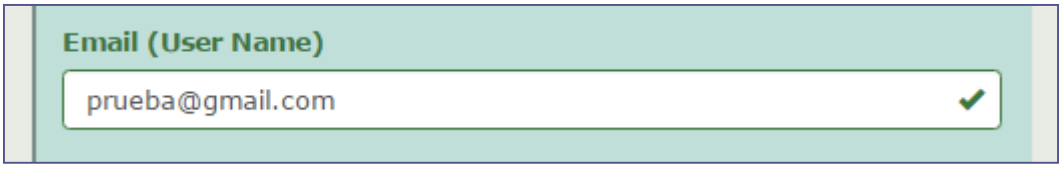

Ilustración 44 - Campo del formulario válido

Como se puede apreciar, mediante esta forma de visualización, al usuario le queda completamente claro cuando algo sale bien o mal. A continuación se detallan las funciones javascript utilizadas para implementar esta interfaz.

```
<script>
function addError(inputId){
	$("#"+inputId+"Input").addClass("has-
feedback").removeClass("has-success").addClass("has-error");
	$("#"+inputId+"Success").addClass("hide");
	$("#"+inputId+"Error").removeClass("hide");
}


function addSuccess(inputId){
	$("#"+inputId+"Input").addClass("has-
feedback").removeClass("has-error").addClass("has-success");
	$("#"+inputId+"Error").addClass("hide");
	$("#"+inputId+"Success").removeClass("hide");
}


function hideErrors(inputId){
	$("#"+inputId+"ErrorMessageBlank").addClass("hide");
	$("#"+inputId+"ErrorMessageInvalidEmail").addClass("hide");
	$("#"+inputId+"ErrorMessageShortPassword").addClass("hide");
	$("#"+inputId+"ErrorMessageDifferentPassword").addClass("hide");
}
</script>
```

Ilustración 45 - Código de visualización de la validación del formulario

Estas funciones son llamadas dentro del código asociado a la validación del formulario, detallado en el punto 5.5.2. En las dos primeras se trabaja con las clases predefinidas en Bootstrap, *has-feedback* y *has-success/has-error*, las cuales construyen el aspecto visto en las ilustraciones 43 y 44. Básicamente, los métodos *addError* y *addSuccess* se centran en eliminar el estado anterior del campo (válido o erróneo) y, tras eso, añaden el nuevo. La última de ellas se basa en eliminar los mensajes de error asociados al campo del formulario con identificador equivalente a *inputId*.

Una vez los datos del formulario estén validados, se envían al servidor y éste procede con las comprobaciones oportunas en la base de datos. Si se produce un error (el usuario ya existía previamente o la base de datos está desconectada) también se informa al usuario, de manera muy similar a la anterior.

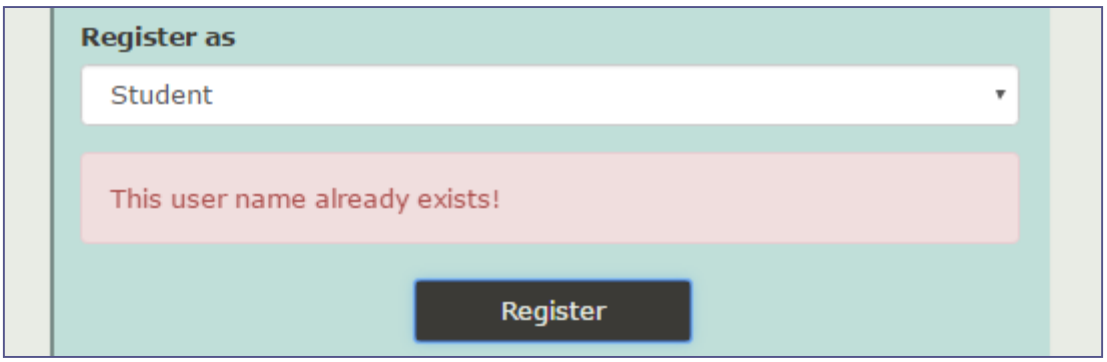

Ilustración 46 - Error en el registro

```
$("#errorBox").fadeOut(function(){
        $("#errorBox").fadeIn("slow");
        $("#errorBox").fadeOut(5000);
});

$("#dbErrorBox").fadeOut(function(){
        $("#dbErrorBox").fadeIn("slow");
        $("#dbErrorBox").fadeOut(5000);
});
```

Ilustración 47 - Código de error en el registro

Como se puede observar, la mecánica es parecida. En este caso, el mensaje de error posee un efecto de desvanecimiento ofrecido por jQuery.

Finalmente, cuando el usuario consigue registrarse satisfactoriamente, W&A muestra por pantalla un panel personalizado informando del éxito. Este elemento se describe con detalle en el apartado 5.4.6.

5.4.4. Página de listados

Si el usuario efectúa correctamente la acción de *login* (explicada en el apartado 5.5.3.), visualizará pantalla de la ilustración 48.

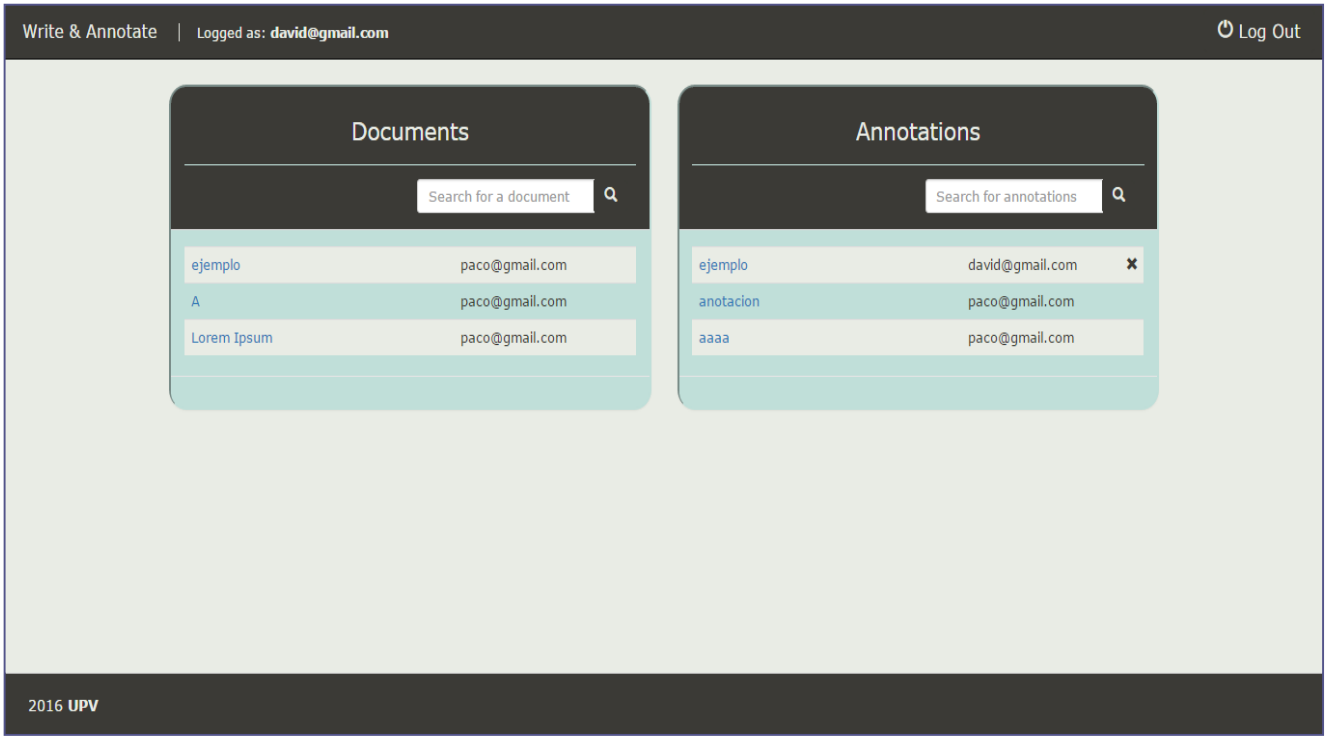

Ilustración 48 - Página de listados (vista alumno)

En la parte superior se encuentra la barra de navegación, compuesta por el nombre de la aplicación, la cuenta con la que el usuario se ha identificado y un botón de *log out*, que sirve para cerrar la sesión. En el centro de la pantalla se observan dos paneles prácticamente

idénticos que conforman el listado de documentos y anotaciones. Ambos poseen, en primer lugar, una cabecera con su título y una opción para buscar elementos de la lista (aunque la funcionalidad no llegó a implementarse). En segundo lugar, se muestran por filas los documentos y anotaciones creadas por los usuarios. Cada fila dispone de un enlace con el nombre del documento (o el texto de la anotación), el autor y un botón de borrado.

Se puede apreciar en la ilustración 48, que únicamente existe un elemento (anotación) con la opción de borrado disponible. Esto es debido al sistema de permisos de W&A. Por un lado, los usuarios registrados como alumnos solamente pueden eliminar sus propias creaciones. Como *david@gmail.com* es un alumno, no tiene permiso para visualizar la cruz de borrado sobre los elementos que no sean suyos. Por otro lado, los profesores pueden eliminar cualquier documento y anotación, sean o no sus propietarios. En la ilustración 49 se muestra la vista que tiene un profesor al identificarse.

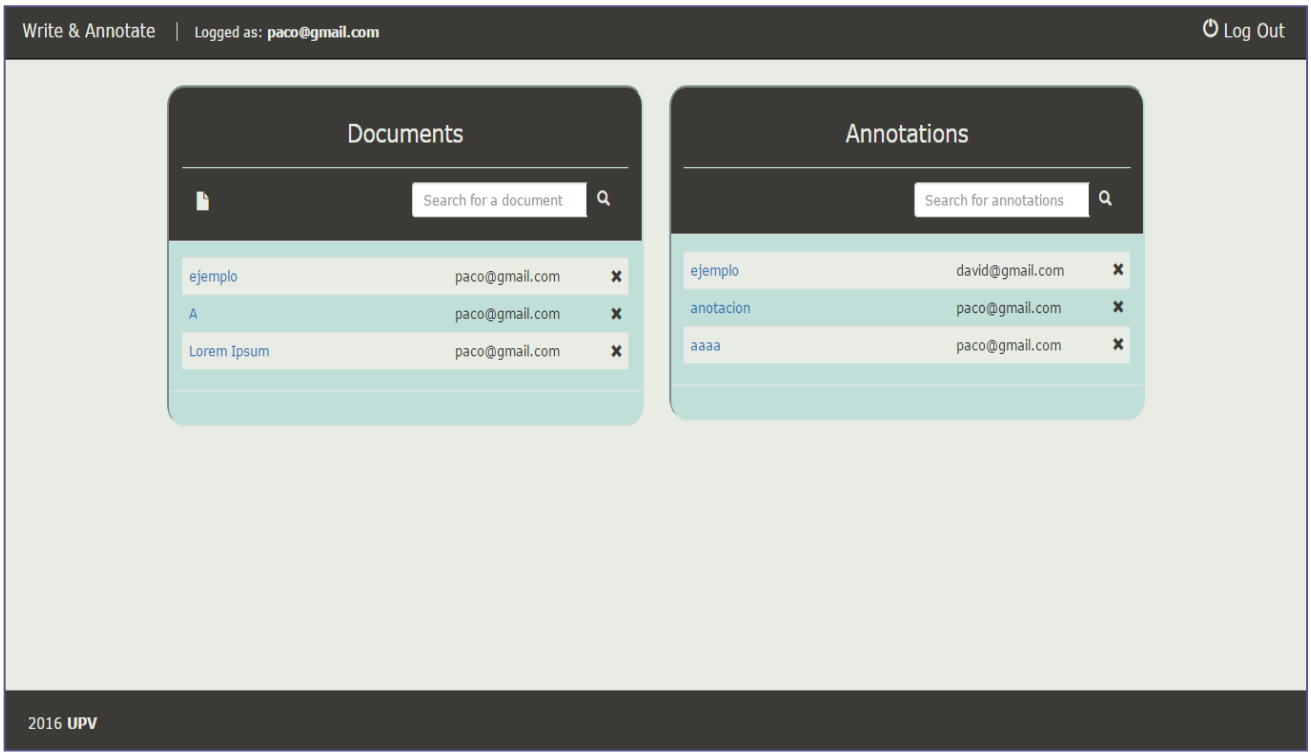

Ilustración 49 - Página de listados (vista profesor)

Se confirma lo comentado anteriormente, ya que el usuario *paco@gmail.com* es un profesor y puede eliminar cualquier elemento de las listas. Una vez pulsado el botón de borrado, aparece un modal de confirmación, el cual se explica en la sección 5.4.6. del documento. A continuación se enseña la función que permite ocultar este botón a los alumnos.

```
function disableDeleteButton(id){
	var userType = "<?php echo $_SESSION['userType'];?>";
	var user = "<?php echo $_SESSION['userEmail'];?>";

	if (userType == "teacher"){}
	else{
		var author = $("#"+id).closest('tr').children(':nth-
child(2)').text();
		if(author != user){
			$("#"+id).closest('tr').children(':nth-
child(3)').children().hide();
		}
	}
}
```

Ilustración 50 - Código para deshabilitar la opción de borrado

Haciendo uso de las variables de sesión se puede conocer el tipo de usuario y su email (nombre de usuario). Para un alumno, se comprueba si es el autor del documento/anotación, y si no lo es, se oculta la opción de borrado asociada a dicho elemento.

En adición, para los profesores aparece un nuevo botón en la cabecera del panel de documentos. Se trata de la opción para subir documentos a la aplicación. Al clicar este botón, el usuario puede seleccionar un archivo de texto de su máquina y subirlo al servidor para que sea visible por todos los integrantes de W&A.

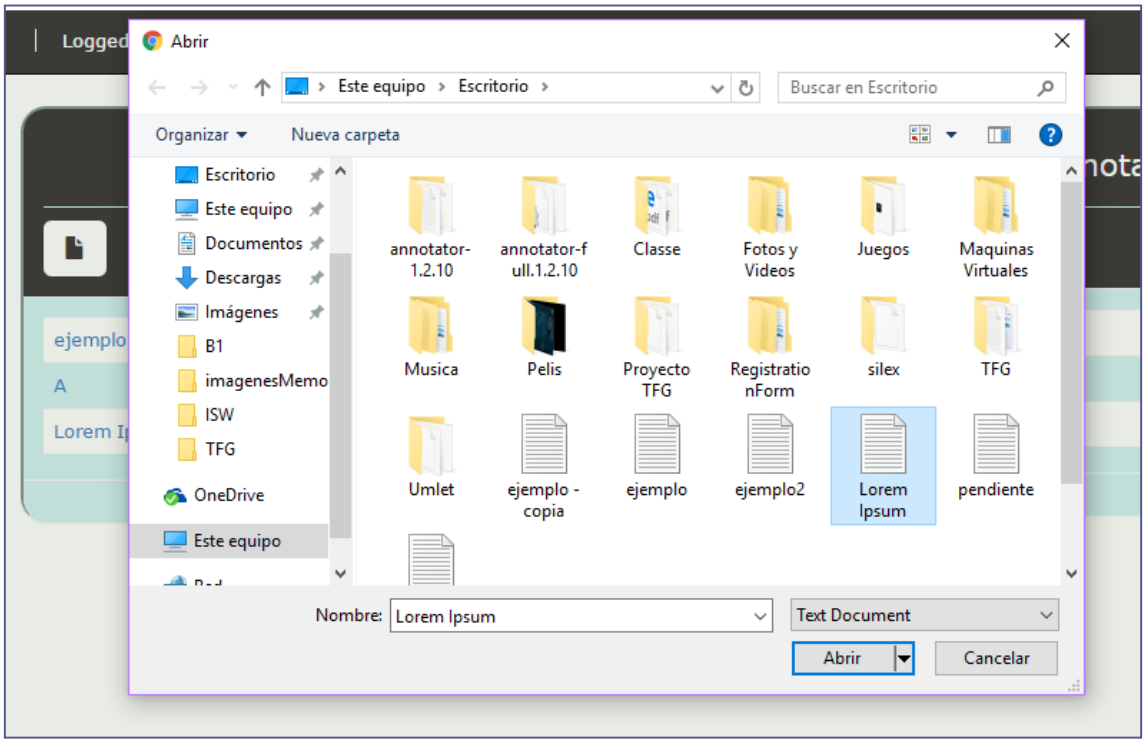

Ilustración 51 - Opción de subida de documentos

Usualmente, este tipo de opciones están compuestas por dos elementos HTML: un botón para seleccionar el archivo y otro para subirlo al servidor. En este caso se ha

programado una alternativa para simular las dos funciones con un solo botón, lo cual acelera el trabajo del usuario.

```
<script>
$("#fileUploadButton").click(function(){
	$("#fileUploadInput").click();
});

$("#fileUploadInput").change(function(){
	$("#fileUploadForm").submit();
});
</script>
```

Ilustración 52 - Código para subir documentos

Esta cadena de manejadores de eventos con jQuery provoca que, al clicar el botón de subida (*id = fileUploadButton*), se "active" el elemento oculto encargado de enviar los datos del formulario al servidor. De esta manera, se muestra únicamente uno de los dos botones que participan en la subida del documento. En los apartados 5.5.8. y 5.5.11. se detalla la implementación interna de la subida y borrado de documentos respectivamente.

5.4.5. Página de documento

Cuando un usuario utiliza alguno de los enlaces disponibles en los listados de documentos y anotaciones, es redirigido a la página mostrada en la ilustración 53.

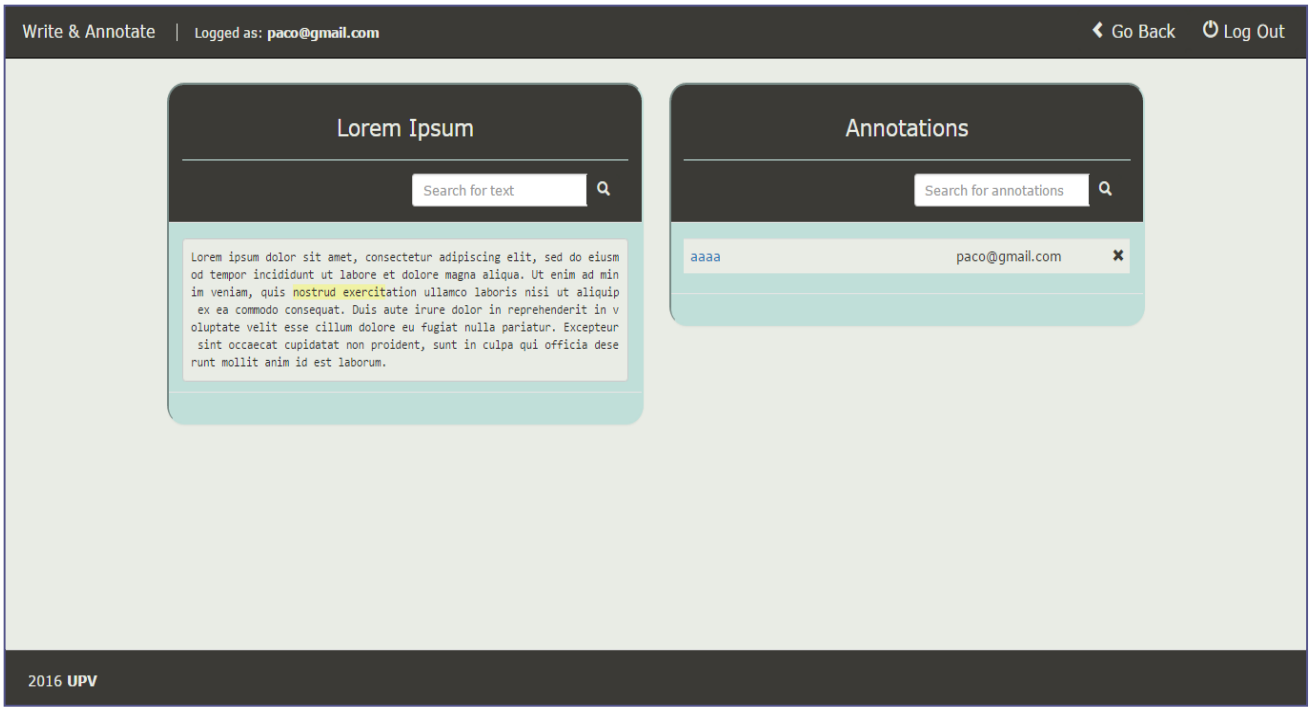

Ilustración 53 - Página de documento

A priori, en esta imagen se observan dos diferencias respecto a la página de listados. En primer lugar, se añade el botón *go back* en la barra de navegación, que permite al usuario volver a la página de listados. En segundo lugar, en vez del listado de documentos, aparece el texto del documento seleccionado por el usuario en la página anterior.

La principal característica de esta página es la integración de la librería *annotator.js* en el texto del documento. Esta herramienta incluye una interfaz propia, bastante intuitiva y fácil de utilizar. El proceso para insertar nuevas anotaciones sobre el contenido del documento se muestra a continuación, en la ilustración 54.

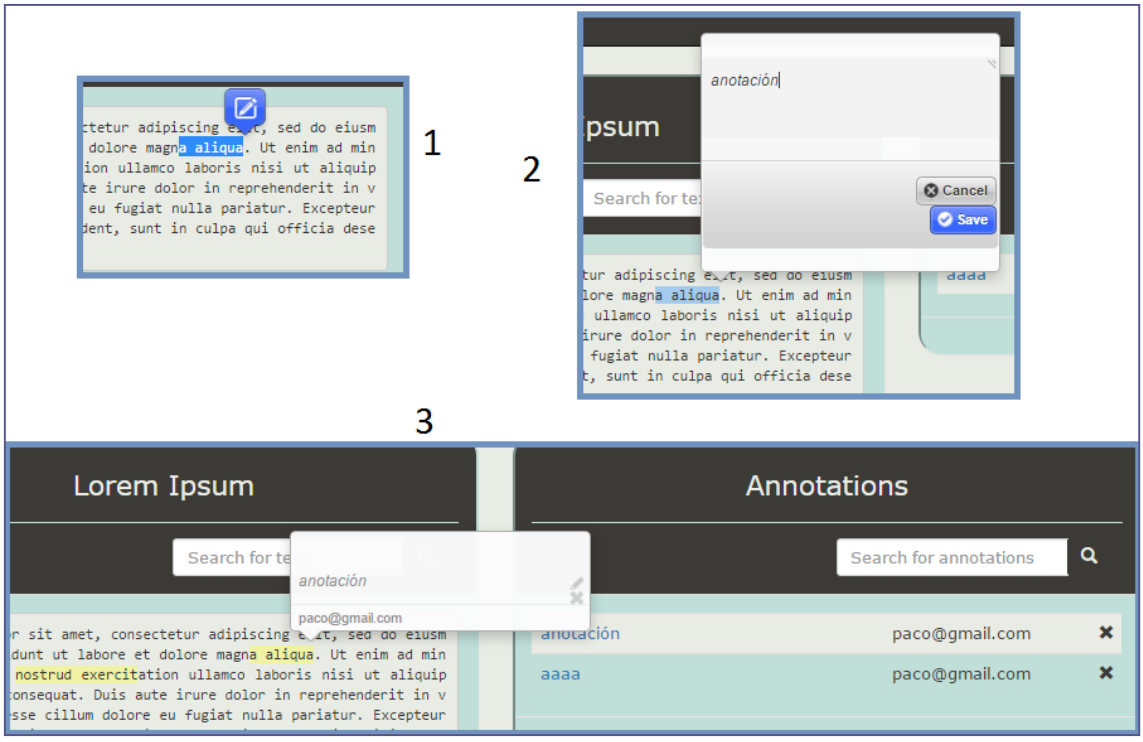

Ilustración 54 - Interfaz *Annotator*

Para empezar, el usuario selecciona un fragmento de texto, dando como resultado la aparición de un pequeño panel con el icono de un lápiz (1). Cuando se pulsa dicho icono, surge una nueva interfaz, mostrando un cuadro de texto donde el usuario puede escribir cualquier comentario y guardarlo, o bien cancelar la acción clicando los botones correspondientes (2). Por último, el texto anotado queda resaltado de color amarillo y la anotación se añade a la lista. Si el usuario coloca el cursor sobre el texto, aparece un nuevo panel informando del contenido de la anotación, el autor y los botones de edición y borrado (3). Si el usuario no es el creador de la anotación, no se mostrarán las opciones para eliminarla o modificarla.

Dado que este capítulo se centra principalmente en la visualización de la interfaz, no se entran en más detalles técnicos sobre el uso de *annotator*. Dicha información se detalla en la sección 5.5.5.

5.4.6. Modales

Con javascript se pueden generar cuadros de texto emergentes con el objetivo de informar al usuario. Estos mensajes cumplen con su función informativa, pero su problema es la apariencia.

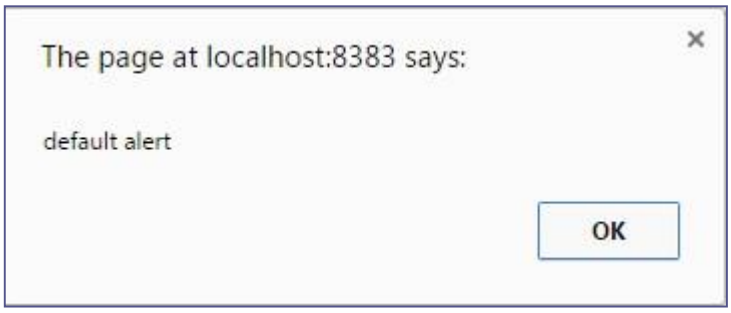

Ilustración 55 - Cuadro de alerta javascript

Se ha comentado varias veces la importancia de tener una interfaz agradable y vistosa para el usuario. Las alertas de javascript no cumplen con ese objetivo y por tanto se ha buscado otra opción para visualizar los mensajes dirigidos al usuario.

Una de las extensiones de Bootstrap ofrece alternativas a las alertas javascript. Se trata de los modales, unos cuadros de diálogo personalizables que aparecen en el centro de la pantalla en respuesta a un evento. Su uso ha sido fundamental para implementar varias funciones de la aplicación. Además, su apariencia se ha programado acorde con la estética del sitio web.

En W&A existen cuatro diferentes modales: *login*, confirmación de borrado, confirmación de *log out* y mensaje informativo al crear una cuenta de usuario. Todos ellos se componen de una cabecera, un cuerpo y un *footer*. Adicionalmente, aplican un efecto de sombreado al resto de la página con el objetivo de fijar en ellos la atención del usuario. A continuación se explican cada uno de ellos.

Login

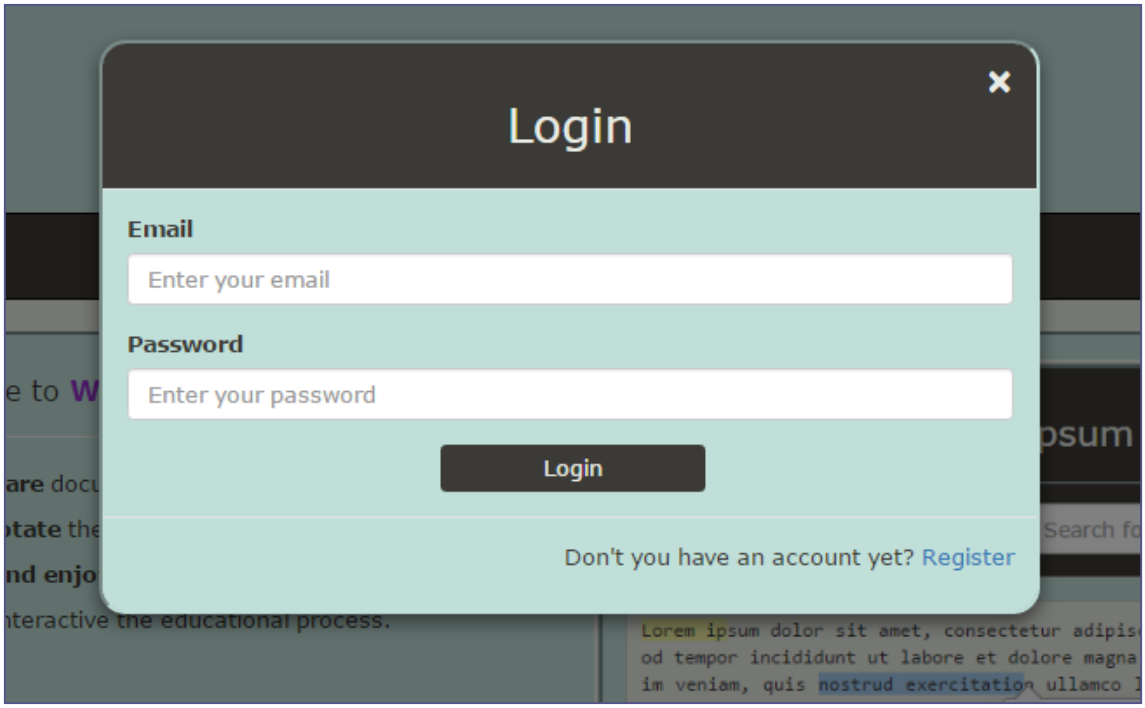

Ilustración 56 - Modal de *login*

Este modal es accesible desde la página de bienvenida o la de registro, y se encarga de mostrar al usuario el formulario de *login*. Para interactuar exitosamente con él, deben insertarse un email y contraseña válidos en los campos correspondientes y pulsar el botón de *login* para enviar los datos al servidor. Tras esto, el usuario será redirigido a la página de listados.

En cambio, si los datos no son correctos, se muestra un mensaje de error, implementado de forma equivalente a la validación del formulario de registro (apartado 5.4.3.).

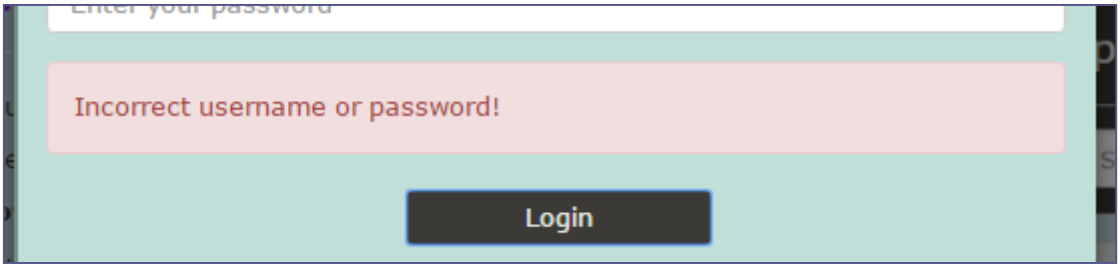

Ilustración 57 - Error en el *login*

Para visualizar el modal, se utiliza el siguiente método jQuery:

```
$(".loginButton").click(function(){
        $("#loginModal").modal();
});
```

Ilustración 58 - Código para mostrar el modal de *login*

Confirmación de borrado

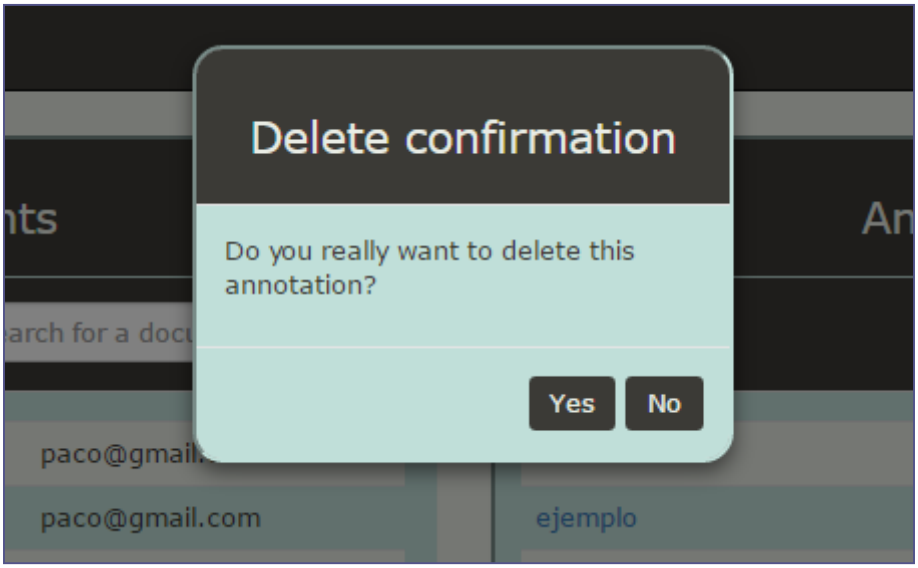

Ilustración 59 - Modal de confirmación de borrado

En esta ocasión, el usuario tiene dos botones a su disposición. Si confirma la acción, el elemento correspondiente será borrado permanentemente de las listas y la base de datos. En cambio, si pulsa el botón "no", la operación se rechaza y se cierra el modal, sin efectuar acción alguna.

A diferencia del modal de *login*, aquí se utiliza una operación de renderizado que modifica el contenido del cuadro según el tipo de elemento, ya que resulta conveniente diferenciar cuando se quiere borrar un documento y cuando una anotación.

```
function render(header, body, footer, id, elementType){
    document.getElementById('customModalHeader').innerHTML = header;
    document.getElementById('customModalBody').innerHTML = body;
    document.getElementById('customModalFooter').innerHTML = footer;
}

function deleteTableRow(thisObj, elementType){
      var elementId =
thisObj.closest('tr').children().children().attr('id');
      var elementContent =
thisObj.closest('tr').children().children().text();

      if (elementType=="document"){
            render('<h3 role="delete-confirm-header">Delete
confirmation</h3>',
                  '<p role="delete-confirm-text">Do you really want
to delete the document: <strong>'+elementContent+'</strong>?</p>',
                  '<button class="btn btn-default btn-grey-hover"
onclick="confirmDelete(\''+elementId+'\',\''+elementType+'\')"
role="delete-button">Yes</button>'+
                   '<button class="btn btn-default btn-grey-hover"
onclick="confirmRefuse()" role="deleteCancel-button">No</button>',
                  elementId, elementType);
      }
      else{
```

```
            render('<h3 role="delete-confirm-header">Delete
confirmation</h3>',
                    '<p role="delete-confirm-text">Do you really want
to delete this '+elementType+'?</p>',
                   '<button class="btn btn-default btn-grey-hover"
onclick="confirmDelete(\''+elementId+'\',\''+elementType+'\')"
role="delete-button">Yes</button>'+
                    '<button class="btn btn-default btn-grey-hover"
onclick="confirmRefuse()" role="deleteCancel-button">No</button>',
                     elementId, elementType);
      }

      $("#customModal").modal({backdrop: 'static', keyboard: false});
}
```

Ilustración 60 - Código para renderizar el modal de confirmación

La primera función de la ilustración 60 se encarga de la visualización final del modal, basándose en la información obtenida de la segunda función.

El método *deleteTableRow(thisObj, elementType)*, distingue que tipo de elemento se quiere eliminar (documento o anotación) y modifica la estructura del modal según corresponda (cabecera, cuerpo y *footer*). El último comando de este método obliga al usuario a cerrar el modal únicamente interactuando con los botones. Por defecto, al clicar fuera del cuadro de diálogo, o al pulsar la tecla *esc*, el modal se cerraría automáticamente.

Confirmación de log out

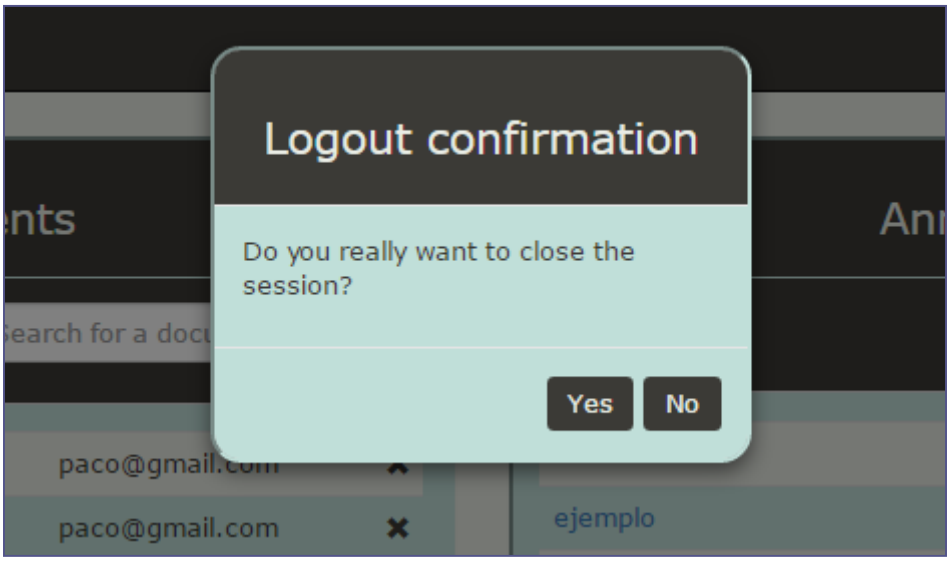

Ilustración 61 - Modal de confirmación de *log out*

Se muestra al clicar el botón de *log out*. Sigue el mismo comportamiento binario que el modal anterior, pero al confirmar, esta vez la función ejecutada consiste en cerrar la sesión del usuario y devolverlo a la página de bienvenida. El código empleado para crear la interfaz es casi idéntico al caso anterior, así que no se entrará en más detalles.

Registro realizado con éxito

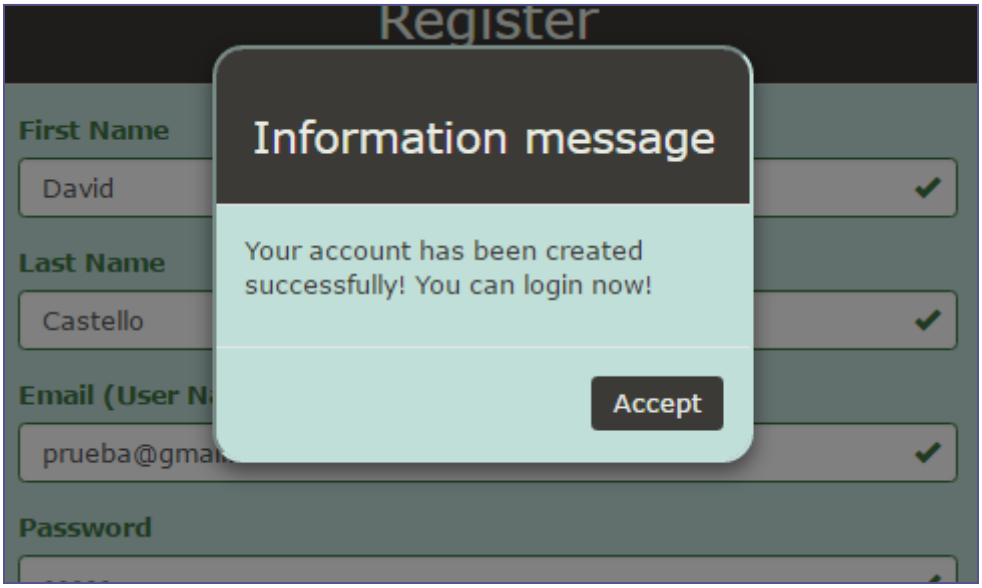

Ilustración 62 - Modal con mensaje informativo

Por último, este modal muestra un mensaje cuando el usuario ha conseguido registrarse satisfactoriamente. En esta ocasión, únicamente se dispone de un botón con el que interactuar, el cual no efectúa ningún cambio en la aplicación, ya que se trata de un mensaje meramente informativo.

5.4.7. *Responsive design*

El *responsive design* es una filosofía de diseño y desarrollo cuyo objetivo es adaptar la apariencia de las páginas web al dispositivo que se esté utilizando para mostrarlas. Hoy en día, las páginas web pueden visualizarse en multitud de dispositivos, como tabletas, móviles, portátiles, PCs, libros electrónicos… Además, dentro de cada tipo existen unas características concretas: tamaño de pantalla, resolución, potencia de la CPU, sistema operativo, capacidad de la memoria… Esta tecnología pretende que con un único diseño web, se obtenga una visualización adecuada para cada dispositivo.

W&A utiliza Bootstrap y su sistema de rejilla para conseguir el *responsive design* en todas las páginas que la forman. Con el objetivo de garantizar esta técnica, se incluye la siguiente etiqueta en la cabecera de las páginas HTML.

```
<meta name="viewport" content="width=device-width, initial-scale=1"/>
```

Ilustración 63 - Etiqueta *meta viewport*

El atributo *width=device-width* provoca que el ancho de la página sea equivalente al ancho del dispositivo utilizado para visualizarla. En segundo lugar, el atributo *initial-*

scale=1 establece el zoom inicial cuando la página es cargada la primera vez por el navegador.

Sistema de rejilla de Bootstrap

Respecto al sistema de rejilla de Bootstrap, permite distribuir el contenido de la página en filas y columnas. Cada fila dispone de un máximo de doce columnas, que pueden tener distintos tamaños. Este sistema tiene predefinidas cuatro clases que representan el tamaño del dispositivo: *xs* (muy pequeño: <768px), *sm* (pequeño: ≥768px y <992px), *md* (mediano: ≥992px y <1200px) y *lg* (grande: ≥1200px). En la ilustración 64 se muestra un ejemplo.

```
<div class="container">
    <div class="panel panel-default" id="menuContent0" role="page-
content">
        <div class="row panel-body row-centered">
            <div class="col-sm-6 col-xs-12">
...
```

Ilustración 64 - Ejemplo: clases del sistema de rejilla de Bootstrap

En el código anterior, todo el contenido HTML insertado dentro de la última etiqueta *div*, ocupará seis columnas en los dispositivos pequeños (*sm*) o de tamaño mayor, y doce columnas en los dispositivos muy pequeños (*xs*).

Media queries de CSS

Otra opción interesante son las *media queries*, un mecanismo incluido en las hojas de estilo CSS utilizado para adaptar el contenido web a las características del dispositivo. Algunos ejemplos usados en W&A se muestran a continuación, en la ilustración 65.

```
@media screen and (max-width: 768px) {
      .navbar-right{
            margin-right:-15px;
      }

      .row-centered{
      text-align:center;
      }

      .col-centered{
      display:inline-block;
      float:none;
      text-align:left;
      margin:0px;
      padding:0px;
      }

      .navbar-form{
            padding-right:5px;
            padding-left:5px;
            padding-top:0px;
            margin:0px;
```

```
            margin-left:20px;
            margin-right:20px;

        }

        .panel-heading{
            padding-bottom:0px;
        }
}
```

Ilustración 65 - Código de las *media queries* de CSS

En el código anterior se muestran varias clases predefinidas en Bootstrap, que se utilizan en algunos elementos HTML para modificar su estilo. Con esta técnica, dichos elementos cambiarán las propiedades listadas arriba cuando el dispositivo sea muy pequeño (<768px).

Colapso de información con Bootstrap

En pantallas grandes, no suele importar la cantidad de elementos de una página, ya que todos (o la mayoría) son visibles para el usuario. En cambio, con los dispositivos pequeños, puede que algunos elementos ocupen mucho espacio en la pantalla, ocultando información relevante del resto de la página y obligando al usuario a utilizar la barra de desplazamiento para visualizarla.

En W&A surgió la necesidad de arreglar esto, ya que la cabecera y el menú de navegación ocupaban casi toda la pantalla de un dispositivo móvil. La solución consistió en introducir un botón de colapso en la barra del menú, que aparece cuando el ancho de la pantalla es muy pequeño. Con este método, las diferentes opciones del menú se convierten en un desplegable que aparece y desaparece al interactuar con el botón de colapso.

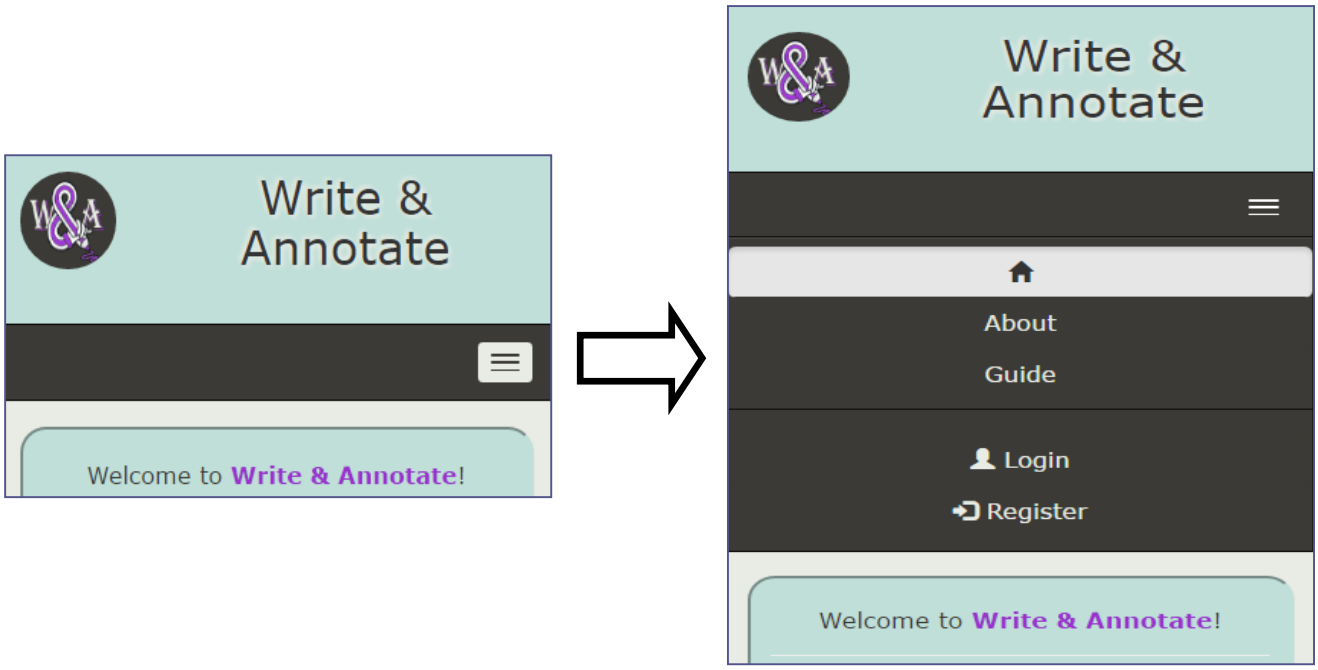

Ilustración 66 - Botón de colapso en dispositivos pequeños

5.4.8. Usabilidad

La usabilidad es la disciplina que estudia la forma de diseñar sitios web para que los usuarios puedan interactuar con ellos de la forma más fácil, cómoda e intuitiva posible. Una interfaz usable es imprescindible para una experiencia positiva del usuario, y por supuesto se ha tenido en cuenta a la hora de crear esta aplicación. W&A sigue las siguientes reglas para garantizar una buena usabilidad:

- Diseño sencillo. W&A presenta una interfaz amigable, con una combinación de colores agradable a la vista. Además, no existen elementos visuales que despisten al usuario o lo aparten de las labores principales del sitio web.

- Estructuración del contenido. Los elementos de la web están separados por espacios, garantizando una correcta distribución del contenido y evitando que los elementos se solapen unos con otros. También se ha organizado la información de manera que lo más importante sea identificado por el usuario a simple vista, sin tener que "esforzarse" en buscarlo.

- Rapidez. La espera para cargar las páginas y sus elementos es bastante aceptable.

- Informar al usuario. El visitante debe conocer en todo momento que está pasando y cuál es el resultado de sus acciones. Para ello se han programado diferentes mecanismos informativos. Por ejemplo, cuando se registra satisfactoriamente aparece un mensaje en forma de modal; al mantener el cursor sobre un botón, éste cambia de color; al enviar un formulario con campos incorrectos, se visualiza un error sobre los elementos

correspondientes; se utilizan colores representativos del éxito o error (verde y rojo respectivamente); cuando se selecciona una opción del menú, ésta queda marcada como "activa"; y un largo etcétera.

- Texto legible. Se utiliza una fuente y tamaño adecuado para el texto. Además es importante evitar los posibles errores ortográficos.

- Estilo consistente. La coherencia de diseño entre las páginas consigue generar en el usuario la impresión de "no haber abandonado el sitio". Se utilizan los mismos fondos, la misma combinación de colores, estilo de texto, botones…

- Documentación. Se le ofrece al usuario una pequeña guía sobre cómo utilizar la aplicación.

- Accesibilidad desde distintos navegadores. El usuario puede acceder a W&A desde el navegador web con el que se sienta más cómodo.

- Accesibilidad para los lectores de pantallas. Los usuarios con discapacidad visual, comúnmente utilizan lectores de pantalla para navegar por la web. Esta aplicación ofrece varias facilidades para ellos. En primer lugar, el uso del atributo *alt* en las imágenes HTML, permite mostrar texto alternativo a la visualización de las mismas. Otro atributo importante para mejorar la usabilidad es *role*, que facilita a los lectores de pantalla la identificación de los elementos más importantes de la página, como el formulario de *login* o el menú de navegación, etcétera. Por último, el atributo *for* utilizado en las etiquetas (*label*), hace posible localizar con que elemento está asociada cada etiqueta.

5.4.9. Problemas encontrados

En este apartado se listan los problemas más destacables que han aparecido al programar la visualización de la interfaz de usuario, así como las soluciones aportadas.

Alertas de javascript personalizadas

Como ya se comentó en el apartado 5.4.6., los cuadros informativos predefinidos en javascript son bastante modestos si se pretende construir una interfaz vistosa y bonita. Por tanto, se decidió buscar una alternativa.

En un principio se crearon alertas personalizadas utilizando CSS. El problema era que no cumplían con el *responsive design*, un aspecto primordial en W&A. El cuadro informativo emergente quedaba anclado al centro de la pantalla permanentemente, pero no se adaptaba al redimensionamiento de la ventana, provocando que se cortase por los extremos en pantallas pequeñas.

Finalmente, se optó por utilizar los modales de Bootstrap vistos en el punto 5.4.6., una herramienta extremadamente útil para mostrar cuadros informativos en respuesta a la interacción del usuario con la interfaz gráfica.

Visualización de la interfaz *annotator*

Al disponer de varias hojas de estilos para crear el diseño de la web, en ocasiones los elementos no se visualizan como se esperaría. Este hecho ocurrió con la interfaz proporcionada por la librería *annotator*, donde los cuadros emergentes quedaban cortados por su contenedor, impidiendo la correcta interacción entre el usuario y la aplicación. La ilustración 67 muestra el problema.

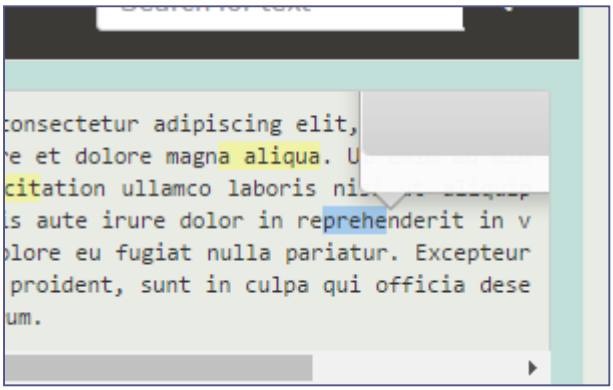

Ilustración 67 - Defecto de visualización de la interfaz *annotator*

La solución a este problema resultó ser bastante sencilla. Con la siguiente línea de código CSS, el elemento seleccionado sobrepasa los límites de su contenedor, permitiendo así una visualización completa y correcta del mismo.

```
.annotableText{
	overflow:visible;
}
```

Ilustración 68 - Código para visualizar correctamente la interfaz *annotator*

Pie de página

El *footer* de una página es un elemento situado habitualmente al final de la misma. Cuando se programó por primera vez, se situaba justo detrás del contenido de la página, generando un espacio vacío entre el *footer* y el final de la web. Este defecto se muestra a continuación en la ilustración 69.

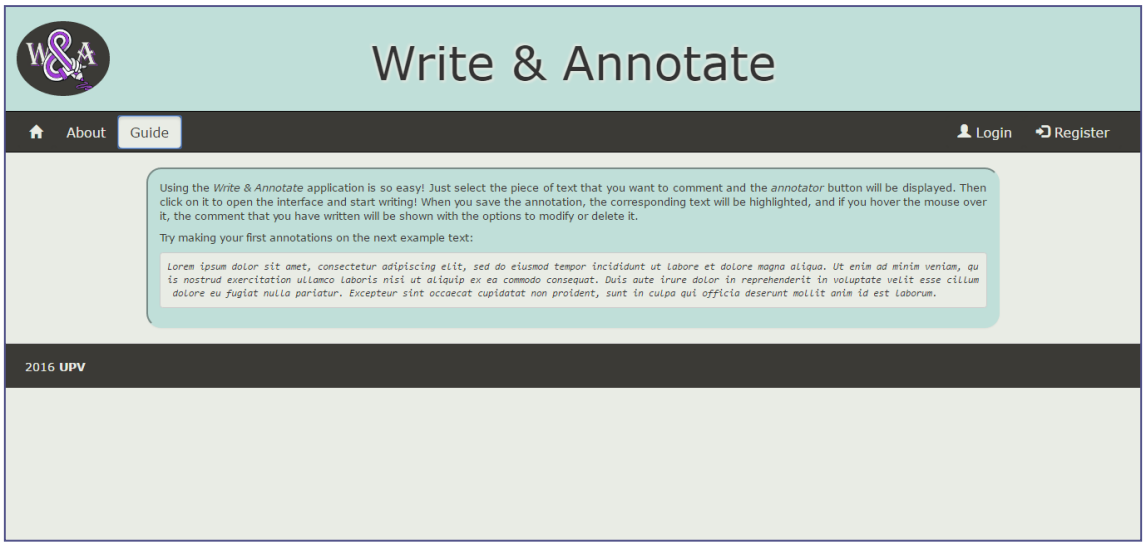

Ilustración 69 - Defecto de visualización del *footer*

Para solucionar este inconveniente se utilizó el siguiente fragmento de código CSS, el cual permite anclar el *footer* al final de la página.

```
html {
  position: relative;
  min-height: 100%;
}

body{
      margin-bottom:80px;
}

.footer{
      margin-bottom:0px;
      position: absolute;
      bottom: 0;
      width: 100%;
      height: 60px;
}
```

Ilustración 70 - Código para visualizar correctamente el *footer*

5.5. Funcionalidad de la aplicación

5.5.1. Introducción

En secciones anteriores del documento se han dado muchas pinceladas introductorias sobre algunas funcionalidades de la aplicación, pero sin detallar demasiado la implementación interna de las mismas. Este apartado se centra en el código de cada función, completando así la información aportada en capítulos previos.

5.5.2. Registro

En el punto 5.4.3. se profundizó en el aspecto visual de la página de registro. Aquí se habla de las dos partes que componen la implementación interna de la función de registro: la validación del formulario y la creación de la cuenta de usuario.

Validación del formulario

Cuando el usuario pulsa el botón para registrarse en el sistema, el cliente realiza las comprobaciones pertinentes con el objetivo de asegurar que todos los campos sean correctos. Específicamente, se verifica que ningún campo esté vacío, que el email tenga un formato correcto, que la contraseña no sea muy corta y que coincida con su confirmación. A continuación se muestra el código que permite la validación.

```
$("#registerFormButton").click(function(){
	var formElementsList =
document.getElementById("registerForm").elements;
	var userPassword = $("#userPassword").val();
	var userPasswordConfirm = $("#userPasswordConfirm").val();
	var userEmail = $("#userEmail").val();
	var emailRegex = /[A-Z0-9._%+-]+@[A-Z0-9.-]+.[A-Z]{2,4}/igm;
	var errCount = 0;

	for(i=0; i<formElementsList.length-2; i++){
		inputValue = formElementsList[i].value;
		inputId = formElementsList[i].id;

		if(inputValue.trim()==''){
			addError(inputId);
			hideErrors(inputId);
			$("#"+inputId+"ErrorMessageBlank").removeClass("hide");
			errCount++;
		}

		else if(inputId == "userEmail" &&
!emailRegex.test(userEmail)){
			addError(inputId);
			hideErrors(inputId);
		$("#"+inputId+"ErrorMessageInvalidEmail").removeClass("hide");
			errCount++;
		}

		else if((inputId == "userPassword" || inputId ==
"userPasswordConfirm") && userPassword != userPasswordConfirm){
			addError(inputId);
			hideErrors(inputId);
	$("#"+inputId+"ErrorMessageDifferentPassword").removeClass("hide");
			errCount++;
		}

		else if((inputId == "userPassword" || inputId ==
"userPasswordConfirm") && userPassword.length<5){
			addError(inputId);
			hideErrors(inputId);
	$("#"+inputId+"ErrorMessageShortPassword").removeClass("hide");
			errCount++;
```

```
            }

            else{
                  hideErrors(inputId);
                  addSuccess(inputId);
            }
      }
      if(errCount==0){
            submitRegForm();
      }
});
```

Ilustración 71 - Código de validación del formulario de registro

Este método recorre uno por uno los elementos del formulario en búsqueda de campos incorrectos, y cuando localiza uno, se muestra el mensaje de error correspondiente y se incrementa el contador de errores. Únicamente si el contador es igual a cero al final de la ejecución se enviarán los datos al servidor.

Un detalle a destacar es la utilización de una expresión regular para validar el formato del email (*/[A-Z0-9._%+-]+@[A-Z0-9.-]+.[A-Z]{2,4}/igm*). Esta expresión fuerza que el email se componga por una cadena de caracteres, una arroba, otra cadena de caracteres, un punto y un conjunto de dos a cuatro letras. Los corchetes delimitan el rango de caracteres posibles.

Creación de la cuenta de usuario

Cuando todos los campos del formulario son válidos, se envían al servidor y éste utiliza el siguiente fichero php para generar la respuesta.

```
<?php
$userName = $_POST["userName"];
$userLastname = $_POST["userLastname"];
$userEmail = $_POST["userEmail"];
$userPassword = $_POST['userPassword'];
$userType = $_POST['userType'];

$m = new Mongo();

$mongoUser = $m->test->users->findOne(array('userEmail' =>
$userEmail));

if (!$mongoUser['userEmail']){
      $m->test->users->insert($_POST, array('safe' => true));
      echo true;
}
else{
      echo false;
}
?>
```

Ilustración 72 - Código del servidor para la operación de registro

Primero, se guardan los valores del formulario utilizando la variable global *$_POST*, predefinida en php. Se trata de un *array* que almacena las variables enviadas al servidor cuando se envía una petición HTTP utilizando el método POST. Tras esto se efectúa una conexión con la base de datos de la aplicación y se busca el email del usuario en la colección *users*. Si el usuario existe, devuelve *false* (desencadenando en la visualización de un mensaje de error sobre la interfaz). De lo contrario, se inserta el nuevo usuario en la base de datos y se le informa con el modal visto en el apartado 5.4.6.

5.5.3. *Login*

Para acceder a la funcionalidad principal de la aplicación, el usuario debe identificarse, proporcionando para ello su email y contraseña en el modal de *login*. De forma similar a la función de registro, se envía una petición al servidor y éste ejecuta el siguiente código php.

```
<?php
$userEmail = $_POST['userEmail'];
$userPassword = $_POST['userPassword'];

$m = new Mongo();

$mongoUser = $m->test->users->findOne(array('userEmail' =>
$userEmail));

if (!$mongoUser['userEmail']){
      echo false;
}
else if($mongoUser['userPassword'] != $userPassword){
      echo false;
}
else{
      session_start();
      $_SESSION['userEmail']=$userEmail;
      $_SESSION['userType']=$mongoUser['userType'];
      $_SESSION['userName']=$mongoUser['userName'];
      echo true;
}
?>
```

Ilustración 73 - Código del servidor para la operación de *login*

Si la información proporcionada (email y contraseña) existe en la base de datos, se crea una nueva sesión y el usuario es redirigido a la página de listados. La sesión se construye almacenando información relevante del usuario en el *array $_SESSION* (predefinido en php).

5.5.4. Control de sesiones

La sesión se define como el período de actividad que un usuario permanece dentro de un sitio web. Esto es realmente útil para efectuar acciones que conciernen solamente a un

usuario concreto, como por ejemplo, deshabilitar la opción de borrado en documentos/anotaciones.

En el apartado anterior se ha visto como se crea una nueva sesión y la información que mantiene. Ahora se van a explicar algunas otras funciones relativas a las sesiones.

Cerrar sesión

Cuando el usuario clica sobre el botón de *log out* y confirma el cierre de sesión, se ejecuta el siguiente código en el lado del cliente.

```
function sessionDestroy(){
	$.ajax({
		url: 'logout.php',
		success:function(){
			window.open("../html/index.html","_self");
		}
	});
}
```

Ilustración 74 - Código del cliente para cerrar la sesión

Como se puede observar, se utiliza una petición AJAX, muy común para interactuar con el servidor, ya que permite la comunicación asíncrona y elimina la necesidad de recargar la página tras cada petición. En este caso, el cliente envía una petición HTTP con el método GET (por defecto) y, tras recibir la respuesta del servidor, el usuario es redirigido a la página de bienvenida. El código ejecutado en el servidor se muestra en la ilustración 75.

```
<?php
session_start();

$_SESSION = array();

if (ini_get("session.use_cookies")) {
    $params = session_get_cookie_params();
    setcookie(session_name(), '', time() - 42000,
        $params["path"], $params["domain"],
        $params["secure"], $params["httponly"]
    );
}

session_destroy();
?>
```

Ilustración 75 - Código del servidor para cerrar la sesión

El principal objetivo de este archivo php es vaciar el *array* de la sesión, eliminar la información de las *cookies* del navegador y finalmente destruir la sesión.

Mecanismo de seguridad

En teoría, solamente los usuarios registrados e identificados pueden acceder a las páginas internas de W&A (listados y documentos), pero un usuario cualquiera podría acceder escribiendo la dirección de estas páginas directamente en la barra de búsqueda del navegador. Para evitar esto, se utiliza el siguiente fragmento de código.

```
<?php
	session_start();
	if(!isset($_SESSION['userEmail'])){
		exit("ACCESS DENIED");
	}
?>
```

Ilustración 76 - Código para denegar el acceso a usuarios no identificados

Básicamente, comprueba si existe una sesión iniciada. Si no es así, muestra un mensaje de error al usuario que intentaba acceder, en vez de la página solicitada.

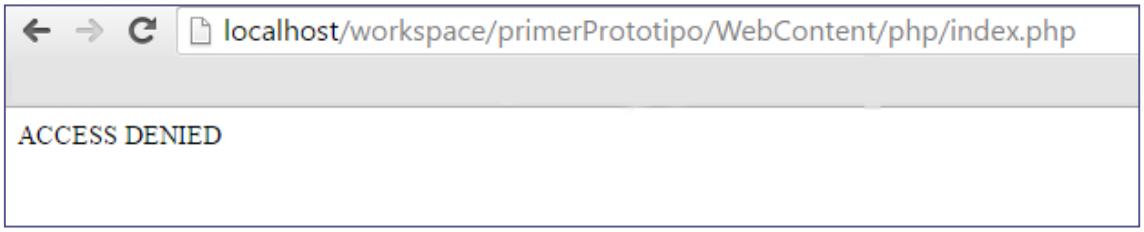

Ilustración 77 - Acceso denegado a las páginas internas

5.5.5. Librería *Annotator.js*

Annotator es el núcleo de la aplicación. Con esta librería, el usuario puede crear anotaciones sobre un fragmento de texto, editarlas y borrarlas. Para disfrutar de su funcionalidad, primero se debe cargar la librería en la cabecera de la página HTML.

```
<link rel="stylesheet" href="../annotator-1.2.10/css/annotator.css">

<script src="../annotator-1.2.10/lib/vendor/showdown.js"></script>
<script src="../annotator-1.2.10/pkg/annotator-full.min.js"></script>
```

Ilustración 78 - Código para cargar la librería *annotator*

A parte de la librería *annotator.js*, es necesario cargar también la hoja de estilos que genera la interfaz del componente y la librería *showdown.js*, utilizada por la extensión *markdown* para convertir el texto plano de las anotaciones en texto HTML.

Después, hay que activar el *annotator* sobre el elemento HTML donde se quieran insertar las anotaciones. Esto se consigue con una simple línea de código, mostrada en la ilustración 79.

```
Var annotator = $('#annotator').annotator().data('annotator');
```

Ilustración 79 - Código para la activación del *annotator*

Finalmente, solo resta añadir las extensiones que resulten convenientes para la funcionalidad definitiva de la aplicación

```
annotator.addPlugin('Markdown');

annotator.addPlugin('Permissions',{
      user: '<?php echo $_SESSION["userEmail"];?>',
      permissions: {
            'read':   [],
            'update': ['<?php echo $_SESSION["userEmail"];?>'],
            'delete': ['<?php echo $_SESSION["userEmail"];?>'],
            'admin':  ['<?php echo $_SESSION["userEmail"];?>']
      },
      showViewPermissionsCheckbox: false,
      showEditPermissionsCheckbox: false
});

annotator.addPlugin('Unsupported', {
      message: "We're sorry the Annotator is not supported by this
browser"
});

annotator.addPlugin('Store', {
      prefix: 'annotationStorageApi.php',
      annotationData: {
            'documentId':'<?php echo $documentId;?>'
      },
      loadFromSearch:{
            'limit':0,
            'all_fields':1,
            'documentId':'<?php echo $documentId;?>'
      }
});
```

Ilustración 80 - Código de las extensiones del *annotator*

- *Markdown*. Se encarga de convertir el texto plano de las anotaciones en texto HTML.

- *Permissions*. Da la posibilidad de otorgar permisos de lectura, edición, borrado y administración sobre las anotaciones. En el código anterior, se ha programado esta extensión para que todo el mundo pueda leer las anotaciones, pero solo el autor de las mismas puede editarlas y borrarlas.

- *Unsupported*. Muestra una notificación cuando la librería *annotator* no está soportada por el navegador.

- *Store*. Permite almacenar las anotaciones en una base de datos y realizar acciones de lectura, edición y borrado sobre ellas. El campo *prefix* indica la dirección de la API donde están escritas las funciones que implementan toda la interacción. El segundo

campo, *annotationData*, constituye un listado de atributos que se añaden a la información de cada anotación. En este caso se inserta el identificador del documento al cual pertenece la anotación. Por último, *loadFromSearch* proporciona un sistema alternativo para listar aquellas anotaciones que cumplen con un criterio de búsqueda determinado.

En este documento no se detalla la implementación interna de *annotator*, ya que se trata de una herramienta existente que simplemente se ha integrado en W&A, pero sí se explica cómo se ha adaptado a la aplicación para conseguir implementar las funciones que permiten al usuario crear, editar y borrar sus propias anotaciones sobre textos en línea.

5.5.6. API para el manejo de documentos y anotaciones

W&A se centra principalmente en realizar anotaciones sobre documentos de texto. Por tanto, es de vital importancia la creación de funciones que permitan operar con estos elementos. Con este objetivo en mente, se han programado dos archivos, uno formado por las operaciones básicas sobre documentos (listar, crear y borrar) y el otro referente a las anotaciones (listar, crear, borrar, modificar y filtrar).

Para la creación de estas dos APIs se ha utilizado Silex, un *micro-framework* escrito en php. Esta herramienta ha permitido programar una serie de controladores que se ejecutan cuando el cliente envía peticiones HTTP a una ruta determinada del servidor. La estructura principal de las APIs implementadas se muestra a continuación, en la ilustración 81.

```php
<?php
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;

require_once __DIR__.'/silex/vendor/autoload.php';

$app = new Silex\Application();

$app->before(function (Request $request) use ($app) {
      if (0 === strpos($request->headers->get('Content-Type'),
'application/json')) {
            $app['data'] = json_decode($request->getContent(), true);
      }
});

$app->get('/', function () use ($app) {
      $out = array(
                  'name'    => "Documents Store API (PHP)",
                  'version' => '1.0.0',
                  'author'  => 'Joan David Castello'
      );
      return $app->json($out);
});

/*Código del resto de controladores.*/
$app->run();
?>
```

Ilustración 81 – Estructura de una aplicación Silex

En primer lugar se importan las librerías necesarias para tratar con las peticiones y respuestas HTTP. En segundo lugar, se incluye el *autoloader*, que carga la configuración requerida para utilizar Silex. Tras esto se crea la aplicación.

La primera función permite a la aplicación trabajar con el formato JSON. Luego, se encuentra el primer controlador del archivo, que se ejecuta cuando el cliente manda una petición GET a la ruta raíz, es decir, a la ruta del documento (*../documentStorageApi.php*). Cuando esto ocurre, el servidor responde con un *array* formado por la información de la API, la cual será mostrada en el navegador.

En las siguientes secciones se habla del resto de controladores que permiten realizar las distintas operaciones sobre los documentos y anotaciones de la aplicación.

5.5.7.Listar documentos y anotaciones

Los listados de documentos y anotaciones se generan automáticamente cuando el usuario accede a la página de listados, sin necesidad de efectuar acciones adicionales.

En este proceso se cargan en el navegador todos los documentos y anotaciones existentes de la base de datos. Esto permite al usuario tener una visión global del estado de la aplicación y posteriormente elegir que acción desea realizar. Los métodos que implementan esta funcionalidad se muestran en la ilustración 82.

```
function loadDocuments(){
   $.get("documentStorageApi.php/documents", function(data){
      var documentId;
      var documentName;

      $.each(data, function(i, val) {
         documentId = data[i]['id'];
         documentName = data[i]['name'];
         documentAuthor = data[i]['user'];

         addDocument(documentId, documentName, documentAuthor);
         disableDeleteButton(documentId);
      });
   });
}

function loadAnnotations(){
   $.get("annotationStorageApi.php/annotations", function(data){
      var annotationId;
      var annotationText;
      var documentId;

      $.each(data, function(i, val) {
          annotationId = data[i]['id'];
          annotationText = data[i]['text'];
          documentId = data[i]['documentId'];
          annotationAuthor = data[i]['user'];

          addAnnotation(annotationId, annotationText, documentId,
```

```
annotationAuthor);
            disableDeleteButton(annotationId);
        });
    });
}
```

Ilustración 82 - Código del cliente para listar documentos y anotaciones

Ambas funciones envían una petición GET a la API correspondiente. Cuando reciben la respuesta del servidor, generan línea a línea los listados de documentos y anotaciones con los datos obtenidos. El código del servidor utilizado para generar las respuestas se muestra a continuación, en la ilustración 83.

```
$app->get('/documents', function () use ($app) {
        $out = array();
        $m = new Mongo();

        $documentsList = $m->test->documents->find();

        foreach ($documentsList as $document){
                $document['id'] = (string) $document['_id'];
                unset($document['_id']);
                $out[] = $document;
        }

        return $app->json($out);
});
```

Ilustración 83 - Código del servidor para listar documentos

La ruta *../documentStorageApi.php/documents* recibe una petición GET activando este controlador. Entonces el servidor conecta con la base de datos para recuperar todos los documentos disponibles y devolverlos al cliente en forma de *array*.

```
$app->get('/annotations', function () use ($app) {
        $out = array();
        $m = new Mongo();

        $annotationsList = $m->test->annotations->find();

        foreach ($annotationsList as $annotation){
                $annotation['id'] = (string) $annotation['_id'];
                unset($annotation['_id']);
                $out[] = $annotation;
        }

        return $app->json($out);
});
```

Ilustración 84 - Código del servidor para listar anotaciones

De manera análoga, el servidor recupera todas las anotaciones de la base de datos para mandárselas al cliente, como se observa en el código de la ilustración 84.

5.5.8. Crear documentos

Un usuario profesor puede crear documentos en la aplicación utilizando el botón de "subida" disponible en la página de listados. Una vez enviado el formulario al servidor, se ejecuta el código de la ilustración 85.

```
function createDocument(thisObj, user){
      fileData = new FormData(thisObj);
      fileData.append('user', user);
      $.ajax({
          url: 'documentStorageApi.php/documents',
          type: 'POST',
          data: fileData,
          processData: false,
          contentType: false,
          success: function(data){
            var documentId = data['id'];
            var documentName = data['name'];
            var documentAuthor = data['user'];

            addDocument(documentId, documentName, documentAuthor);
            $('#'+documentId).parentsUntil("tbody").hide();
            $('#'+documentId).parentsUntil("tbody").fadeIn();
          }
    });
}
```

Ilustración 85 - Código del cliente para crear documentos

El cliente realiza una petición POST mediante AJAX al servidor, enviando en ella la información relativa al documento subido (título, contenido y autor). Al recibir la respuesta, el cliente inserta el documento en la lista correspondiente. Un aspecto importante de esta operación es que el documento se añade dinámicamente sin necesidad de recargar la página. Esto es debido a la comunicación asíncrona con el servidor permitida por AJAX.

En el lado del servidor se ha implementado un controlador que se ejecuta cuando la ruta *../documentStorageApi.php/documents* recibe una petición POST. Se muestra a continuación, en la ilustración 86.

```
$app->post('/documents', function () use ($app) {
      $filename = pathinfo($_FILES['datafile']['name'],
PATHINFO_FILENAME);
      $filecontent =
file_get_contents($_FILES['datafile']['tmp_name']);
      $fileuser = $_REQUEST['user'];
      $document = array("name"=>"$filename",
"content"=>"$filecontent", "user"=>"$fileuser");

      $m = new Mongo();
      $m->test->documents->insert($document, array('safe' => true));
      $document['id'] = (string) $document['_id'];
      unset($document['_id']);
      return $app->json($document);
});
```

Ilustración 86 - Código del servidor para crear documentos

Utilizando la variable global *$_FILES*, predefinida en php, se obtienen los datos del archivo subido al servidor. Con esta información, se inserta el documento en la base de datos y es devuelto al cliente en formato JSON.

5.5.9. Crear anotaciones

Cualquier usuario puede añadir anotaciones en un documento seleccionando un fragmento de texto y escribiendo lo que desee. Cuando se guarda un nuevo comentario, la librería *annotator*, internamente, envía una petición POST a la ruta *../annotationStorageApi.php/annotations* de la API. Entonces, se ejecuta el siguiente código en el servidor.

```
$app->post('/annotations', function () use ($app) {
	$annotation = $app['data'];
	$m = new Mongo();

	$m->test->annotations->insert($annotation, array('safe' =>
true));
	$annotation['id'] = (string) $annotation['_id'];
	unset($annotation['_id']);

	return $app->json($annotation);
});
```

Ilustración 87 - Código del servidor para crear anotaciones

Sigue el mismo procedimiento que en el caso de la creación de documentos. La única diferencia reside en que los datos relativos a la anotación son generados por la librería *annotator*, los cuales se insertan en la base de datos y son devueltos al cliente.

5.5.10. Filtrar anotaciones

Filtrar es una variante de la acción de listado, en la que solamente se muestran aquellas anotaciones que cumplen con un criterio de búsqueda. En W&A se ejecuta esta operación de filtrado de dos formas distintas. En primer lugar, cuando un usuario accede a la página de un documento, solo se listarán las anotaciones asociadas a dicho documento. En segundo lugar, al borrar un archivo, también se eliminarán todas las anotaciones que haya escritas en él.

La siguiente función mostrada en la ilustración 88 envía una petición GET a la ruta *../annotationStorageApi.php/search* para filtrar las anotaciones según el identificador de uno de los documentos.

```
function filterAnnotationsByDocument(documentId){
	$.ajax({
		url: 'annotationStorageApi.php/search',
		type: 'GET',
		data: {'limit':0, 'all_fields':1,'documentId':documentId},
```

```
            contentType: false,
            success: function(data){
                  $.each(data.rows,function(i){
                        var documentId =
data['rows'][i]['documentId'];
                        var annotationText = data['rows'][i]['text'];
                        var annotationId = data['rows'][i]['id'];
                        var annotationAuthor =
data['rows'][i]['user'];

                        addAnnotation(annotationId, annotationText,
documentId, annotationAuthor);
                        disableDeleteButton(annotationId);
                  });
            }
      });
}
```

Ilustración 88 - Código del cliente para filtrar anotaciones

Al recibir la respuesta, el cliente creará la lista con las anotaciones filtradas y las mostrará al usuario en el panel correspondiente. Respecto al lado del servidor, se ejecutará el siguiente código.

```
$app->get('/search', function (Request $request) use ($app) {
      $out = array();
      $searchQuery=array('documentId' => $request->get('documentId'));
      $m = new Mongo();

      $annotationsList = $m->test->annotations->find($searchQuery);

      foreach ($annotationsList as $annotation){
            $annotation['id'] = (string) $annotation['_id'];
            unset($annotation['_id']);
            $out[] = $annotation;
      }

      return $app->json(['total'=>count($out),'rows'=>$out]);
});
```

Ilustración 89 - Código del servidor para filtrar anotaciones

Se puede observar que el servidor recibe el identificador del documento en la petición, el cual utiliza para crear el criterio de búsqueda en la base de datos y, posteriormente, devolver la lista al cliente.

5.5.11. Borrar documentos

En el listado de documentos, cada uno de ellos tiene asociado un botón de borrado que puede ser utilizado por los usuarios para eliminarlo de la aplicación. Cuando se confirma la opción de borrado, se ejecuta el siguiente código en el cliente.

```
function deleteDocument(id){
	$.ajax({
		url: 'documentStorageApi.php/documents/'+id,
		type: 'POST',
		success: function(){
			$('#'+id).parentsUntil("tbody").fadeOut(function(){
				$('#'+id).parentsUntil("tbody").remove();
			});

			$.ajax({
				url: 'annotationStorageApi.php/search',
				type: 'GET',
				data: {'limit':0,
'all_fields':1,'documentId':id},
				contentType: false,
				success: function(data){
					$.each(data.rows,function(i){
						var annotationId =
data['rows'][i]['id'];

						deleteAnnotation(annotationId);
					});
				}
			});
		}
	});
}
```

Ilustración 90 - Código del cliente para borrar documentos

En esta ocasión, se realiza una petición POST vía AJAX a la ruta *../documentStorageApi.php/documents/{id}*. Al recibir la respuesta, se elimina el documento de la correspondiente lista y, además, se envía una nueva petición al servidor para filtrar aquellas anotaciones asociadas al documento eliminado para borrarlas también.

Por su lado, el servidor ejecuta en respuesta el siguiente controlador mostrado en la ilustración 91.

```
$app->match('/documents/{id}', function (Request $request, $id) use
($app) {
	$m = new Mongo();

	$m->test->documents->remove(
		array('_id' => new MongoId($id))
	);

	$m->test->annotations->remove(
			array('documentId' => $id)
	);

	return new Response('', 204);
})
->method('DELETE|POST');
```

Ilustración 91 - Código del servidor para borrar documentos

La función *match* de Silex sirve de comodín para enmascarar los métodos HTTP. En este caso, aunque se recibe una petición POST, se ejecuta la operación de borrado sobre la base de datos. El hecho de no utilizar directamente el método DELETE se debe a que algunos navegadores no lo soportan, y por tanto no se podría efectuar correctamente esta operación.

5.5.12. Borrar anotaciones

Los usuarios pueden realizar esta acción de dos formas distintas. La primera es equivalente al caso de los documentos, interactuando con la opción de borrado de las listas. La segunda posibilidad es utilizando la interfaz de *annotator*, que aparece al pasar el ratón sobre un fragmento de texto comentado. En la ilustración 92 se muestra la función que permite el borrado de anotaciones.

```
function deleteAnnotation(id){
	$.ajax({
		url: 'annotationStorageApi.php/annotations/'+id,
		type: 'POST',
		success: function(){
			$('#'+id).parentsUntil("tbody").fadeOut(function(){
				$('#'+id).parentsUntil("tbody").remove();
			});
		}
	});
}
```

Ilustración 92 - Código del cliente para borrar anotaciones

En la imagen anterior se observa otra petición POST al servidor, en concreto a la ruta *annotationStorageApi.php/annotations/{id}*. Como respuesta, el servidor ejecuta las siguientes líneas de código.

```
$app->match('/annotations/{id}', function (Request $request, $id) use
($app) {
	$m = new Mongo();

	$m->test->annotations->remove(
		array('_id' => new MongoId($id))
	);

	return new Response('', 204);
})
->method('DELETE|POST');
```

Ilustración 93 - Código del servidor para borrar anotaciones

De nuevo se usa el método match y se utiliza el identificador de la anotación para localizarla en la base de datos y eliminarla del sistema.

5.5.13. Modificar anotaciones

Para editar una anotación, el usuario debe utilizar la interfaz que aparece al mantener el cursor sobre el texto marcado, escribir el nuevo comentario y guardarlo. Una vez hecho esto, se envía una petición PUT a la ruta *../annotationStorageApi.php/annotations/{id}* y se ejecuta el siguiente controlador en el servidor, mostrado en la ilustración 94.

```
$app->put('/annotations/{id}', function (Request $request, $id) use
($app) {
	$annotation = $app['data'];
	$m = new Mongo();

	unset($annotation['id']);

	$m->test->annotations->update(
		array('_id' => new MongoId($id)),
		array('$set' => $annotation)
	);

	return new Response('', 303, array('Location' => $request-
>getUri()));
});
```

Ilustración 94 - Código del servidor para modificar anotaciones

5.5.14. Buenas prácticas de programación

Tener buenas costumbres al programar es muy importante en cualquier proyecto software para mejorar la calidad del mismo. Esta sección se centra en describir los distintos hábitos que se han seguido para desarrollar W&A.

Comentar el código

Escribir texto explicativo en los documentos ayuda al correcto entendimiento de las funciones. Este procedimiento resulta especialmente útil para el aprendizaje de aquellos programadores que no han participado en la escritura del código, sobre todo si su intención es utilizar las funciones para mejorar el trabajo o usarlas en proyectos propios.

Hojas de estilo externas

El código CSS puede insertarse de tres maneras distintas en una página: cargando archivos externos, incluyendo el código dentro de la propia página o utilizar el atributo *style* para cada elemento HTML. La primera opción es la que sigue W&A, ya que, además de ser la práctica más generalizada, permite mejorar la reusabilidad del código en aplicaciones web formadas por muchas páginas con el mismo estilo (o similar).

Por otro lado, incluir CSS en los documentos HTML serviría en webs donde no se comparte el mismo estilo entre páginas, o cuando el número de páginas es muy limitado. Por otro lado, el atributo *style* solo debería utilizarse en casos muy particulares donde se

pretende modificar el aspecto de un elemento HTML específico. En otros casos, esta opción dificulta enormemente el trabajo.

En W&A se han utilizado un par de hojas de estilo. La primera corresponde a las páginas accesibles por los usuarios no registrados (bienvenida y registro), y la segunda se usa en aquellas disponibles para los usuarios registrados e identificados (listados y documentos). El motivo de utilizar solo dos hojas de estilo se debe a la enorme cantidad de elementos comunes entre los dos grupos nombrados, es decir, resultaba totalmente innecesario escribir cuatro archivos cuando únicamente cambia el diseño de unos pocos elementos.

Reutilización de código HTML

Utilizar la misma estructura HTML una y otra vez en las distintas páginas que conforman un sitio es algo muy común en el desarrollo web. W&A no es una excepción. Con el objetivo de mejorar la reusabilidad del código, se ha programado un archivo javascript que carga los elementos HTML más utilizados, evitando así la necesidad de reescribirlos en todas las páginas. En la ilustración 95 se muestra un ejemplo.

```
function footer(){
      document.write('<div class="panel-footer navbar footer"
role="footer">');
      document.write('  <footer>');
      document.write('        <p class="footerText">2016
<strong>UPV</strong></p>');
      document.write('  </footer>');
      document.write('</div>');
}
```

Ilustración 95 - Código para cargar el footer de una página

Esta función es invocada desde el documento HTML para generar el pie de página. Otros elementos repetitivos en la aplicación son: la cabecera, el modal de *login*, la barra de navegación…

Considerar navegadores incompatibles con javascript

Existen navegadores que solamente visualizan texto, no teniendo disponibles algunas tecnologías como javascript. En lugar de interpretar los *scripts*, estos navegadores asumen que son texto de la propia página y lo muestran por pantalla al usuario, un comportamiento bastante indeseable.

Para evitar esta situación, en W&A se proponen dos alternativas:

- Utilizar la etiqueta *noscript* de HTML. Con esta opción, cuando el navegador no entienda el lenguaje, mostrará por pantalla el contenido de esta etiqueta.

- Comentar los scripts. Este método provoca que los navegadores incompatibles con javascript ignoren completamente el código.

5.5.15. Problemas encontrados

Trabajar con la librería *annotator* ha sido un trabajo complicado, ya que se trata de una herramienta que utiliza una programación bastante avanzada. Además, al ser necesario el entendimiento de la misma, se ha invertido bastante tiempo para tal propósito.

El inconveniente u obstáculo más destacado al trabajar con la librería surgió cuando se querían filtrar las anotaciones por documento. Al entrar en la página de un texto determinado, aparecían amontonadas absolutamente todas las anotaciones guardadas en la aplicación, en vez de mostrarse únicamente los comentarios asociados a dicho documento en concreto. Este hecho presentaba un problema bastante grave para el desarrollo, ya que afectaba al núcleo principal de la aplicación.

La solución se obtuvo dentro de la propia librería. Utilizando la opción loadFromS*earch*, incluida en la extensión *store*, es posible crear un criterio de filtrado alternativo a la operación de lectura por defecto, seleccionando únicamente aquellas anotaciones que siguen dicho criterio. Aplicando esta función, se consiguieron visualizar correctamente los documentos y sus respectivas anotaciones.

5.6. Almacenamiento de datos

Como se comentó en el apartado 5.2.7., W&A utiliza un sistema de gestión de bases de datos MongoDB. Este método de almacenamiento otorga gran flexibilidad a las colecciones (el equivalente de tabla en SQL), ya que no es necesario definir un esquema previo, permitiendo así tener en la misma colección varios elementos con distintos atributos.

En W&A están presentes tres colecciones en la base de datos, que se encargan de almacenar la información relativa a los usuarios, documentos y anotaciones. A continuación, en las ilustraciones 96, 97 y 98 se muestra como están estructuradas cada una de ellas.

Usuarios

```
[
  {
    "id": "identificador",
    "userName": "nombre del usuario",
    "userLastname": "apellido del usuario",
    "userEmail": "email del usuario",
    "userPassword": "contraseña del usuario",
    "userType": "tipo de usuario (alumno/profesor)"
  }
  {
```

```
    …
  }
]
```

Ilustración 96 - Formato de la colección usuarios

Documentos

```
[
  {
    "id": "identificador",
    "name": "título del documento",
    "content": "contenido del documento",
    "user": "autor del documento"
  }
  {
    …
  }
]
```

Ilustración 97 - Formato de la colección documentos

Anotaciones

```
[
  {
    "id": "identificador",
    "permissions":
       {
        "read":["usuario1","usuario2","…"],
        "update":["usuario1","usuario2","…"],
        "delete":["usuario1","usuario2","…"],
        "admin":["usuario1","usuario2","…"]
       },
    "ranges":
       {
        "start":"no utilizado",
        "startOffset":"carácter donde empieza la anotación",
        "end":"no utilizado",
        "endOffset":"carácter donde termina la anotación"
       },
    "quote": "texto anotado",
    "text": "contenido de la anotación",
    "documentId": "identificador del documento"
    "user": "autor de la anotación"
  }
  {
    …
  }
]
```

Ilustración 98 - Formato de la colección anotaciones

Como se puede observar, las colecciones MongoDB son listados de objetos con atributos clave-valor. Este modo de mostrar la información facilita enormemente la lectura

de los datos por parte de los lenguajes de programación utilizados en el cliente (javascript) y el servidor (php).

6. Evaluación

6.1. Introducción

La fase de evaluación, pese a ser la última en terminar, comienza en paralelo a la etapa inicial de todo proyecto y está presente hasta la finalización del mismo. Esto es debido a la necesidad de conocer en todo momento el rumbo que está tomando el trabajo. No es permisible posponer la etapa de evaluación al final de un proyecto, ya que desde el inicio y a medida que avanza el proyecto, es crucial revisar: la redacción de documentos escritos, los bocetos preliminares de la interfaz, los requisitos que deben satisfacerse en la aplicación (ERS), el correcto funcionamiento de cada una de las funciones implementadas, el cumplimiento de las necesidades del diseño, la satisfacción del usuario, y un largo etcétera. Básicamente, debe evaluarse todo lo realizado en un proyecto.

En este capítulo se explican las pruebas realizadas al final de la implementación, como son: comprobar la existencia de enlaces rotos, validar las hojas de estilo CSS y probar pantallas de diferentes tamaños.

6.2. Pruebas

6.2.1. Enlaces rotos

Un enlace roto representa una dirección web que ya no puede ser consultada. Es necesario eliminar este tipo de enlaces para mejorar la usabilidad de la aplicación. Se ha utilizado la herramienta *dead link checker* (disponible en línea: http://www.deadlinkchecker.com/) para revisar las páginas HTML de la aplicación, en búsqueda de enlaces no operativos. A continuación, en las ilustraciones 99 y 100 se muestran los resultados.

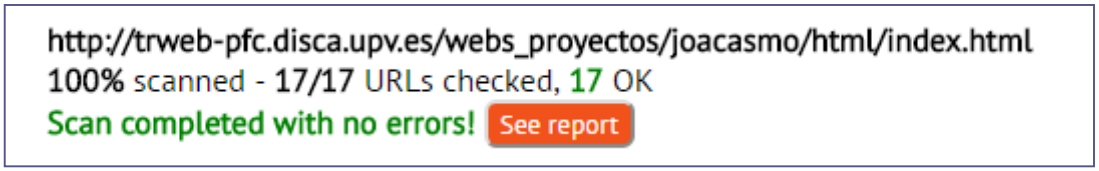

Ilustración 99 - Revisión de enlaces rotos (archivo index.html)

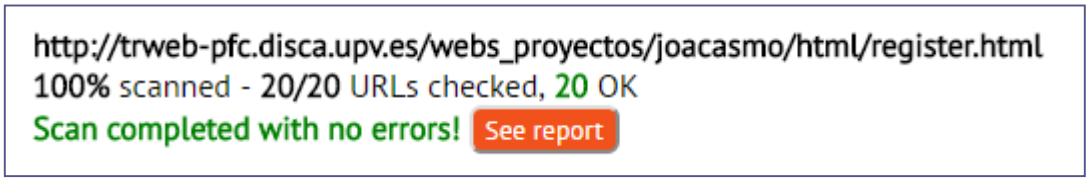

Ilustración 100 - Revisión de enlaces rotos (archivo register.html)

Las pruebas indican que no existen enlaces rotos en W&A, por tanto se puede afirmar que el primer experimento de evaluación ha sido todo un éxito.

6.2.2. Validación CSS

La validación CSS tiene como objetivo ayudar a los desarrolladores web a comprobar si sus hojas de estilo son correctas. Existen herramientas, como *css-validator* (disponible en línea: http://jigsaw.w3.org/css-validator/), que comparan las hojas de estilo con las especificaciones CSS, ayudando a encontrar errores comunes, errores tipográficos, usos incorrectos del código o riesgos de usabilidad.

En W&A se utilizan varias hojas de estilo predefinidas por Bootstrap, jQuery y la librería *annotator*. Estos archivos no son de creación propia y, por tanto, no han sido sometidos a evaluación.

Por otro lado, se han escrito un par de archivos CSS propios que complementan a los anteriores para crear el estilo de la interfaz de la aplicación. Estas hojas de estilo han sido evaluadas utilizando la herramienta css-validator y el resultado se muestra en las ilustraciones 101 y 102.

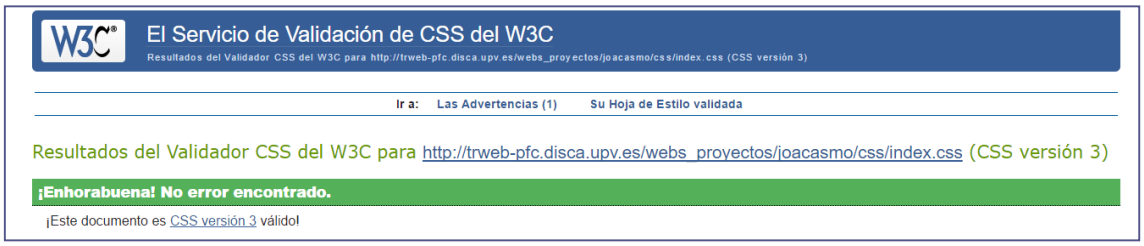

Ilustración 101 - Validación CSS (archivo index.css)

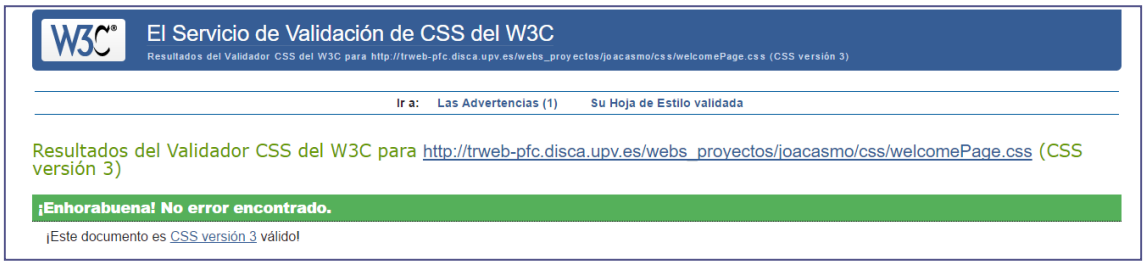

Ilustración 102 - Validación CSS (archivo welcomePage.css)

Como se puede observar, no se ha detectado ningún error en las hojas de estilo. Por tanto, esta prueba también se ha realizado exitosamente.

6.2.3. Tamaño de pantalla

Hoy en día es inmensa la variedad de dispositivos con los que los usuarios pueden acceder a la red. Cada uno de estos dispositivos tiene unas características que lo diferencia del resto: la cantidad de memoria, capacidad de procesamiento, resolución… En este apartado se habla del tamaño de la pantalla y de cómo se adapta la interfaz de W&A a cada uno de ellos.

En el apartado 5.4.7. se habló de la adaptabilidad de la aplicación a los distintos dispositivos. En este punto se pretende mostrar gráficamente el resultado obtenido al probar la interfaz en un portátil, una tableta y un móvil. A continuación se muestran las principales pantallas de W&A visualizadas en todos ellos.

Página de bienvenida

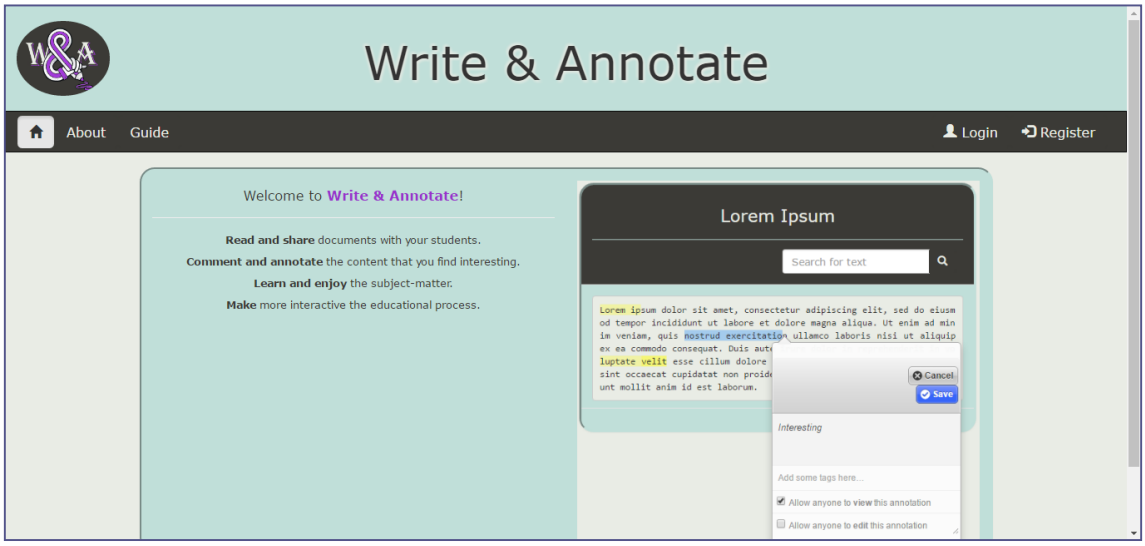

Ilustración 103 - Interfaz desde el portátil (página de bienvenida)

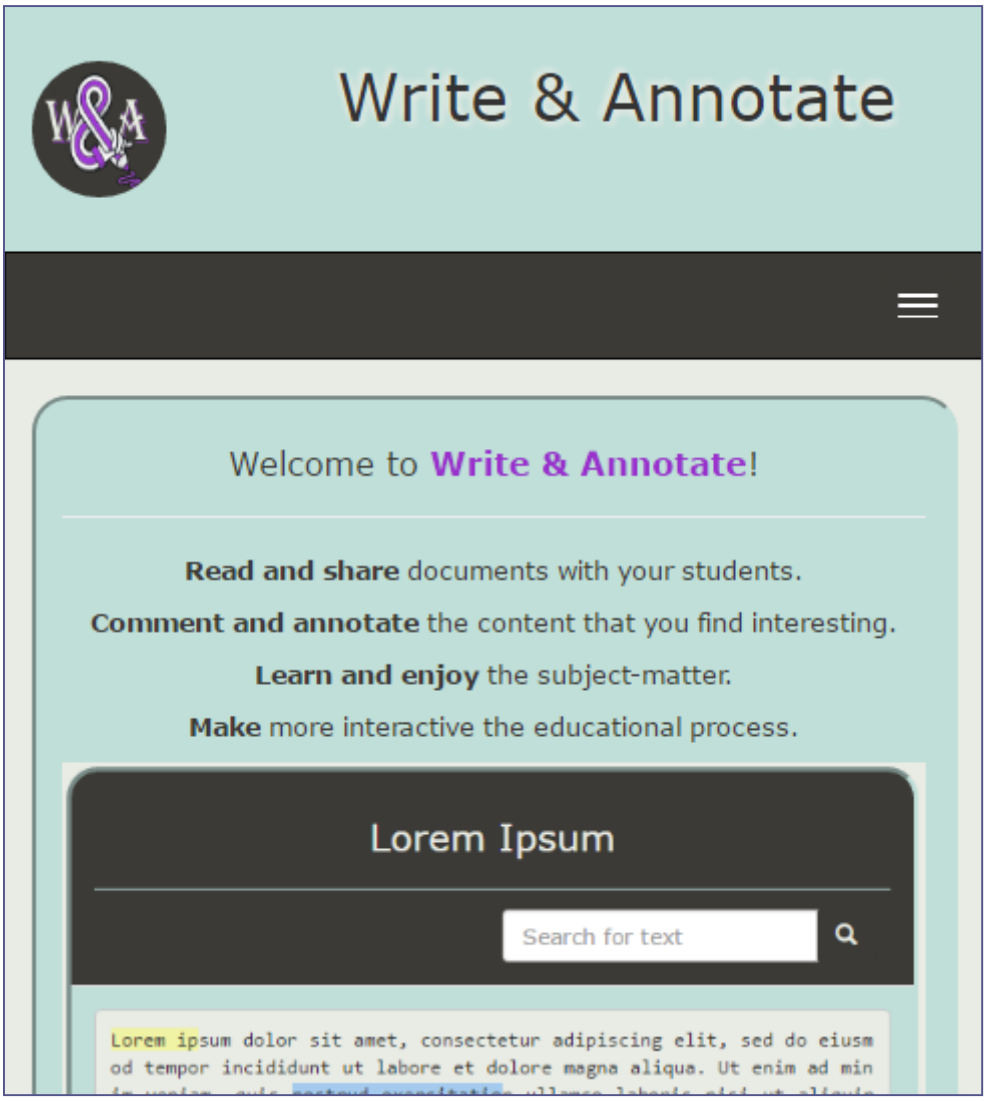

Ilustración 104 - Interfaz desde la tableta (página de bienvenida)

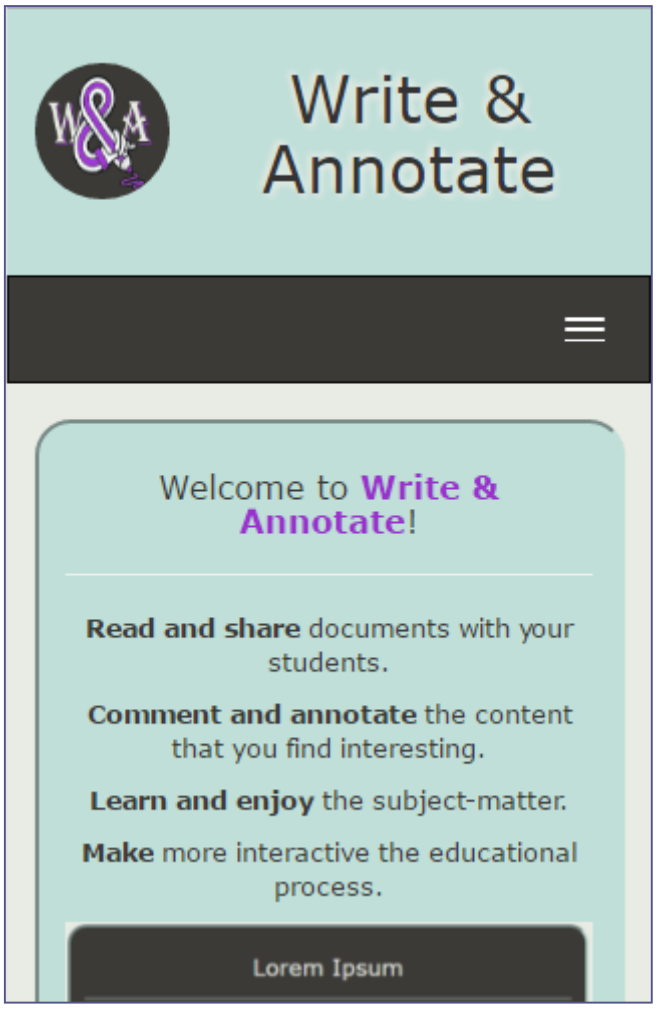

Ilustración 105 - Interfaz desde el móvil (página de bienvenida)

Se puede apreciar que en las pantallas más pequeñas, las opciones del menú de navegación se comprimen en una lista desplegable accionada por un botón de colapso. Este hecho evita que el menú oculte prácticamente media interfaz.

Página de registro

Ilustración 106 - Interfaz desde el portátil (página de registro)

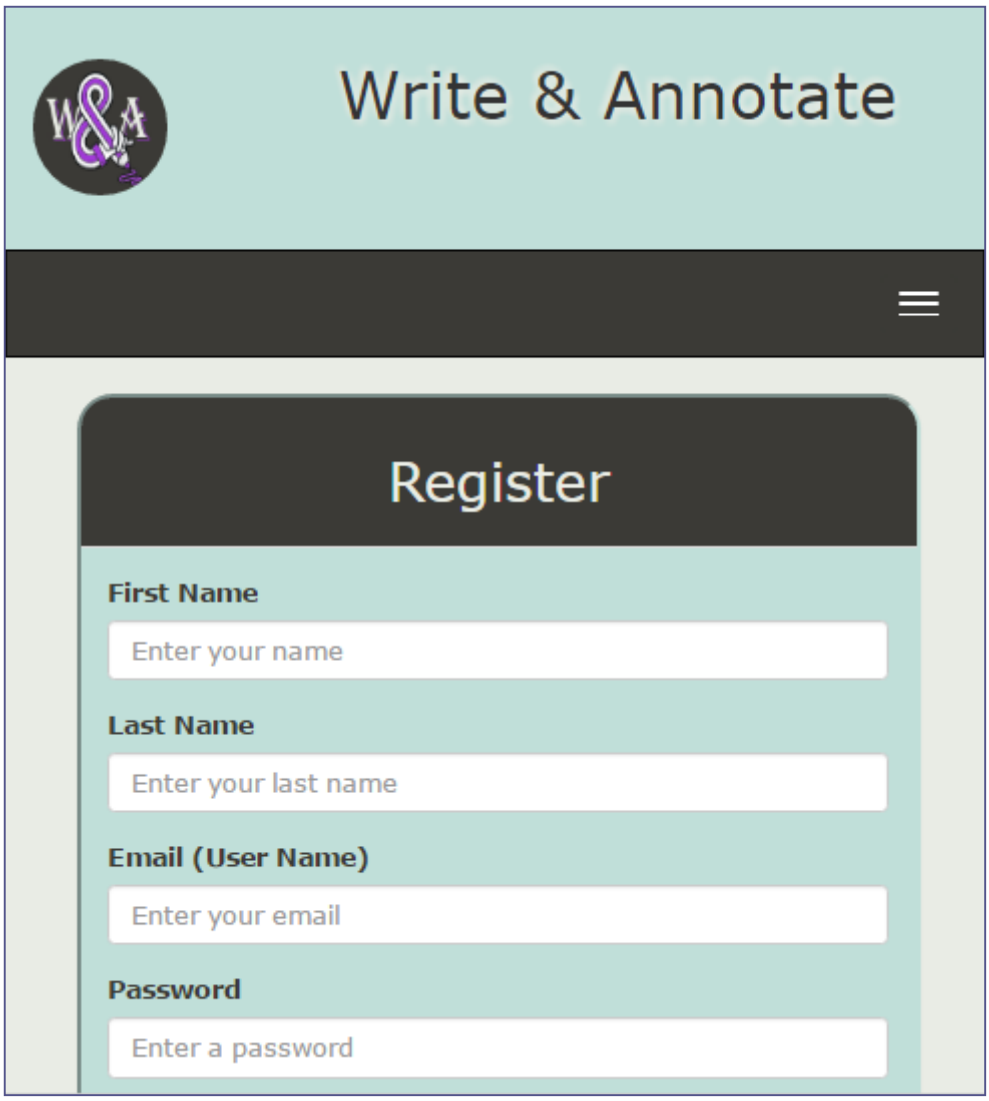

Ilustración 107 - Interfaz desde la tableta (página de registro)

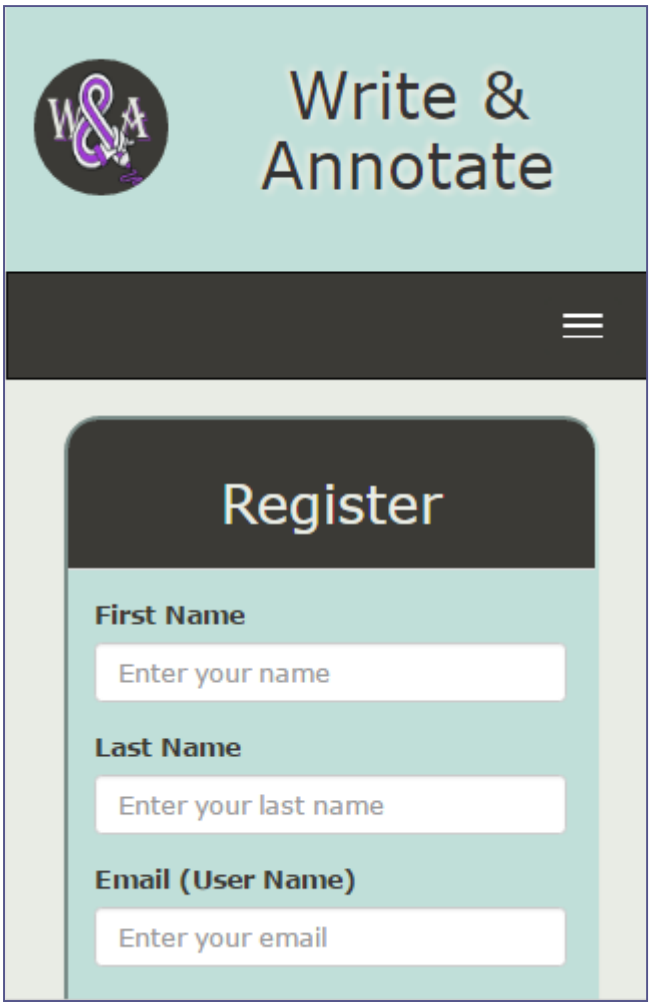

Ilustración 108 - Interfaz desde el móvil (página de registro)

El logo y el título de la aplicación modifican sus dimensiones para ajustarse a la pantalla. Además, el panel de registro y sus elementos se adaptan al tamaño del dispositivo sin perder información y manteniendo la calidad de presentación.

Página de listados

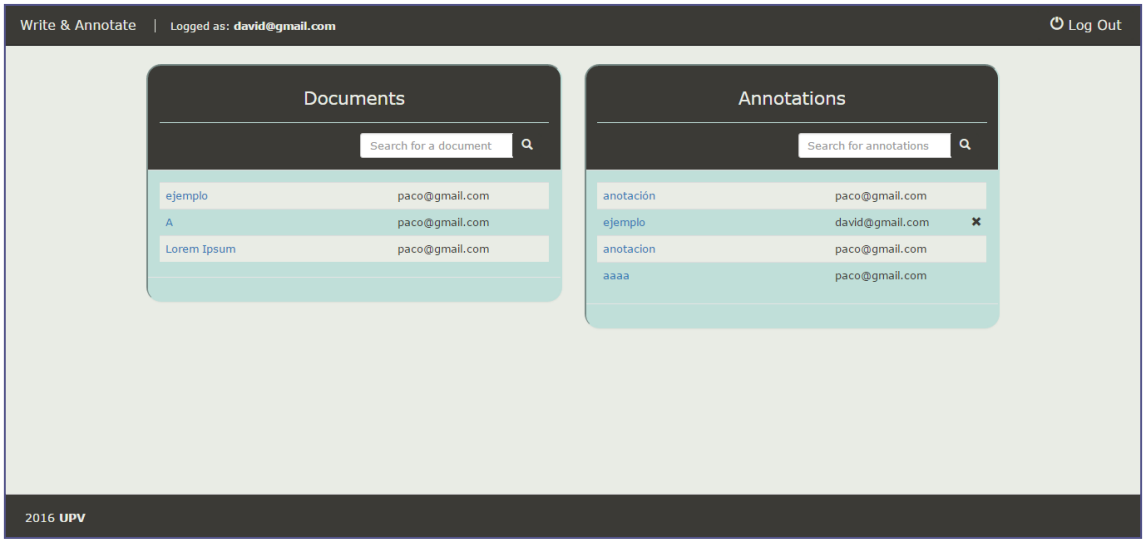

Ilustración 109 - Interfaz desde el portátil (página de listados)

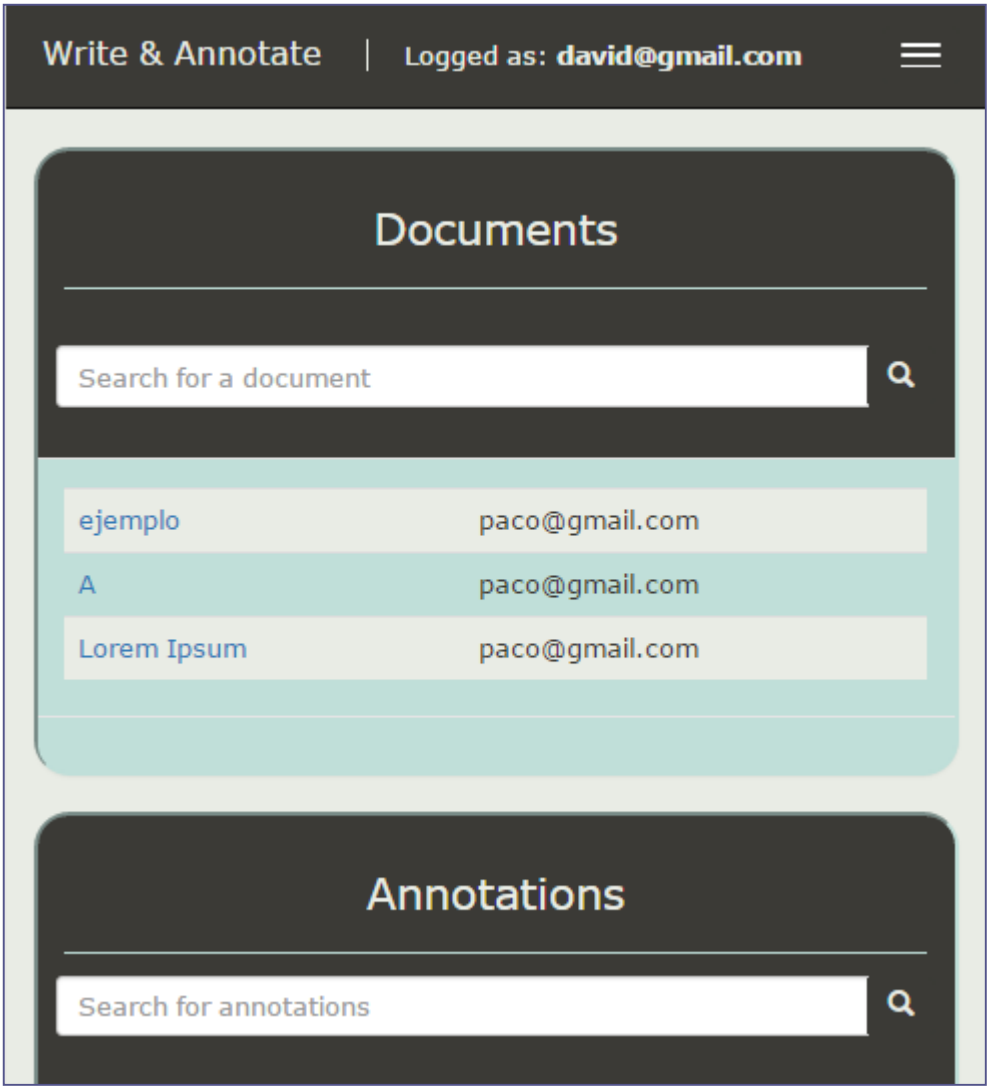

Ilustración 110 - Interfaz desde la tableta (página de listados)

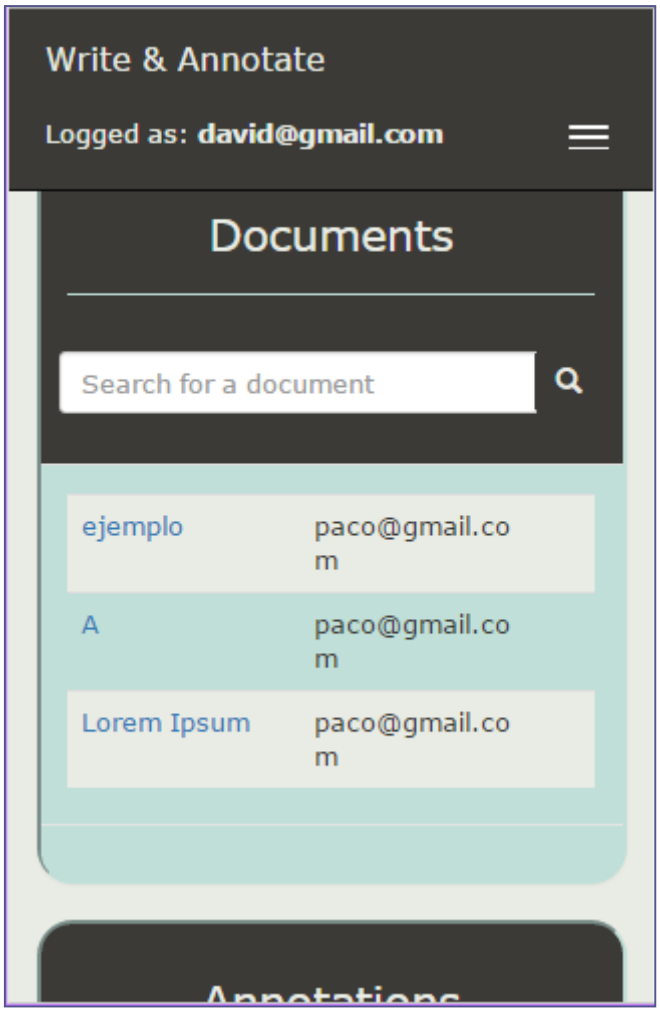

Ilustración 111 - Interfaz desde el móvil (página de listados)

Cada línea de las listas se ajusta al tamaño del panel, sin perder información y sin permitir que el texto sobresalga de los límites de las filas. También se puede observar como en los dispositivos pequeños, los paneles se posicionan uno encima de otro automáticamente, facilitando la navegación con un solo movimiento vertical (suprimiendo el deslizamiento horizontal por la página).

Página de documento

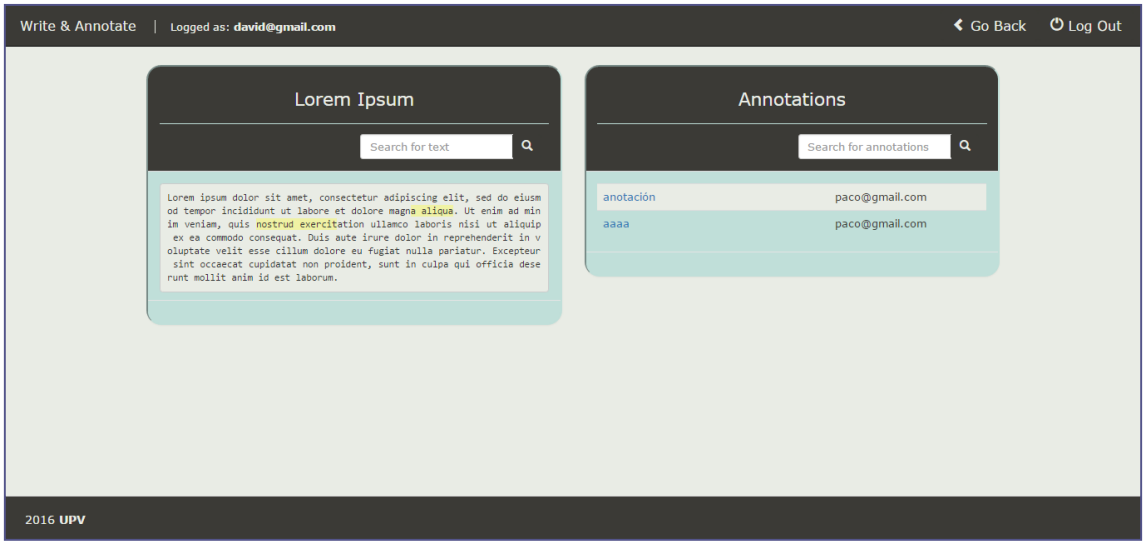

Ilustración 112 - Interfaz desde el portátil (página de documento)

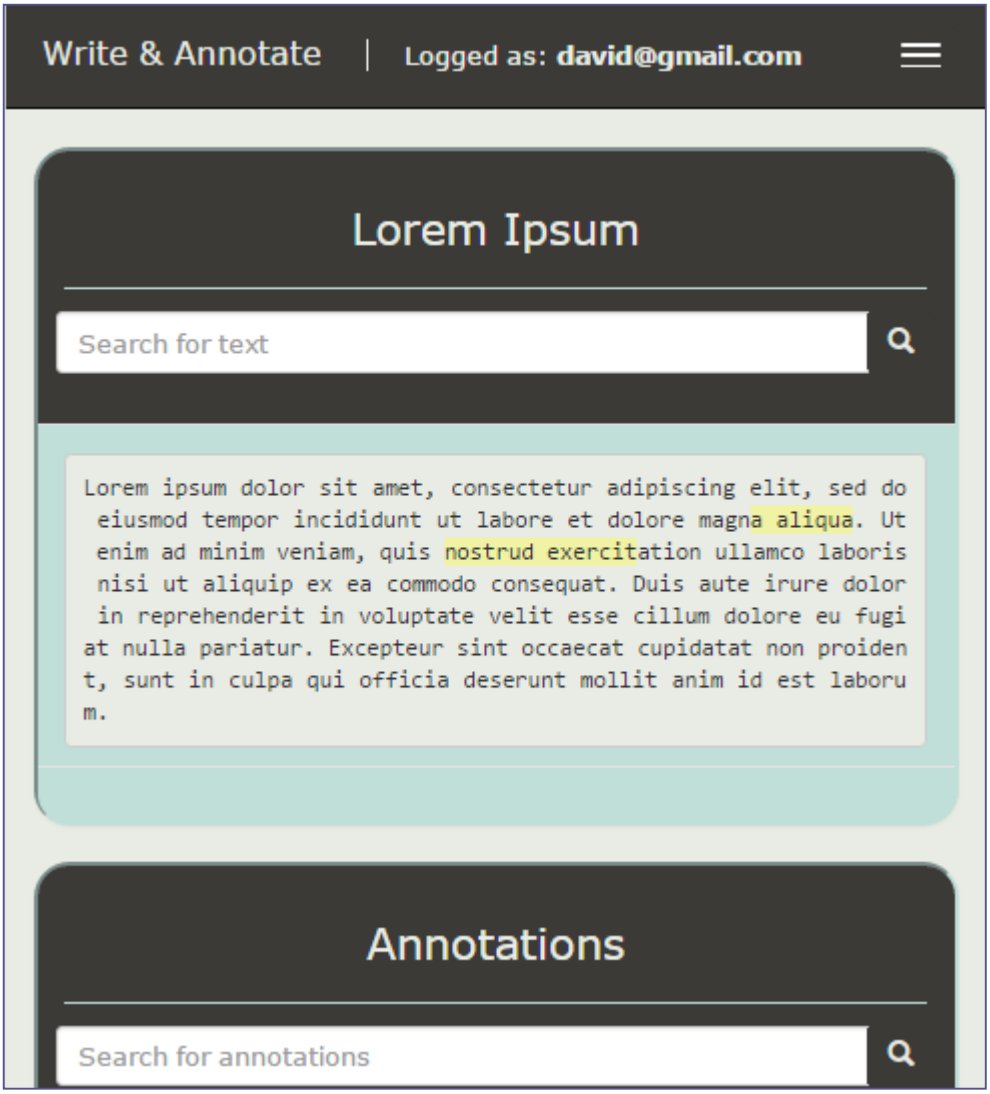

Ilustración 113 - Interfaz desde la tableta (página de documento)

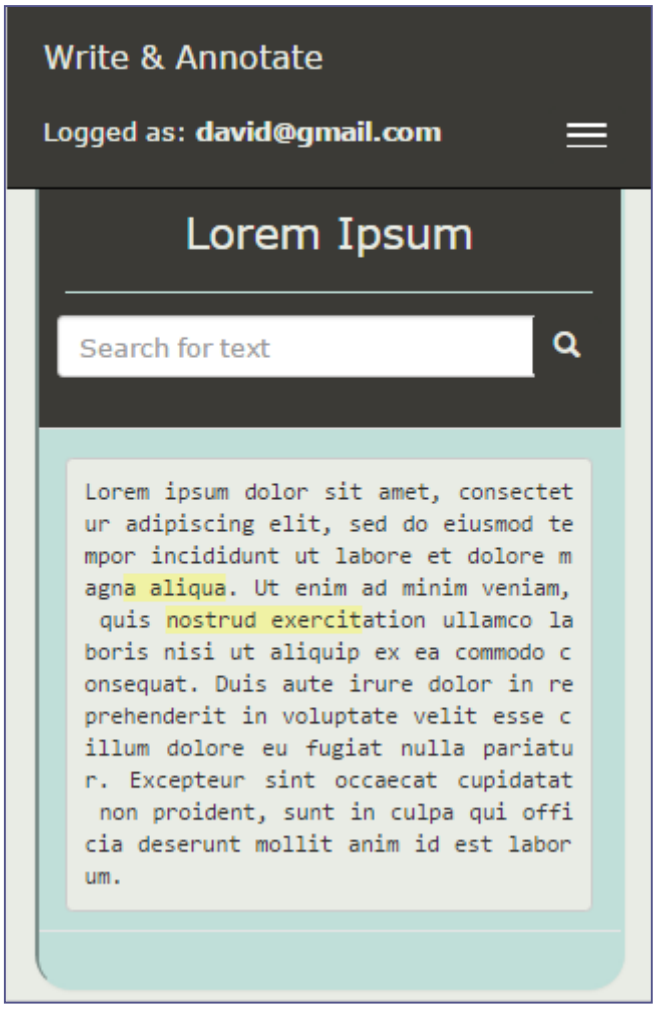

Ilustración 114 - Interfaz desde el móvil (página de documento)

Como se puede observar, la información mostrada se adapta perfectamente al tamaño de la pantalla, permitiendo así que los usuarios puedan acceder a la aplicación desde cualquier dispositivo.

Con esta última prueba termina la fase de evaluación y, por tanto, también concluye el proyecto de W&A. Ahora solo queda añadir las conclusiones y la bibliografía revisada para el desarrollo del trabajo.

7. Conclusiones

7.1. Acerca de la aplicación

W&A surgió con el objetivo de convertirse en una aplicación destinada a usuarios reales, y no limitarse a ser un ejemplo de exhibición de proyecto final de grado en el repositorio de la UPV. Pese a que todavía se podría sacar mucho más partido a este trabajo, el objetivo principal se ha cumplido, y cualquier usuario que tenga instalado alguno de los navegadores actuales puede acceder a la aplicación entrando en la siguiente url: http://trweb-pfc.disca.upv.es/webs_proyectos/joacasmo/html/index.html.

La implementación de un sistema de *login* y registro, una interfaz sencilla que sigue los cánones del *responsive design*, un sistema de gestión de bases de datos fiable donde se pueden almacenar los documentos y las anotaciones creadas por los usuarios, la capacidad de anotar cualquier texto… hacen de W&A una aplicación intuitiva, usable y útil para todos aquellos profesores y estudiantes que quieran compartir información con sus compañeros en el ámbito académico.

7.2. Valoración personal

Cuando empecé con este proyecto, no tenía muy claro cuál era el objetivo de la aplicación a implementar. Tampoco poseía experiencia en algunas de las tecnologías utilizadas, como Annotator.js, MongoDB o PHP. Mi única base era lo aprendido a lo largo de la carrera y mi pasión por el desarrollo web. Pero, tras un período de aprendizaje, comencé a ver el potencial que podría tener la aplicación y conseguí aclarar las ideas relativas a su implementación.

La experiencia ha sido muy buena, desde las fases iniciales del proyecto hasta su conclusión he disfrutado trabajando y observando cómo los objetivos se iban cumpliendo uno por uno. También ha sido interesante aprender a utilizar nuevas tecnologías para posteriormente ensamblarlas y formar el resultado final. Además, con los consejos y motivación proporcionada por mi tutor, las partes más “duras” (como la redacción de la memoria o la fase inicial del proyecto, donde todo era incertidumbre) han acabado siendo mucho más llevaderas.

Si hay que nombrar la parte negativa de esta última temporada de la carrera, destacaría en primer lugar la dificultad para compaginar el TFG con el trabajo, la familia, los amigos y otros asuntos personales, ya que el tiempo disponible era bastante limitado para conseguir todos los propósitos que tenía en mente para la aplicación. En segundo lugar, el estrés y el sentimiento de frustración sufrido cuando las cosas no salían como deseaba, especialmente en el período de implementación, me han consumido muchas horas sin conseguir avanzar. A pesar de ello, la satisfacción al solucionar los problemas compensaba el tiempo dedicado.

Para terminar, he de comentar que, en general, ha sido una grata experiencia haber podido participar en este proyecto. Con la motivación que me ha aportado el trabajo final de grado, espero sinceramente poder dedicarme profesionalmente al diseño y desarrollo web en un futuro no muy lejano.

7.3. Trabajos futuros

W&A tiene un amplio potencial para integrar extensiones y mejoras de su funcionalidad. Este apartado se centra en describir una lista de posibles características que podrían añadirse a la aplicación en un futuro.

- Traducción de idioma. El contenido de la aplicación inicial está escrito en inglés, pero sería interesante añadir una opción para la traducción del texto a otros lenguajes, como el español, francés o alemán.

- Servidor SFTP. Tener un sistema de mensajería interno en W&A podría ser de gran utilidad en varios aspectos. Por ejemplo, se podrían incluir mensajes de confirmación y activación de cuentas al registrar un nuevo usuario. También cabría la posibilidad de contemplar un sistema de notificaciones automatizado que enviase un correo al usuario cuando se creasen nuevas anotaciones o documentos, y un largo etcétera.

- Gestión de grupos. Podría ser interesante incluir la opción de añadir grupos de usuarios para determinadas acciones, como por ejemplo, crear un nuevo documento visible únicamente para los profesores. Si se tuviese creado un grupo "profesores", la acción podría realizarse con un solo clic, en vez de ir seleccionándolos uno por uno.

- Historial de documentos. Esta idea se presentó en la ERS de la memoria, pero nunca llegó a implementarse. Tener un listado de todas las modificaciones sobre un mismo documento, podría ser de utilidad a los usuarios para observar cómo han ido evolucionando los textos y sus anotaciones a lo largo del tiempo.

- Anotaciones privadas. Permitiría al usuario tener sus propios apuntes personales sobre los documentos académicos en línea.

- Manipulación de PDF. W&A trabaja con documentos de extensión *.txt* únicamente. Sería interesante permitir al usuario subir documentos con otros formatos (como PDF) a la aplicación.

- Búsqueda de anotaciones y documentos. Aunque la interfaz para ello está presente en la aplicación, no se pudo implementar la funcionalidad por falta de tiempo. La capacidad de filtrar los documentos y anotaciones por nombre, contenido, autor o fecha de

creación es siempre una característica muy útil para ahorrar tiempo de búsqueda a los usuarios.

- Paginar las listas. A medida que crezca el número de documentos, y sobre todo de anotaciones, podría ser conveniente añadir esta funcionalidad para evitar que la página (las listas) se extienda en exceso hacia abajo y oculte información al usuario.

- Página de perfil. En un principio esta característica estaba pensada para implementarse, pero dada la información no demasiado extensa que se dispone del usuario, parecía irrelevante crear una página de perfil. En un futuro podrían añadirse datos adicionales sobre los usuarios, y por consiguiente darle sentido a esta funcionalidad.

- *Plugins* de *annotator*. La cantidad de extensiones que posee esta librería es inmensa. W&A utiliza solo unas pocas. En general, muchas de ellas tendrían cabida en la aplicación, como por ejemplo: la extensión *Tags*, que ofrece la posibilidad de añadir etiquetas a las anotaciones; o *Filter*, que permite cargar solo aquellas anotaciones que coinciden con un criterio de búsqueda determinado…

- Interfaz móvil. La visualización desde un dispositivo móvil es acorde a las expectativas de un *responsive design*, pero la funcionalidad central de *annotator* no se adapta correctamente a la pantalla táctil de los móviles actuales. Por tanto, añadir una mejora en este aspecto haría de W&A una aplicación mucho más accesible por los usuarios.

8. Bibliografía

[1] EDITORIAL UPV (REF.247). *Una Guía para la Realización y Supervisión de Proyectos Final de Carrera (PFC) en el ámbito de la Web.* ISBN 978-84-8363-325-0.

[2] IEEE. *Guía del IEEE para la Especificación de Requisitos*. Std. 830-1998.

[3] JOHNSON, T. E., ARCHIBALD, T. N., & TENENBAUM, G. *Individual and team annotation effects on students' reading comprehension, critical thinking, and meta-cognitive skills*. Computers in Human Behavior, 26(6), p. 1496–1507, 2010.

[4] NOVAK, E., RAZZOUK, R., & JOHNSON, T. E. *The educational use of social annotation tools in higher education: A literature review*. The Internet and Higher Education, 15(1), p. 39-49, 2012.

[5] ORGANIZACIÓN MONGODB, INC. *Documentación MongoDB* [en línea]. © 2008-2016. Actualizado el 16 de mayo de 2016 [revisado durante todo el proceso de desarrollo: diciembre 2015 - mayo 2016]. Disponible en: https://docs.mongodb.com/v2.6/.

[6] ORGANIZACIÓN MOZILLA DEVELOPER NETWORK. *Documentación javascript* [en línea]. © 2005-2016. Actualizado el 12 de mayo de 2016 [revisado durante todo el proceso de desarrollo: diciembre 2015 - mayo 2016]. Disponible en: https://developer.mozilla.org/es/docs/Web/JavaScript.

[7] ORGANIZACIÓN REFSNES DATA. *Tutoriales W3CSchools* [en línea]. © 1999-2016. [Revisado durante todo el proceso de desarrollo: diciembre 2015 - mayo 2016]. Disponible en: www.w3schools.com.

[8] ORGANIZACIÓN SENSIOLABS. *Documentación Silex* [en línea]. © 2010-2016. Actualizado el 8 de abril de 2016 [revisado durante todo el proceso de desarrollo: diciembre 2015 - mayo 2016]. Disponible en: http://silex.sensiolabs.org/.

[9] ORGANIZACIÓN THE ANNOTATOR PROJECT. *Documentación Annotator.js* [en línea]. © 2014. Actualizado el 31 de marzo de 2016 [revisado durante todo el proceso de desarrollo: diciembre 2015 - mayo 2016]. Disponible en: http://docs.annotatorjs.org/en/v1.2.x/.

[10] ORGANIZACIÓN THE JQUERY FOUNDATION. *Documentación jQuery* [en línea]. © 2016. Actualizado el 26 de mayo de 2016 [revisado durante todo el proceso de desarrollo: diciembre 2015 - mayo 2016]. Disponible en: http://api.jquery.com/.

[11] ORGANIZACIÓN THE PHP GROUP. *Documentación PHP* [en línea]. © 2001-2016. Actualizado el 10 de mayo de [revisado el 20 de mayo de 2016: diciembre 2015 - mayo 2016]. Disponible en: http://php.net/manual/es/.

[12] RAZON, J. TURNER, T.E. JOHNSON, G. ARSAL, G. TENENBAUM. *Effects of a collaborative annotation method on students' learning and learning-related motivation and affect*. Computers in Human Behavior, 28 (2), p. 350–359, 2012.

[13] VARIOS AUTORES. *Foros de consulta StackOverflow* [en línea]. [Revisado durante todo el proceso de desarrollo: diciembre 2015 - mayo 2016]. Disponible en: http://stackoverflow.com/.

[14] VARIOS AUTORES. *Ejemplos de TFG* [en línea]. [Revisado durante toda la redacción de la memoria: mayo – junio 2016]. Disponible en: https://riunet.upv.es/.

[15] VARIOS AUTORES. *Wikipedia* [en línea]. [Revisado durante toda la redacción de la memoria: mayo – junio 2016]. Disponible en: https://es.wikipedia.org.

[16] WOLFE, J. *Annotation technologies: A software and research review*. Computers and Composition, 19,p. 471–497, 2002.