FORO PARA ENTORNOS EDUCATIVOS VIRTUALES EN 3D

En OpenWonderland

Irene Pérez Encinar

Tutora: María Blanca Ibáñez Espiga

Agradecimientos

La primera persona a la que quiero dar las gracias es mi padre, no ya porque sin su ayuda no habría podido llevar a cabo este proyecto, sino porque siempre me empuja a confiar en mí misma y a exigirme más y cree en mis posibilidades incluso cuando yo no lo hago.

Por supuesto a mi madre, mi inspiración, el mejor ejemplo de fuerza y valentía. Alguien a quien debo ser como soy, el entusiasmo que pongo en cada pequeña cosa, la pasión en lo que hago.

A mi hermano, que es mi soporte; una de las personas a las que más admiro, no sólo por su inteligencia y madurez, sino porque no acepta un "no" por respuesta, nada se cruza en su camino cuando se propone hacer algo.

Cómo no, a toda mi familia por haber estado ahí en los momentos duros, porque sin su apoyo yo hoy no estaría aquí.

A mis amigos de Extremadura, por seguir estando ahí aunque pasen los años, porque hoy en día es de agradecer que haya cosas que no cambien. También a los que he hecho en la universidad, especialmente a Marta, por ser mi eterna compañera de fatigas, a Estefanía y a Rubén. Y, por supuesto, a mis amigos de la residencia por haberme acompañado en todas mis alegrías y tristezas a lo largo de este "trayecto" y por haberme hecho crecer como persona.

Y para finalizar, a mi tutora, María Blanca Ibáñez, por brindarme la posibilidad de trabajar en un proyecto tan innovador y fascinante.

Resumen

En este trabajo se presenta el diseño de un foro como objeto de aprendizaje en un entorno 3D y su implementación sobre la plataforma de mundos virtuales OpenWonderland. Se utiliza como metáfora del foro un "oráculo" al que los estudiantes acuden para plantear sus dudas sobre una serie de temas previamente definidos. El foro es, además, un instrumento de evaluación formativa, ya que se fundamenta en la colaboración entre estudiantes, siendo ellos mismos los encargados de responder las preguntas de sus compañeros. Al profesor se le permite definir qué temas se van a tratar en la asignatura (mediante una serie de etiquetas) y conocer la participación y el nivel de los alumnos en cada uno.

Índice

1. Introducción

1.1. Motivación

En los últimos años se ha incrementado la existencia de mundos virtuales, los cuales han comenzado a cambiar desde el propósito inicial de entretenimiento a ser utilizados en otros contextos, principalmente de colaboración en tiempo real. [**1**] Así, han ido apareciendo mundos virtuales con fines profesionales de aprendizaje (simuladores de vuelo), en el entorno médico o incluso en el empresarial, donde se utilizan para comercializar contenidos y productos. También comienzan a ser utilizados en el entorno educativo, el cual es actualmente uno de los más prometedores.

Esto no es de extrañar dadas las características de esta tecnología, que es una combinación de un entorno gráfico 3D y un sistema de interacción social basado en chat, integrando otras herramientas de colaboración online como comunicación por voz, blogs, etc., que permiten salvar las distancias entre los distintos usuarios y hacer la participación inmersiva (también se conoce a estos mundos como Entornos Virtuales Multi-Usuario, MUVEs) [**2**] .

Los espacios virtuales también aportan flexibilidad (pueden adaptarse a las necesidades específicas de cada tarea, permitiendo crear los escenarios adecuados), la posibilidad de innovar minimizando el riesgo (creación y prueba de prototipos) y facilitan el trabajo en equipo (sensación de presencia, de comunidad) [**3**]. Estas características hacen de los mundos virtuales una valiosa herramienta para el desarrollo de experiencias educativas basadas en la exploración, la simulación o el modelado.

Sin embargo, para que den una cobertura global a todas las actividades involucradas en el aprendizaje, faltan metáforas para herramientas que faciliten las actividades sociales propias de los entornos educativos; entre ellas, un foro. Este es un tipo de aplicación que siempre está presente en cualquiera de los entornos educativos en 2D, un ejemplo de las cuales podría ser *Moodle*, muy utilizado en la actualidad. [**4**]

Por esta razón, el planteamiento principal de este trabajo se basa en cómo desplegar un foro en entornos 3D, qué funcionalidad requiere y cómo puede, además, usarse en la actividad formativa como recurso de evaluación.

Entre las herramientas para la creación de mundos virtuales que existen en la actualidad, como pueden ser *Second Life* o *Croquet*, una de las más importantes es *OpenWonderland*, plataforma de código abierto con la que se está trabajando en el Departamento de Ingeniería Telemática de la Universidad Carlos III de Madrid y que, por lo tanto, hace que este aporte sea incluso de mayor utilidad en el propio entorno universitario. [**5**]

1.2. Conceptos básicos de los mundos virtuales

Los mundos virtuales son entornos 3D multi-usuario que típicamente evocan al mundo real, con reglas como gravedad, locomoción, acciones en tiempo real o comunicación. Muchos de los conceptos de estos mundos son herencia de los videojuegos, de donde surgieron inicialmente; entre ellos, está la representación de los usuarios mediante avatares. Los avatares pueden personalizarse (por ejemplo, elegir su género) y son la vía a través de la cual interactuar con el entorno –desplazarse por él – y relacionarse con el resto de usuarios.

Otro tipo de personaje muy importante en los juegos y en los mundos virtuales es el *NPC* (personaje no jugador, *non-player character*) que pueden actuar como simples robots o llegar a ser entes con cierta inteligencia, capaces de interactuar con los usuarios, de forma que se cree la sensación de estar comunicándose con otra persona.

Ambos tipos de personajes no son más que casos particulares de objetos 3D, es decir, todos aquellos elementos visuales con una funcionalidad específica asociada (permiten interacción). Estos objetos constan, por tanto, de dos partes: el modelo 3D (también llamado *artwork*), es decir, la parte gráfica, el cómo se dibujan (o *renderizan*[1]) en el mundo; y las propiedades o capacidades que se le pueden asociar y que definen su comportamiento. Algunas de éstas pueden definirse de forma implícita para un objeto concreto y otras pueden ser genéricas, de forma que puedan asociarse a cualquiera de ellos.

En cuanto al modelo 3D, también puede utilizarse por sí solo (no necesariamente asociado a un objeto) para crear escenarios – pueden ser edificios, mesas, sillas, etc. Son entidades que no interactúan con los usuarios y cuyas propiedades se reducen a posición, tamaño, opacidad…

Gracias a todos estos elementos, los usuarios de los mundos virtuales pueden construir nuevos entornos y experiencias, que pueden ser compartidos con otras personas. Estos entornos multi-usuario se sustentan mediante arquitecturas cliente-servidor, que no sólo permiten la interacción a través de la red, sino que además brindan a estos mundos la persistencia que requieren.

1.3. Objetivos

El objetivo central de este trabajo se resume en diseñar e implementar un foro con fines educativos para entornos virtuales 3D, integrado dentro de la plataforma *OpenWonderland* de creación de dichos entornos. Otros objetivos de este trabajo son:

- Que el foro permita, además de la funcionalidad que habitualmente brinda en entornos 2D, la evaluación formativa de los estudiantes que hace uso de él.

- Motivar a los estudiantes y hacerles más atractiva la aplicación para fomentar su participación. Este objetivo llevado a la realidad 3D se materializa en dos elementos:

[1] Se documentan ampliamente, en textos en Internet, los calcos lingüísticos *renderizar* (con su participio *renderizado*) y *renderización*, que de momento no recoge ningún diccionario general. Consulta realizada a la Real Academia Española de la Lengua.

- La representación metafórica del foro en forma de un *Oráculo* al que los estudiantes, por medio de sus avatares, pueden acudir para resolver sus dudas. Esta metáfora se inspira en el *oráculo de Delfos*, lugar al que acudían los griegos para preguntar a los dioses sobre cuestiones inquietantes. [**6**]

- La implementación de un mecanismo auxiliar que detecte qué cuestiones siguen sin resolver y a quiénes deben ser planteadas, incluso en el caso de que los estudiantes no acudan al *Oráculo* de forma voluntaria. La metáfora visual de este mecanismo se realiza mediante un *fantasma* que persigue al avatar del alumno.

- Realizar un diseño que permita que la aplicación sea fácilmente ampliable y mejorable, cumpliendo con la filosofía de las tecnologías de código libre y abriendo así una ventana a que la comunidad de usuarios de la herramienta pueda incrementarse en un futuro.

2. Arquitectura de la plataforma OpenWonderland

OpenWonderland es una herramienta de código abierto en *Java* que permite la creación de mundos virtuales 3D multi-usuario, pensada principalmente para entornos colaborativos, entre los que los educativos tienen gran importancia. Puede ser utilizada por cualquier usuario para crear escenarios y experiencias de aprendizaje, pero, además, está diseñada para que cualquier desarrollador familiarizado con *Java* pueda extender su funcionalidad mediante la implementación de *módulos* (*plugins)*. [**5**]

En la actualidad todavía se encuentra en fase de desarrollo, por lo que hay muchas aplicaciones que todavía no están definidas para esta plataforma; un ejemplo es el foro objeto de este trabajo.

Surgió, al igual que el concepto de mundo virtual, de la integración de varios proyectos distintos: por una lado, el proyecto *Darkstar*[2], una tecnología *middleware* cliente-servidor que permite la comunicación multi-usuario; por otro, un motor gráfico en Java para videojuegos en 3D (*game engine*), llamado *JMonkey;* y, además, el proyecto *JVoice*, por el interés en el uso comercial de la herramienta para comunicar grupos de trabajo mediante un audio inmersivo.

Para entender la importancia de cada uno de estos proyectos dentro de la arquitectura, es necesario comprender primero que se trata de un entorno multi-usuario donde cada mundo virtual que se cree estará alojado en un servidor y a él podrán conectarse tantos clientes como se quiera. El servidor tiene la función de almacenar datos sobre el estado del mundo: qué objetos hay, en qué posición y que características y funcionalidad tienen; información sobre el escenario, etc. Y también tendrá que actualizarla a medida que los clientes interactúen con los distintos elementos, a la vez que progagar esos cambios al resto de clientes. Este tipo de tareas, entre otras, son función del servidor middleware citado, *Darkstar*.

Asimismo, este servidor se encarga de garantizar la persistencia de los objetos 3D por medio de una jerarquía de documentos *xml* que se almacenan en disco y que guardan toda la información sobre éstos. Siempre que se producen cambios en el estado del mundo (por ejemplo, si se mueve un objeto de lugar), se actualiza la información de estos ficheros. Un detalle importante es que la información del *artwork* utilizado para crear escenarios no se almacena así, para ello es necesario crear (y eso debe hacerlo el usuario) un tipo de fichero especial, llamado *snapshot*.

Para poder llevar a cabo tal acción, existe un servidor web de configuración de *OpenWonderland*, *Glassfish v3* [**7**] de tal forma que, desde cualquier navegador, un cliente autorizado puede conectarse y llevar a cabo ésta u otras tareas de administración.

Por otra parte, es importante mencionar que en el servidor todos estos objetos 3D, modelos para la creación de escenarios o avatares de los clientes conectados en cada momento, no son más que una representación en memoria, es decir, un conjunto de datos que son compartidos por todos los

[2] De forma externa a la plataforma *OpenWonderland*, el proyecto *Darkstar* ha evolucionado y en la actualidad se denomina *RedDwarf* [**23**]. No obstante, en el contexto de *OpenWonderland* se sigue utilizando todavía la denominación *Darkstar* y se respetará así en la redacción de esta memoria.

clientes. Realmente, la *renderización* es tarea del motor gráfico mencionado*, JmonkeyEngine,* [**8**] y sólo acontece en el cliente, quien se descarga el objeto o modelo del servidor y se crea su copia local para este propósito. De ahí que sea muy importante sincronizar, siempre a través del servidor (ya que los clientes no se conectan entre sí de forma directa), los cambios que se produzcan en uno de ellos, actualizando las instancias de los demás.

La aplicación que se utiliza en el cliente para acceder al mundo virtual creado con *OpenWonderland* es la *Java WebStart*, que se lanza desde la página web de acceso al servidor. Una vez que se identifica, al cliente se le muestra una ventana en la que aparece su avatar delante de una porción del mundo virtual en la cual puede posicionar tanto modelos 3D como objetos que ya estén instalados en el servidor. También puede personalizar su avatar o añadir características (capacidades, es decir, cierta funcionalidad) a objetos 3D que ya haya. Los modelos 3D que se utilizan para crear los escenarios pueden tomarse del repositorio de contenido de Google (ya que la herramienta es compatible con el formato *.kmz* de *GoogleMaps*) o ser creados con herramientas como *Sketch Up* (después tendría que exportarse el modelo a *kmz*, opción que viene incorporada en la propia aplicación). [**9**]

Un ejemplo de lo que vería un cliente que ha posicionado en el mundo virtual un objeto sería:

Figura 1. Visión en el cliente del mundo virtual creado con *OpenWonderland* [10**]**

2.1. Desarrollo modular

Para que sea un sistema escalable y sencillo de expandir, *OpenWonderland* está definido como una parte central (*core*) y un conjunto de módulos, que se encargan de ir añadiendo funcionalidad, ya sea comunicarse con aplicaciones externas, representar un objeto en el mundo, permitir que haya una nueva capacidad que pueda concedérsele a distintos objetos o incluso regular la autentificación o la seguridad.

Así, cada vez que se instala un nuevo módulo, éste se descompone en las diferentes partes que lo forman y se distribuye entre servidor y cliente según corresponda. Esto se implementa mediante paquetes Java: el del cliente, el del servidor y el común. Evidentemente, no es necesario que en todos los módulos estén presentes los tres; algunas aplicaciones pueden tener que actuar simplemente en el servidor, sin necesidad de que exista una contraparte en el cliente (un ejemplo puede ser un servicio de acceso a una base de datos).

No obstante, lo más común es encontrarnos con objetos 3D o capacidades, para los que tendremos que definir clases en los tres paquetes que se encarguen de:

Paquete cliente	Paquete común	Paquete servidor
• Creación del objeto 3D o capacidad • *Renderización*	• *Estado* del objeto 3D en el cliente y en el servidor • Tipos de datos comunes, como la definición de los mensajes que se vayan a intercambiar, etc	• Representación del objeto 3D o capacidad en el servidor • Comunicación con la base de datos (si la hubiera)

Figura 2. Tipos de clases java presentes en un módulo

Estas clases podrán estar organizadas, además, en sub-paquetes dentro de los tres principales, lo cual simplemente implica la definición de una jerarquía de directorios.

En definitiva, un módulo es una forma de encapsular todo el conjunto de clases que definen una unidad funcional, así como los recursos gráficos utilizados para la *renderización* (*artwork*), para que pueda ser distribuida e incorporada a cualquier servidor *OpenWonderland* de manera sencilla.

Es importante hacer una aclaración para que no haya dudas con la terminología: en Java, las instancias de una clase se denominan objetos, pero no debe confundirse esto con los objetos 3D, que son entidades presentes en el mundo con las que se puede interactuar.

2.2. Celdas y componentes

Una celda (*cell)* en *OpenWonderland* es una entidad que representa a un objeto del mundo virtual y sus propiedades como, por ejemplo, si se puede mover o no, qué posición y tamaño tiene en el mundo, su forma, color, textura (modelado 3D)...

Los componentes (*capabilities, capacidades*) son propiedades que pueden añadirse como atributos de las celdas, pero se desarrollan de forma independiente porque pueden ser cualidades genéricas que se añadan a diferentes tipos de objetos.

Un ejemplo para entender mejor estos conceptos podría ser imaginar que se tiene una mesa en 3D en el mundo virtual. Al hacer click sobre ella, la mesa cambia y, en lugar de ser de color blanco, pasa a ser de color negro. Además, la mesa no se puede atravesar, si un avatar se acerca a ella y sigue avanzando, se chocará.

En este ejemplo, se tendría una celda que representa a la mesa y que tendría un atributo –o capacidad- *color*. Además, para poder *renderizarla* de dos formas distintas, tendría asociados dos modelos 3D: uno sería el de la mesa blanca y otro el de la mesa negra. De esta forma, la celda

engloba todo lo que rodea a la mesa: un escuchador de evento de ratón, los modelos 3D, la lógica que regula el cambio al producirse el click y la lógica que regula que no se pueda colisionar con ella. Pero que no se pueda atravesar un objeto opaco no es sólo aplicable a una mesa: también a una silla, un edificio, etc, por lo que tiene sentido desarrollar esa funcionalidad como una *capacidad* que pueda ser añadida a cualquier celda.

Una celda necesita, por una parte, tener una representación en el servidor, que sea común a todos los clientes y que mantenga coherencia entre ellos. Es decir, si un cliente está visualizando una celda con unas determinadas características de tamaño o posición, esto debe ser así para todos los demás también. En el ejemplo anterior, si un cliente ve la mesa de color negro, todos los clientes deben verla así. Pero cada cliente tiene su propia instancia en la que realiza los cambios de forma local, por lo que esas interactuaciones tendrán que propagarse al servidor y, de ahí, al resto de clientes, para mantener la coherencia (si alguien hace click y la mesa ahora es blanca, debe ser blanca para todos).

Gráficamente se podría representar como:

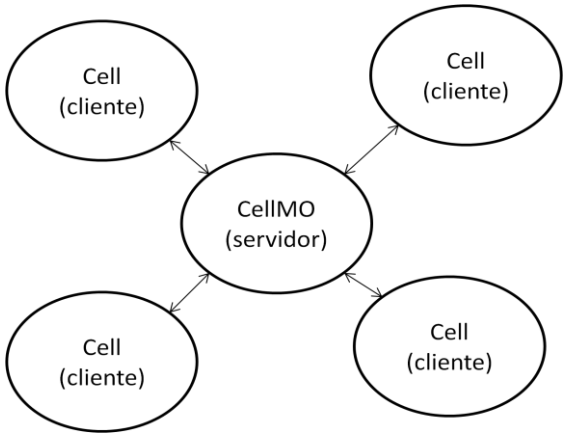


Figura 3. Celdas en *OpenWonderland*

En el diagrama se observa que las instancias de las celdas en los clientes se identifican con *Cell*, por implementar esta clase, mientras que en el servidor aparece *CellMO*. La principal diferencia radica en que el objeto del servidor debe ser persistente ("MO" significa *Managed Object*), es decir, tiene que encargarse de guardar la información relativa a la celda en el correspondiente fichero xml. Volviendo al ejemplo anterior, es necesario almacenar si la mesa en un momento concreto es blanca o negra.

Por otra parte, al entrar un nuevo cliente en el mundo, la clase *Cell* tiene la función de comunicarse con su correspondiente *CellMO* y descargarse la información de estado del objeto para *renderizarlo* correctamente. Las flechas de la Figura 3 representan el canal de comunicación asociado a cada cliente, del que también se genera una copia para cada uno (es decir, cada *Cell* tiene su propio canal de comunicación con la *CellMO*). En el servidor, para identificar a cada cliente, existe un campo que es necesario enviar en todos los mensajes que se intercambien, llamado *cellID.* Éste se genera automáticamente cada vez que un nuevo cliente inicia sesión y es único durante toda su estancia en el servidor.

En cuanto a los componentes, al igual que las celdas, se implementan como clases Java y su estructura y comunicación cliente servidor es idéntica a lo explicado para éstas. Los componentes normalmente se desarrollan de dos formas: bien en el mismo módulo en el que se desarrolla la celda, para añadir cierta funcionalidad a ésa en concreto; bien en un módulo aparte si es una característica genérica que se podrá añadir posteriormente a varios tipos de celda.

Terminando con el ejemplo, la funcionalidad asociada al cambio de color de la mesa es una *capacidad* que se implementaría en el mismo módulo que la celda mesa en sí, mientras que la característica de colisión se haría en un módulo separado para que sea independiente y se pueda incluir en un servidor sin necesidad de que exista la mesa.

2.3. Arquitectura de comunicaciones

Además del canal y el tipo de mensaje asociado a una celda, surgen algunos tipos de datos que es necesario comunicar aunque no haya ninguna, por lo que *OpenWonderland* proporciona una arquitectura de comunicaciones fácil de extender y que le da mucha libertad a los desarrolladores para personalizarla mediante la extensión de clases Java o implementación de interfaces.

Esta arquitectura se compone de 3 niveles: el nivel más bajo, o de *protocolo*, un nivel intermedio, o de *sesión* y el nivel más alto, o de *conexión*. [**11**]

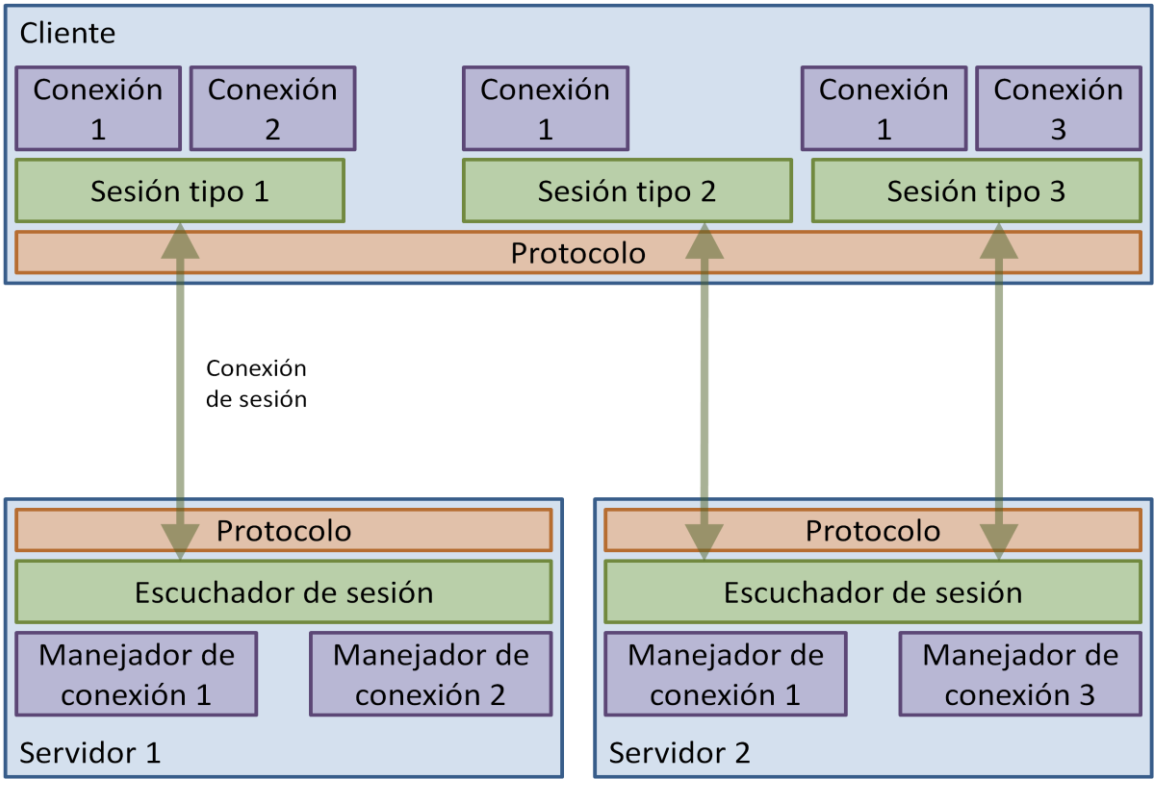


Figura 4. Arquitectura de Comunicaciones. Del tutorial [11]

2.3.1. Nivel de protocolo

El nivel de protocolo se define para que los clientes puedan elegir qué tipo de comunicaciones quieren usar. Lo más habitual es utilizar el protocolo por defecto, cuya única limitación es la

exigencia de que los mensajes transmitidos deben ser objetos *Java serializables* (ver *Glosario*). Así, sólo sería necesario usar un protocolo diferente en aplicaciones que no cumplan esta condición.

2.3.2. Nivel de sesión

Es el nivel intermedio de la comunicación y está pensado para facilitar que un cliente pueda conectarse a la vez a más de un servidor o que pueda establecer varias sesiones con el mismo. Dentro de cada sesión luego se pueden crear diferentes tipos de conexiones según la naturaleza de los datos a enviar. Por ejemplo, existen unos elementos en los mundos virtuales llamados *portales* que pueden teletransportar a los usuarios de un mundo a otro (o entre zonas del mismo mundo). Estos elementos, por tanto, pueden requerir el establecimiento de sesiones en más de un servidor, pero los datos que se comunican a cada uno son los mismos, por lo que se pueden utilizar instancias del mismo tipo de conexión en cada sesión.

De las sesiones es importante saber que tienen un ciclo de vida que es necesario *escuchar* para poder crear conexiones sobre ellas. Las sesiones se inicializan como "desconectadas" en el servidor y pasan al estado de "conectadas" cuando un cliente hace *login* de forma exitosa. A partir de ese momento, pueden añadirse sobre ellas las conexiones que se requieran. Finalmente, las sesiones caducan al desconectarse los clientes del servidor.

Al igual que ocurre con el protocolo, a menos que se busque un comportamiento específico, lo normal es aprovechar las sesiones ya existentes; en ese caso, se define un escuchador (*SessionLifecycleListener*) que avisa cuando la sesión se conecta y ya está disponible para añadirle conexiones.

2.3.3. Nivel de conexión

Finalmente, el nivel de conexión es el más importante de cara a este trabajo, ya que es el que va a permitir definir un tipo de comunicación personalizada, tanto el tipo de mensaje que se envía como el comportamiento que deben tener tanto cliente como servidor para gestionarlo.

En una conexión, los mensajes entre cliente y servidor pueden ser enviados de forma unidireccional (por ambas partes), como pregunta-respuesta (bloqueante) o también el servidor tiene la capacidad de enviar un mensaje a todos los clientes (*broadcast*). Lo que no está contemplado es que un cliente se pueda comunicar con los demás de forma directa (tiene que hacerlo a través del servidor).

Como se puede observar en la Figura 4, se pueden definir tantos tipos de conexión como se quiera, aunque la idea es que haya uno para cada tipo de datos; por ejemplo, una conexión podría ser para enviar datos asociados con la celda, otra para los de voz, etc. La única restricción es que sólo puede haber una instancia de cada tipo de conexión por sesión. Pero un tipo de conexión puede utilizarse en todas las diferentes sesiones que se quiera.

Así, definir un nuevo tipo de conexión requiere básicamente 4 pasos:

- Crear el tipo de mensaje que se vaya usar para transmitir la información (extendiendo la clase *Message*)

- Definir el tipo de conexión mediante la extensión de la clase *ConnectionType*
- Crear una clase heredera de *ClientConnection* en el cliente donde se definan las acciones a realizar cuando se reciban mensajes y también un método para enviarlos.
- Crear la contraparte en el servidor, una clase que extienda *ClientConnectionHandler*

De acuerdo con la estructura de clases en un módulo explicada en la tabla de la Figura 2, las dos primeras deben pertenecer al paquete común, ya que deben instalarse tanto en el cliente como en el servidor; por su parte, las dos últimas pertenecerán a los paquetes de cliente y servidor, respectivamente.

3. Funcionalidad y diseño del "Oráculo"

Existen dos tipos de actores en la interacción con el foro: alumnos y profesores de la asignatura, y la funcionalidad que la aplicación debe brindar a cada uno es diferente. Mientras que los primeros sólo participan en el uso de la herramienta, los segundos, además, tienen que poder configurarla.

La funcionalidad principal del *Oráculo* es, como en cualquier foro, permitir a los alumnos consultar preguntas, formularlas y responderlas. Además, como se trata de una herramienta de evaluación, debe incorporar mecanismos y criterios para tal efecto.

Partiendo de estas bases, se definen ciertos criterios organizativos y funcionales del foro, que van a afectar a las tareas que pueden realizar ambos tipos de actores:

- Para poder trabajar con preguntas y respuestas es necesario llevar un registro de ellas, es decir, definir algún tipo de mecanismo de catalogación y búsqueda. Se ha optado por utilizar un tesauro, esto es, un conjunto de palabras clave o etiquetas (referidas como *tags* en el código realizado) con las que relacionar las preguntas. Estas etiquetas podrán referirse a temas de una asignatura, a conceptos clave dentro de ella, etc. (por ejemplo, en el ámbito de las telecomunicaciones las palabras clave podrían ser nombres de protocolos de comunicación). Las respuestas de una pregunta heredan sus etiquetas.

- Para evitar la presencia de textos inadecuados en el foro, se define un mecanismo de validación de preguntas y respuestas. También puede utilizarse para evitar contenidos repetidos o que se salgan de los objetivos de la asignatura.

- Aparte de esto, preguntas y respuestas llevan asociada una calificación, que es completamente independiente de si son válidas o no. Esta calificación le sirve a los profesores como referencia para evaluar a los alumnos, pero también a estos últimos como medida de calidad a la hora de estudiar.

- Basados en las calificaciones, se definen niveles de conocimiento para los alumnos, con el fin de motivar la participación y de tener una referencia sencilla y visible para todos de quién sabe de qué temas. En este sentido, se definen alumnos "expertos", con nivel "medio" y con nivel "bajo" en un área.

3.1. Funcionalidad para el alumno

El alumno puede *ir al Oráculo* para consultar las preguntas y sus correspondientes respuestas relacionadas con uno o varios temas (*tags*) de la lista disponible. Ésta es una lista cerrada, definida por el profesor, por lo que el alumno no tiene la opción de inventar nuevas etiquetas. La razón de esta decisión es que esta aplicación no es un foro de propósito general, sino que es una herramienta de aprendizaje y evaluación de unos conocimientos específicos en una materia. No obstante, dentro de la lista disponible, no hay restricción en la cantidad de etiquetas con las que relacionar una pregunta (la única es exigir que al menos sea una).

Como resultado de esta búsqueda sólo se incluyen aquellas que ya hayan sido validadas por el profesor, estén o no calificadas.

Aparte de consultar las respuestas existentes para cada pregunta, el alumno también puede responder si considera que falta información importante. Así, una pregunta puede tener un número cualquiera de respuestas, en función de las distintas aportaciones que hagan los alumnos.

Por otro lado, el alumno también puede plantear una nueva pregunta, que deberá ir relacionada con al menos uno de los *tags* de la lista.

Además, dado el carácter evaluador del foro, otra funcionalidad importante para el alumno es poder conocer su progreso en la asignatura, esto es, sus calificaciones en las distintas preguntas y respuestas que ha hecho y sus *estadísticas* por tema (*tag*). En este caso, *estadísticas* se refiere a número y nota media de preguntas y respuestas y nivel (experto, medio, bajo).

En esta área personal del alumno, éste tendrá acceso a todas sus preguntas y respuestas, independientemente de si son válidas o no o de si están calificadas. Además, podrá editar o borrar aquellas que no tengan nota aún y, en el caso de las preguntas, si todavía no han sido respondidas.

En el siguiente diagrama de casos de uso se sintetiza la funcionalidad anteriormente descrita:

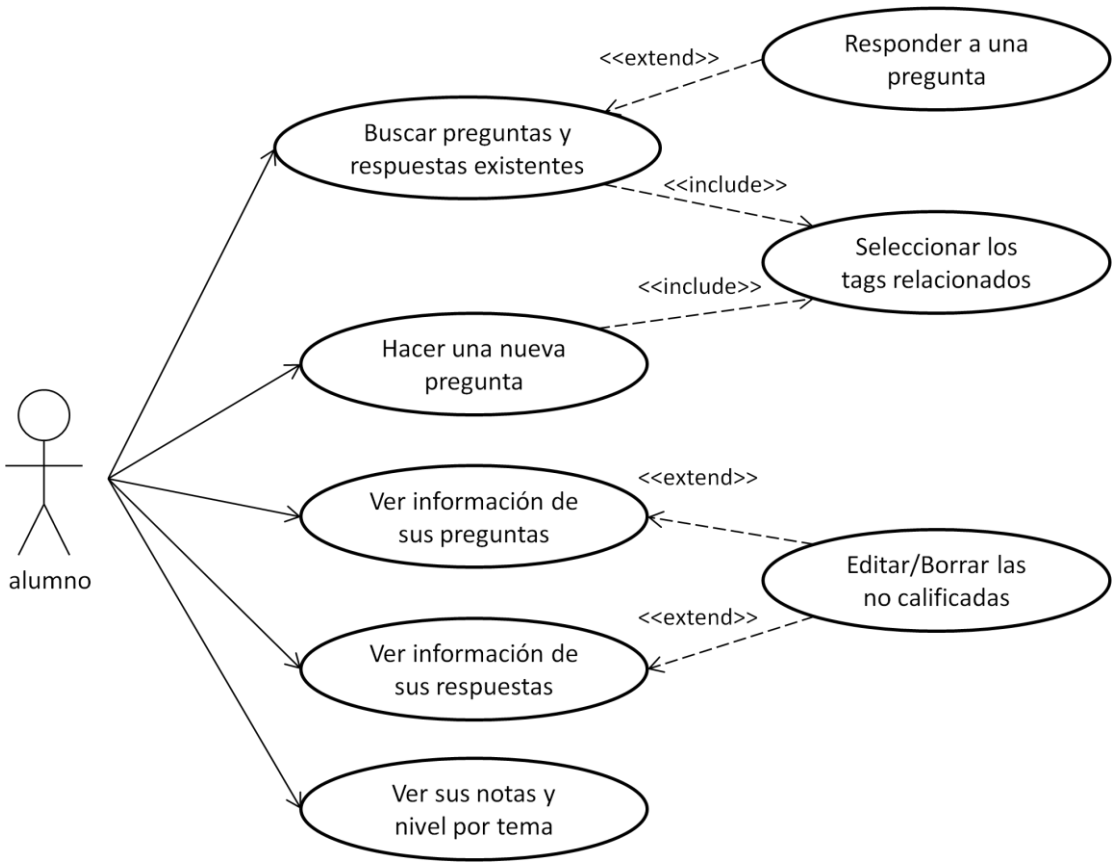


Figura 5. Diagrama de casos de uso para el alumno

En él sólo se han incluido las principales tareas y se ha evitado plasmar información redundante (para ampliar, ver capítulo 5, *Interfaz del alumno*).

Además de esta parte central del foro, se ha añadido una funcionalidad novedosa adicional orientada a fomentar la participación de los alumnos. Ésta consiste en plantearles preguntas que sigan sin resolver mientras permanezcan en el mundo virtual, independientemente de si están cerca del *Oráculo* o realizando otra actividad. Este mecanismo se representa visualmente mediante un mensaje de aviso y un *fantasma* que persigue al avatar del alumno mientras se desplaza por el mundo y que sólo desaparece en el caso de que el alumno acepte consultar la lista de preguntas o desactive este tipo de avisos.

3.2. Funcionalidad para el profesor

Las tareas que tiene que realizar el profesor se dividen en dos grupos: las requeridas en la etapa de instalación y configuración, cuando se debe dar de alta el sistema y se deben introducir los datos iniciales, y las relacionadas con el uso habitual de la herramienta, que proporciona la funcionalidad principal.

Tras crear el *Oráculo*, lo primero que el profesor tiene que hacer es definir la lista de alumnos de la asignatura. Los alumnos que no hayan sido registrados por el profesor, no tendrán acceso a la herramienta. Además, en esta primera etapa de configuración, también debe definirse el tesauro, por lo que el profesor tendrá que decidir qué lista de etiquetas comprende una asignatura. Estos dos pasos previos son necesarios para que los alumnos puedan comenzar a utilizar la aplicación. Posteriormente, durante el uso habitual de la herramienta, también se podrá acceder a las listas de alumnos y etiquetas para modificarlas.

Otras de las tareas principales del profesor para que la aplicación funcione correctamente son validar y calificar las preguntas y respuestas de los alumnos. Hasta que una pregunta o respuesta no sea válida, no será visible a los alumnos, para permitir así al profesor que controle los contenidos del foro.

Relacionada con la tarea de calificar, está la de definir los umbrales para los niveles de “experto”, “medio” y “bajo”. Éstos tienen un valor por defecto que puede ser modificado en cualquier momento por el profesor. Por ejemplo, podría ser que en algunas asignaturas para ser “experto” deba tenerse una nota superior a un 8 y, sin embargo, en otras más complicadas con tener un 7 ya sea suficiente.

Por otra parte, el profesor también tiene la capacidad de plantear nuevas preguntas que considere interesantes en la asignatura, bien para evaluar los conocimientos de los alumnos en ciertos conceptos, bien para dinamizar el foro. También puede consultar qué preguntas formuladas por los alumnos siguen sin respuesta y proporcionar él mismo una nueva si lo considera necesario.

En el siguiente diagrama de casos de uso se sintetiza la funcionalidad anteriormente descrita:

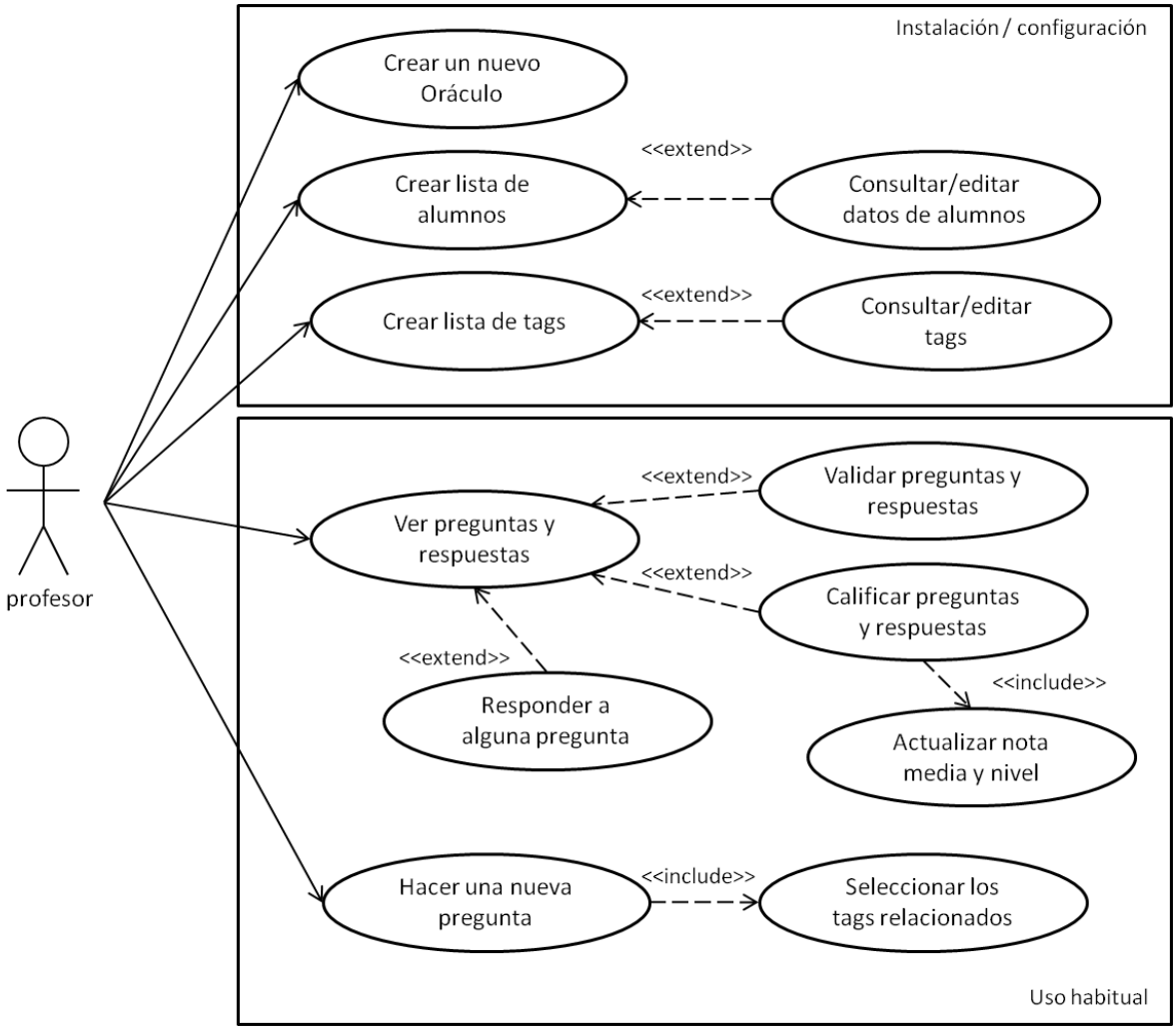


Figura 6. Diagrama de casos de uso para el profesor

En él se han incluido sólo las tareas principales para dar una idea general más compacta. Información más detallada se encuentra en el capítulo 6, *Interfaz del profesor*.

3.3. Diseño de la aplicación

En las secciones anteriores se han analizado la funcionalidad de la aplicación y los elementos principales de la plataforma *OpenWonderland;* estos dos factores son los principales motivadores del diseño elegido.

Como se ha podido comprobar, profesores y alumnos trabajan con los mismos datos (preguntas, respuestas, etiquetas, y toda la información que las rodea) pero las acciones que realizan sobre ellos son muy diferentes. Esto implica que la aplicación puede estructurarse de forma que compartan el acceso a los datos, pero las interfaces sean separadas.

Realmente, el objetivo central de este proyecto es la interacción del alumno, porque lo que se busca principalmente con los mundos virtuales es hacer el aprendizaje mucho más inmersivo y

colaborativo. Sin embargo, la interacción con el profesor requiere, por una parte, la configuración inicial de la herramienta y, por otra, un uso más complejo de los datos donde prima eficiencia sobre presentación.

Así, la parte del foro para entornos virtuales 3D, implementada en la plataforma *OpenWonderland* , es la del alumno, mientras que la del profesor es una herramienta 2D secundaria implementada en *PHP*, donde se ha perseguido una mayor simplicidad y un diseño similar a otras aplicaciones que el profesor está habituado a utilizar (como por ejemplo *Aula Global* en esta universidad).

Esto se observa de forma esquemática en la siguiente figura:

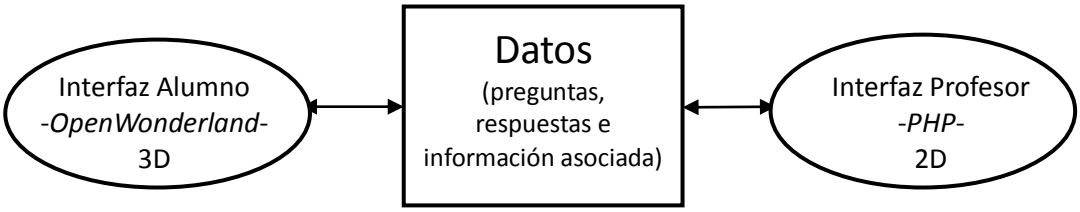


Figura 7. Estructura de la aplicación

Estas *interfaces* abarcan, por una parte, la lógica que define la recuperación y tratamiento de los datos y su comunicación entre los clientes y el servidor y, por otra, la forma de ser éstos presentados al usuario.

3.3.1. Interfaz del alumno

Siguiendo el modelo de desarrollo de la plataforma *OpenWonderland*, la implementación de la interfaz del alumno se ha realizado como un módulo, que se puede añadir a cualquier servidor para utilizar la aplicación.

El diseño del módulo se inspira en la representación metafórica del foro como un ser u oráculo, por ser el lugar al que acudir para resolver las preguntas que los alumnos tengan. Esta misma metáfora también ha inspirado el nombre dado al módulo, *Delfos*, en honor al oráculo griego dedicado principalmente al dios Apolo. [**6**]

Este concepto llevado al mundo virtual puede representarse mediante un *personaje no jugador*, en inglés *non-player character* o *NPC*. En *OpenWonderland* ya existe un módulo que implementa este tipo de personaje como un caso particular de celda (*cell*, ver capítulo 2, *Arquitectura de la plataforma OpenWonderland*).

Una opción de desarrollo del foro podría haber sido utilizar este módulo *NPC* y añadirle la funcionalidad específica (descrita en el capítulo 3, *Funcionalidad y diseño del "Oráculo"*). Sin embargo, esto tiene el inconveniente de que si otros desarrolladores mejoran el *NPC* existente, el *Oráculo* no podría aprovecharse de ello y quedaría obsoleto.

Teniendo esto en cuenta, se llegó a la conclusión de que el mejor diseño es implementar la funcionalidad del foro como un módulo independiente, consistente en un componente (*capability*

dentro de la terminología de *OpenWonderland*) que se le pueda añadir a cualquier objeto (*cell*), entre ellos, el *NPC.* Esta aproximación tiene ventajas no sólo desde el punto de vista del desarrollo, sino también de la utilización de la aplicación en experiencias educativas. Por ejemplo, podría ser que alguien prefiera utilizar como metáfora visual un objeto relacionado con la asignatura en lugar de un personaje, algo como un matraz para química, un ábaco para matemáticas, etc.

Se ha concebido que sólo haya un foro por mundo virtual, es decir, aunque el componente *Delfos* implementado se le añadiera a más de un objeto, todos actuarían de la misma forma y accederían a la misma base de datos, por lo que de cara al usuario (alumno) sería como si sólo hubiera uno.

Toda la estructura y funcionamiento detallado de este módulo se describe en el capítulo 5, *Interfaz del alumno.*

3.3.2. Interfaz del profesor

La interacción del profesor con la aplicación implica dos etapas: la etapa inicial de configuración y, posteriormente, el uso habitual de la herramienta para revisar las preguntas y respuestas de los alumnos, evaluarlas, introducir nuevas preguntas para dinamizar el foro, etc.

En primer lugar, para poder dar de alta un nuevo *Oráculo*, el profesor necesita llevar a cabo una serie de pasos, tanto dentro como fuera de *OpenWonderland*. Esta etapa implica la creación de la base de datos, el acceso al mundo virtual con poder administrativo (poder para insertar objetos, conferirles una serie de capacidades, etc) y la configuración inicial a través de la interfaz *PHP* provista para tal fin (inserción en la base de datos de información como la lista de alumnos o de etiquetas). Todo ello requiere una serie de condiciones previas y deber ser llevado a cabo siguiendo unos pasos concretos, por lo que el capítulo siguiente, de Instalación del Oráculo, se dedicará a ese propósito.

Por otra parte, para el uso habitual de la herramienta se dedica otra parte de la interfaz *PHP,* que está diseñada con el fin de ser intuitiva y similar a otras aplicaciones similares. En el capítulo 6, *Interfaz del profesor*, se hace un análisis más detallado de la estructura y de la forma de uso de esta interfaz.

La interfaz *PHP* del profesor, por tanto, cumple dos funciones: configuración y uso habitual de la herramienta, pero se implementa como una sola porque en ambos casos se realizan algunas tareas similares y para que la presentación al usuario sea homogénea.

3.3.3. Modelo de datos

Para poder soportar toda la funcionalidad descrita anteriormente es necesaria una base de datos que contenga toda la información y las relaciones entre preguntas, respuestas, etiquetas con las que están relacionadas, estudiantes y su nivel, etc.

En esta sección se presenta las tablas que componen esta base de datos, explicando brevemente la función de cada una y justificando su existencia de cara a su interacción con las interfaces de alumno y profesor (qué tipo de consultas realizan y qué tipo de modificaciones de los datos).

Por este motivo, en este capítulo no se describe en detalle la estructura de la base de datos, ni la totalidad de los campos que componen cada tabla, esto se incluye en el capítulo de *Anexos*, en el punto 10.3 *Estructura de la base de datos*.

Tabla de etiquetas

Esta tabla almacena la lista de las etiquetas disponibles en el foro (tesauro). En la siguiente matriz se muestra su relación con las tareas que realiza cada actor de la aplicación:

	Interfaz del alumno	**Interfaz del profesor**
Consultas (lectura de datos)	Es consultada por ambas interfaces para recuperar la lista de *tags*, tanto cuando se quiere realizar una búsqueda relacionada con algunos, como cuando se quiere formular una nueva pregunta.	
Modificaciones	No puede realizar ninguna	Permite añadir y borrar etiquetas cuando no estén relacionadas todavía con ninguna pregunta.

Tabla de preguntas

Almacena información relacionada con las preguntas, definiendo un identificador único para cada una. Es una de las tablas centrales de la base de datos y se ve afectada en la mayoría de las interacciones, de las cuales, las más importantes, se presentan en la siguiente matriz:

	Interfaz del alumno	**Interfaz del profesor**
Consultas (lectura de datos)	Preguntas realizadas por un alumno, datos completos de una pregunta a partir de su identificador, etc.	Preguntas sin calificar o sin validar, preguntas sin respuesta, preguntas de un alumno concreto, etc.
Modificaciones	Guardar una nueva pregunta, editar o borrar una pregunta propia (no calificada), etc.	Guardar una nueva pregunta, validar y calificar preguntas de alumnos, etc.

Tabla de respuestas

Almacena información relacionada con las respuestas. La funcionalidad básica que cubre esta tabla para cada interfaz se muestra en la siguiente matriz:

	Interfaz del alumno	Interfaz del profesor
Consultas (lectura de datos)	Respuestas a una pregunta concreta, respuestas dadas por el propio alumno, etc.	Respuestas a una pregunta concreta, respuestas desde la última visita, etc.
Modificaciones	Guardar una nueva respuesta, editar una respuesta propia (no calificada).	Guardar una nueva respuesta, validar y calificar respuestas de alumnos, etc.

Tabla de preguntas-etiquetas

Para garantizar que una pregunta pueda relacionarse con un número indefinido de etiquetas se define esta tabla, en la que cada entrada vincula un "*tag*" y un "*identificador de pregunta*". Así, para cada pregunta, se definirán tantas entradas en la tabla como etiquetas con las que esté relacionada.

En la siguiente matriz se muestra la interacción de cada interfaz con esta tabla:

	Interfaz del alumno	Interfaz del profesor
Consultas (lectura de datos)	Búsqueda de preguntas relacionadas con una o más etiquetas, etiquetas relacionadas con una pregunta concreta.	
Modificaciones	Cada vez que se guarda una pregunta en la correspondiente tabla, en ésta también debe crearse un registro por cada *tag* con el que esté relacionada.	

Tabla de alumnos

En ella se almacena el listado de alumnos de la asignatura, identificados unívocamente por su '*login*', que debe corresponderse con el nombre de su avatar, es decir, su *login* dentro del servidor donde se cree el mundo virtual con *OpenWonderland*.

Esta tabla es necesaria para que sólo puedan utilizar la aplicación los alumnos autorizados, que no tienen por qué ser el total de alumnos dados de alta en dicho servidor. La principal funcionalidad que aporta a ambas interfaces se muestra en la siguiente matriz:

	Interfaz del alumno	Interfaz del profesor
Consultas (lectura de datos)	Al entrar en el mundo, se comprueba la existencia del alumno en esta tabla. Sólo podrá acceder a la aplicación si el resultado es positivo.	Listado de alumnos pertenecientes a la asignatura.
Modificaciones	No puede realizar ninguna.	Crear la lista de alumnos y modificar sus datos (el *login* solamente si el alumno no ha participado aún).

Tabla de expertos

En esta tabla se almacena a los alumnos junto con sus calificaciones y nivel (*experto, medio o bajo*) para cada etiqueta. Esto no se puede hacer en la tabla anterior porque para cada estudiante tiene que haber tantas entradas como etiquetas definidas en la aplicación. En la siguiente matriz se muestran algunos ejemplos de la información que se puede extraer y guardar aquí:

	Interfaz del alumno	Interfaz del profesor
Consultas (lectura de datos)	Estadísticas propias para un alumno, quiénes son los alumnos expertos en cada tema, etc.	Qué alumno tiene qué nivel en cada tema, nota media de cada alumno, etc.
Modificaciones	No puede realizar ninguna (sólo se actualiza con preguntas y respuestas calificadas).	Cuando el profesor califica una pregunta o respuesta, todos los registros afectados se actualizan (según con qué *tags* estén relacionadas)

Tabla de preguntas no respondidas

Aquí se almacenan las preguntas que se han planteado a los alumnos mediante el mecanismo del "fantasma" y que éstos no han sido capaces de responder. Cada entrada relaciona un *identificador de pregunta* y un *login*, de forma que cada dupla es única, pero un mismo alumno aparecerá tantas veces como preguntas no haya sabido responder y una pregunta, como alumnos a los que se les haya planteado sin éxito.

	Interfaz del alumno	Interfaz del profesor
Consultas (lectura de datos)	Algoritmo del *fantasma*: conocer qué preguntas ya se han realizado a un alumno para no repetir.	Conocer qué preguntas no ha sido capaz de responder un alumno concreto o cuáles no ha respondido ninguno.
Modificaciones	Algoritmo del *fantasma*: cada vez que realiza una pregunta a un alumno y éste no la responde, escribe un nuevo registro en esta tabla.	No realiza ninguna.

Tabla de profesores

Sólo se usa en la interfaz del profesor y permite conocer cuándo fue la última vez que accedió al sistema para poder comprobar si hay nuevas preguntas o respuestas desde esa última visita.

Tabla de parámetros de configuración

Estos parámetros definen cuándo considerar a un alumno experto, con nivel medio o con nivel bajo y también cuántas veces deben ser formuladas las preguntas a cada alumno por el algoritmo del *fantasma* (ver capítulos 5 y 6).

	Interfaz del alumno	Interfaz del profesor
Consultas (lectura de datos)	Algoritmo del *fantasma*: conocer si una pregunta debe ser planteada a un alumno según ciertos criterios (ver capítulo 5, *Interfaz del alumno*).	Cuando se califica una respuesta, se recalcula la media del alumno y se consulta esta tabla para saber si este cambio afecta a su nivel. En cualquier momento el profesor puede saber el valor de estos parámetros.
Modificaciones	No puede realizar ninguna.	Todos los parámetros de la tabla se pueden modificar desde la Interfaz del Profesor.

4. Requisitos previos e instalación del "Oráculo"

4.1. Requisitos

La estructura de la aplicación (ver Figura 7) requiere, además del servidor de *OpenWonderland* para soportar la interfaz del alumno, otros dos servidores: el gestor de la base de datos y un servidor web para utilizar la interfaz del profesor.

Dada su amplia utilización, especialmente dentro del mundo del software libre, se ha decidido utilizar tecnologías '*LAMP*', es decir, *Linux – Apache – MySQL – PHP* para este propósito, ya que su combinación cubre los aspectos necesarios y son subsistemas software de código abierto y cuyo uso está muy extendido.

Por esta razón, en este documento se obviarán los detalles de la instalación de los servidores *MySQL* [**12**], como gestor de la base de datos, y *Apache* [**13**], como servidor web, y se cubrirán los requisitos del servidor *OpenWonderland*.

En primer lugar, es necesario tener un servidor en la versión 5 de *OpenWonderland* funcionando, ya sea ejecutándolo desde un archivo binario [**14**] o a partir del código fuente [**15**]. En ambos casos, además, para que la aplicación pueda ser usada, este servidor debe tener instalados (y funcionando correctamente) los módulos del servicio sql (*sql-service.jar*) y de autentificación (*security-session-auth.jar*). En los siguientes párrafos se explica el por qué y se proporciona una pequeña introducción de cómo usarlos (en los *Anexos* se puede encontrar información más completa).

4.1.1. Servicio SQL

Hasta la realización de este trabajo, no se había contemplado la comunicación con una base de datos *MySQL* desde la plataforma *OpenWonderland*, por lo que inicialmente no había ningún camino que seguir.

Así, ya que *OpenWonderland* está escrito en Java, la primera aproximación de este proyecto fue utilizar las librerías ya desarrolladas en este lenguaje, el API *Java Data-Base Connection, jdbc,* que proporciona un conjunto de clases necesarias para interactuar con el gestor de la base de datos *MySQL*.

Sin embargo, utilizar estas librerías tal cual no es posible, ya que el servidor *Darkstar*, el *middleware* sobre el que descansa *OpenWonderland,* impone una serie de restricciones en la realización de una transacción:

- La transacción tiene que realizarse en menos de 100ms o expirará un contador
- Puede producirse un fallo en cualquier momento si una transacción entra en conflicto con cualquier otra
- En cualquiera de los casos anteriores, la transacción se volverá a repetir sucesivas veces hasta que se realice con éxito.

Estas reglas afectan al tipo de código que se puede ejecutar en el servidor. Por ejemplo, en el caso de la base de datos, podríamos estar haciendo una inserción repetidas veces si no se realiza en los 100ms exigidos. Esto genera una inconsistencia en el acceso a los datos que no es permisible, por lo que se hace necesario buscar un mecanismo diferente. [**16**]

Con la colaboración encontrada en los foros que proporciona *OpenWonderland* y la ayuda de los distintos desarrolladores, se logró resolver esta incidencia. Para poder evitar los mecanismos principales de sincronización de *Darkstar*, está pensada la creación de servicios, que no están sujetos a las mismas restricciones. Así, un servicio lo que hace es guardar una lista de acciones que tiene que realizar y, cuando la transacción se realice con éxito, las ejecuta en otro hilo, sin límites en el tiempo.

Aprovechando esta característica, *Jonathan Kaplan* (uno de los principales desarrolladores de *OpenWonderland*) proporcionó un nuevo módulo [**17**] que implementa uno de esos servicios y que permite el acceso a la base de datos de una forma consistente. El funcionamiento consiste en crear una lista de consultas a la base de datos e irlas realizando conforme las anteriores transacciones vayan completándose con éxito. Una vez que todas se han ejecutado, se genera una nueva transacción para devolver los resultados a nuestro código en el servidor.

Así, este módulo lo que hace es proporcionar la herramienta genérica de acceso a la base de datos, de forma que cualquier otro código, a través de éste, pueda definir las consultas concretas y la forma de actuar con los resultados obtenidos.

En la siguiente figura se puede observar en sombreado dónde se enmarcaría este servicio dentro de la estructura de *OpenWonderland* (descrita en el capítulo 2, *Arquitectura de la plataforma OpenWonderland*):

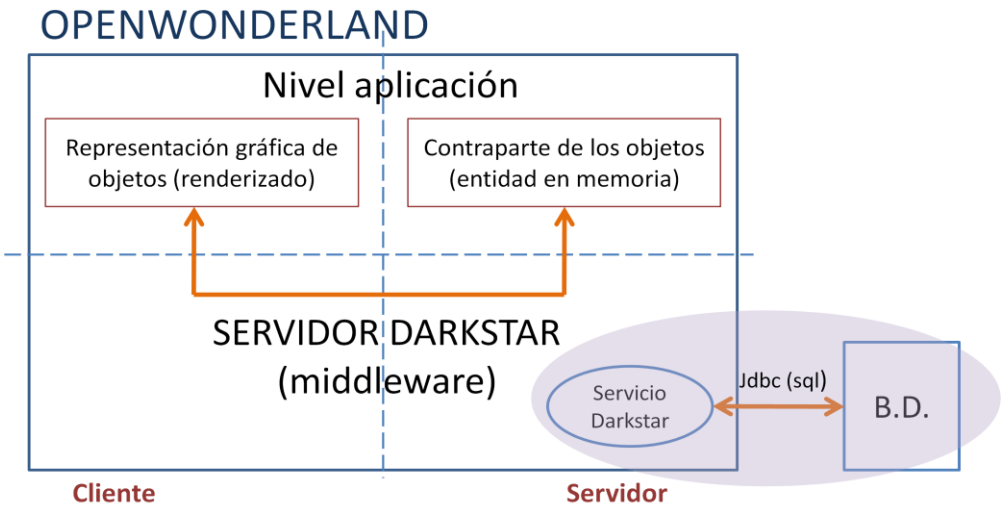


Figura 8. Arquitectura de *OpenWonderland* y *servicio sql*

Para poder usar este servicio, que es una herramienta genérica, sólo hay que particularizar según los requisitos concretos de cada aplicación. Esto se hace por medio de la herencia de clases Java, para lo que existen dos clases abstractas en este módulo que son las que habrá que extender:

- *BaseSqlRunnable<T>*: es la clase en la que se define y se lanza la lista de consultas que se quieren realizar. El valor T que aparece entre los símbolos <> define el tipo de dato que queremos devolver, ya sea *String, Integer*, o cualquier otro tipo de clase que nosotros hayamos definido. Lo que sí es importante es que este tipo de dato debe ser *serializable* (ver *Glosario*).

- *SqlCallback<T>*: es la transacción que se genera para devolver los datos una vez que se ha realizado la consulta con éxito. El servicio está pensado para que sea aquí donde se decida qué hacer con la respuesta de la base de datos, por lo que es necesario implementar el método *handleResult(T result)* donde el tipo de dato tiene que coincidir con el que se especificaba en la llamada anterior.

Una vez que se han definido las clases específicas como extensión de las anteriores, para ejecutar la consulta sólo es necesario recuperar el *SqlManager* de OpenWonderland (en concreto, del servidor *Darkstar*) y llamar al método *execute(BaseSqlRunnable, SqlCallback)* definido también en el servicio (ver capítulo 5, *Interfaz del alumno*).

Para completar la información aquí presentada, en los *Anexos*, en la sección 10.1 *Módulo del servicio sql*, se proporciona un breve resumen de cómo realizar la instalación y configuración de este módulo.

4.1.2. Módulo de autentificación

Como existen grupos específicos de personas que pueden utilizar el sistema y grupos que no, se hace necesario proporcionar un mecanismo de autentificación para acceder al *Oráculo*. Así, no todos los alumnos estarán autorizados a entrar en un determinado mundo virtual y, además, los que sí lo estén, tendrán que identificarse mediante una contraseña para que no puedan ser suplantados.

Sin embargo, en la configuración por defecto de *OpenWonderland* esto no ocurre, lo único que tiene que hacer el cliente es decidir un nombre para su avatar y con eso tiene el acceso garantizado. No obstante, gracias a como está diseñada la arquitectura, existe un módulo (*security-session-auth.jar*) que permite la autentificación y que está bastante depurado e incluso se puede encontrar información detallada de cómo instalarlo y utilizarlo. [**18**]

Este módulo permite dos tipos de configuración: uno en el que siempre se requiere autentificación, para lo que todos los usuarios deberán tener una cuenta válida en el servidor, y otro en el que, además, se permiten invitados que puedan acceder sin necesidad de tener una cuenta. Queda a discreción de quien gestione el servidor *OpenWonderland* decidir cuál de los mecanismos prefiere, aunque en concreto los estudiantes que vayan a utilizar el *Oráculo* sí que deben tener su cuenta con contraseña, ya que es lo que garantiza su autentificación.

Se ha diseñado así considerando que un estudiante, cuando accede a un mundo, probablemente va a realizar más de una actividad y resultaría muy redundante y molesto hacerle autentificarse para cada una de ellas por separado. Además, lo ideal es utilizar los recursos que ya están desarrollados y probados.

Además, estos estudiantes tendrán que formar parte de la tabla de alumnos prevista en la base de datos de la aplicación (Modelo de datos), ya que no todos los usuarios del mundo virtual tienen por qué poder utilizar el *Oráculo* (podría ser, por ejemplo, que fueran de otra asignatura).

Aunque toda la información de cómo instalar, configurar y usar este módulo se puede encontrar en la página de *OpenWonderland*, en los *Anexos* se incluye un breve resumen de los pasos para facilitar la tarea.

4.2. Pasos a seguir para la instalación

Una vez que los servidores *MySQL*, *Apache* y *OpenWonderland* funcionan correctamente y cumplen los requisitos especificados anteriormente, hay que seguir una serie de pasos previos a la utilización de la aplicación.

4.2.1. Creación de la base de datos

En primer lugar, es necesario crear la base de datos, para lo que se necesita un usuario con permisos en el servidor *MySQL* correspondiente. Después, habría que ir creando todas las tablas mencionadas en el capítulo anterior y configurando sus restricciones y relaciones (ver *Anexos*, sección 10.3 *Estructura de la base de datos*). Para que todo ello pueda hacerse de una forma sencilla, se proporciona un *script* de creación (*delfos-en.sql*) que sólo es necesario ejecutar desde un terminal, escribiendo:

```
>>mysql -u usuario -p < delfos-en.sql
```

En el lugar donde pone usuario deberemos escribir el nombre del nuestro y, al pulsar *retorno de carro*, nos pedirá la contraseña correspondiente.

4.2.2. Datos iniciales

Una vez creada la base de datos, antes de que pueda usarse la aplicación es necesario definir qué alumnos van a poder hacerlo y cuáles van a ser las palabras clave (etiquetas, *tags*). Para ello, se ha dedicado una parte de la *Interfaz del profesor* a dar de alta un nuevo *Oráculo,* guiando al usuario a través de los distintos pasos.

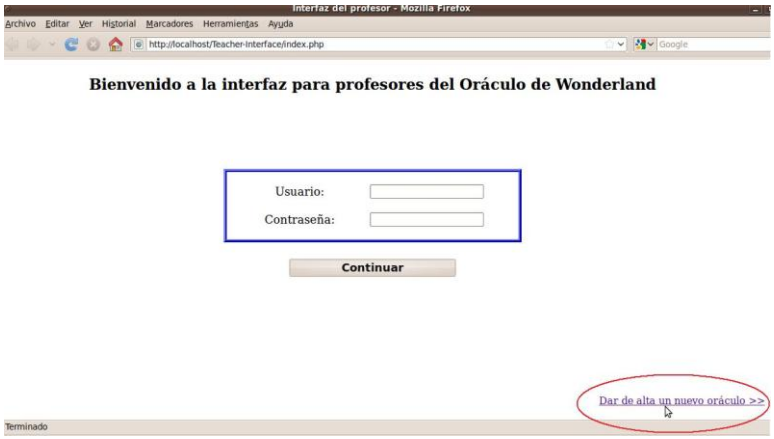


Figura 9. Pantalla principal de la Interfaz del Profesor

Lo primero que habrá que hacer será dar acceso a esta aplicación a la base de datos creada, para lo que necesitará un usuario con contraseña que tenga permisos de acceso en el servidor *MySQL* y el nombre de la base de datos.

A continuación, habrá que crear un usuario nuevo, que será el que se utilice posteriormente para autentificarse como profesor en esta interfaz y poder acceder a ella. Estos datos que se introducen en la etapa de creación se almacenan creando un nuevo fichero *PHP* que tiene por nombre el *login* que haya elegido el profesor y cuyo contenido es una función que devuelve los datos de acceso a la base de datos. Por otra parte, para poder autentificar al profesor, existe una tabla *"teachers"* donde se almacena su nombre de usuario y contraseña.

Cuando ya se han completado estos dos pasos, se accede a la pantalla en la que se introducen los datos de los alumnos, necesarios para que éstos puedan empezar a utilizar el *Oráculo* en *OpenWonderland*. Aquí el profesor puede añadir nuevos alumnos, así como editar o borrar aquellos que considere necesario.

Figura 10. Crear lista de alumnos

Una vez completa la lista de alumnos, el siguiente paso es seguir el enlace en la parte inferior derecha de la pantalla, que permite crear la lista de etiquetas (a las que a partir de ahora nos

referiremos como *tags*). Este paso también es indispensable en la instalación ya que, si no hay etiquetas, no se pueden catalogar las preguntas, imposibilitando su posterior búsqueda. Por este motivo, existe la restricción en la interfaz del alumno de que, incluso en el caso de que un estudiante ya pertenezca a la base de datos, no se le permitirá interactuar hasta que no existan *tags*.

La lógica de esta pantalla es similar a la anterior, con idea de que sea más fácil interactuar con ella. Por una parte, existe un botón en la parte derecha que permite añadir nuevos *tags*, de forma que se va generando una tabla en la parte izquierda con los que ya existen. Además, en esta misma tabla se pueden seguir los enlaces para editar o borrar los *tags*. Una vez editados, se guardan los cambios con el botón *Grabar* que aparece en la parte inferior, justo debajo de la tabla.

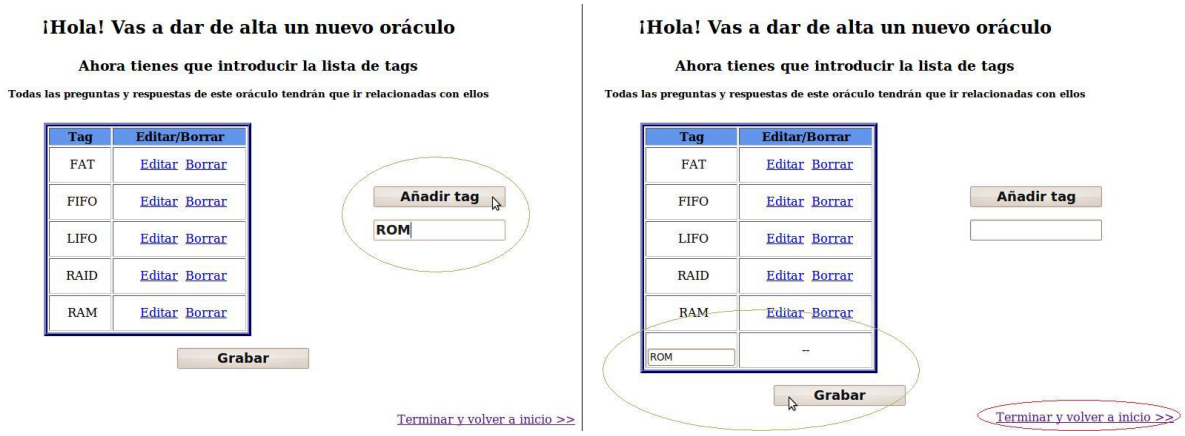


Figura 11. Creación de la lista de *tags*

Para finalizar, basta con cerrar el navegador o, si se quiere comprobar que todo funciona correctamente, seguir el enlace inferior que vuelve a la página principal donde habrá que autentificarse. A partir de ahí, la interacción con esta interfaz está descrita en el capítulo 6, *Interfaz del profesor.*

4.2.3. Instalación del módulo en *OpenWonderland*

Por último, para que la instalación se complete con éxito, el profesor tendrá que acceder al mundo virtual de *OpenWonderland* (versión 5) donde quiera situar el *Oráculo*. Previamente, deberá asegurarse, en la pantalla de administración del servidor, de que los módulos necesarios están instalados. Estos son, además de los mencionados en la sección 4.1. *Requisitos (servicio sql y autentificación)*, el módulo desarrollado en este trabajo, *delfos-component-module.jar,* y el que implemente el objeto al que queramos conferirle la capacidad de *Oráculo* que, como sugerencia, podría ser un NPC (*npc.jar*).

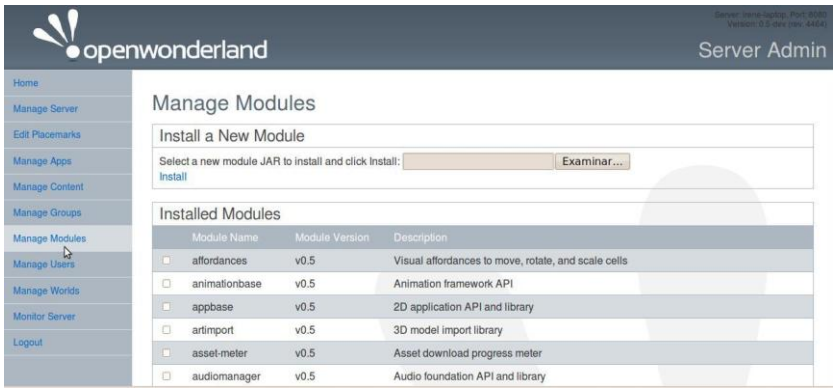


Figura 12. Pantalla de instalación de módulos de *OpenWonderland*

Una vez el profesor se ha autentificado y ha entrado en el mundo, tendrá que situarse en el lugar donde quiera que esté el *Oráculo* y, pulsando la opción *Insertar>Objeto* del menú superior, seleccionar el objeto 3D al que quiera concederle esta capacidad (como ejemplo, en esta memoria se utiliza un *NPc)* y darle a aceptar.

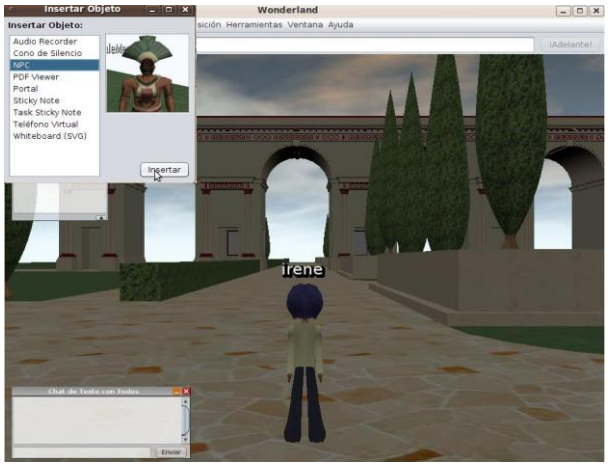


Figura 13. Creación de un NPC

Ahora que el objeto se *renderiza* en el mundo, el siguiente paso es hacer *click* con el botón derecho del ratón sobre él y seleccionar *Propiedades*, que hará aparecer una ventana donde se le pueden añadir capacidades a este objeto.

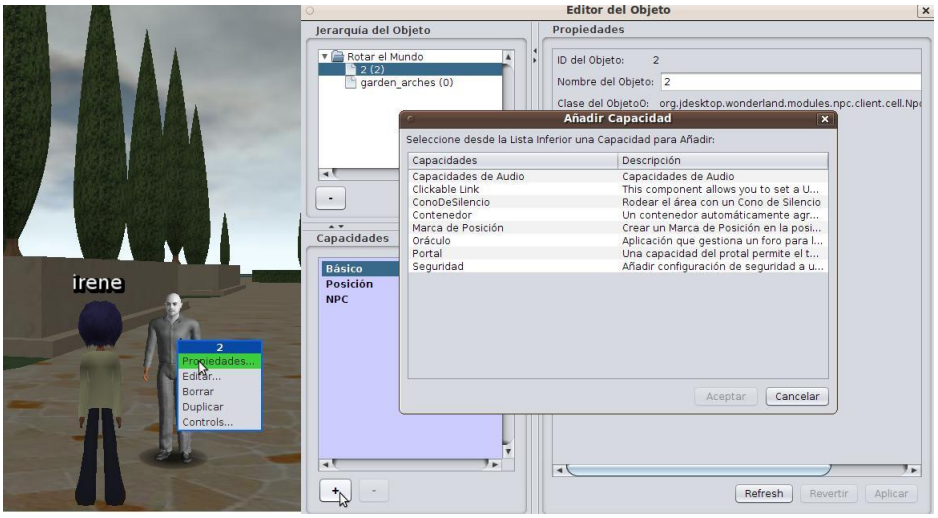


Figura 14. Añadir capacidades a un objeto

Habrá que añadir dos capacidades: por un lado, la capacidad *Oráculo*, de la que habrá que introducir el texto que se debe mostrar al posar el ratón sobre el objeto (*tooltip*); y, por otro, también habrá que añadirle *Seguridad*, de forma que sólo el profesor pueda modificar las propiedades del *NPC* (o el objeto elegido) y ningún alumno esté autorizado a hacerlo.

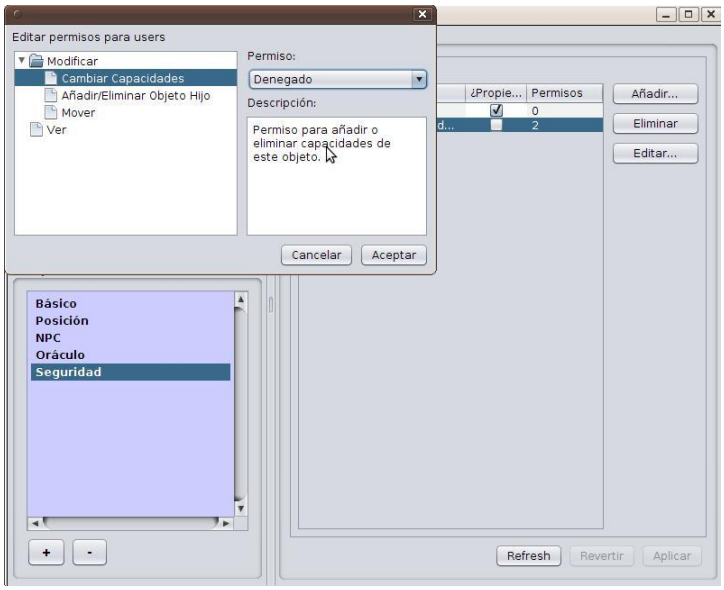


Figura 15. Configurar los permisos de un objeto

El módulo que permite conferirle seguridad a cualquier objeto del mundo está instalado por defecto en el servidor, por lo que no requiere ningún paso previo.

5. Interfaz del alumno

En este capítulo se completan todos los detalles de la funcionalidad y el diseño de la interfaz del alumno y se explica cómo se ha llevado a cabo su implementación.

A modo de resumen de la funcionalidad y estructura de esta parte (introducidos en el capítulo 3, *Funcionalidad y diseño del "Oráculo"*), se incluye el siguiente esquema:

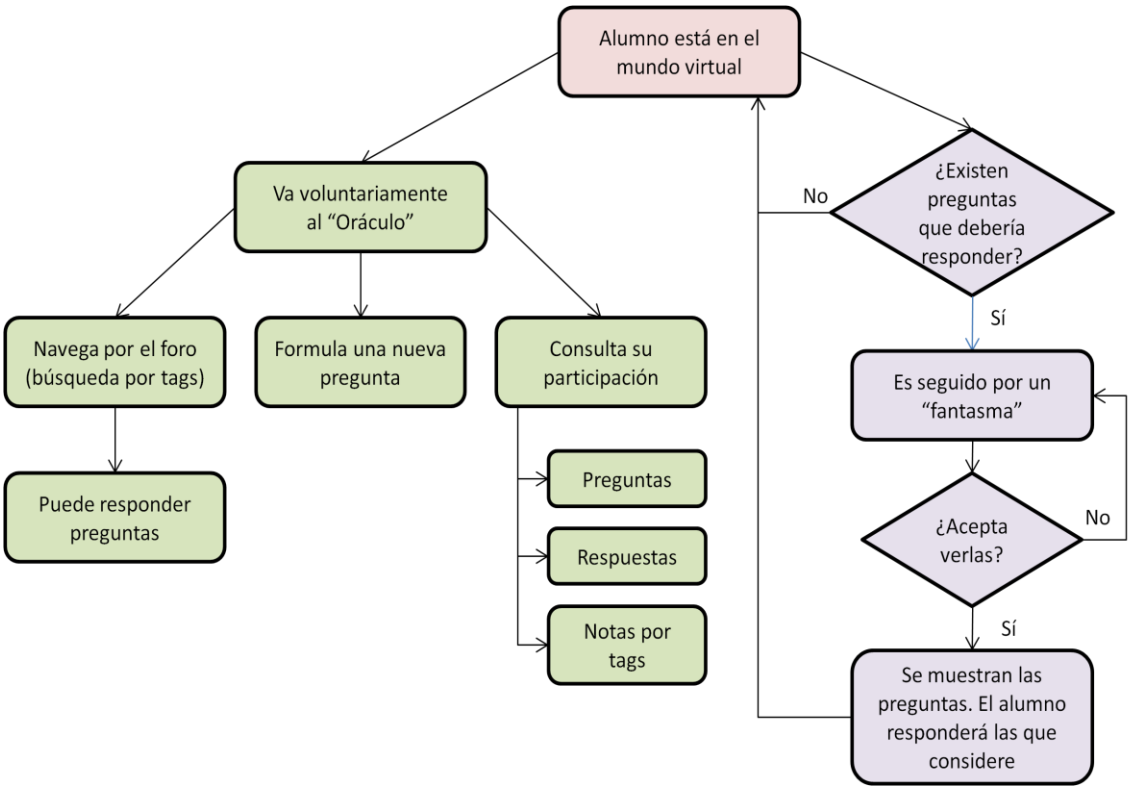


Figura 16. Interfaz del alumno

Donde vemos sombreada en verde la parte correspondiente a la interacción voluntaria del alumno y en morado la parte automática, que intenta hacer participar a los alumnos incluso aunque éstos no lo tuvieran planeado.

No obstante, aunque sean dos caminos diferenciados, los dos siguen un patrón de diseño similar y comparten recursos y porciones de código. Por tanto, el primer paso del presente capítulo será presentar la lógica que se ha seguido en la implementación de ambos y los principales tipos de clases *Java* que se encargan de llevar a cabo la funcionalidad.

5.1. Estructura

En primer lugar, es importante distinguir los 3 niveles en los que se estructura esta parte de la aplicación:

- Nivel inferior, que se encarga de la comunicación con la base de datos y, por tanto, tiene lugar en el servidor.
- Nivel intermedio, en el que se gestiona el paso de mensajes entre cliente y servidor y que depende de la tecnología *middleware* de *OpenWonderland*, el servidor *Darkstar.*
- Nivel superior, interfaz gráfica, conjunto de ventanas y elementos visuales que le permiten desde hacer una pregunta, hasta consultar otras que se hayan hecho, ver sus estadísticas, etc.

Así, aunque todos están relacionados, se persigue que su implementación sea lo más independiente posible, de forma que, a medida que la tecnología avance, puedan mejorarse por separado sin que ello afecte al resto de la aplicación. Este tipo de diseño atiende a un conocido patrón: el *modelo-vista-controlador*, que se analizará a continuación.

5.1.1. Patrón modelo-vista-controlador

Este patrón de diseño trata de distinguir tres bloques funcionales diferenciados de forma que su desarrollo y posterior mantenimiento sea lo más independiente posible. Estos bloques son:

- *Vista:* se refiere a la interfaz gráfica, es decir, a la forma visual de presentar los datos, que en nuestro caso será un conjunto de ventanas y modelos 3D.
- *Modelo*: contiene los datos y su procesamiento, por lo tanto, será la parte que se comunique con la base de datos
- *Controlador*: es el bloque que captura las peticiones del usuario y las redirige por el camino adecuado, ya sea comunicándose con la vista o con el modelo.

A continuación se explica cómo se aplica este patrón a la *interfaz del alumno*, para lo cual es importante saber que ésta implementa una serie de interacciones que siguen el siguiente flujo:

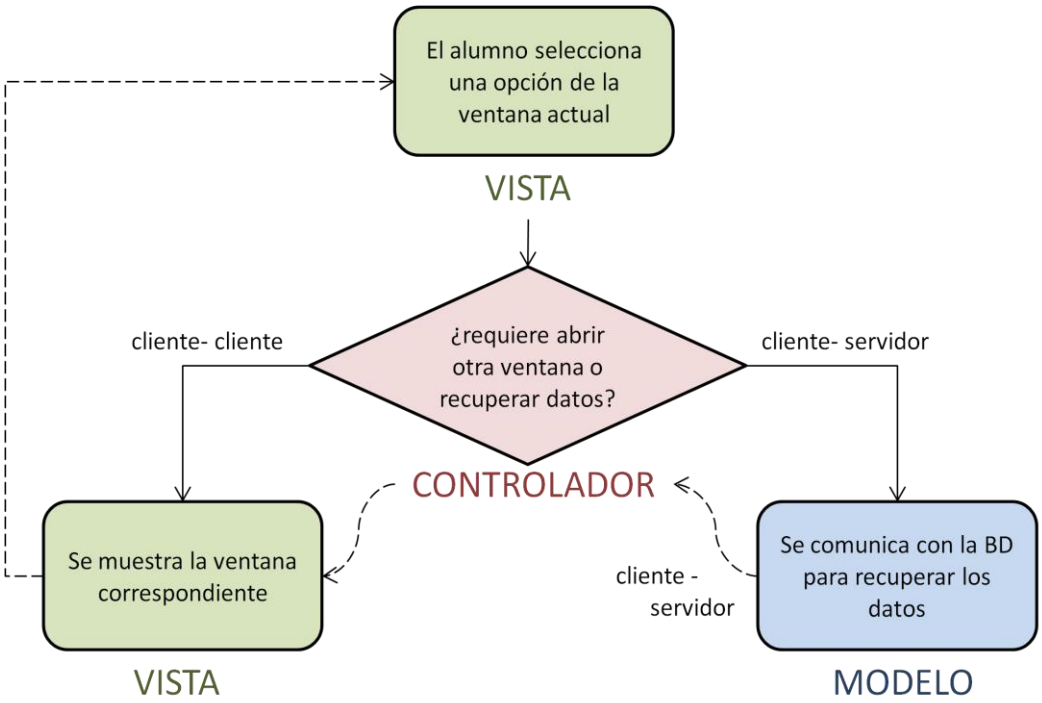


Figura 17. Aplicación del patrón modelo-vista-controlador

Así, se podría decir que, de los niveles citados al principio de la sección, el nivel inferior pertenece al *modelo*, ya que se encarga de la comunicación con la base de datos, mientras que los otros dos niveles son compartidos. El nivel intermedio, es decir, la comunicación cliente-servidor, involucra tanto a *controlador* como a *modelo* y, por su parte, el nivel superior, aunque principalmente es la representación gráfica y, por tanto, la *vista*, también conlleva la definición de escuchadores de las peticiones del alumno, correspondientes al *controlador*.

Analizando en mayor profundidad el módulo desarrollado[3], éste se estructura en una serie de clases *Java* que se agrupan en estos tres bloques (el listado completo de las clases se incluye en los *Anexos*). A continuación se presentan de forma esquemática las principales clases que soportan la funcionalidad, organizadas según la estructura de módulo que define la plataforma *OpenWonderland* (ver capítulo 2).

[3] Para el desarrollo de este módulo se tomó como punto de partida uno ya existente, el "*Tooltip-component*" de *Jordan Slott*, desarrollador de *OpenWonderland* [**24**]. Dicho módulo consiste en una *capacidad* que permite mostrar un "tooltip" o "descripción emergente" en el HUD (ver *Glosario*) al posar el ratón sobre el objeto 3D al que se le añade. Sobre esta base, se ha implementado la funcionalidad completa del foro en clases separadas, aprovechando principalmente la definición y creación de un *componente* o *capacidad* ya existente. Esto se especifica en el listado de clases incluido al final del trabajo.

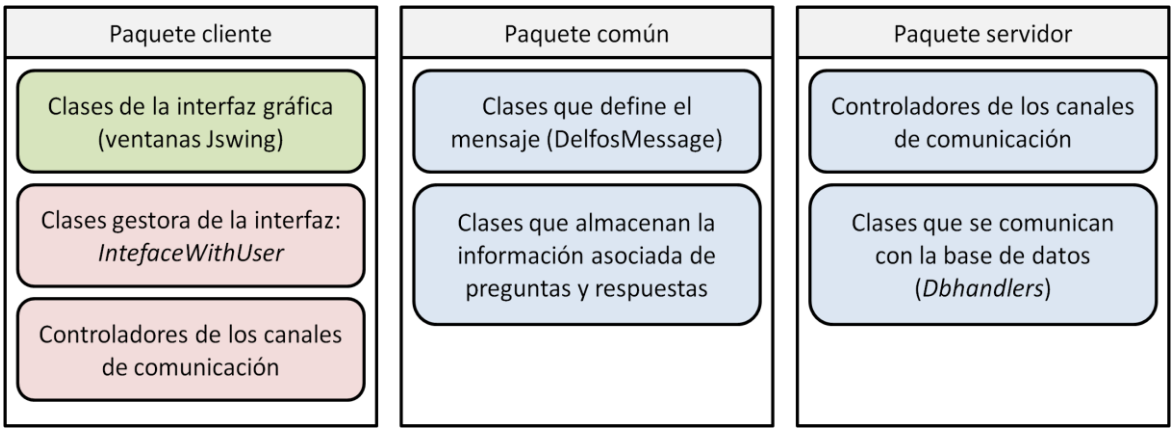


Figura 18. Estructura del módulo desarrollado

En la imagen están representados los bloques mediante los mismos colores que en la Figura 17 (*vista* en verde, *controlador* en rojo y *modelo* en azul). Las clases que se han representado en la columna del paquete común definen la forma de almacenar la información en memoria y transmitirla entre cliente y servidor, por lo que forman parte del modelado de los datos. No obstante, están en dicho paquete porque es necesario que se instalen también en el cliente, ya que tanto la *vista* como el *controlador* deben ser capaces de interpretar la información que les llega.

Una vez conocida esta estructura, se puede explicar con mayor detalle cada tipo de clase *Java. E*n los *Anexos*, sección 10.4 (*Listado completo de las clases de la Interfaz del Alumno*), se incluye un esquema del flujo de ejecución de estas clases:

- Clases que se comunican con la base de datos (*DBhandlers*). Se definen varias, según el tipo de consulta y de datos que se necesite recuperar. Todas ellas utilizan el servicio *SQL* proporcionado por *Jonathan Kaplan* (capítulo 4) para la comunicación con este tidpo de bases de datos, por lo que deben extender las clases *BaseSqlRunnable<T>* y *SqlCallback<T>* de dicho servicio.

- Controladores de los canales en el servidor (*ClientConnectionHandlers, 2.3.3, Nivel de conexión).* Hay dos, correspondientes a las dos partes que se mencionaban anteriormente (iniciativa por parte del alumno o por parte del *Oráculo*). Cada uno tiene la misión de escuchar el canal y, cuando se reciban mensajes de los clientes, analizarlos y llamar al *DBhandler* correspondiente para recuperar la información requerida. Una vez completo el mensaje con los datos (para lo que pueden ser necesarias llamadas encadenadas a varios *DBhandlers*), es de nuevo esta clase la que envía el mensaje al cliente de vuelta).

  Al llamar a un *DBhandler*, es importante que el controlador le pase la dupla (cliente, mensaje) y que ésta permanezca unida mientras dure el procesamiento en el servidor para que no haya problemas de concurrencia. Así, las peticiones de los distintos clientes se atienden de forma secuencial, ya que el acceso a la base de datos lo es de todas formas.

- *DelfosMessage*. Es una clase *Java* que extiende a *Message* (definida en el *core* de *OpenWonderland*, ver 2.3.3, *Nivel de conexión*) y que tiene atributos para almacenar y transmitir los diferentes tipos de datos de la aplicación (cadenas de caracteres, vectores,

enteros…). Una característica importante es que todas las instancias de *DelfosMessage* deben crearse con un identificador numérico asociado (*messageID*) para saber su funcionalidad. Los posibles valores del identificador se definen en esta clase como atributos finales.

- Información asociada a preguntas y respuestas *(QuestionData* y *AnswerData).* Son dos clases Java creadas para esta aplicación de forma que todos los datos asociados a preguntas y respuestas (texto, identificador, nota, alumno que la formuló…) se puedan manejar en conjunto de manera más cómoda. Al ser ésta una información que se transmite por un canal de comunicación y también se recupera de la base de datos, estas clases deben ser *serializables* (ver *Glosario*).

- Controladores de los canales en el cliente (*ClientConnection, Nivel* de conexión*)*. Su funcionalidad básica es enviar y recibir mensajes del servidor (del correspondiente *ClientConnectionHandler*). Igualmente hay dos, uno por canal, ya que cada uno realiza acciones diferentes. De hecho, aquí es necesario especificar que, en el caso del canal del *fantasma*, parte de la lógica del *modelo* se lleva a cabo aquí, ya que es un procesamiento algo más complicado y el servidor recibe muchas peticiones, por lo que se ha considerado mejor implementar ciertos algoritmos en el cliente y así evitar sobrecargas (en la sección 5.3, *Interacción por iniciativa del Oráculo: Fantasma* se explica esto con mayor detalle).

- *InterfaceWithUser*. La función de esta clase es gestionar las diferentes ventanas que conforman la interfaz y hacer de nexo entre éstas y los canales de comunicación. Tiene un atributo de tipo *HUD* (*Head-up-display*), es decir, un tipo de componente visual que siempre se observa en la pantalla, incluso aunque nos movamos con el avatar por el mundo (ver *Glosario*). En este HUD es donde se van poniendo los diferentes paneles definidos en las clases de *JSwing*. Así, cuando se recibe un mensaje en el cliente, el *ClientConnection* se lo pasa a *InterfaceWithUser*, que, a su vez, lo procesa y crea el *Jpanel* correspondiente y lo carga en el HUD.

- Clases de JSwing. Cada una define un *JPanel* en el que se presenta la información correspondiente y se proporcionan una serie de botones y otros elementos de selección como *checkboxes* para que el alumno pueda ir interactuando con la herramienta. Para unificar y simplificar el código, siempre reciben como parámetro una referencia a la *InterfaceWithUser*, es decir, una referencia al *controlador,* y un *DelfosMessage*, de donde cada una ya extraerá los datos correspondientes.

Todo el código desarrollado se ha llevado a cabo en inglés para que pueda ser reutilizado y extendido por cualquier miembro de la comunidad de *OpenWonderland*, que es internacional. No obstante, todos los textos que aparecen en la interfaz con el alumno, es decir, en las diferentes ventanas, pueden estar en cualquier otro idioma, gracias a un recurso provisto por Java: el *BUNDLE*. Éste es un tipo de fichero donde se van definiendo los pares clave-valor, de forma que, aunque las claves de los textos están en inglés, su valor puede estar en el idioma que se quiera para simplificar

las traducciones sin tener que modificar el resto del código. Concretamente, en el módulo desarrollado se han incluido dos ficheros de este tipo: uno en español y otro en inglés.

5.2. Interacción por iniciativa del alumno

Una vez conocida la estructura del módulo y la funcionalidad básica del foro, se pueden ir analizando las diferentes ventanas de la aplicación y cómo el alumno puede interactuar con ellas. Esta parte corresponde a la sección sombreada en verde en la Figura 16, al principio del presente capítulo; una versión ampliada y detallada se muestra a continuación:

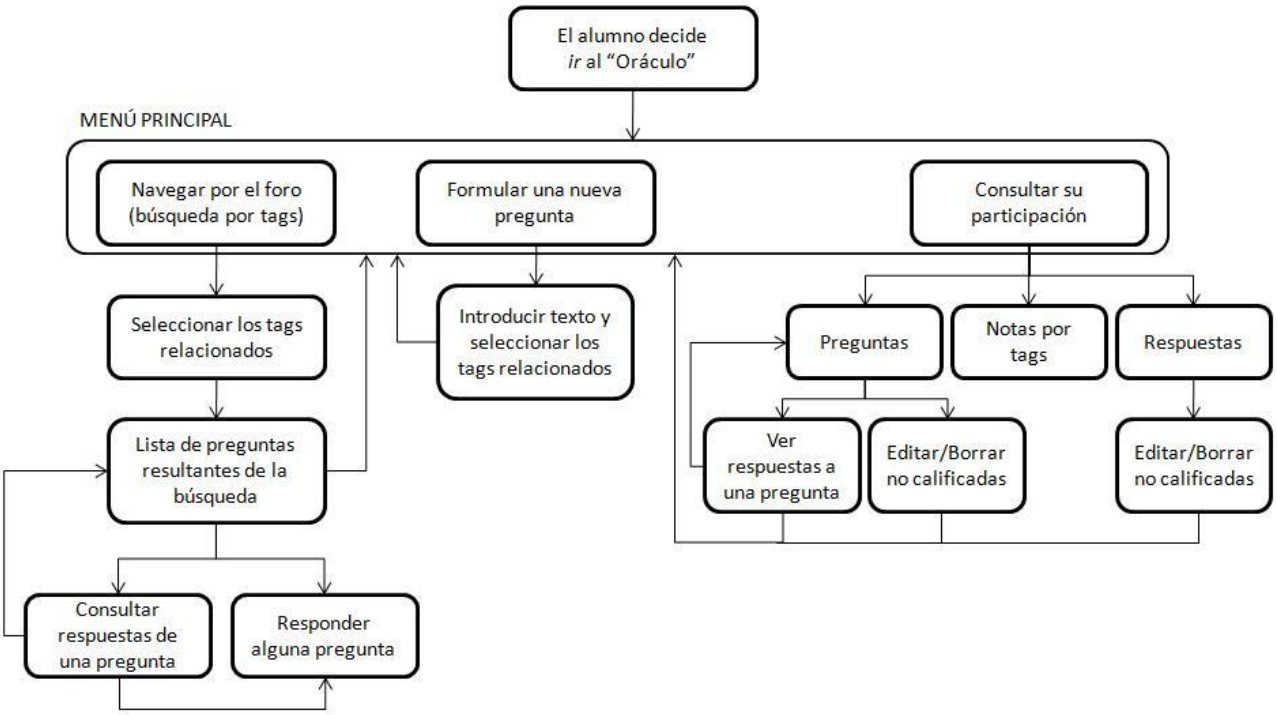


Figura 19. Interacción por iniciativa del alumno

En este diagrama sólo aparece lo que percibe el alumno, pero no los procesos internos que tienen que tener lugar para poder proporcionar esta funcionalidad. Por eso, a continuación se irán recorriendo las distintas ramas mostrando, por una parte, cómo visualiza esto el alumno realmente y, por otra, la serie de tareas que se desencadenan con cada interacción.

En primer lugar, el alumno se acerca al *Oráculo* y, si posa el ratón sobre él, aparecerá un texto, a definir por el profesor, que se sugiere que sea algo similar a "*Soy el Oráculo, haz click para comenzar*". Cuando el alumno hace *click*, lo que ocurre internamente es que el *controlador* percibe esta acción y envía un mensaje al servidor para que el *modelo* compruebe si pertenece a la asignatura, tras lo que se enviará el resultado a la *vista*. Si no pertenece, se muestra un mensaje temporal de error y se termina el proceso. En caso contrario, se visualiza entonces el panel del menú principal, que contiene tres botones, como se observa en el diagrama.

Si se selecciona la primera opción: "*Navegar por el foro*", vuelve a darse un proceso similar: el *controlador* se comunica con el *modelo* para recuperar la lista de *tags* (etiquetas) y, una vez

obtenida de la base de datos, ésta se envía de vuelta al cliente para que se muestre en la ventana correspondiente. Una peculiaridad que se ha añadido a esta lista de *tags* es que, junto a cada uno, aparezca el nombre de los *expertos* en ese tema (Figura 20). Recordando la definición de "experto" proporcionada en el capítulo 3, éste se define como el alumno que en un cierto tema (identificado por un *tag*) tiene una nota mayor que una dada (elegida por el profesor). Este tipo de distintivo, y el hacerlo público ante el resto de alumnos, es un intento de motivar la participación, así como una forma de identificar a quién se le debe preguntar cuando se tengan dudas en ciertos temas que otros no han sabido responder.

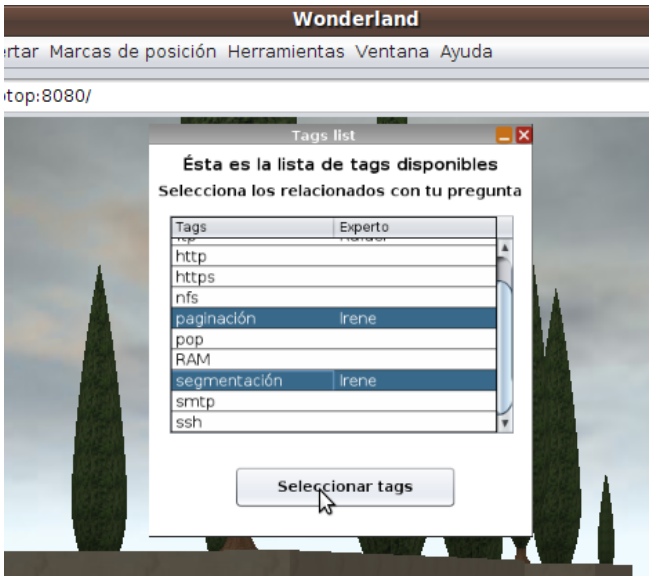


Figura 20. Lista y selección de *tags*

El siguiente paso es seleccionar los *tags* con los que quiera relacionarse la búsqueda, de forma que aparecerá una lista de las preguntas ya existentes en el foro y así el alumno puede ver si su duda ya está resuelta. No hay límite en el número de *tags* con que se puede relacionar una pregunta y, por tanto, tampoco lo hay para el criterio de búsqueda. Cuando se hace la selección, se envía un mensaje al servidor y se recuperan de la base de datos todas las preguntas relacionadas con cualquiera de los *tags* seleccionados. Después, se ordenan en función del número de *tags*, es decir, primero se muestran las que estén relacionadas con todos los seleccionados, después las que tengan uno menos, etc. (Figura 21). Así se reduce la posibilidad de que una búsqueda dé resultados vacíos y tenga que realizarse de nuevo, haciendo más lenta la interacción. Además, hay que tener en cuenta que los propios alumnos son los que catalogan sus preguntas y podría ser que los criterios de unos y otros a la hora de relacionar un texto con ciertas etiquetas no fueran los mismos.

El aspecto de la lista de preguntas sería similar al siguiente:

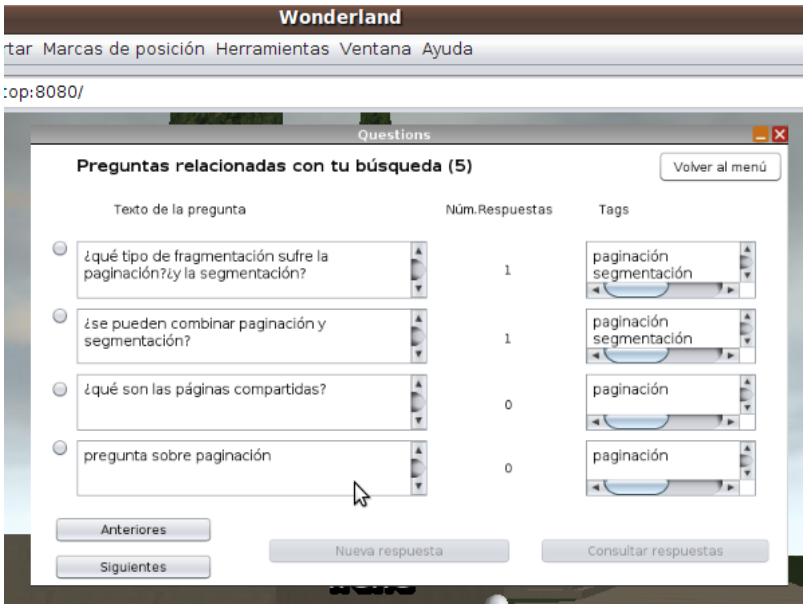


Figura 21. Preguntas relacionadas con una serie de *tags*

En la Figura 21 se observa que en la ventana aparece el texto de la pregunta junto con el número de respuestas y los *tags* con los que está relacionada (para saber si son todos los requeridos o sólo parte de ellos). Además, en la esquina superior derecha hay un botón para volver al menú principal (que se encuentra también en otros paneles) y, en la parte inferior, otros cuatro botones. Por un lado, están los de "Anteriores" y "Siguientes", que permiten moverse a través de la lista de preguntas (el número total es el que aparece entre paréntesis junto al título del panel). Por otro, aparecen dos botones desactivados, llamados "Nueva respuesta" y "Consultar respuestas", que sólo se activan cuando se selecciona una pregunta (usando los *radiobuttons* a la izquierda de cada una). Es más, en el caso del segundo, no se activa a menos que la pregunta tenga respuestas.

Las preguntas que se muestran como resultado de esta búsqueda son sólo las válidas (ver capítulo 3, Funcionalidad y diseño del "Oráculo"), por tanto, puede que haya más preguntas relacionadas con los temas elegidos que no aparezcan. De igual modo, una pregunta puede tener ya respuestas, pero si no están validadas, el número que aparece a su derecha seguirá siendo cero y no podrán consultarse. Con esto, se entiende que en la tabla de preguntas de la base de datos existan varios campos relativos al número de respuestas, ya que hay que contabilizar tanto las válidas, como las no válidas y las pendientes de validación. En el apartado siguiente y en el capítulo dedicado a la interfaz del profesor se profundizará en la utilidad de estos campos.

Volviendo al análisis del panel, el siguiente paso dependerá de la opción elegida por el alumno:

- Volver al menú principal: en este caso, el *controlador* (*InterfaceWithUser*) no necesita comunicarse con el *modelo*, ya que lo único que hay que hacer es mostrar otra ventana, lo cual es tarea de la *vista*.
- Hacer una nueva pregunta: igual que en el anterior, en este caso sólo es necesaria la siguiente ventana, que permite introducir texto.
- Consultar respuestas: aquí es necesario comunicarse con el servidor, con lo que el *controlador* le envía un mensaje al *modelo* para que recupere las respuestas de una

pregunta dada (conocida gracias a un identificador único de pregunta que también se almacena en la tabla de respuestas). Después se envía el mensaje de vuelta al cliente y se muestra la ventana correspondiente (Figura 22).

Siguiendo la tercera alternativa, el resultado obtenido sería similar a:

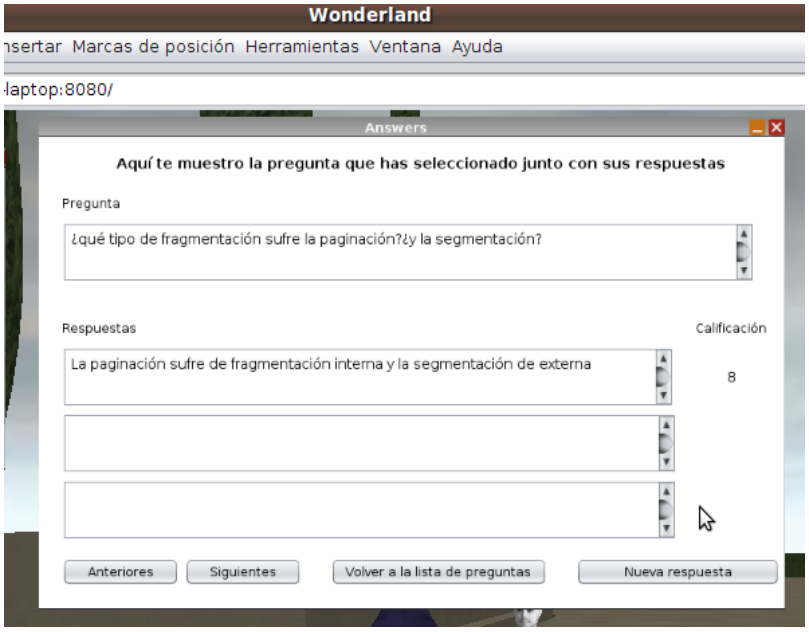


Figura 22. Pregunta junto con sus respuestas

En esta ventana se observa la pregunta seleccionada acompañada de las respuestas que tenga, igualmente para desplazarse por ellas se usarían los botones “Anteriores” y “Siguientes”. Los otros dos botones presentes son el de “Nueva respuesta”, cuya funcionalidad es idéntica al de la ventana anterior y el de “Volver a la lista de preguntas”, que lleva de nuevo al panel anterior (sin necesidad de pasar por el servidor esta vez, porque en el cliente se guarda una copia de la lista de preguntas recuperadas en la búsqueda actual; se irá sobrescribiendo con posteriores búsquedas).

Si en alguno de los dos paneles el alumno decide dar una nueva respuesta, simplemente le aparecerá una ventana donde introducir texto y, una vez lo haga, se enviará un mensaje al servidor para que almacene el texto en la base de datos. Esto afecta tanto a la tabla de respuestas, como a la de preguntas, donde el contador del número de respuestas (sin validar) se incrementa. Si la operación se realiza con éxito, en la pantalla se observará un mensaje temporal confirmándolo y nos volverá a mostrar el menú principal.

Por otra parte, continuado por otra de las ramas del menú (Figura 19) se proporciona la opción de introducir una nueva pregunta en el foro. En este caso es necesaria la comunicación con el *modelo* para recuperar la lista de *tags*. Así, al alumno se le presenta una ventana con un área para introducir texto y con las etiquetas con que puede relacionar su pregunta. Si deja el espacio en

blanco o no selecciona ningún tema, no se activará el botón de “Guardar”. El aspecto de esta ventana sería similar al siguiente:

Figura 23. Panel para formular una nueva pregunta

Al igual que en el caso de dar una respuesta, una vez se ha guardado la información (en la tabla de preguntas por una parte y en la tabla que relaciona los *tags* con el identificador de pregunta por otra) aparecerá un mensaje de éxito y de nuevo la ventana del menú principal.

Por último, la tercera rama del menú (Figura 19) está pensada para que el alumno pueda ver cómo va su participación en el foro: qué nivel tiene en cada tema, qué calificación tiene en sus preguntas y respuestas, etc. Así, desde el menú principal, al pulsar el botón “Mi participación” simplemente aparece una nueva ventana, para la que no hay que recuperar información:

Figura 24. Participación del alumno

Una vez aquí, si el alumno selecciona “Mis preguntas”, igual que en los procesos descritos anteriormente, se envía un mensaje al servidor, se recuperan las preguntas de la base de datos y se crea el panel específico para mostrarlas:

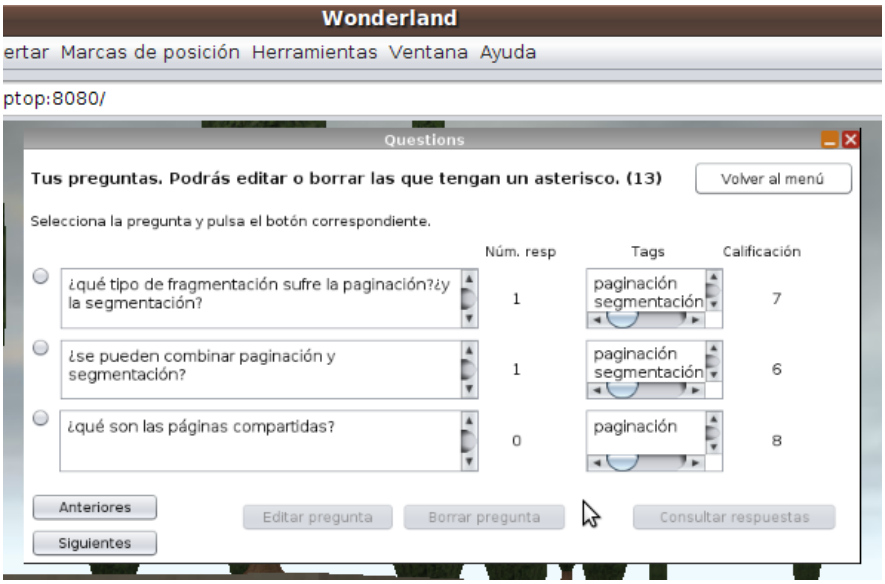


Figura 25. Preguntas formuladas por un alumno

Aunque esta ventana (Figura 25) trata de ser similar a la ventana de preguntas que se muestra en la búsqueda por *tags* (Figura 21), hay algunas diferencias, como que aquí sí se muestra la calificación y aparecen los botones "Editar pregunta" y "Borrar pregunta" en lugar del botón de "Respuesta nueva". No obstante, el funcionamiento es similar, sólo se activan los botones cuando la pregunta seleccionada cumpla los requisitos, bien que tenga respuestas, bien que pueda ser editada o borrada. Lo segundo se permite siempre que la pregunta no haya sido calificada ni respondida (podría ocurrir que el profesor no le dé una nota a una pregunta, pero la valide para que aparezca en el foro ante los demás alumnos, posibilitando así que tuviera respuestas aunque no estuviera calificada).

En el caso de que el alumno seleccione "Mis respuestas" la ventana que obtendrá será similar a ésta:

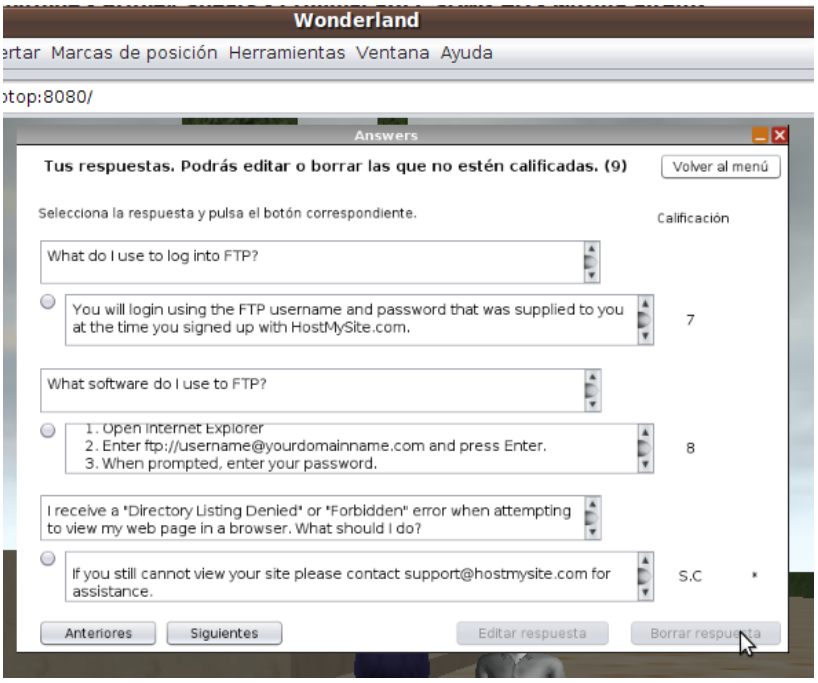


Figura 26. Lista de respuestas de un alumno

Esta ventana la única peculiaridad que posee es que encima de cada respuesta dada por este alumno aparece la pregunta a la que corresponde. Para que no haya lugar a equívoco, la respuesta está indentada y precedida por un *radiobutton* para poder seleccionarla. Siguiendo una lógica similar al caso anterior, una respuesta puede ser editada o borrada sólo cuando no está calificada (aparecerán las siglas S.C. como se observa en la Figura 26).

Por último, el alumno puede observar sus promedios, es decir, el número de preguntas y respuestas que ha formulado de cada tema, así como su nota media y nivel correspondiente (bajo, medio o experto). El aspecto de este panel es similar al siguiente:

Figura 27. Promedios de un alumno por temas

Como una pregunta o respuesta puede estar relacionada con más de un tema, inevitablemente en estos promedios se contará más de una vez; con lo cual, la suma de preguntas o respuestas de todos los temas no tiene por qué corresponderse con el total: éste puede observarse en las ventanas correspondientes entre paréntesis al lado del título.

Cualquier ventana puede cerrarse por el procedimiento habitual, en el aspa de la esquina superior derecha. Esta acción tiene como resultado la interrupción del proceso que estuviera en marcha, es decir, si, por ejemplo, el alumno estaba introduciendo una nueva pregunta y cierra la ventana antes de darle al botón "Guardar", la pregunta no se almacena en la base de datos.

5.3. Interacción por iniciativa del *Oráculo*: *Fantasma*

En este apartado se describe la otra parte de la interfaz del alumno, una aportación novedosa al foro que tiene como fin fomentar la participación de los alumnos mediante un algoritmo que, de forma automática, les puede plantear preguntas no respondidas al entrar éstos en el mundo virtual.

Siguiendo el mismo patrón que en el apartado anterior, se analizarán paso a paso los procesos y algoritmos que tienen lugar internamente y los resultados visibles para el alumno en el orden que se van sucediendo.

En primer lugar, al entrar el alumno en el mundo, se toma su *login* y se comprueba si está en la base de datos (el proceso sigue los mismos pasos descritos hasta ahora: el *controlador* captura la entrada del alumno y se comunica con el *modelo*, el cual realiza una serie de acciones dependiendo del resultado de la consulta). En caso negativo, simplemente no se hace nada más. En caso positivo, el siguiente paso es averiguar qué preguntas no tienen respuesta y cuál es el nivel del alumno en cuestión en los temas (*tags*) con los que están relacionadas.

Para saber si una pregunta no tiene respuestas es necesario mirar dos campos en la tabla de preguntas: el de respuestas válidas y el de respuestas pendientes de validación; ambos deben estar a 0.

El algoritmo es el siguiente:

- Primero se debe motivar a aquellos alumnos que no suelen participar
- Después a los que sí participan, pero sus resultados no son muy buenos
- Posteriormente, si ninguno de estos grupos ha sido capaz de responder, se acude a un alumno "experto", suponiendo que, si tiene buenos resultados normalmente, tendrá más capacidad para responder, aunque la pregunta sea difícil.
- Finalmente, en caso de que ni siquiera los expertos sean capaces de responder, se entiende la pregunta puede estar mal planteada y, por tanto, deber ser el profesor quién la revise (este último caso tiene lugar en su interfaz correspondiente, con lo que no afecta al algoritmo de esta parte).

Para controlar cuántas veces se pregunta a cada grupo de alumnos y cuándo se considera que éstos tienen *malos resultados* o son *expertos*, existe una tabla de configuración en la que cada profesor puede introducir los parámetros que considere correctos para su asignatura (en el capítulo siguiente se explicará cómo). Por otra parte, para que este algoritmo funcione, el sistema se encarga de almacenar en la base de datos información como cuántas veces ha sido planteada una pregunta y a qué alumnos (de forma que no se le pregunte siempre al mismo si entra varias veces seguidas).

Así, el *modelo* se encarga primero de recuperar toda esta información, es decir:

- Preguntas sin responder (en la misma tabla aparece cuántas veces ha sido preguntada cada una) que no hayan sido planteadas a este alumno
- Nivel de este alumno en cada tema
- Parámetros de configuración (veces que se pregunta a cada tipo de alumno)

y después de enviársela al cliente para que la procese (de ahí que en el primer apartado se comentase que ciertas partes del cliente también forman parte del *modelo*). Aunque, de esta forma, quizás se estén enviando más datos de los necesarios a través de la red (puede que el algoritmo en el cliente descarte varias de las preguntas), es el precio que hay que pagar para aliviar la carga del servidor, quien procesa muchas peticiones a la vez, por una sola por cliente.

Una vez recibido el mensaje, tienen lugar las siguientes acciones:

- Se crean en memoria dos vectores: uno de *tags* y otro del correspondiente nivel del alumno en cada uno de ellos (se relacionan por los índices).

- Se guardan también las variables del número de veces que hay que preguntar a cada grupo de alumno; llamémoslas:
    - A: número de veces que hay que preguntar a alumnos que no han respondido nunca
    - B: número de veces a alumnos cuyas respuestas han sido malas
    - C: número de veces a *expertos*

- Para cada pregunta, se recorren todos sus *tags* y se ve si para, al menos, uno de ellos, se cumple:
    - El nivel en el *tag* es "*nunca ha respondido*" y la pregunta ha sido formulada menos de A veces.
    - El nivel es "*ha respondido mal*" y la pregunta ha sido formulada entre A y A+B veces
    - El nivel es "*experto*" y la pregunta ha sido formulada entre A+B y A+B+C veces

- Si no se da ninguno de esos casos para ningún *tag*, la pregunta se descarta. Cuando ya se han procesado todas, si existen resultados para mostrar al alumno, aparecerá un mensaje en pantalla y se *renderizará* un fantasma detrás del avatar:

Figura 28. Interacción por iniciativa del "Oráculo": fantasma

Mientras el alumno no decida mirar las preguntas pendientes, el *fantasma* seguirá al avatar conforme se vaya desplazando por el mundo virtual. Para poder implementar esta funcionalidad, inicialmente se pensó en utilizar un nuevo objeto 3D que fuera obteniendo constantemente las coordenadas del avatar y adaptando su posición en consecuencia. Pero se observó que esta opción no es viable porque produce una gran sobrecarga en el sistema, además de plantear otra serie de inconvenientes. Entre ellos, el hecho de que el fantasma sólo podría seguir a un alumno cada vez, ya que los objetos son celdas y, por tanto, su representación en el servidor es única para todos los clientes. Esto significa que, si un objeto cambia su posición, todas las réplicas de los distintos clientes tienen que reproducir ese cambio para mantener la coherencia.

La solución encontrada finalmente resuelve el problema de forma mucho más sencilla: en lugar de usar un objeto 3D, se utiliza un modelo, es decir, una simple imagen 3D, pero que no tiene poder de interactuación. Los modelos sólo se cargan en el cliente, no tienen contraparte en el servidor, de manera que cada alumno tiene su propio fantasma y sólo lo visualiza él. Además, el modelo 3D puede ser añadido como nodo hijo del avatar. Éste es un concepto propio de motores gráficos, en los que los elementos que se visualizan se definen como nodos o entidades que forman un árbol; por ejemplo, una rama típica podría ser una silla dentro de una habitación, que a su vez está dentro del mundo virtual. El mundo sería el nodo padre, la habitación un nodo hijo de éste y la silla, a su vez, un nodo hijo de la habitación. Esto implica que uno contiene a otro y, si cambia su posición, cambiarán con ella las posiciones de los elementos que albergue. Así, abstrayendo esta idea, podemos obtener de forma sencilla y, sin sobrecargar el sistema, la sensación de que el fantasma está siguiendo al avatar.

Este mecanismo del “fantasma” se activa de forma periódica, gracias a un temporizador que se da de alta en el cliente al iniciar sesión. Así, si durante el tiempo que el alumno está en el mundo, nuevas preguntas son validadas por el profesor, la aplicación puede detectarlo y poner en marcha el algoritmo. Los pasos en este caso serían exactamente los mismos que se han descrito anteriormente. Es necesario usar este mecanismo de *polling*, en lugar de recurrir a interrupciones,

porque la interfaz del profesor es externa y su única vía de comunicación con la del alumno es a través de la base de datos.

Con todo esto, se puede apreciar que es una lógica suficientemente compleja como para requerir una separación de toda la interacción normal (por iniciativa del alumno). No obstante, necesitan ser compatibles y compartir ciertas partes del código, ya que la base de datos es la misma para las dos y, por otra parte, parece redundante redefinir la lógica, por ejemplo, de responder a una pregunta.

Por esta razón, se definen dos canales de comunicación separados[4], con sus respectivos *ClientConnection* y *ClientConnectionHandler* (ver primera sección del capítulo), cuya lógica de actuación al recibir un mensaje es completamente distinta. Sin embargo, el resto de los componentes de la aplicación son compartidos, los procesos que se han descrito hasta ahora en las otras secciones del capítulo, también se aplican aquí: el tipo de mensaje es el mismo, las clases que controlan la base de datos, el controlador de la interfaz gráfica, etc.

Así, cuando el alumno pulsa "Ver preguntas" en la ventana que acompaña al fantasma, ambos desaparecen y, en su lugar, se muestra una ventana en la que se visualizan las primeras preguntas seleccionadas. En este caso, no es necesario pasar por el servidor para recuperarlas, ya que las preguntas las tenemos en el cliente como se ha explicado anteriormente. No obstante, sí hay que enviarle un mensaje al servidor para que se actualicen en la base de datos los campos que controlan el número de veces que se ha formulado cada pregunta y a qué alumnos. Sólo se actualizan las preguntas que, efectivamente, el alumno está visualizando en ese momento, por si en cualquier momento cierra la ventana y sólo ha visto parte de la lista. Así, será cada vez que se pulse el botón "Siguiente" para moverse por las preguntas, cuando se enviará uno de estos mensajes de actualización.

El aspecto de este panel de preguntas es similar a los anteriores, la única novedad que incluye es que al lado de la pregunta aparece el nivel del alumno en el tema relacionado:

[4] La creación de ambos canales es independiente de que exista un objeto en el mundo con *capacidad* de "Oráculo", lo cual viene impuesto por el protocolo de comunicaciones. En el caso del canal principal, no es un problema, ya que la comunicación comienza al interactuar el alumno con el "Oráculo" y esto no es posible si no existe. Sin embargo, en el canal del "fantasma" la interacción la comienza el sistema, por lo que ha sido necesario buscar un mecanismo para detectar si existe o no un "Oráculo" en el mundo. Esto se ha logrado aprovechando las características de persistencia de *OpenWonderland*, asignando una clave textual al objeto "Delfos-Component" en la creación y comprobando la existencia de esta clave siempre antes de iniciar la comunicación en el canal del "fantasma".

Figura 29. Preguntas para que el alumno responda

Como se puede observar, el procedimiento para responder a una de las preguntas, es idéntico a los ya mencionados: se selecciona la pregunta en cuestión, el botón “Responder” se activa entonces y, si lo pulsamos, nos aparece una ventana donde introducir texto. En este momento, internamente se ha producido un cambio de canal, ya que el proceso de responder es idéntico en ambos casos, con lo que se aprovecha el código ya implementado.

Por otra parte, volviendo a la ventana inicial que acompaña al fantasma, vemos que existe otro botón llamado “Desactivar avisos”. Éste le permite al alumno bloquear el fantasma para que no aparezca durante la sesión actual, si prevé que va a estar realizando otras tareas que requieran no ser interrumpido. Finalmente, si el alumno no quiere responder a ninguna pregunta en un momento concreto, pero quiere que el fantasma permanezca como recordatorio de que en algún momento debe hacerlo, basta con cerrar la ventana de avisos por el procedimiento habitual (aspas en la esquina superior).

6. Interfaz del profesor

Igual que en el capítulo anterior se ha analizado la interfaz del alumno, en este se describirá la del profesor, que está implementada en *PHP* y *Javascript* para simplificar su diseño y desarrollo. Ya que ésta es una interfaz secundaria, el presente capítulo se centrará en describir su funcionalidad y las distintas pantallas de que consta, sin que requiera un análisis tan exhaustivo como la interfaz del alumno. En todo el diseño se ha intentado imitar a plataformas más conocidas y usadas para que la interactuación sea más sencilla e intuitiva.

6.1. Estructura

PHP es un lenguaje de programación que facilita la creación de páginas web. Permite tanto orientación a objetos como programación procedural, por lo que se ha aprovechado esta ventaja y ambos paradigmas se utilizan en esta aplicación. Existen cuatro tipos diferentes de ficheros:

- Para la comunicación con la base de datos, se define una clase llamada *db.php*. Todos los accesos a los datos, tanto para consulta como para modificación, están encapsulados en los métodos de esta clase. Además, los atributos de esta clase son los datos de la cuenta MySQL (*login* y *contraseña*) y el nombre de la base de datos a utilizar. Estos tres atributos se recuperan de un fichero creado en la inicialización del *Oráculo* por el profesor.

- Por otra parte, existe otro fichero genérico en el que están las principales funciones que se utilizan con frecuencia en la aplicación; en este caso se trata de programación procedural. Básicamente, son funciones para definir un menú lateral, para estructurar las tablas donde se muestran preguntas y respuestas o para guardar las calificaciones y validaciones.

- El otro tipo de fichero genérico presente en la aplicación es la hoja de estilos (css) que permite definir todo lo relativo a la visualización de la interfaz, de manera que contenido y forma se separan como es habitual en la mayoría de aplicaciones web (*definición de forma* en la Figura 30).

- Todos los demás ficheros contienen código escrito en *html, PHP* y *Javascript* para implementar la funcionalidad (*definición de contenido* en la Figura 30). Los hay de varios tipos:

    - Páginas de bienvenida: la ventana de inicio, en la que el profesor debe autentificarse, y otra que se muestra al entrar, con las estadísticas generales de la aplicación y los datos de configuración (nota para ser experto, veces que se pregunta a cada tipo de alumno…).

    - Páginas interactivas: muestran una serie de preguntas o respuestas según el criterio elegido (que sean recientes, que estén sin responder…). Cada uno de estos ficheros define su búsqueda concreta y luego utiliza las funciones genéricas para que todas las páginas tengan un mismo aspecto.

- Ventanas *pop-ups*: permiten borrar o editar cierta información (de un alumno, pregunta o respuesta, pero sólo en determinados casos). También muestran la información más completa y detallada de preguntas y respuestas.

La siguiente figura muestra la estructura descrita en los párrafos anteriores de forma más sintetizada y mostrando las relaciones entre los distintos ficheros:

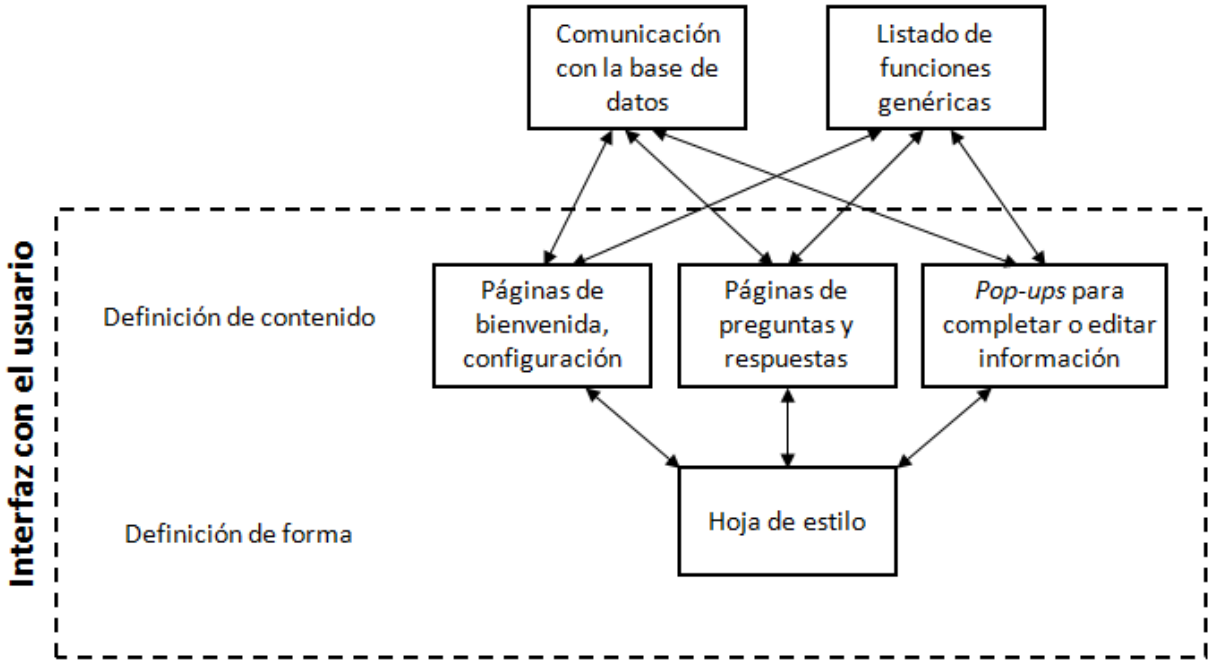


Figura 30. Esquema de la estructura de la interfaz del profesor

En cuanto al uso de *Javascript*, en general su función es controlar el lanzamiento de estos *pop-ups*, asegurarse de que el profesor no abandona una página sin guardar los datos y dirigir la mayoría de enlaces de la aplicación a la ventana correspondiente.

6.2. Uso de la aplicación

A continuación se irán analizando las diferentes pantallas de que consta, explicando qué ofrecen, cómo interactuar con ellas y, muy brevemente, cómo están implementadas.

En primer lugar, el profesor tendrá que introducir sus datos de acceso y, una vez validados contra la base de datos, accederá a la primera página de la interfaz, la página de bienvenida. En ella, lo primero que puede observarse es un menú lateral, que aparece en todas las páginas para poder navegar de forma más sencilla a través de ellas. El menú consta de los siguientes enlaces:

- Página inicial o de bienvenida
- Lista de estudiantes
- Lista de *tags*
- Búsqueda por *tags*
- Preguntas sin calificar
- Respuestas sin calificar
- Preguntas nuevas
- Respuestas nuevas
- Preguntas sin respuesta
- Preguntas que los alumnos no son capaces de responder
- Lista completa de preguntas
- Lista completa de respuestas
- Hacer una nueva pregunta
- Cerrar sesión

A continuación, se explican las distintas páginas mencionadas en el orden en que aparecen en el menú.

6.2.1. Página de bienvenida

Ésta es una página diseñada para mostrar las estadísticas más importantes del *Oráculo*, así como la información más relevante de configuración. Esto se presenta en distintas tablas:

- Primero las de estadísticas, tales como: cuántas preguntas y respuestas hay en total, cuántas desde la última vez que se entró y cuántas preguntas siguen sin responder. En estas tablas, los números son enlaces que llevan a las páginas correspondientes (seguir estos enlaces es idéntico a seguir los del menú).

- Y después las de configuración, en las que aparece el valor por defecto en un cuadro de texto que permite cambiarlo. Son dos:
    - La primera permite definir parámetros relacionados con el nivel de los alumnos. El profesor puede decidir, por un lado, a partir de qué nota media en un tema se considera a un alumno "*experto*" y cuántas respuestas tiene que haber dado como mínimo y, por otro, por debajo de qué nota se considera que un alumno ha respondido mal (ver Interacción por iniciativa del *Oráculo*: *Fantasma– Interacción por iniciativa del Oráculo: Fantasma*).

    - La segunda permite configurar otros parámetros del algoritmo del *fantasma*, esto es, el número de veces que hay que formular una pregunta sin respuesta a cada tipo de alumnos (primero a los que no hayan respondido nunca, después a los que hayan respondido mal y, finalmente, a los considerados expertos en el tema).

Para guardar los cambios es necesario pulsar un botón al final de la página, llamado "Grabar". Si el profesor intenta salir de la página sin pulsar este botón, habiendo realizado algún cambio, saltará un aviso en una ventana, generado por una función de *Javascript,* para evitar que se pierdan las

modificaciones. Éste es un mecanismo que se incluye en todas las páginas en las que se guarda información.

En la Figura 31 se pueden observar las tablas de estadísticas (las dos primeras) y de configuración (las dos últimas).

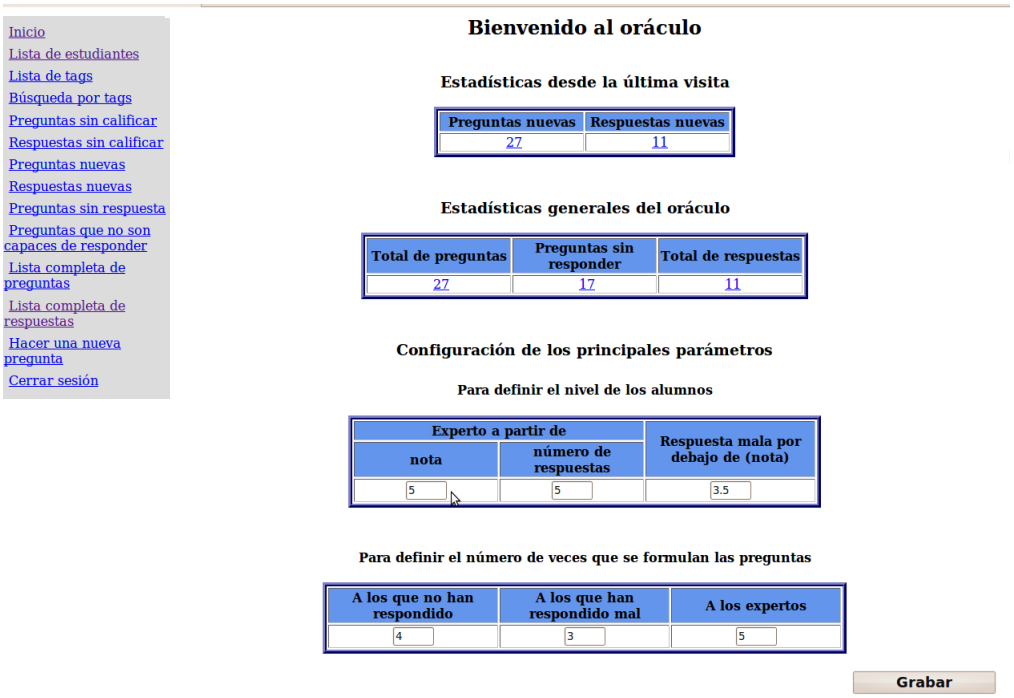


Figura 31. Página inicial interfaz del profesor

6.2.2. Lista de estudiantes

Esta página permite al profesor ver la lista de alumnos de la asignatura, así como quiénes de ellos son expertos en qué temas. También permite añadir estudiantes si en la etapa de configuración no se introdujeron todos (ver sección 4.2.2, *Datos iniciales*) o ha habido algún cambio posterior. En la siguiente imagen se puede observar su aspecto:

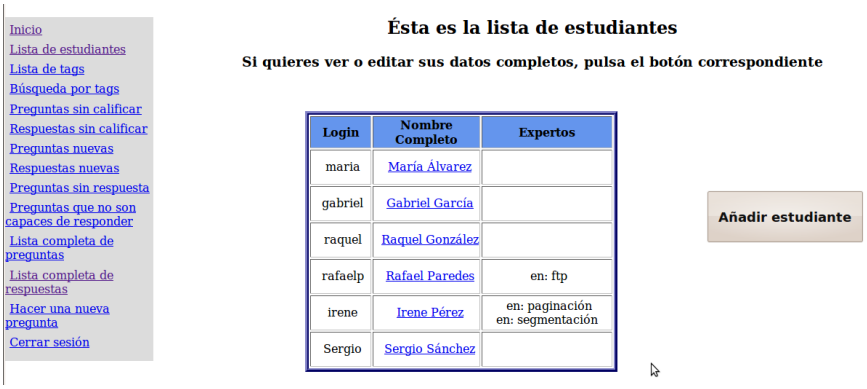


Figura 32. Lista de estudiantes

En la lista de estudiantes, el nombre de cada uno es un enlace que lleva a una página con información de su progreso: se puede ver cuántas preguntas ha hecho, cuántas no ha sabido responder y cuántas respuestas ha dado (en los tres casos, los números son enlaces que llevan a las listas correspondientes). También incluye una tabla con las estadísticas para cada tema: número de preguntas y respuestas formuladas, nota media y nivel en cada uno.

Un ejemplo de página de un estudiante podría ser:

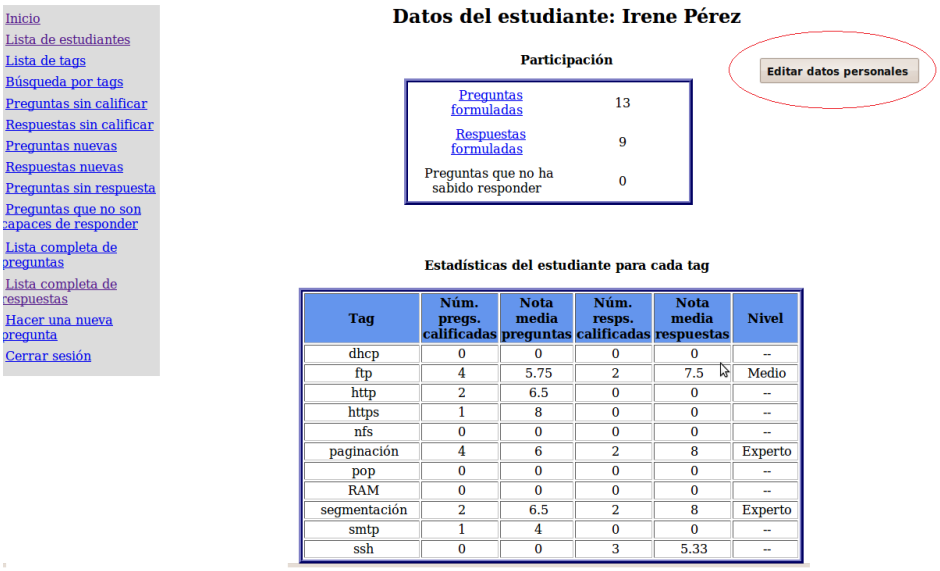


Figura 33. Página personal de un alumno

Como se observa en la figura, rodeado en rojo, otra acción que permite la página de cada estudiante es editar sus datos personales mediante una ventana *pop-up* que aparece al pulsar el botón. Una peculiaridad es que siempre se van a poder editar el nombre y apellidos, pero no el *login*. Éste último sólo se puede cambiar si el alumno nunca ha participado en el foro, ya que es el identificador que se utiliza en todas las tablas de preguntas, respuestas, expertos… Por otra parte, este criterio también se sigue a la hora de borrar un alumno: sólo en el caso de que éste nunca haya participado, aparecerá el botón correspondiente:

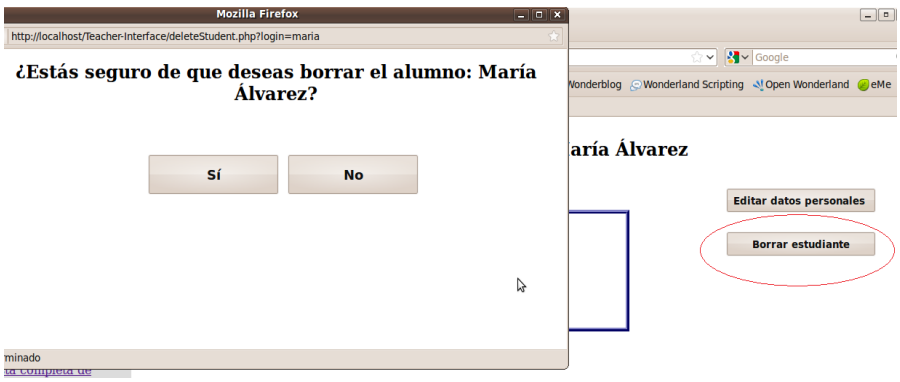


Figura 34. Borrado de un estudiante cuando está permitido

6.2.3. Lista de etiquetas (*tags*)

Consiste en una tabla en la que se muestra la lista de *tags* existentes en la base de datos y que se acompaña de unos enlaces para editarlos o borrarlos en el caso de que no hayan sido usados todavía.

Asimismo, también incluye un recuadro para añadir un nuevo *tag*, que se guardará al pulsar el botón correspondiente. El aspecto de esta página se puede observar en la siguiente figura:

Figura 35. Lista de *tags*

Los enlaces de editar y borrar son algunos de los ejemplos que están implementados con *Javascript*, ya que no tienen simplemente que redirigir a otra página, sino que actúan como los botones de un formulario, ya que se requiere pasar información entre páginas, en este caso el nombre del *tag*.

Cuando se pulsa editar, el nombre del *tag* se sustituye por un recuadro de texto donde se puede escribir el nuevo valor. Una vez hecho, habrá que bajar por la página hasta el final para pulsar el botón “Grabar”. En cuanto a borrar, si se pulsa, aparecerá una ventana *pop-up* similar a la de borrar un alumno. En ambos casos, cuando se elige el “Sí”, se elimina el elemento de la base de datos, se cierra el *pop-up* y se actualiza la página con la lista para comprobar que el cambio se ha hecho efectivo.

En todas las páginas y para todos los tipos de datos se sigue siempre el mismo proceso para guardar o eliminar información, de forma que el sistema sea más intuitivo. El funcionamiento también es el mismo para todas las páginas cuando se intenta salir de la página sin guardar habiendo hecho alguna modificación: siempre salta un mensaje de aviso.

6.2.4. Búsqueda por *tags*

Al igual que los alumnos pueden acudir al foro para buscar qué preguntas y respuestas hay relacionadas con unos temas específicos, el profesor puede querer obtener una información similar y así controlar, por ejemplo, en qué temas los estudiantes se desenvuelven mejor y en cuáles hay menor participación.

El funcionamiento consiste en recuperar los *tags* existentes de la base de datos y se mostrarlos en un recuadro junto con un *checkbox* al lado de cada uno para seleccionar los que se quieran (no hay límite en el número).

Una vez que el profesor ha elegido los *tags*, pulsa el botón correspondiente y se realiza la búsqueda de las preguntas en la base de datos. En este caso, como este tipo de aplicación *web* no tiene tantas limitaciones en cuanto a velocidad o envío de datos como *OpenWonderland*, no hace falta incluir en la búsqueda aquellas preguntas que estén relacionadas sólo con algunos de los *tags* (ver sección 5.2, *Interacción por iniciativa del alumno*). Por esta razón, los resultados serán únicamente los que cumplan el criterio completo.

Si hay preguntas que mostrar, aparecerá una tabla debajo mostrándolas como se puede observar en la figura:

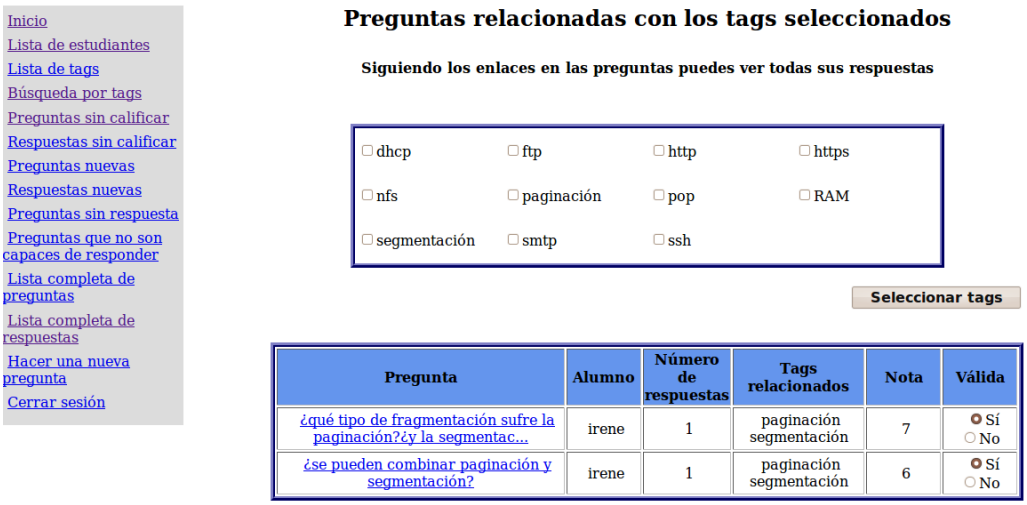


Figura 36. Búsqueda de preguntas por tags

Esta tabla se construye utilizando una función del fichero genérico. Así, todas las tablas de preguntas y respuestas que existen en las otras páginas se hacen de la misma forma, aunque eso implique que a veces incluyan información redundante. Pero es el coste que hay que pagar para que haya cierta uniformidad y así el uso sea más intuitivo y no requiera aprenderse cada pantalla específica.

Otro detalle importante es que el texto de la pregunta aparece sesgado y en un enlace. Esto se ha hecho así para evitar que las tablas sean demasiado largas y no permitan a simple vista hacerse una idea general. Por ello, se limita a dos líneas de texto y se incluye el enlace para completar la información de cada una si se quiere, datos que aparecerán en una ventana *pop-up* (ver sección 6.2.10, *Información completa de preguntas y respuestas*).

6.2.5. Preguntas y respuestas sin calificar

Otras dos pantallas que se ofrecen al profesor son las correspondientes a las preguntas y respuestas que aún siguen sin calificar, en las cuales aparecerá un cuadro de texto en lugar de la nota numérica para que el profesor pueda introducir el valor que considere.

Una aclaración importante, que ya se ha mencionado en capítulos anteriores, es que tanto preguntas como respuestas tienen que pasar dos procesos distintos y separados: el de validación y el de calificación. Una pregunta o respuesta válida significa que es candidata a ser mostrada a los alumnos, porque el profesor ha considerado que su contenido es apto o que enriquece la asignatura. Y esto puede ocurrir independientemente de la nota que tenga. Por otra parte, una pregunta puede estar calificada pero no ser válida, por ejemplo, por repetir conceptos de otra o por estar mal planteada. No obstante, el hecho de que no se muestre a otros alumnos, no significa que no valga como mecanismo de evaluación para el profesor, de ahí que calificación y validación sean completamente independientes.

El funcionamiento en todas las páginas de visualización de preguntas y respuestas es el mismo:

- Cada página define su criterio de búsqueda específico (sin calificar, sin responder, etc) y realiza la consulta a la base de datos.
- Una vez recuperado el conjunto de preguntas o respuestas de la base de datos, se almacenan en un vector y se le pasan a una función genérica para que construya la tabla.

Así, la forma de presentar los datos es igual para todos los casos. Esto también simplifica el funcionamiento de la calificación y la validación de preguntas y respuestas: también se definen algoritmos genéricos, de forma que en todas las páginas el proceso es el mismo. De esta forma, para calificar, por ejemplo, una pregunta, el profesor no tiene por qué acceder necesariamente a la página de preguntas sin calificar: si la ha encontrado como resultado de la búsqueda por *tags*, también puede poner la nota desde esa página.

Un ejemplo de visualización de preguntas sin calificar podría ser:

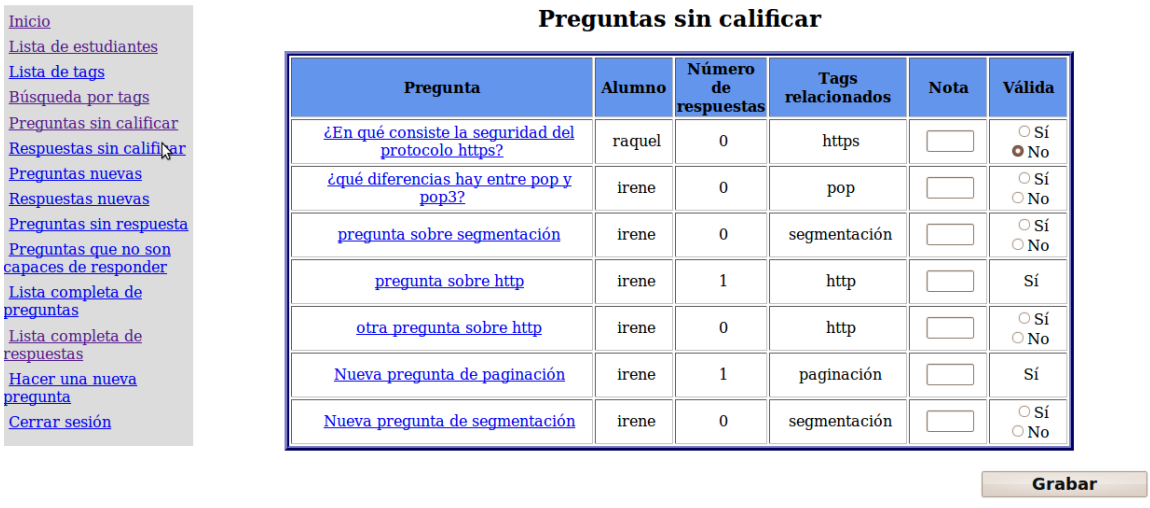


Figura 37. Preguntas sin calificar

En la figura se observan los cuadros mencionados para introducir la calificación y los *radiobuttons* para la validación (una pregunta puede estar "pendiente de validación" – cuando ninguno está seleccionado-, ser "válida" – estará seleccionado el *radiobutton* del "Sí"- o ser "no válida" – estará seleccionado el "No"). Siempre se puede cambiar una pregunta a "válida", pero no se podrá invalidar si ya tiene respuestas, caso en el que, en lugar de los *radiobuttons*, aparece la palabra "Sí" en el campo *válida*.

Una vez que el profesor introduzca las notas o seleccione las preguntas que quiera validar o invalidar, deberá pulsar el botón "Grabar" al final de la página para almacenar los cambios. De nuevo, si intenta salir antes de hacerlo, le saldrá una ventana de aviso.

La calificación introducida debe ser un dato numérico, aunque no se restringe la escala. Debe ser así en aras a poder calcular las medias y, con ellas, los niveles de los alumnos. Si se introduce cualquier otro carácter, el sistema lo detectará y mostrará un mensaje de error en la página.

Una vez guardados los cambios, las preguntas calificadas ya no aparecerán más en esta página, pero se podrán consultar en otras como "Búsqueda por *tags*" o "Lista completa de preguntas", donde, en lugar del recuadro, aparecerá ya la nota asignada.

Por otra parte, en cuanto a las respuestas sin calificar, se les pueden aplicar todos los conceptos vistos para las preguntas, el único cambio en la interfaz es que en este caso, existe una nueva columna en la tabla y ésta es la pregunta a la que se está respondiendo.

Al igual que ocurre con las preguntas, las respuestas también aparecen truncadas para simplificar la vista general. Para completar su información, basta con seguir el enlace en el texto (ver sección 6.2.10, *Información completa de preguntas y respuestas*).

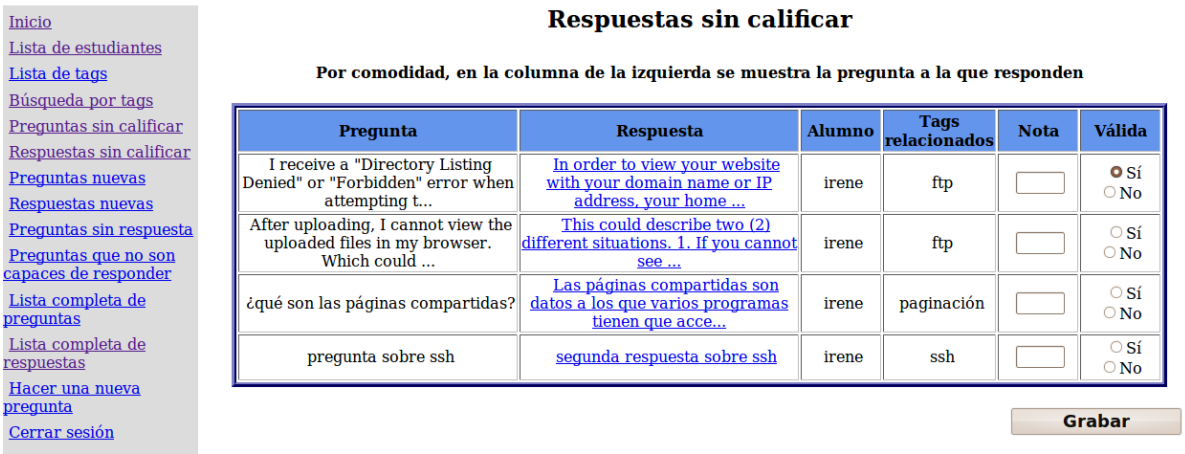


Figura 38. Respuestas sin calificar

6.2.6. Preguntas y respuestas nuevas

Esta sección es similar a la anterior, la única peculiaridad es el concepto de "nuevas", que se explica a continuación.

“Nuevas” se refiere a preguntas y respuestas que el profesor todavía no ha podido ver porque han sido formuladas con posterioridad a la última vez que entró en la interfaz. Para controlar esto, en la base de datos se almacena la fecha de la última sesión del profesor. También se almacena un registro temporal de cuándo fueron hechas tanto preguntas como respuestas.

De esta forma, basta comparar ambos datos para saber cuáles deben ser mostradas en esta página.

6.2.7. Preguntas sin respuesta y preguntas que los alumnos no son capaces de responder

Al igual que la sección anterior, ésta sólo pretende esclarecer la diferencia entre ambos tipos de preguntas sin respuestas, ya que la visualización y las acciones que se pueden realizar sobre las preguntas son idénticas a las del 60

Así, “preguntas sin respuesta” engloba todas aquellas que todavía no tienen respuestas, ni válidas, ni pendientes de validación, pero no existe ningún tipo de restricción en cuanto al número de alumnos a los que ya se les han formulado.

Sin embargo, la sección “preguntas que los alumnos no son capaces de responder” es la terminación del algoritmo del *fantasma* (ver sección 5.3, *Interacción por iniciativa del Oráculo: Fantasma*) y contiene sólo aquellas pregunta que ya han sido formuladas las veces especificadas en la configuración a los tres tipos de alumnos: los que no han respondido nunca en esos temas, los que han respondido mal y los expertos. Por tanto, éstas se consideran ya preguntas “para el profesor”, ya que se entiende que debe haber algún problema con ellas cuando ningún alumno ha sido capaz de responderlas.

6.2.8. Lista completa de preguntas y lista completa de respuestas

Finalmente, para visualizar todas las preguntas y respuestas presentes en la base de datos, se proporcionan estas páginas en las que se incorporan nuevos enlaces para facilitar la navegación. Ambas son similares, la única diferencia es que, en el caso de las respuestas, la tabla contiene una columna más para albergar la pregunta a la que se responde. Así, lo que se explique en los siguientes párrafos sobre la página de preguntas aplica también para la de respuestas.

Las páginas explicadas en apartados anteriores muestran un subconjunto muy reducido de las preguntas del foro, con lo cual no se hace necesario restringir el número de ellas que se visualizan por página. Sin embargo, en este caso podemos encontrarnos con una consulta muy grande a la base de datos, por lo que es bueno proporcionar un mecanismo que vaya recuperando las preguntas poco a poco y así evitar sobrecargas en el sistema.

Así, en esta pantalla se ofrecen ambas alternativas: visualizar todo el conjunto de preguntas seguidas o visualizarlas de 20 en 20. Para ello, basta con seguir el enlace que se encuentra en la esquina superior derecha y que permite cambiar de un modo a otro. Este tipo de enlace es otro de los ejemplos que se deben implementar en *javascript*, ya que requiere enviar datos de un formulario.

En el caso de que se quieran visualizar las preguntas por bloques de 20, para ayudar a la navegación se incorpora un nuevo conjunto de enlaces que son simplemente los números de página y los símbolos de anterior y siguiente.

En la siguiente figura podemos observar todo lo anterior:

Figura 39. Lista completa de preguntas

Como se observa, por lo demás ésta vuelve a ser una página similar a las anteriores.

6.2.9. Hacer una nueva pregunta

Finalmente, el último enlace que aparece en el menú es el que permite al profesor formular una nueva pregunta, ya que puede darse el caso de que quiera evaluar las respuestas que den los alumnos a una duda concreta o simplemente introducir preguntas inicialmente para que el foro empiece a funcionar con algo de fluidez.

Estas preguntas se consideran “maestras” y, por tanto, el formularlas implica automáticamente su validación y calificación con la nota más alta. Además, para contemplar el caso de que hubiera más de un profesor en la asignatura, el *login* relacionado con ellas es la palabra clave “profesor” y así evitar posibles confusiones.

Para formularlas se incluye un cuadro de texto y la lista de *tags* disponibles para poder seleccionar los que correspondan. Si la pregunta está en blanco o no hay ninguna etiqueta asociada, pulsar el botón “Guardar” no tendrá ningún efecto. Si se intenta salir antes de salvar los cambios, aparecerá un mensaje de aviso.

En la siguiente figura se observa la apariencia de esta página:

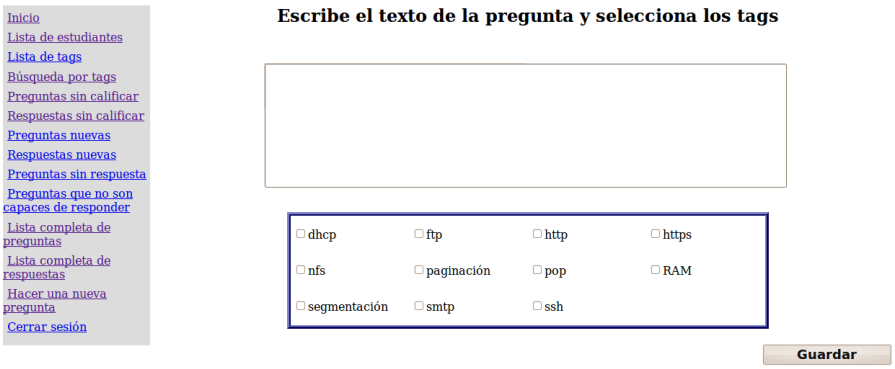


Figura 40. Pantalla para que el profesor pueda hacer una nueva pregunta

6.2.10. Información completa de preguntas y respuestas

En todas las tablas de preguntas y respuestas, éstas se han truncado y convertido en enlaces para simplificar la visualización inicial. Para ver toda la información completa, basta entonces con seguir dicho enlace.

En el caso de las preguntas, la ventana *pop-up* que aparece sería similar a ésta (en el caso de que la pregunta esté formulada por un alumno):

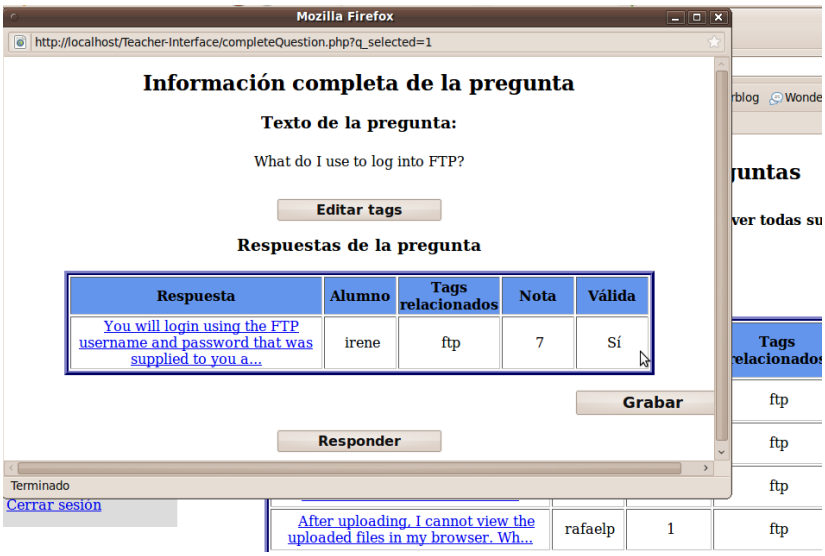


Figura 41. Información completa de una pregunta de un alumno

En la figura se observa inicialmente el texto de la pregunta, seguido de un botón para editar los *tags*, la lista de respuestas (donde la tabla es similar a las demás, sólo que esta vez no aparece la pregunta) y, finalmente, un botón para responder.

Esta tabla de respuestas, al ser idéntica en funcionalidad a las de las otras páginas, también permite validarlas y calificarlas si todavía no lo estuvieran.

Por otra parte, se incluye el botón “Responder” para el caso que se describe en la sección 6.2.7 (*Preguntas sin respuesta y preguntas que los alumnos no son capaces de responder*), sobre las preguntas que los alumnos no han sido capaces de responder, pero también por si el profesor considera necesario en algún momento aportar una respuesta “maestra” a alguna pregunta.

En cuanto al botón que resta, el de edición de *tags*, la razón de incluirlo es que son los propios estudiantes los que catalogan las preguntas con una serie de *tags*, los que ellos quieran, sin límite. El problema es que si todavía no tienen suficientes conocimientos, quizá un texto esté más relacionado con otro tema distinto o le sobren etiquetas. Por esta razón, el profesor tiene la capacidad de editar los *tags* asignados a una pregunta. Sin embargo, no se contempla que la edite porque las preguntas sirven para evaluar a los alumnos y al editarla, puede que la calificación dad pierda sentido. La solución es que si el profesor considera que esa pregunta no es lo suficientemente buena como para que sea visible al resto de alumnos, no debe validarla y, si lo desea, en su lugar puede plantear él una nueva que se adapte mejor a lo que busca.

En cuanto a las preguntas que hayan sido planteadas por el profesor, la ventana cambia ligeramente ya que, en ese caso, sí que puede modificar el texto de la misma (siempre y cuando no existan ya respuestas de los alumnos). Esto se permite para solucionar el caso en el que el profesor haya cometido algún error al plantear la pregunta.

Simplemente, se incluye un nuevo botón que, al ser pulsado, convierte el texto de la pregunta en un área de texto donde poder editarlo. Después hay que guardar los cambios:

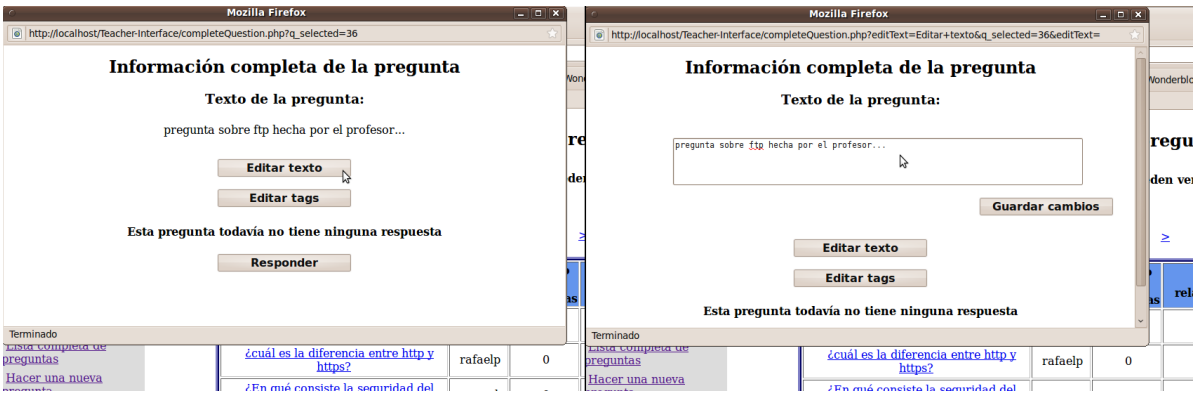


Figura 42. Información completa de una pregunta del profesor

Finalmente, la ventana *pop-up* donde se muestra la información completa de una respuesta es incluso más sencilla. En ella aparece el texto de la respuesta precedido de la pregunta a la que responde. En el caso de las respuestas, los *tags* no se editan ya que se heredan de la pregunta correspondiente; para editarlos, por tanto, habría que editar los de la pregunta.

Al igual que para las preguntas, en el caso de que sea una respuesta del profesor, se incluye un mecanismo de edición, cuyo funcionamiento es idéntico a lo descrito anteriormente.

7. Conclusiones y trabajo futuro

Como se ha discutido a lo largo de este trabajo, los mundos virtuales 3D suponen un gran avance en entornos educativos gracias a sus características de inmersión y colaboración principalmente, ya que el alumno puede interactuar con el entorno y con sus compañeros a través de su avatar, de forma que se crea una sensación de comunidad que favorece la motivación y el trabajo en equipo [**19**].

No obstante, ésta es un área todavía en fase de desarrollo, por lo que siguen existiendo desafíos como puedan ser: la creación de una serie de herramientas necesarias en todo entorno educativo; la presentación de contenidos dentro del mundo virtual de forma que sean visibles desde cualquier punto o bien que sólo los perciban algunos usuarios; mejorar la interacción por medio de interfaces diferentes del teclado o el ratón, etc.

Con este punto de partida, el foro 3D desarrollado constituye una innovación importante al no existir aún este tipo de herramienta, habitual en los entornos educativos 2D, para la plataforma de mundos virtuales elegida, *OpenWonderland*. Esta plataforma se encuentra todavía en fase de desarrollo, lo cual ha planteado diversos retos a la hora de llevar a cabo la implementación del foro.

En primer lugar, se ha proporcionado una metáfora atractiva para representar el foro en entornos 3D, como ha sido su personalización en un "Oráculo". Esto se ha hecho, además, teniendo en cuenta la flexibilidad y escalabilidad necesarias para este tipo de aplicación, implementando la funcionalidad como un componente que se puede añadir a cualquier objeto del mundo.

Por esta misma razón, se han separado las interfaces de alumno y profesor, atendiendo a las diferencias en la funcionalidad requerida para cada uno; en el alumno prima la motivación, la inmersión y la colaboración; mientras que el profesor necesita una herramienta más eficaz, que presente más información aunque sea menos atractiva.

Además, se ha conseguido sincronizar la actuación de ambas partes de la herramienta aunque sean independientes y estén realizadas en lenguajes diferentes, lo cual se sustenta en el diseño de un modelo de datos capaz de soportar toda la funcionalidad y el acceso desde ambas interfaces.

En ese sentido, se ha logrado un gran aporte a la comunidad de *OpenWonderland* en colaboración con uno de sus principales desarrolladores, *Jonathan Kaplan*, ya que, hasta la realización de este trabajo, no era posible acceder a una base de datos en *MySQL*, que es una tecnología ampliamente usada.

Por otra parte, se ha cuidado la presentación de los datos al cliente del mundo virtual (alumnos) utilizando para ello una ventana HUD – "*Head-up-display*" (ver *Glosario*), de forma que siempre estén visibles independientemente de la posición del avatar y que se garantice que cada alumno visualiza el resultado de su propia consulta al "*Oráculo*", sin que ello interrumpa las de los demás. Además, se proporcionan sendos recursos para que la interfaz del alumno pueda configurarse tanto en inglés como en español.

Aparte de todo el diseño e implementación de la parte central del foro, que se corresponde con la funcionalidad habitual de este tipo de aplicación, se ha buscado un mecanismo innovador para fomentar la participación de los alumnos. Esto ha supuesto desde la definición de unos criterios y el correspondiente algoritmo para la selección de preguntas y estudiantes, hasta la elección de una metáfora visual acorde con el mundo virtual 3D, consistente en un *fantasma* que persigue al avatar del alumno.

La implementación de este mecanismo se ha realizado teniendo en cuenta la arquitectura del motor gráfico utilizado por *OpenWonderland*: *JMonkeyEngine*, el cual estructura en contenido visual en forma de árbol, donde cada nodo representa un objeto en el mundo virtual y puede contener y estar contenido en otros nodos. Así, se ha aprovechado esta cualidad para definir el *fantasma* como nodo hijo del avatar del alumno, de forma que siempre se mueve con él automáticamente. Así, se evitan el coste computacional y la latencia de un algoritmo que estuviera continuamente tomando las coordenadas del avatar y borrando y redibujando el *fantasma* en consecuencia.

Finalmente, se han definido los correspondientes canales de comunicación entre clientes y servidor, capaces de gestionar el algoritmo del *fanstasma* de manera independiente a la interacción habitual (por iniciativa del alumno). La importancia de esto reside en que el canal del *fantasma* debe existir aunque el avatar no se encuentre en un entorno cercano al *Oráculo*, pero solamente si dicho *Oráculo* está dado de alta en alguna parte del mundo.

En definitiva, el trabajo realizado supone una nueva aplicación completa que aporta una visión algo diferente de un foro para entornos educativos, tratando de hacer el aprendizaje más dinámico y colaborativo, lo cual se ha demostrado que mejora los resultados. El diseño ha incluido desde la definición de la funcionalidad hasta la estructura del modelo de datos necesario para sustentarla, pasando por la elección del protocolo de comunicaciones y de la visualización de los datos en pantalla. Todo el diseño y su posterior implementación se ha basado, por una parte, en un análisis de las prestaciones que pueden buscar en un foro ambos tipos de usuarios – alumnos y profesores – y, por otra, en las restricciones impuestas al hallarnos dentro de una plataforma concreta de mundos virtuales, con una arquitectura específica.

No obstante, como en el caso de cualquier aplicación que se desarrolle, existen muchas vías para mejorarla y, parte de ese proceso debe hacerse una vez que se empiece a utilizar la herramienta y se puedan observar realmente cuáles son las necesidades de alumnos y profesores.

En ese sentido, es muy favorecedor el hecho de estar trabajando con una plataforma de código abierto, ya que nos permite incluir el módulo desarrollado como parte del proyecto *OpenWonderland* y así conseguir que personas de todo el mundo puedan beneficiarse de su uso, a la vez que lo mejoren aportando nuevas ideas.

7.1. Trabajo futuro

El primer paso para poder continuar con el trabajo realizado sería probarlo en un entorno similar al de su uso habitual, inicialmente con un grupo reducido de alumnos y un profesor que evalúen las prestaciones, la funcionalidad y la utilidad de la herramienta.

Una vez hecha esta prueba y un análisis de los resultados, hay diversas mejoras que se pueden hacer al código, por ejemplo:

- Mejora de la vista de la aplicación: forma de presentar la información dentro del *HUD*, diferentes menús, más ventanas con información más completa, etc.

- Creación de más de un *Oráculo* por mundo, por ejemplo, uno por asignatura, para lo que habría que introducir algunos cambios en la base de datos y en la forma de acceder (como podría ser asignar prefijos a las tablas).

- Recuperar la lista de posibles alumnos de la base de datos de OpenWonderland y dársela al profesor en la etapa de creación para que seleccione los correspondientes a su asignatura, en lugar de tener que introducirlos de cero.

- Incluir nuevos idiomas de visualización para internacionalizar la herramienta.

Pasada esta fase de mejora del prototipo, el siguiente paso sería probar la herramienta en un entorno menos controlado, es decir, pasar de una red de área local a Internet, con las implicaciones que ello conlleva, como latencia, ancho de banda, número de peticiones al servidor, requisitos gráficos de los equipos cliente, etc.

8. Presupuesto

Se han analizado dos escenarios distintos para el cálculo del presupuesto asociado a este proyecto: por un lado, la tarea realizada hasta ahora, es decir, la fase de desarrollo; por el otro, al ser ésta una aplicación ideada para ser utilizada en un entorno educativo, se ha considerado interesante estimar cuál sería el presupuesto necesario para su funcionamiento.

8.1. Escenario de desarrollo

El presupuesto asociado a este escenario se desglosa en los siguientes costes:

8.1.1. Costes de personal

Para llevar a cabo este cálculo es necesario realizar un análisis de las distintas tareas en que se divide este trabajo y estimar la duración de cada una de ellas:

	Dedicación en meses	**Coste (Euros)**
Fase 1. Estudio de las tecnologías	2	5388,78
Fase2. Diseño de la aplicación	2	5388,78
Fase3. Implementación de la Interfaz del Alumno	4	10777,56
Fase4. Implementación de la Interfaz del Profesor	1	2694,39
Fase 5. Escritura de la memoria y posterior revisión	2	5388,78
Total	11	29638,29

Los datos considerados han sido 21 días laborables por cada mes y jornada de 8 horas diarias. Todos los roles necesarios para el desarrollo del proyecto (diseño, programación en distintos lenguajes, etc.) fueron asumidos por la autora. El salario considerado ha sido el establecido en las plantillas de la universidad para un Ingeniero, esto es, 2694,39€ por hombre y mes.

Aunque la estimación se ha hecho distinguiendo claramente las fases por simplicidad, en realidad se fueron introduciendo cambios en el diseño también en la fase de implementación al comprobar de forma práctica ciertas mejoras o límites. De igual modo, la estimación en meses es orientativa, ya que no sólo se ha trabajado en días laborables y también durante algunas fases del proyecto no se ha desarrollado la jornada completa, por lo que su duración real ha sido mayor.

Al total obtenido hay que realizarle un ajuste, teniendo en cuenta que el ingeniero realizador del proyecto no está aún en posesión del título, por lo que se le puede aplicar un factor del 70% en el cálculo. Con esto, el coste de personal asciende a 20746,8 euros.

Es importante mencionar la existencia de un consultor externo, *Jonathan Kaplan*, cuya inestimable ayuda ha sido necesaria para el acceso a la base de datos desde *OpenWonderland*. Una consulta de este tipo habría subido el presupuesto de personal, al ser éste un Ingeniero Senior, con un mayor salario. Sin embargo, esta colaboración se ha hecho de forma altruista en los foros de la plataforma.

8.1.2. Costes de material

Se dividen en costes de hardware, el cual ha sido un ordenador portátil personal valorado en 700 euros, y en costes de software. En este caso, al tratarse todas las tecnologías involucradas de software libre, los gastos asociados son nulos.

8.1.3. Costes indirectos

Se calculan como el 20% del coste total y engloban todos aquellos gastos como electricidad, conexión a Internet, etc. En este proyecto ascienden a 4289,36 euros.

Con todo ello, el presupuesto total para el escenario de desarrollo es de 25736,16 euros.

8.2. Escenario de producción

Se contempla el uso inicial de la aplicación durante un cuatrimestre y con un grupo de 20 alumnos. Teniendo en cuenta este escenario, el presupuesto destinado para la utilización de la herramienta se puede desglosar en:

8.2.1. Costes de personal

Contemplan la existencia de un ingeniero encargado del mantenimiento de los servidores. Para el cálculo se considera una dedicación de 1 hora semanal, lo cual implica, aproximadamente, 4 horas al mes y, por tanto, 16 horas al cuatrimestre.

Por otra parte, hay que considerar la instalación de los servidores *OpenWonderland, MySQL y Apache* y la puesta a punto para su funcionamiento. Esta tarea puede llevar dos días con jornada de 8 horas, lo cual supone un total de 16 horas.

Contando para ambas tareas el salario de un Ingeniero, estimado a partir de los datos del apartado anterior, el presupuesto total para el personal ascendería a 135 euros.

8.1.2. Costes de material

Al igual que en el apartado anterior, sólo es necesario tener en cuenta los gastos de hardware.

En primer lugar, es necesario un servidor que soporte las 3 tecnologías mencionadas en el apartado anterior y que dé servicio a unos 20 clientes. Una máquina de estas características tiene un coste de aproximadamente 1500 euros.

En segundo lugar, se consideran 20 estaciones de trabajo para los alumnos, las cuales pueden ser equipos de 500 euros, resultando así un total de 10000 euros.

8.1.3. Costes indirectos

Se calculan como el 20% del coste total y engloban todos aquellos gastos como electricidad, conexión a Internet, etc. En este caso ascienden a 2327 euros.

Con todo ello, el presupuesto total para el escenario de producción es de 13962 euros.

9. Glosario

Serialización / Serializable

En lenguajes orientados a objetos, la serialización es el escribir el estado de un objeto en un flujo de bytes de manera que pueda salvarse el estado del programa en un área de almacenamiento persistente, como un fichero. También se necesita para algunos tipos de comunicaciones o, por ejemplo, en java, para implementar el Método de Invocación Remota (RMI). Más tarde ser pueden recuperar estos objetos usando el proceso de deserialización.

Esto se consigue mediante la implementación de la interfaz *Serializable,* que exige que todos los atributos de la clase que queremos hacer serializable, también lo sean. Otra característica importante es que si una clase es *Serializable*, todas sus subclases lo son también. [**20**]

Singleton/instancia única

En lenguajes orientados a objetos, el patrón *singleton* restringe la creación de objetos de una clase a una única instancia. Se suele utilizar cuando la clase controla el acceso a un recurso físico único o cuando cierto tipo de datos debe estar disponible para todos los demás objetos de la aplicación. La forma de conseguir que haya una única instancia global es haciendo el constructor privado y controlando la creación en la propia clase; si esto se hace en un método de clase (*static*) se permite, además, que el acceso sea global.

HUD (head-up-display)

El nombre proviene de los dispositivos transparentes presentes en la cabina de los aviones y que permiten a los pilotos ver información relevante sin tener que mover la cabeza y dejar de mirar al frente. Hoy en día se utilizan en muchos ámbitos, como por ejemplo el de los videojuegos. Análogamente, en *OpenWonderland* está definido este tipo de componente y desarrollado como un módulo de los que se instalan por defecto. De hecho, son *HUD*s las ventanas de chat y de usuarios conectados que siempre aparecen por defecto mientras estamos en el mundo virtual.

10. Anexos

10.1. Módulo del servicio sql

Todas las instrucciones para instalar y configurar este servicio se encuentran en el *README* incluido en el módulo [**21**]. No obstante, de forma breve, los pasos son:

- Descargar el driver *mysql-connector-java-5.__-bin.java* (después del 5. vendría la versión; para este proyecto se ha usado la 16, pero siempre se pueden utilizar versiones posteriores) y colocarlo en el directorio */sql-service/lib*.

- Ejecutar el comando "*ant dist*" desde el directorio /sql-service. De esta forma se construye el *.jar* que hay que instalar en el servidor de *OpenWonderland*.

- Instalar el módulo por el procedimiento habitual.

- Crear el fichero de configuración (se puede llamar, por ejemplo, *mysql.properties*) en el que deben constar las siguientes líneas:

```
Driver=com.mysql.jdbc.Driver
url=jdbc:mysql://___
username=___
password=___
```

  Donde la url debe ser la de la base de datos de la aplicación, como, por ejemplo, en este proyecto, que sería `localhost/delfos-en`. El usuario y la contraseña deberán ser los que utilicemos para gestionar nuestra base de datos en mysql.

- Copiar dicho fichero a un directorio que sea accesible para el servidor Darkstar, por ejemplo, en: *~/.wonderland-server/0.5-dev/*

- Configurar el servidor Darkstar a través de la interfaz web para que incluya este fichero. En el menú *Manage Servers*, hay que pinchar al lado del Darkstar, donde pone *edit*, y en uno de los cuadros de propiedades añadir:

Sqlservice.properties.file	~/.wonderland-server/0.5-dev/mysql.properties

10.2. Módulo de autentificación

Lo primero que se necesita para poder configurar la autentificación es el módulo *security session-auth*. [**22**] Antes de instalarlo hay que desinstalar el módulo por defecto, *security-session-noauth*. Este proceso es diferente si se lanza el servidor desde el archivo binario (*.jar*) o con el comando *ant* desde el código fuente; ambos casos vienen detallados en el tutorial de [**18**].

En ambos casos, a partir de ahí la configuración es idéntica y conlleva los siguientes pasos:

- Una vez arrancado el servidor, hay que entrar en la interfaz web y pulsar el botón *Server Admin,* el cual pedirá un usuario y contraseña. El usuario por defecto es "admin" y la contraseña también "admin".

- Lo primero será cambiar esa contraseña por una más segura, que habrá que escribir en un fichero en blanco y guardar bajo un nombre como *wonderland.password* (es orientativo, en realidad el nombre del fichero podría ser cualquiera).

- Al igual que con el fichero de configuración del *servicio sql,* éste puede situarse en cualquier directorio, pero es recomendable tener todos estos archivos en el mismo sitio, por ejemplo, *~/.wonderland-server/0.5-dev.*

- Lo siguiente es avisar a los distintos servidores de esta nueva contraseña. Se puede poner una diferente para cada uno, simplemente creando nuevos ficheros como el anterior. Por simplicidad, aquí se usa la misma para todos.
  En el menú *Manage Servers* hay que ir pinchando al lado del nombre de los servidores, donde pone *edit* y en cada uno de ellos añadir la propiedad:

  Darkstar

sgs.password.file	~/.wonderland-server/0.5-dev/wonderland.password

  Shared Applications Server

sas.password.file	~/.wonderland-server/0.5-dev/wonderland.password

  Voice Bridge

voicebridge.password.file	~/.wonderland-server/0.5-dev/wonderland.password

  Para configurar el servidor web el procedimiento es diferente. Hay que añadir la siguiente línea en el fichero *my.run.properties* (que estará en el directorio *Wonderland* junto con el resto del código):

  ```
  wonderland.webserver.password.file=
          ~/.wonderland-server/0.5dev/wonderland.password
  ```

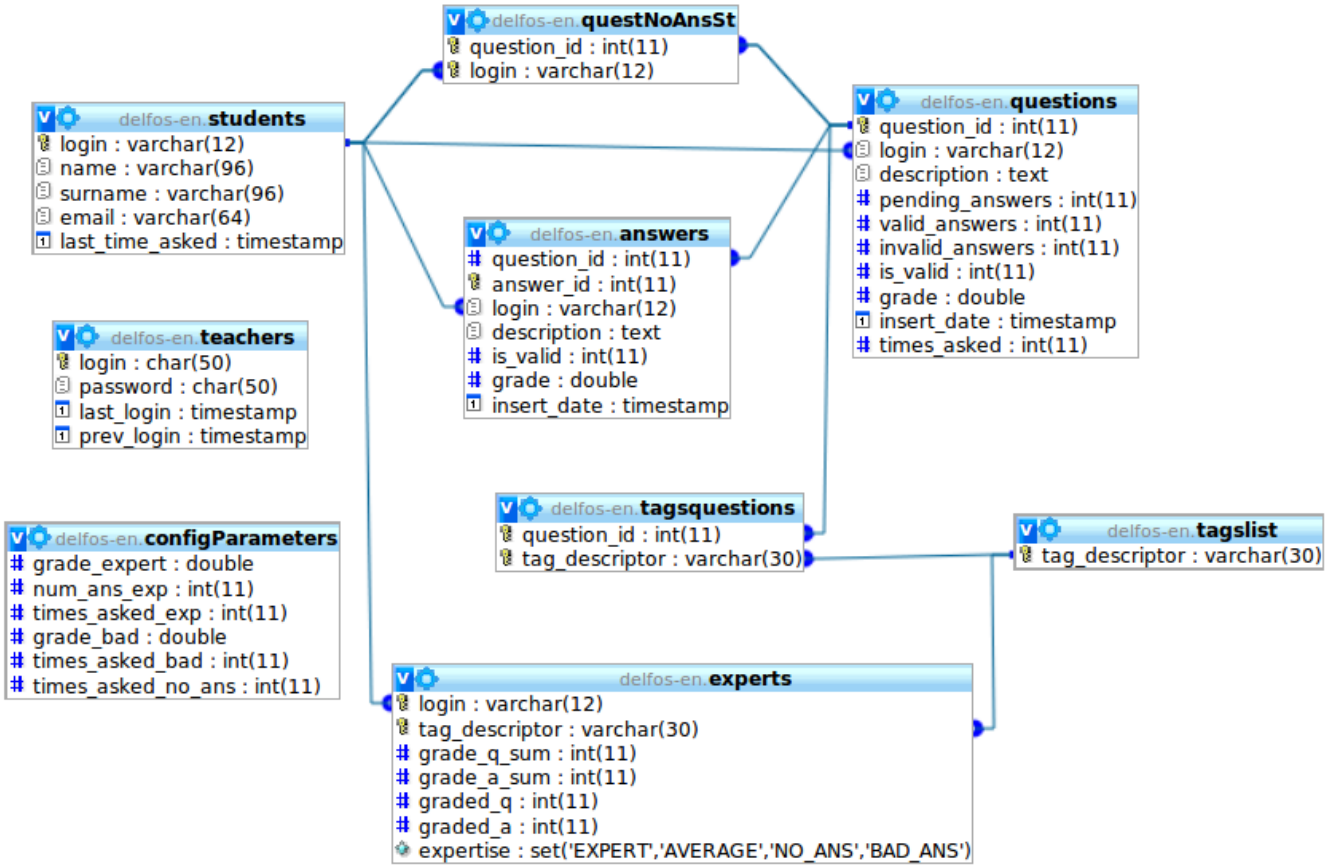
- Finalmente, en el menú *Manage Users* habrá que editar los usuarios que hay inicialmente (que son los servidores) con la nueva contraseña. Éste es el menú en el que luego hay que dar de alta a todos los usuarios a los que se quiera permitir el acceso a los mundos del servidor.

A partir de aquí, para lanzar el servidor *OpenWonderland* correctamente, habrá que ejecutar:

```
java –jar Wonderland.jar ~/my.run.properties
```

10.3. Estructura de la base de datos

En la siguiente figura se puede observar un esquema completo de las tablas que componen la base de datos de la aplicación y las relaciones entre ellas:



En las siguientes tablas se incluye la información que almacena cada campo:

Tabla de alumnos – 'students'

Campo	Tipo	Uso
login	varchar(12) **Clave primaria**	Identificador único para el alumno. En las demás tablas, cualquier *login* que aparezca tiene que pertenecer a ésta.
name	varchar(96)	Nombre propio del alumno (opcional)
surname	varchar(96)	Apellido(s) del alumno (opcional)
email	varchar(64)	Correo electrónico del alumno (opcional)
last_time_asked	timestamp	Fecha y hora de la última vez que se le formuló alguna pregunta mediante el algoritmo del "fantasma".

Tabla de preguntas – 'questions'

Campo	Tipo	Uso
question_id	int (11) Autoincremental **Clave primaria**	Identificador único para la pregunta, cualquier id. de pregunta en otras tablas tiene que pertenecer a ésta.
login	varchar(12) students>login	Identificador del alumno que ha hecho la pregunta (obligatorio)
description	text	Texto de la pregunta (obligatorio)
pending_answers	int (11)	Número de respuestas de esta pregunta que el profesor no ha revisado todavía (0 por defecto).
valid_answers	int (11)	Número de respuestas válidas de esta pregunta (0 por defecto).
invalid_answers	int (11)	Número de respuestas no válidas de esta pregunta (0 por defecto).
is_valid	int (11)	Indica si la pregunta es válida (1), no válida (-1) o no ha sido validada todavía (0, valor por defecto).
grade	double	Nota de la pregunta (0 por defecto).
insert_date	timestamp	Fecha y hora de cuando se formuló la pregunta.
times_asked	int (11)	Número de veces que esta pregunta ha sido formulada mediante el algoritmo del "fantasma".

Tabla de respuestas – 'answers'

Campo	Tipo	Uso
question_id	int (11) questions>question_id	Identificador de la pregunta a la que corresponde esta respuesta (obligatorio).
answer_id	int(11) Autoincremental **Clave primaria**	Identificador único de la respuesta.
login	varchar(12) students>login	Identificador del alumno que ha hecho la pregunta (obligatorio)
description	text	Texto de la respuesta (obligatorio).
is_valid	int (11)	Indica si la respuesta es válida (1), no válida (-1) o no ha sido validada todavía (0, valor por defecto).
grade	double	Nota de la respuesta (0 por defecto).
insert_date	timestamp	Fecha y hora de cuando se dio la respuesta.

Tabla de etiquetas – 'tagslist'

Campo	Tipo	Uso
tag_descriptor	varchar (30) **Clave primaria**	Nombre de la etiqueta o 'tag' que sirve como identificador único. Cualquier 'tag' que aparezca en otras tablas tiene que pertenecer a ésta.

Tabla de etiquetas-preguntas – 'tagsquestions'

Campo	Tipo	Uso
tag_descriptor	varchar (30) tagslist>tag_descriptor	Etiqueta o 'tag' con la que está relacionada la pregunta.
question_id	int (11) questions>question_id	Identificador de la pregunta relacionada con el 'tag'. Aparecerá tantas veces en la tabla como 'tags' con los que esté relacionada.

En este caso, la clave primaria son el conjunto de ambos campos, ya que cada etiqueta puede aparecer tantas veces como preguntas con las que esté relacionada y lo mismo ocurre para los identificadores de pregunta. Sin embargo, cada dupla (*tag_descriptor, question_id*) es única.

Tabla de preguntas no respondidas por los alumnos – 'questNoAnsSt'

Se refiere sólo a las preguntas que les han sido planteadas por el fantasma. El diseño de esta tabla es idéntico al de la anterior.

Campo	Tipo	Uso
login	varchar (12) students>login	'Login' del alumno que no ha respondido a alguna de las preguntas formuladas por el fantasma.
question_id	int (11) questions>question_id	Identificador de la pregunta que no ha respondido este alumno.

Tabla de expertos – 'experts'

Al igual que en las dos tablas anteriores, en ésta la clave primaria también es el conjunto de dos datos: 'login' y 'tag_descriptor', ya que, para cada alumno, habrá tantas entradas en esta tabla como 'tags' en la lista (tabla 'tagslist').

Campo	Tipo	Uso
login	varchar(12) students>login	Identificador del alumno cuyas calificaciones y nivel se almacenan en el registro.
tag_descriptor	varchar (30) tagslist>tag_descriptor	Etiqueta o 'tag' con respecto a la cual se calcula la nota media y nivel del alumno.
grade_q_sum	double	Suma acumulada de todas las notas en preguntas del alumno relacionadas con el 'tag'.
grade_a_sum	double	Suma acumulada de todas las notas en respuestas del alumno relacionadas con el 'tag'.
graded_q	int (11)	Número de preguntas calificadas de este alumno relacionadas con este 'tag'.
graded_a	int (11)	Número de respuestas calificadas de este alumno relacionadas con este 'tag'.
expertise	set ('EXPERT', 'AVERAGE', 'NO_ANS', 'BAD_ANS')	Nivel del alumno para el 'tag'; se actualiza calculando la nota media de respuestas y comparando con los parámetros de configuración.

Tabla de parámetros de configuración – 'configParameters'

Esta tabla sólo tiene un registro, en el código se controla que no se inserte sin haber borrado. Almacena los parámetros de configuración editables por el profesor (ver capítulo 6, *Interfaz del profesor*).

Campo	Tipo	Uso
grade_expert	double	Nota umbral a partir de la cual se considera a un alumno 'experto' (por defecto, 7'5)
grade_bad	double	Nota umbral por debajo de la cual se considera a un alumno con nivel bajo ('bad_ans'). (por defecto, 3'5)
num_ans_exp	int (11)	Número mínimo de respuestas calificadas para poder considerar que un alumno es experto (por defecto, 5)
times_asked_exp	int (11)	Número de veces que se formula una pregunta a expertos en el algoritmo del fantasma (por defecto, 3)
times_asked_bad	int (11)	Número de veces que se formula una pregunta a estudiantes con nivel bajo en el algoritmo del fantasma (por defecto, 5)
times_asked_no_ans	int (11)	Número de veces que se formula una pregunta a estudiantes que nunca han respondido en el algoritmo del fantasma (por defecto, 5)

Tabla de profesores – 'teachers'

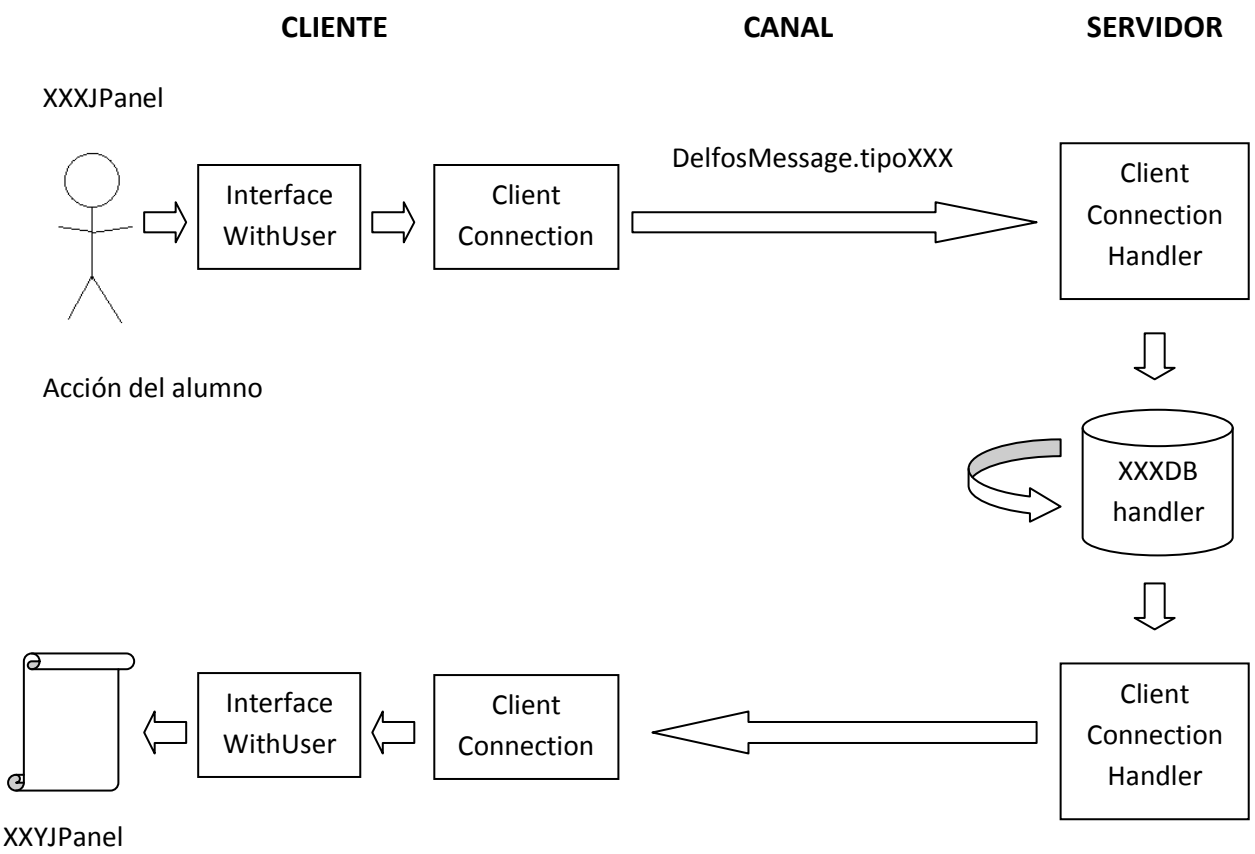
Campo	Tipo	Uso
login	varchar(12) **Clave primaria**	Identificador único para el profesor con el que se autentifica al entrar en su interfaz.
password	varchar(12)	Contraseña de acceso a la interfaz del profesor.
last_login	timestamp	Fecha y hora de la última vez que el profesor entró en el sistema (o si está dentro, la actual).
prev_login	timestamp	Fecha y hora de la anterior vez que el profesor entró en el sistema.

10.4. Listado completo de las clases de la Interfaz del Alumno

Paquete cliente	Paquete común	Paquete servidor
*DelfosCellComponent**	*DelfosCellComponentClientState**	*DelfosCellComponentMO**
*DelfosCellComponentFactory**	*DelfosCellComponentServerState**	
*DelfosCellComponentProperties**		
*TooltipClientPlugin**		
*TooltipJPanel**		
DelfosClientPlugin		DelfosServerPlugin
DelfosMainConnection	DelfosMainConnectionType	DelfosMainConnectionHandler
DelfosDaemonConnection	DelfosDaemonConnectionType	DelfosDaemonConnectionHandler
InterfaceWithUser	DelfosMessage	
DaemonInfoJPanel		CheckExpertiseDBhandler
InfoJPanel	QuestionData	CompleteQuestionDataDBhandler
MainMenuJPanel	AnswerData	CompleteQuestionTagsDBhandler
MyAnswersJPanel		GetExpertsDBhandler
MyPartJPanel		GetMyStatsDBhandler
MyQuestionsJPanel		SaveAnswerDBhandler
MyStatsJPanel		SaveQuestionDBhandler
NewDataJPanel		SearchAnswersDBhandler
NewQuestionJPanel		SearchIDquestionByTagsDBhandler
NewQuestionsJPanel		SearchQuestionsDBhandler
ShowAnswersJpanel		StudentsDBhandler
ShowQuestionsJPanel		TagsListDBhandler
TagsListJPanel		UpdateTimesAskedDBhandler

Las clases en cursiva y acompañadas con un asterisco son las que ya existían previamente en el módulo (cambiando el nombre y añadiendo algún atributo en algunas) desarrollado por Jordan Slott [**23**]. Son las que se encargan con la creación del *Componente* (*capability*) y del *tooltip*.

Las demás, como se comentó en el Interfaz del alumno, implementan la funcionalidad principal de la aplicación, siguiendo el siguiente hilo de ejecución:



La única excepción a esta situación ocurre en el caso del algoritmo del *fantasma* (clases *Daemon*), donde el hilo no se dispara por una acción del alumno sino por un aviso de un temporizador.

La flecha semicircular en las consultas a la base de datos significa que puede darse más de una en algunos casos, por ejemplo, cuando se realiza una búsqueda de preguntas relacionadas con una serie de *tags*, primero se recuperan los identificadores de pregunta (*SearchIDquestionByTagsDBhandler*) y, con esta información, se completan el resto de datos de las preguntas recuperadas (*CompleteQuestionsDataDBhandler*).

11.Referencias

[1] Mundos Virtuales Online: Mini-Guía. [Citado el 13/10/11]. http://www.masternewmedia.org/es/2007/04/11/mundos_virtuales_online_miniguia.htm

[2] Artículo sobre Mundos Virtuales en Wikipedia. [Citado el 13/10/11]. http://es.wikipedia.org/wiki/Mundo_virtual

[3] Espacios virtuales para el aprendizaje. [Citado el 13/10/11]. http://www.masternewmedia.org/2004/09/27/3d_virtual_spaces_for_learning.htm

[4] Moodle. [Citado el 13/10/11]. http://moodle.org/

[5] OpenWonderland. [Citado el 13/10/11]. http://openwonderland.org/

[6] Artículo de Delfos en Wikipedia. [Citado el 13/10/11]. http://es.wikipedia.org/wiki/Or%C3%A1culo_de_Delfos.

[7] Glassfish. [Citado el 13/10/11]. http://glassfish.java.net

[8] Motor gráfico JMonkeyEngine. [Citado el 13/10/11]. http://jmonkeyengine.com/

[9] Galería de modelos 3D de Google. [Citado el 13/10/11]. http://sketchup.google.com/3dwarehouse/.

[10] Tutoriales de creación de una nueva celda. [Citado el 13/10/11]. http://wiki.java.net/bin/view/Javadesktop/ProjectWonderlandDevelopingNewCell05Part1

[11] Tutorial sobre la arquitectura de comunicaciones de OpenWonderland. [Citado el 13/10/11]. http://wiki.java.net/bin/view/Javadesktop/ProjectWonderlandCommsArch

[12] Sitio web de MySQL. [Citado el 13/10/11]. http://dev.mysql.com/

[13] Sitio oficial de Apache. [Citado el 13/10/11]. http://www.apache.org/

[14] Archivo binario de OpenWonderland. [Citado el 13/10/11]. http://openwonderland.org/download

[15] Código fuente de OpenWonderland. [Citado el 13/10/11]. http://code.google.com/p/openwonderland/wiki/DownloadBuildSource05

[16] Consulta en el foro de OpenWonderland sobre el trabajo con una base de datos. [Citado el 13/10/11]. http://groups.google.com/group/openwonderland/browse_thread/thread/691b727e8d2fa924/e62336432cc2e5ff?q=working+with+a+database&lnk=ol&.

[17] Módulo sql-service. [Citado el 13/10/11]. http://code.google.com/p/openwonderland-modules/source/browse/trunk/0.5/unstable/sql-service/#sql-service

[18] Autentificación en OpenWonderland. [Citado el 13/10/11]. http://wiki.java.net/bin/view/Javadesktop/ProjectWonderlandAuthentication05

[19] Marja Kankaanranta, Pekka Neittaanmäki Ahmer Iqbal, "Participation of the young ones in virtual worlds: a look at experiences and motivations," *World Journal on Educational Technology*, vol. 3, no. 1, 2011.

[20] Herbert. Schildt, *Java 2: Manual de referencia.*.

[21] README del servicio SQL de OpenWonderland. [Citado el 13/10/11]. http://code.google.com/p/openwonderland-modules/source/browse/trunk/0.5/unstable/sql-service/README

[22] Módulo de autentificación de OpenWonderland. [Citado el 13/10/11]. http://openwonderland.org/module-warehouse/module-warehouse/doc_details/190-authentication?cat=add_ons&lang=en

[23] Servidor RedDwarf. [Citado el 13/10/11]. http://www.reddwarfserver.org/?q=content/open-source-online-gaming-universe

[24] Tooltip-component (openwonderland.org). [Citado el 13/10/11]. http://openwonderland.org/module-warehouse/module-warehouse/doc_details/208-tooltip?cat=add_ons&Itemid=123