Trabajo Final de Máster 19 de enero 2014

Ampliación de funcionalidades para la red Kpax

Desarrollo Web – Comercio Electrónico

Consultor Externo: Daniel Riera Terrén

Profesor: Francisco Javier Noguera Otero

Alumna: Eugenia Carrera Formoso

Esta obra está licenciada bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported. Para ver una copia de esta licencia, visita http://creativecommons.org/licenses/by-nc-sa/3.0/ o envía una carta a Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

RESUMEN DEL PROYECTO

El proyecto de la plataforma Kpax, ha sido desarrollada e implementada sobre la plataforma open source Elgg por un grupo de la escuela de Estudios de Informática, Multimedia y Telecomunicaciones de la propia Universitat Oberta de Catalunya .

La motivación del proyecto es desarrollar una red social para el aprendizaje basado en juegos. El campo de los llamados juegos serios es muy amplio. Muchos de ellos se diseñan para entrenar ciertas capacidades desde un punto de vista práctico, y abarcan cualquier campo de la enseñanza en la actualidad. La red social estará en producción en esta url Kpax – http://Kpax.uoc.edu/Elgg

Como punto de partida del desarrollo se ha utilizado una plataforma de servicios para red social de código abierto llamada Elgg [1], que proporciona una API muy rica para la configuración y personalización de la plataforma. Elgg debe ejecutarse como aplicación en un entorno LAMP [2], es decir sobre un servidor web, principalmente Apache, que contenga módulos PHP, y con un servidor MySQL. Las especificaciones de Elgg indican que no está soportada ninguna otra base de datos.

Las ampliaciones añadidas a Elgg deben hacerse bajo la forma de plugins y con una estructura muy concreta. El grupo de trabajo ha añadido un plugin Kpax con multitud de funcionalidades orientadas a la compartición de juegos.

El entorno de trabajo de Kpax incluye una aplicación web corriendo en un servidor JBoss que proporciona servicios web de tipo REST a la aplicación Kpax, constituida por un conjunto de plugins que corren sobre Elgg. Esta aplicación web proporciona acceso a una segunda base de datos MySQL, gestionada en exclusiva desde el plugin Kpax, en la que se guardarán distintas características de los juegos, usuarios y desarrolladores.

Nuestro proyecto consiste en la creación de un plugin que permita añadir juegos a la plataforma, interaccionando con la base de datos Kpax a través del servicio web svrKpax, hacer una búsqueda una vez añadidos, en función del nivel de acceso del usuario y que permita además la gestión por parte de los administradores de la red.

Para ello hemos tenido que realizar los siguientes pasos:

- Estudio y comprensión de la estructura de Kpax, tanto del plugin como del servicio web.
- Programación de plugins que implementen funcionalidades de Kpax
- Modificación de tablas en MySQL, mapeo en sus correspondientes clases de Java y revisión del acceso a estas tablas a través del servicio web.
- Integración en Kpax de los plugins programados.
- Pruebas funcionales de los plugins añadidos a la plataforma

Como conclusión tenemos que comentar que dada la complejidad del entorno y la variedad de componentes implicados, el estudio del sistema se ha alargado. Nos hemos enrocado en buscar una solución a un problema que luego ha sido solucionado utilizando una estrategia diferente. Por estas razones no hemos podido cumplir con los plazos de entregas parciales, y además el desarrollo entregado no cumple con la totalidad de los requisitos propuestos.

Tabla de contenidos

RESUMEN DEL PROYECTO 3
INTRODUCCION....5
PRESENTACIÓN 5
OBJETIVOS 5
REQUERIMIENTOS....5
DESCRIPCIÓN DE LAS TECNOLOGÍAS IMPLICADAS....7
PLATAFORMA DE RED SOCIAL ELGG 8
ENTORNO DE DESARROLLO 9
Eclipse....9
Maven....10
GitHub....10
SERVICIO WEB SVRKPAX 11
JBoss....13
Servicios Web 13
Jersey 14
Spring 15
Hibernate....15
PLANIFICACIÓN DE TAREAS....17
ESTUDIO DE VIABILIDAD....19
ANALISIS DEL SISTEMA....21
COMPONENTES DEL SISTEMA....21
ENTORNO DE DESARROLLO....22
PLUGIN KPAX EN ELGG....24
BASE DE DATOS KPAX....29
DISEÑO 31
Diagrama de Casos de Uso....31
Añadir un juego....31
Editar un juego....31
Borrar mi juego 31
Visualizar lista de mis juegos....32
Visualizar lista de todos los juegos:....32
Editar información de un juego:....32
Esquema de las páginas del plugin....33
DESARROLLO....36
PRIMER DESARROLLO FALLIDO....36
DESARROLLO DEFINITIVO....38
BASE DE DATOS KPAX....38
APLICACIÓN JAVA....39
ENTORNO ELGG....40
Página intro 40
Página add game....41
Página all games....43
Página my games....44
Página delete....45
Página edit game....46
Página game item....47
IMPLANTACION Y MANTENIMIENTO....50
CONCLUSIONES....51
REFERENCIAS BIBLIOGRÁFICAS....53

INTRODUCCION

PRESENTACIÓN

El proyecto de la red social Kpax [3], ha sido desarrollado e implementado sobre la plataforma open source Elgg por un grupo de la escuela de Estudios de Informática, Multimedia y Telecomunicaciones de la propia Universitat Oberta de Catalunya .

La motivación del proyecto es conseguir una plataforma de aprendizaje mediante los llamados juegos serios, pensados para entrenar ciertas capacidades desde un punto de vista práctico.
Se ha utilizado una plataforma de servicios para red social de código abierto como punto de partida del desarrollo. Se busca que sea flexible en cuanto a:

- Conexión on-line y off-line desde una variedad de dispositivos
- Aprendizaje y entrenamiento en distintas áreas.
- Facilidad para que los desarrolladores publiquen sus juegos.
- Flexibilidad para constituir comunidades y grupos tanto de aprendizaje como de desarrollo de juegos orientados a la enseñanza, posibilitando la creación de proyectos innovadores y multidisciplinares.

La parte principal del desarrollo está ya implementada [4], pero faltan muchas funcionalidades que pueden ser añadidas mediante plugins, debido a que la arquitectura de la plataforma Elgg integra de esta forma la ampliación de funciones.

OBJETIVOS

Queremos incorporar un plugin a a Kpax para que las personas que desarrollen un juego tengan disponibles todas las opciones de interacción para la publicación y gestión de juegos serios en la plataforma Kpax.

Se pretende que la introducción de nuevos juegos quede completamente abierta tanto a la comunidad UOC (profesores, alumnos, personal de gestión, etc) como a programadores externos a la universidad, de forma que cualquiera pueda darse de alta.

REQUERIMIENTOS

El desarrollo será en PHP para el plugin y en Java para la modificación del servicio web. Una vez validado el usuario, el plugin Gameserver debe comprobar si el usuario es administrador o no para habilitar distintas pestañas y botones en el entorno de trabajo de la siguiente manera :

Diferenciaremos dos tipos de usuario: el tipo A no tendrá privilegios ni roles especiales, el tipo B será administrador o usuario privilegiado. Dependiendo de los privilegios del usuario validado, el plugin tendrá distintos requisitos:

Tipo A: Visualizan una pestaña nombrada como "Desarrolladores".

Tipo B: Visualizan una pestaña nombrada como "Nuevos juegos".
Finalmente pactamos implementar una única pestaña “Desarrolladores” para ambos.

2 Cualquier usuario tipo A puede añadir juegos. Así, cuando seleccione la pestaña "Desarrolladores", verá un botón "Añadir juego", otro "Mis juegos" y una explicación de cómo se puede colaborar con la plataforma. El formulario tendrá que permitir durante el proceso de añadir el juego a Kpax, validar la identidad del desarrollador generando una clave pública/privada para comunicarse con la plataforma de manera cifrada.

3 (A) En caso de hacer clic en el botón “Añadir juego” se abrirá una nueva página en la que se mostrará un formulario con la información necesaria para añadir un juego a la plataforma. Tras la edición, habrá un botón para confirmar que ya se ha rellenado la información y se quiere enviar a Kpax.

4 (A) Una vez pulsado “enviar”, se guardarán los datos en la base de datos y se mostrará una pantalla con la lista de juegos del desarrollador (la misma que en el punto 5).

5 (A) Si se pulsa “Mis juegos”, se hará una consulta a la base de datos Kpax solicitando los juegos desarrollados por el usuario y se listarán, mostrando el estado (aprobado, pendiente, denegado), las estadísticas (número de jugadores, puntuación, etc.) y edición (se podrán editar algunos de los campos del formulario de entrada, previa aceptación por parte de los administradores).

6 (B) Los administradores podrán ver la lista de todos juegos existentes en el sistema en forma de cuadro de control editable y ordenable.

6 (B) La página para administradores debe tener un filtro configurable que permita mostrar sólo los juegos pendientes de aprobación, los que tienen cambios pendientes de aprobar, todos, etc.

6 (B) Para cada juego, se debe mostrar un botón para ver toda la información del juego, así como otro botón que permita aprobar, o denegar el juego. En caso de denegación, se podrá editar un cuadro de texto con la explicación correspondiente.

DESCRIPCIÓN DE LAS TECNOLOGÍAS IMPLICADAS

El estudio del campo de los juegos serios por su amplitud queda descartado. Sí es necesario comentar que no se ha encontrado una implantación similar con esa temática. Foros, blogs, organizaciones de interesados en juegos de casi cualquier campo de aplicación (docencia, formación empresarial) empresas desarrolladoras y otros forman parte del amplio espectro que abarca esta temática, pero ninguno integrado en una plataforma de red social.

En este caso la descripción del estado del arte va a centrarse en la presentación de las tecnologías que participan en los distintos aspectos del proyecto: JBoss, Apache, Hibernate, Maven, Jersey, Spring, MySQL, Eclipse, etc. La mayoría de estas tecnologías están desarrolladas en Java y son de código abierto . Debemos tener en cuenta un par de cuestiones previas:

Las tecnologías web no son productos estáticos. La aparición de nuevos dispositivos, nuevos medios para la comunicación, nuevos sistemas operativos, y en general el desarrollo y mejora de las herramientas por parte de la comunidad y de las empresas desarrolladoras, lleva a que estos productos estén en constante evolución.

Por otra parte, muchas de ellas se ofrecen no como aplicaciones más o menos terminadas, sino como frameworks, como conjuntos de piezas relacionadas que podemos ensamblar a nuestro antojo, respetando sus dependencias, algunas incluyen interfaces de servicio para modificar los comportamientos de algunos componentes.

Esto es así para muchos aspectos del código de fuente abierta, pero al hablar del lenguaje Java, nos encontramos que las tecnologías nos son presentadas no sólo como conjuntos de librerías de funciones que pueden ser utilizadas, tipo API (Application Programming Interface) sino con una parte llamada SPI (Service Provider Interface) que permite modificar la forma en la que trabajan las funciones de la API.

Vamos a echar un vistazo general al sistema Kpax, que podemos dividir en tres partes principales:

- La plataforma Elgg en la que se ejecutan las funciones de red social, y que actúa como cliente de un servicio web.

- El entorno de desarrollo para el servicio REST. Para el desarrollo del plugin se ha utilizado simplemente un editor avanzado de texto, Notepad++ y una maqueta de la propia plataforma.

- El servicio web Kpaxsrv de tipo REST, que consta de una aplicación accesible a través de Http de modo que permite la ejecución remota de ciertos métodos y funciones publicados.

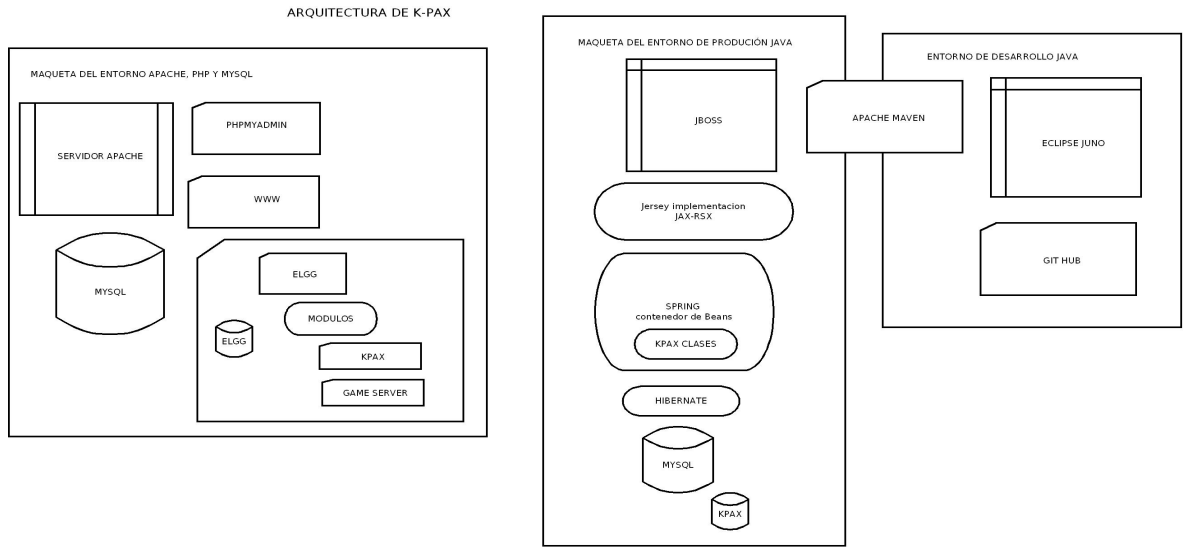

Figura 1 Esquema general del sistema

PLATAFORMA DE RED SOCIAL ELGG

Sobre un servidor Apache con soporte para PHP, asociado a un servidor de base de datos MySQL, se ejecuta la aplicación Elgg. Como componentes o plugins de dicha aplicación encontramos el plugin Kpax y el plugin Game server, objeto del presente desarrollo. Ambos complementan el sistema Elgg de modo que Elgg se ocupa de funciones como el login y los plugins proporcionan funcionalidades extra dentro del sistema. Parte de las funciones lanzadas desde Elgg son peticiones que recibirá el servicio web, devolviendo resultados que serán tratados por la lógica de los plugins para la presentación final en el navegador del cliente autenticado.

A continuación vemos un esquema de esta parte del sistema:

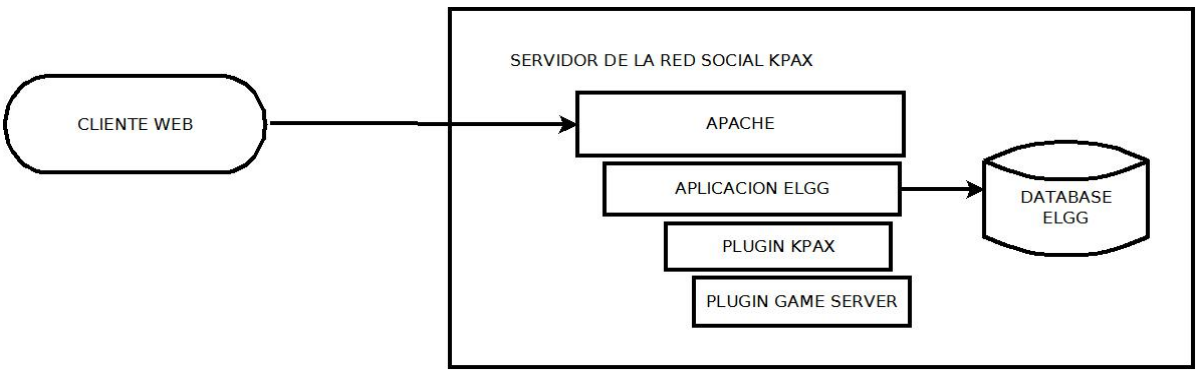

Figura 2 Esquema del sistema Elgg

Elgg es un sistema que cumple varios requisitos clave que descartan la elección de otras plataformas similares 5] disponibles:

- Es de código abierto lo que descarta sistemas tan sólidos como SocialEngine, que también está desarrollado con PHP, pero el coste del motor básico son 300 dólares. Es necesario comentar que Elgg admite, mediante un sistema de suscripción de pago, sobre 50 dólares , la contratación de servicios profesionales de soporte y

consultoría a través de la empresa Thematic Networks, empresa que absorbió a Curverider [6] en 2010, esta última creada por los desarrolladores originales de Elgg.

- Es más customizable que otras herramientas de código abierto como Icow o Phpsocial, que son más bien un conjunto de scripts que incorporan funcionalidades de red social a un site o como ApleSeed, que aporta un servidor menos customizable.

El conjunto Apache con PHP y servidor de base de datos MySQL se ha convertido casi en un estándar de facto. Apache es el servidor más utilizado para servir páginas estáticas, o como frontend para arquitecturas mas complejas en las que mediante AJP, Apache Jserv Protocol, se puede balancear carga entre varios servidores.

PHP es un lenguaje sencillo y robusto para la generación de páginas html dinámicas, es muy fácil embeberlo en html al ser de tipo script, interpretado y no compilado, y sus funciones para el acceso a bases de datos postgre y MySQL son eficientes y rápidas. En nuestro caso el lenguaje ya viene impuesto por Elgg.

Respecto a los sistemas de gestión de bases de datos relacionales de fuente abierta, entre los más utilizados están Postgre y MySQL, a los que últimamente se ha añadido Mariadb, una derivación de MySQL por parte del equipo original de desarrollo. Ahora MySQL es mantenida por Oracle, aunque por ahora se mantiene como producto de código abierto.
En nuestro caso, dado que Elgg está desarrollada en PHP y sólo admite como base de datos MySQL [1][7], no existen otras posibilidades de elección.

Sería suficiente con un servidor que soportase la función de reescritura de URLs, en el caso de Apache ese componente es mod-rewrite, y contuviese un intérprete PHP, no tiene porque ser Apache. Para poner en producción un proyecto pequeño o mediano sería suficiente, si la carga para el servidor aumentase, quizá habría que proponer otras soluciones.

ENTORNO DE DESARROLLO

El entorno de desarrollo para la aplicación Java se compone de un entorno de desarrollo integrado o IDE Eclipse Juno (versión 4.2) , un gestor de proyectos Apache Maven y un repositorio en red servido a través de GitHub.

Eclipse

Eclipse es un IDE [8] creado inicialmente por IBM, pero es mantenido por la Eclipse Foundation desde 2004 y respaldado por una comunidad de voluntarios. Se distribuye con licencia [9] propia EPL 1.0, que incluye varios puntos sobre reconocimiento de patentes del propio IDE, por ello no es compatible con GPL pero sí copyleft, lo que permite distribuir copias del IDE libremente, y no obliga a que el trabajo derivado de su uso sea GPL.

Un IDE consta como mínimo de editor de texto, compilador, intérprete, depurador y un cliente. Eclipse incluye todo lo anterior y muchas funcionalidades más. Las distribuciones o proyectos de Eclipse pueden ser de varios tipos, y estar organizados en conjuntos de herramientas elegidas para dar una funcionalidad más completa. En nuestro caso se ha utilizado el proyecto para herramientas web que incluye entre otros, editores para los lenguajes script más usados en la web (html, js, xml...), asistentes para servicios web,

servidor y cliente web y todo el soporte para Java Enterprise Edition.

Maven

Maven [10] es una herramienta de construcción de proyectos que abarca los procesos de compilación, empaquetado, despliegue o test, mediante un proceso de aplicación de patrones o plantillas, predefinidas o creadas ad-hoc por el usuario. Además posee otras funciones como gestión de dependencias, gestión de versiones o distribución y despliegue del proyecto. Es un producto de Apache Software Foundation para gestión de proyectos de desarrollo de software en Java.

Maven describe el proyecto Java en forma de modelo de objetos para el proyecto o Project Object Model POM [11] utilizando un fichero xml en el que se listan sus partes componentes, las dependencias entre ellas, las librerías externas al proyecto requeridas y sus ubicaciones en repositorios locales y on line.

Por otra parte realiza el control de las dependencias de forma totalmente desatendida. En proyectos con múltiples tecnologías, o con varios participantes en el desarrollo, mediante repositorios en red sincronizándose con repositorios locales, permite mantener la coherencia de los conjuntos de librerías auxiliares jar, descargándolas automáticamente a repositorios locales de los que puede tomarlas en el momento de compilar. La filosofía de Maven es no almacenar las dependencias [12] en el mismo lugar que el código, y a la vez, mantener las dependencias actualizadas y disponibles para operaciones como la compilación.

Las funcionalidades de Maven [13] se pueden ampliar mediante plugins, por ejemplo podría generar los mapas de Hibernate a partir de una base de datos, o desplegar la aplicación en el servidor por defecto. Puede trabajar compartiendo repositorio con Eclipse y tomar como origen de la compilación esos ficheros.

GitHub

Githhub [14] es uno de los principales servicios de alojamiento y publicación para proyectos de código empleados en la actualidad. Es una plataforma de código abierto. Está basado en Git, la herramienta creada por Linus Torvalds para gestionar el repositorio del núcleo de linux.

Además permite que varios desarrolladores creen un repositorio, lo compartan y trabajen sobre él a la vez, creando un sistema en cascada de actualizaciones de modo que no se borren cambios guardados por otros, y proporcionando un sistema de verificación de la persona que realiza actualizaciones en los repositorios comunes mediante RSA.

En nuestro caso nos han pedido que creemos un fork a partir del desarrollo principal para poder añadir el código. De esta forma si fuese necesario añadirlo al repositorio principal, se hará tras una revisión por parte del tutor de prácticas.

SERVICIO WEB SVRKPAX

Este sistema está compuesto por:

- Una base de datos MySQL llamada Kpax con información sobre los juegos.
- Una aplicación Java Kpaxsrv
- El servidor JBoss en la que desplegar la aplicación.

La aplicación, desarrollada en Java, se compila desde el entorno de desarrollo generando un fichero .war, que contiene todo el código Java y las librerías y dependencias principales. JBoss recibe una copia de este fichero para ser desplegado, o lo que es lo mismo, JBoss ejecuta la aplicación Java, que principalmente ofrece servicios vía protocolo HTTP a la aplicación Kpax que se ejecuta en el entorno Elgg.

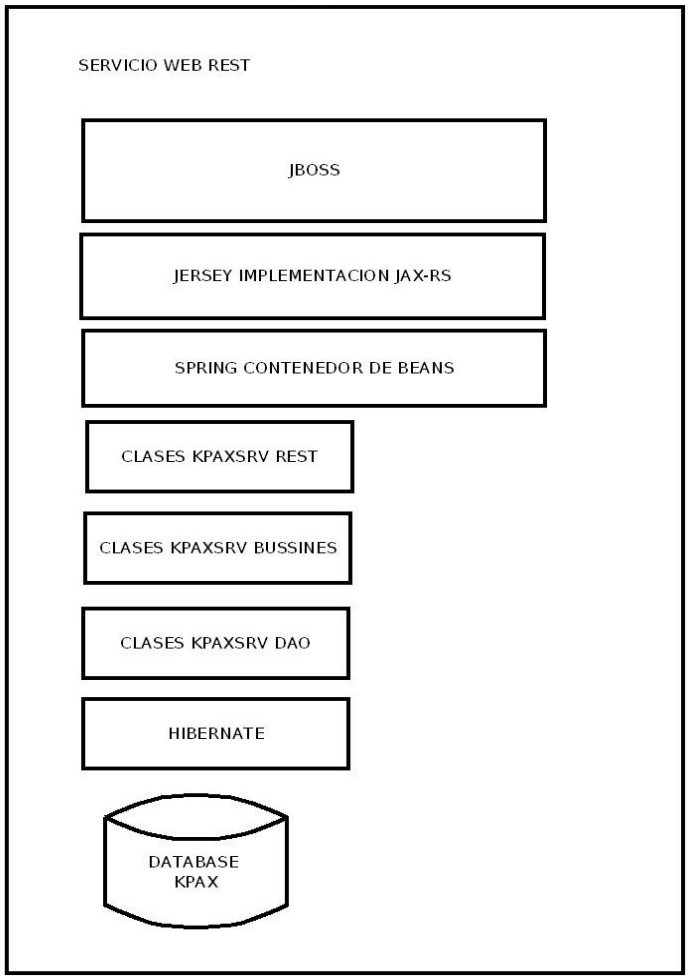

Figura 3: Esquema del servicio Web srvKpax

La forma en la que el servidor JBoss transforma el war en servicios web se basa en los siguientes componentes que se van activando durante el arranque del servidor:

- Un contenedor de beans llamado Spring [15] junto con un servicio de nombres JNDI permite localizar los bean y mantenerlos en memoria, aunque sus funciones van

mucho más allá: incluye inversión de control a través de la inyección de dependencias.

- Una implementación de servicios web tipo REST llamada Jersey que recibe y emite las peticiones [16] destinadas a los beans.

- Una herramienta de mapeo objeto – relacional (ORM) llamada Hibernate [17] permite que las peticiones que reciben los beans en memoria se apliquen a la base de datos Kpax.

- Es más que un conector JDBC o un lenguaje HQL, es un sistema de mapeo completo.

Entraremos con más profundidad en posteriores apartados, ahora daremos una explicación introductoria, teniendo en cuenta que los componentes se comunican en vertical, cada uno con el inmediatamente superior o inferior:

La base de datos Kpax en MySQL está compuesta de al menos 15 tablas que cubren diferentes aspectos de la organización y compartición de juegos. Podemos citar entre ellas Game, con todas las características de un juego (id, nombre, fecha de creación, etc) o User o Category, con las categorías en las que se puede agrupar. Para cada una de las tablas relevantes se crean en el proyecto web al menos tres ficheros Java:

- gameVO es la definición de la tabla como un objeto. Se describe cada una de las columnas y se establecen dos métodos por columna uno para ver el valor y otro para escribirlo, get y set.

- GameBO contiene la definición de funciones que tienen como objeto esa tabla. En esta clase se definirían funciones que podrían requerir para su resultado una consulta sql sobre la tabla que representan.

- GameDAO es la clase que ejecutaría en efecto la consulta sql que requiere la función definida en bussines para obtener su resultado, pero no utilizará sql, sino que se comunicará con la Base de Datos a través del componente Hibernate, utilizando un lenguaje propio llamado HQL.

Además del código Java propiamente dicho, los ficheros incluyen anotaciones Java [18] , esto es, tags precedidos de @ que el compilador de Java no interpreta pero mantiene en los ficheros compilados, de modo que durante el despliegue de la aplicación, son leídas por los componentes Spring [19] , Jersey e Hibernate. Cada uno interpreta su propio juego de anotaciones:

- A Spring le sirven para conocer principalmente las dependencias entre las clases

- A Jersey le sirven para conocer entre otros, los tipos mime del servicio web, o los retornos de datos esperados.

- A Hibernate le sirven para conocer los nombres de las tablas, sus campos y sus relaciones (foreing key, campos autoincrementales, etc)

El contenedor de beans Spring, crea una instancia de cada una de las tres clases anteriores. Estas instancias o Beans se mantienen en memoria durante todo el tiempo de ejecución del servidor JBoss. Spring implementa la funcionalidad inversión de control, que consiste en que los beans se referencian entre ellos para darse servicios mutuamente,

como si se autoorganizasen.

Los beans de tipo DAO, están escritos en lenguaje HQL, un lenguaje que Hibernate interpreta para poder enviar y recibir consultas de una base de datos relacional como MySQL.

Finalmente existe en el proyecto Java una clase que contiene un listado de los servicios web que se van a ofrecer desde el servidor, la principal de ellas es la clase games. Java del paquete Rest , que contiene la definición de servicios ofrecidos desde Jersey. Estos servicios serán invocados desde el cliente validado en Elgg, a través de las funciones declaradas y llamadas desde sus plugins.

Tras esta explicación general de la parte Java del proyecto, podemos profundizar un poco en las principales tecnologías mencionadas:

JBoss

Es un servidor de aplicaciones [20] de código abierto escrito en Java. Marc Fleury comenzó el proyecto en 1999, y en 2001 lo convirtió en empresa de servicios. En 2006 la compañía fue adquirida por RedHat, que ha mantenido la filosofía de JBoss como desarrollo de código abierto, con una comunidad de desarrolladores voluntarios por detrás y comercializando servicios de soporte y consultoría. Está licenciado bajo la LGPL , (GNU Lesser General Public License).

JBoss AS 4.2.3 es un servidor [23] de aplicaciones compatible con las especificaciones de Java EE 1.4. Esta versión incluye un contenedor de Servlets Apache Tomcat 6 y un contenedor de beans propio llamado Enterprise JavaBeans 3.0. También incluye de forma nativa:

- Servicios web tipo JAX-RPC.
- Soporte para Hibernate
- JNDI Java Naming and Directory Interface un conjunto API/SPI customizable para conectar con cualquier servicio de nombres disponible (DNS, LDAP,etc)
- Funcionalidades para agrupar en cluster varias instancias de JBoss actuando colaborativamente.

Servicios Web

Un servicio web es un sistema que ofrece datos o funciones a otros sistemas distantes a través del protocolo HTTP. Por ejemplo, es la forma en la que una base de datos pública podría ser consultada por aplicaciones de terceros sin necesidad de comprometer su seguridad publicándola en la web. Los servicios Web [24] pueden ser de dos tipos:

Simple Object Access Protocol (SOAP) Protocolo de acceso a objetos simples.
Representational State Transfer (REST). Transferencia de estado representacional

SOAP
Es un protocolo aceptado por W3C que define, utilizando notación xml, como debe ser la estructura de los mensajes intercambiados entre dos aplicaciones web, utilizando los métodos Get y Post de http. En su evolución ha incluído WSDL (Web Services Description Language) un estándar que define la forma de describir en XML un servicio web dado. De modo que la publicación de aplicaciones complejas se estandariza y el consumidor del

servicio se pliega también a ese estándar. Este protocolo se utiliza para servicios complejos.

REST
Se trata de una arquitectura o filosofía [25] que proporciona directrices sobre la forma en la que varias aplicaciones intercambian datos mediante HTTP y como eso influye en la forma de programar dichas aplicaciones. Es decir el diseño de servicios web debe centrarse en los recursos del sistema, cada uno de ellos con su propia URI, y teniendo en cuenta que los estados de los recursos sólo pueden modificarse a través de los métodos PUT, GET, POST, o DELETE de HTTP.

Rest requiere que se utilicen peticiones completas e independientes, de modo que el servidor no necesita almacenar el estado [26] de las mismas. Es más ligero puesto que no incluye cabeceras HTTP y más fácil de implementar.

Los servicios REST pueden utilizar para la información intercambiada o bien XML dentro del cuerpo de las peticiones o respuestas HTTP o bien JSON [27], una notación para colecciones de datos mucho más simple. En este proyecto se usará JSON para la obtención de datos desde el servicio web.

Ejemplo datos JSON:

```
{
"employees": [
{ "firstName":"John" , "lastName":"Doe" },
{ "firstName":"Anna" , "lastName":"Smith" },
] }
```

Ejemplo datos XML:

```
<employees>
        <employee>
                <firstName>John</firstName>
                <lastName>Doe</lastName>
        </employee>
        <employee>
                <firstName>Anna</firstName>
                <lastName>Smith</lastName>
        </employee>
</employees>
```

Los servicios REST o JAX-RS [29] tienen en la actualidad varias implementaciones, una de las más conocidas es Jersey, de la que hablaremos a continuación, puesto que es la que se utiliza en este proyecto.

Jersey

Jersey [30] es una implementación para servicios web ligeros RESTful. En nuestro caso se integra con JBoss como una aplicación web que forma parte del paquete desplegado y que se carga antes que las clases desplegadas. También se podría añadir como componente al propio servidor. El framework está escrito en Java y es de código abierto. La versión más reciente de Jersey es 2.5. De acuerdo a la documentación, Jersey se construye con Maven.

Apoyándose en los componentes Jackson, encargados de la manipulación de datos Json, las funciones de Jersey consisten en recibir las peticiones HTTP destinadas al web service y enviárselas al gestor de Beans Spring, así como emitir las respuestas HTTP. Ente los módulos [31] disponibles podemos destacar:

- El core de la aplicación: jersey-server, jersey-client y jersey-common
- Módulos para el envío, recepción y procesado de json:
- Jersey-media-json-jackson, html-json o jersey-media-json-processing

Spring

Spring framework [32] es una plataforma Java de código abierto, desarrollada inicialmente por Rod Johnson. Su primera versión fue liberada en 2003 bajo la licencia Apache 2.0. Su filosofía de producto prefiere el uso de POJOs [33] es decir Plain Old Java Objects, o ficheros de clases Java muy poco dependientes de otras clases o librerías. Permite aplicar servicios de forma no invasiva a estas clases, mediante inyección de dependencias a través de anotaciones Java.

Entre las características de Spring, hay que mencionar al menos dos que tienen relación con el presente proyecto:

Inversión de control
Es un método [34] de programación en el que el flujo de ejecución de un programa se invierte respecto a los métodos de programación tradicionales, en los que desde la ejecución principal se realizan llamadas a procedimientos o funciones auxiliares.

En la inversión de control se especifican respuestas deseadas a solicitudes de datos concretos, dejando que las acciones de control que se requieran para obtener esos resultados sean llevadas a cabo por entidades externas, como otras clases o librerías implicadas en el desarrollo.

Inyección de dependencias
Es un patrón de diseño [35] orientado a objetos, en el que se suministran objetos a una clase en lugar de ser la propia clase quien cree el objeto. El término fue acuñado por primera vez por Martin Fowler. Es un caso particular de inversión de control. Spring [36]implementa esta característica actuando como contenedor de clases o beans e inyectando a cada objeto referencias a los objetos necesarios según las relaciones plasmadas en un fichero de configuración.

Otros módulos que contiene Spring:
herramientas de acceso a datos mucho menos potentes que Hibernate. .
Gestión de transacciones: unifica distintas APIs de gestión y coordina las transacciones para los objetos Java.
Modelo vista controlador: Un framework basado en HTTP y servlets, que provee herramientas para la extensión y personalización de aplicaciones web y servicios web REST sin que sea necesario Jersey

Spring a pesar de ser Open Source, es un producto propietario de la empresa SpringSource [40] que como RedHat o JBoss, se apoyan en una comunidad de desarrolladores voluntarios, y proveen a su vez servicios profesionales de consultoría y formación.

Hibernate

Hibernate [17] es una herramienta de mapeo objeto-relacional (ORM) para Java. Fue creado desde cero en 2001 por un grupo de desarrolladores liderados por Gavin King. Ahora el proyecto forma parte de Red Hat. Hibernate es software libre, distribuido bajo los términos de la licencia GNU LGPL. No se basaron en otras especificaciones ni en estándares, con lo que el framework es un producto sin dependencias de otras herramientas.

Hibernate mapea sólo bases de datos relacionales a objetos, de modo que cada tabla componente de la base de datos se representa por una clase Java, de tipo POJO (plain old Java object). Para el mapeo puede utilizar archivos xml o anotaciones, mediante el componente Hibernate Annotations. Cualquiera de estos mecanismos permite reflejar las relaciones que las tablas tienen en la base de datos entre los objetos que las representan.

Hibernate [42] ofrece también un potente lenguaje de consulta de datos llamado HQL (Hibernate Query Language), similar al SQL clásico y a diferencia de él, actúa sobre objetos y no sobre tablas. También incluye una API llamada Criteria para construir consultas desarrolladas en un lenguaje orientado a objetos. A continuación mostramos un esquema de Hibernate:

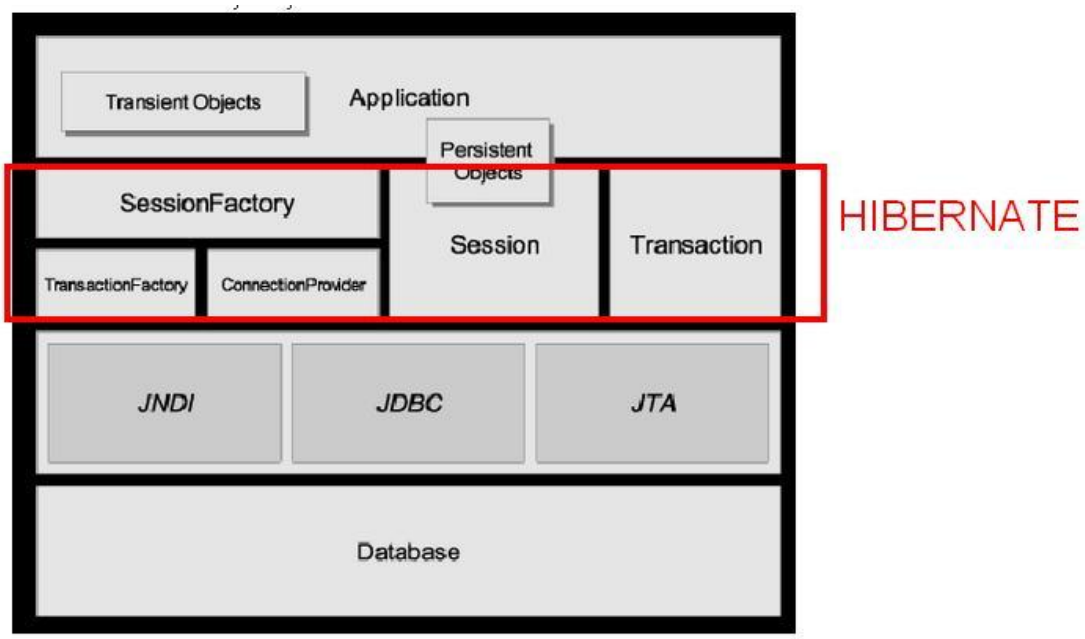

Figura 4 Esquema de Hibernate, obtenida de [43]

PLANIFICACIÓN DE TAREAS

Nombre	Fecha de inicio	Fecha de fin	Duración (días)
Preparar entorno de desarrollo	01/10/2013	15/10/2013	10
Acceso y creación de un repositorio derivado (fork)	01/10/2013	05/10/2013	4
Preparación del entorno, revisión de documentación	07/10/2013	15/10/2013	6
Estudiar el desarrollo ya hecho	14/10/2013	01/11/2013	14
Revisar que ficheros hay que tocar para integrar el código añadido	14/10/2013	25/10/2013	9
Localizar ficheros de formato y apariencia css	15/10/2013	30/10/2013	11
Revisar la base de datos añadiendo campos o tablas si es necesario	21/10/2013	01/11/2013	9
Iniciar el desarrollo de la 1º entrega	25/10/2013	26/11/2013	22
Desarrollo de código	25/10/2013	22/11/2013	20
Añadir función discriminadora de grupo de usuario	25/10/2013	02/11/2013	6
Añadir botón desarrolladores	25/10/2013	02/11/2013	6
Crear vista con dos botones llamada por botón desarrolladores	25/10/2013	02/11/2013	6
Crear formulario para botón mis juegos	28/10/2013	12/11/2013	11
Crear formulario para botón añadir juego	28/10/2013	19/11/2013	16
Crear formularios de operación correcta / operación fallida	28/10/2013	19/11/2013	16
Crear vista con todos los juegos añadidos por usuario	28/10/2013	22/11/2013	19
Documentación	11/11/2013	26/11/2013	11
Pruebas	14/11/2013	26/11/2013	8
Ficheros entregables	25/11/2013	26/11/2013	1
Iniciar el desarrollo de la 2º entrega	25/11/2013	04/01/2014	30
Desarrollo de código	25/11/2013	25/12/2013	22
Crear botón nuevos juegos en la vista del usuario	25/11/2013	30/11/2013	5
Crear vista con tabla ordenable y editable	25/11/2013	25/12/2013	22
Crear formularios que permitan la actualización de la base de datos a través de dicha vista.	25/11/2013	25/12/2013	22
Documentación	05/12/2013	25/12/2013	14
Parón Navideño	23/12/2013	02/01/2014	8
Pruebas	09/12/2013	04/01/2014	20
Ficheros entregables	03/01/2014	04/01/2014	1
Aceptación de desarrollo	20/12/2013	11/01/2014	16
Preparación de la memoria fin de prácticas externas	13/12/2013	18/01/2014	26
Entrega Final	20/01/2014	21/01/2014	1

Existen al menos cuatro grupos de tareas:

Los que abarcan la preparación del entorno de desarrollo:

- Instalación de las aplicaciones, alta y descarga del repositorio, generación de un fork o proyecto derivado del principal, para no sobreescribir el código sin revisión.
- Estudio del sistema ya implantado para mantener la unicidad del aspecto de las ventanas, en la codificación, localización de ficheros, estudio del sistema en general.
- Preparación de un entorno de pruebas o preproducción en el que validar los desarrollos.

Las tareas que corresponden a la primera entrega o hito, funcionalidades para usuarios no privilegiados:

- Programar que en función del tipo de usuario muestre una pestaña o ambas.
- Crear la página desarrolladores, con los dos botones que llamen a nuevos formularios.
- Crear en la base de datos de la plataforma los campos y las restricciones que aplicará el formulario a la hora de añadir un juego nuevo.
- Crear el formulario que permita añadir un juego nuevo.
- Crear la pagina que genere la consulta a base de datos sobre los juegos propios, llamada desde el botón mis juegos
- Pruebas y documentación.

Las tareas que corresponden a la segunda entrega o hito, funcionalidades para usuarios administradores:

- Crear la página nuevos juegos solo visible para administradores, con los dos botones que llamen a nuevos formularios.
- Crear en la base de datos de la plataforma los campos y las restricciones referidos al estado de aceptación o denegación en el estado de un juego (publicado, pendiente de publicación...etc)
- Crear la pagina que genere la consulta a base de datos sobre los juegos del sistema, llamada desde el botón nuevos juegos
- Presentar los resultados a los administradores en una tabla ordenable y editable.
- Pruebas y documentación.

Finalmente las tareas de aceptación del desarrollo y entrega de memoria general del proyecto.

Se han propuesto dos hitos:

Una primera entrega el **25 de noviembre de 2013** con el desarrollo correspondiente a las funcionalidades para los usuarios no privilegiados.

Una segunda entrega el **3 de enero de 2014** con el desarrollo correspondiente a los usuarios administradores.

En la planificación [44] se marca el hito de la entrega final de la memoria el 20 de enero, aunque la fecha real fijada es el 19 de enero.

ESTUDIO DE VIABILIDAD

Kpax ha sido creada a partir de la plataforma Elgg, Open Source , con una comunidad muy activa detrás y totalmente desarrollada en PHP. Esta plataforma funciona sobre un entorno LAMP, que abarca un servidor Web Apache con PHP y una base de datos MySQL. Las funcionalidades extras deben ser implementadas mediante plugins, debido a que la plataforma Elgg integra de esta forma el software añadido. Vamos a presentar a continuación la situación inicial del sistema.

Kpax posee la funcionalidad de salvar y mostrar juegos. Nos han pedido que traslademos esta funcionalidad a un plugin nuevo y que esta funcionalidad sea ampliada con nuevas funciones y con una característica a mayores que aporte nueva información. Además debe integrarse con la plataforma Elgg y con los cinco plugins que pertenecen al proyecto Kpax:

- apiadmin
- kpax
- likekpax (que exige deshabilitar el modulo like de Elgg)
- kpax_theme_responsive
- html5

También debemos implementar los métodos necesarios para la petición de servicios a la aplicación Java Kpax servida desde JBoss en forma de servicio web.

Hemos comprobado que el plugin Gameserver va a tener fuertes dependencias con el plugin Kpax, puesto que el fichero en el que se declaran los métodos, es una clase PHP definida para el plugin Kpax, que no se puede portar ni extender, según las pruebas preliminares realizadas.

Kpax implementa un tipo nuevo de objeto en Elgg, el juego. El objeto juego se deriva del objeto blog y se da de alta en la base de datos Elgg como nueva entidad. Parte de la gestión del plugin Kpax sobre los juegos tiene lugar a través de la bases de datos de Elgg puesto que se crea previamente en Elgg un objeto juego, antes de que sea salvado en la base de datos Kpax.

Hemos utilizado como premisa de la gestión de juegos que no se utilice la base de datos Elgg, sino la base de datos Kpax. Al establecer la descripción del sistema, tal y como está dispuesto el entorno de desarrollo, observamos que ambas bases de datos residen en el mismo servidor MySQL. Esto es por comodidad, pero nos pareció que se establecía a modo de restricción en el sistema que la base de datos Kpax sólo podía ser accedida desde la capa DAO de la aplicación Java.

De la premisa anterior se dedujo que la base de datos Elgg sólo podría ser accedida desde la red social, y realmente esto no es así, hay varias funcionalidades del servicio web que lanzan operaciones contra la base de datos Elgg, así como componentes de Kpax que posteriormente han sido portados a Gameserver que también la acceden.

El plugin Kpax define una clase Kpaxsrv.php que contiene parte de los métodos REST que están publicados en el servicio web. En concreto define los métodos que serán llamados desde el plugin Kpax. La opción disponible para la utilización de este fichero es referenciarlo desde el nuevo plugin utilizando la ruta original que apunta al plugin Kpax mod/Kpax/lib.

Esto nos ha permitido aprovechar parte de las funciones portadas desde Kpax a Gameserver, pero nos ha obligado a dar de alta los nuevos métodos añadidos en esta

clase Kpaxsrv.php y a indicar la dependencia en el fichero manifest.xml del nuevo plugin, puesto que Elgg dará un fallo si se intenta habilitar el plugin sin haber activado antes Kpax.

Como conclusiones del estudio de viabilidad podemos comentar que la creación de un nuevo plugin que se ajuste a los requerimientos y que contenga parte de la funcionalidad de Kpax es factible, pero dará lugar a un plugin con una fuerte dependencia del original.

ANALISIS DEL SISTEMA

COMPONENTES DEL SISTEMA

Tras recibir los requisitos, comenzamos a analizar los componentes del sistema. Se nos proporciona una máquina virtual que contiene una maqueta del sistema completamente funcional, lo que acelera el análisis.

Esta maqueta está compuesta de:

- Un entorno de desarrollo Java, en concreto Eclipse Juno, que contiene una aplicación Java llamada Kpaxsrv.
- La herramienta de gestión de proyectos Apache Maven.
- Un servidor de aplicaciones JBoss en el que está desplegada esta aplicación en forma de servicio web, junto con los componentes Jersey, Spring e Hibernate que requiere para operar.
- Un entorno Apache con módulos PHP y base de datos MySQL que contiene:
  la plataforma Elgg desplegada con el nombre Kpax.
- El conjunto de plugins que forman la aplicación Kpax corriendo en la plataforma Elgg Kpax: Apiadmin, Kpax, LikeKpax (que exige deshabilitar el modulo like de Elgg), Kpax_theme_responsive y HTML5
- La base de datos correspondiente a la plataforma Elgg.
- Una segunda base de datos Kpax cuyo esquema de tablas se mostrará más adelante.

Como ya comentamos, ambas bases de datos, Elgg y Kpax residen en el servidor MySQL del sistema LAMP, pero desde el punto de vista funcional, sólo la aplicación Java Kpax accede a la base de datos Kpax.

Existen pues dos componentes principales del proyecto Kpax:

- Una aplicación Java Kpax:
  Es servida desde un web server JBoss.
  Accede a una base de datos MySQL a través de Hibernate, una herramienta que mapea objetos a cada una de las tablas de una base de datos relacional como MySQL. Estos mapeos se definen mediante ficheros xml y facilitan el envío y recuperación de datos desde las clases Java.
- Una aplicación PHP Kpax:
  Está integrada dentro de la plataforma Elgg.
  Utiliza una clase php en la que se definen los métodos publicados por el servicio web. El intercambio de información en formato Json tiene lugar a través del protocolo HTTP.
  Genera la parte de visualización desde Elgg.

ENTORNO DE DESARROLLO

Este es un esquema de todo el sistema:

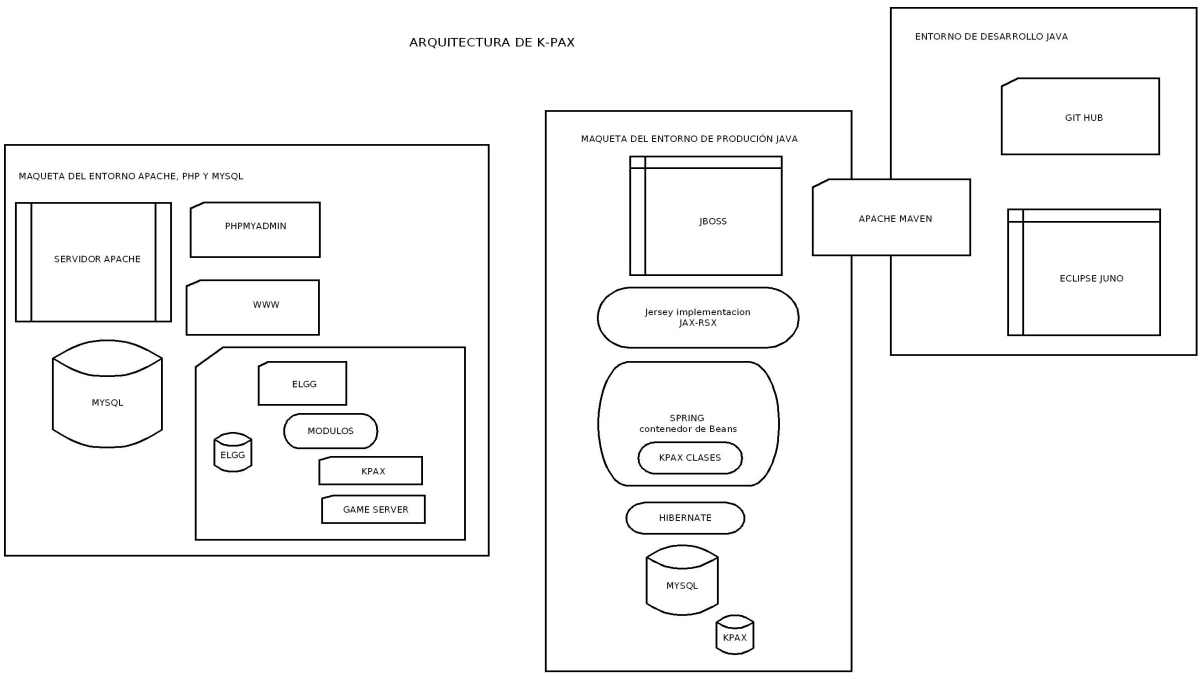

Figura 5 Esquema del sistema, incluyendo las herramientas de desarrollo

La aplicación Java se ha desarrollado desde un IDE [8] Eclipse. Se ha utilizado Maven para la gestión de dependencias de esta aplicación, de modo que las librerías requeridas que no estén disponibles, puedan ser obtenidas desde la web de un modo casi desatendido. La aplicación se compone de 5 paquetes principales de clases:

uoc.edu.svrKpax.busines	Conjunto de clases con la lógica de la aplicación.
uoc.edu.svrKpax.dao	Conjunto de clases utilizadas para el acceso a datos a través de Hibernate.
uoc.edu.svrKpax.vo	Conjunto de clases en las que se mapean cada una de las tablas de la base de datos. Todas tienen una estructura similar, se definen los nombres de los campos (columnas) como variables y se definen un método get y otro set para cada uno de los campos.
uoc.edu.svrKpax.rest	Conjunto de clases utilizadas para el acceso http gestionado por Jersey.
uoc.edu.svrKpax.utils	Conjunto de herramientas de seguridad para la autenticación y certificación de los juegos.

Dentro del paquete uoc.edu.svrKpax.rest, existen tres clases imprescindibles para el funcionamiento del servicio web, que se referencian a continuación. El plugin Gameserver

utilizará principalmente games.java:

La clase games.java	Se definen los métodos publicados por el servicio para que sean accedidos desde Elgg.
La clase jsonp.java	Se ocupa de las peticiones desde otras redes sociales a través de la funcionalidad jason with padding
La clase user.java	Se ocupa de la validación contra la base de datos Elgg y contra otras plataformas desde las que se admite el acceso.

Bajo la carpeta Referenced Libraries se recogen las librerías gestionadas por Maven:

- Componentes Hibernate para mapear las querys sql a objetos y viceversa.
- Componentes de acceso a MySQL tipo ODBC/JDBC.
- Servicios web, utilizando una implementación de servicios Web JAX-RS: Java API for RESTful Web Services llamada Jersey, que hemos comentado en la descripción de tecnologías.

Para que todo el sistema funcione tras la compilación del código y el despliegue en el servidor JBoss es necesario que las instancias de las clases se mantengan en memoria y además que se tengan en cuenta las relaciones entre ellas, es decir, que clase necesita a otra sin que se produzcan relaciones circulares. Estas funciones las lleva a cabo el componente Spring. Por una parte lee las anotaciones java insertadas en el propio código de cada clase y por otra parte utiliza el fichero xml de configuración llamado ApplicationContext.xml, que se encuentra en el directorio src/main/sources.

Este fichero debe ser modificado en caso de crear nuevas clases en la aplicación Java. Es necesario definir nuevas dependencias de clases para que Spring sepa de que forma se relacionan las nuevas clases con las ya existentes.

A continuación mostramos un fragmento de este fichero, donde podemos ver las relaciones entre los objetos o beans instanciados a partir de las clases. Existen dos tipos de declaraciones, para objetos de acceso a datos Dao y para objetos de la capa Bussines:

```
  <!-- DAOS -->
  <bean id="sDao" class="uoc.edu.svrKpax.dao.SessionDaoImpl">
    <property name="sessionFactory" ref="sessionFactory" />
  </bean>

   <bean id="gDao" class="uoc.edu.svrKpax.dao.GameDaoImpl">
    <property name="sessionFactory" ref="sessionFactory" />
  </bean>
….................<!-- Resto de declaraciones -->
<!-- GAMES -->
  <bean id="gBo" class="uoc.edu.svrKpax.bussines.GameBOImp">
    <property name="sBo" ref="sBo" />
    <property name="gDao" ref="gDao" />
    <property name="gvDao" ref="gvDao" />
    <property name="cDao" ref="cDao" />
  </bean>
….................<!-- Resto de declaraciones -->
</beans>
```

La información de este fichero se completa con anotaciones java, por ejemplo, la clase Games.java del paquete REST, en el que se definen los métodos ofrecidos por el servicio Web, lleva numerosas anotaciones @inject con las clases necesarias para cada método definido.

PLUGIN KPAX EN ELGG

Esta es una vista de la pantalla del plugin Kpax previamente desarrollado. Muestra indicaciones para los desarrolladores de juegos y mediante un par de botones, da acceso a la lista de juegos propios y al formulario desde el que se dan de alta nuevos juegos:

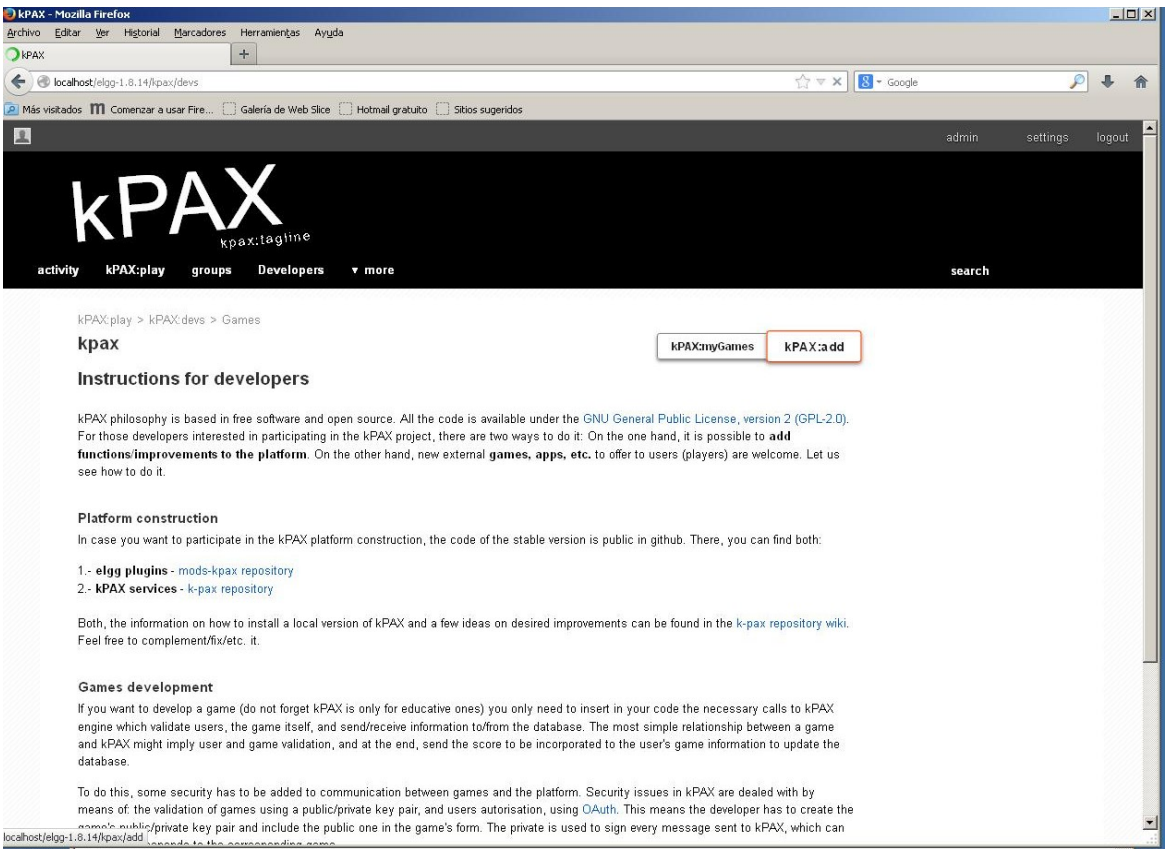

Figura 8 Captura de imagen del plugin Kpax ejecutándose en Elgg

Elgg sigue un modelo MVC [45] para su desarrollo, además de requerir ciertas directrices para el desarrollo de componentes en PHP, entre las más importantes está la estructura de carpetas y ficheros obligatoria y subcarpetas con el nombre del plugin bajo cada una, para que distintos plugins puedan tener ficheros con nombres comunes como add, delete o save sin interferirse unos a otros. Todos los plugins deben situarse en la carpeta /mod y la carpeta mod/nombre-del-plugin se considera su directorio raíz.

Estos son los principales ficheros y carpetas:

Fichero manifest.xml Fichero informativo con nombre, versión y otras características del plugin. En él se definen las dependencias de otros plugins

Fichero start.php Fichero que posee principalmente dos funciones:

- Función init, en la que se registran los componentes activos para todas las páginas del plugin: librerías, acciones, eventos, entidades, elementos activos de las páginas como botones o pestañas, y la propia función page_handler.
- Función page handler, en la que se define y registra la forma de acceder a cada página que forma parte del plugin.

Carpeta Actions Contiene los ficheros php que ejecutarán las acciones definidas en el plugin.

Carpeta Languages Como mínimo contiene el fichero en.php.

Carpeta Pages Contiene las páginas definidas en el fichero init

Carpeta Views Contiene las vistas, formularios y ficheros css, toda la parte de presentación de la información

Carpeta Lib Puede contener clases php o ficheros con funciones auxiliares que pueden ser llamadas desde varios puntos.

Sobre esta estructura se podrán añadir las distintas funcionalidades durante el desarrollo. Esta es una vista de la estructura de carpetas y ficheros de la aplicación Kpax en PHP:

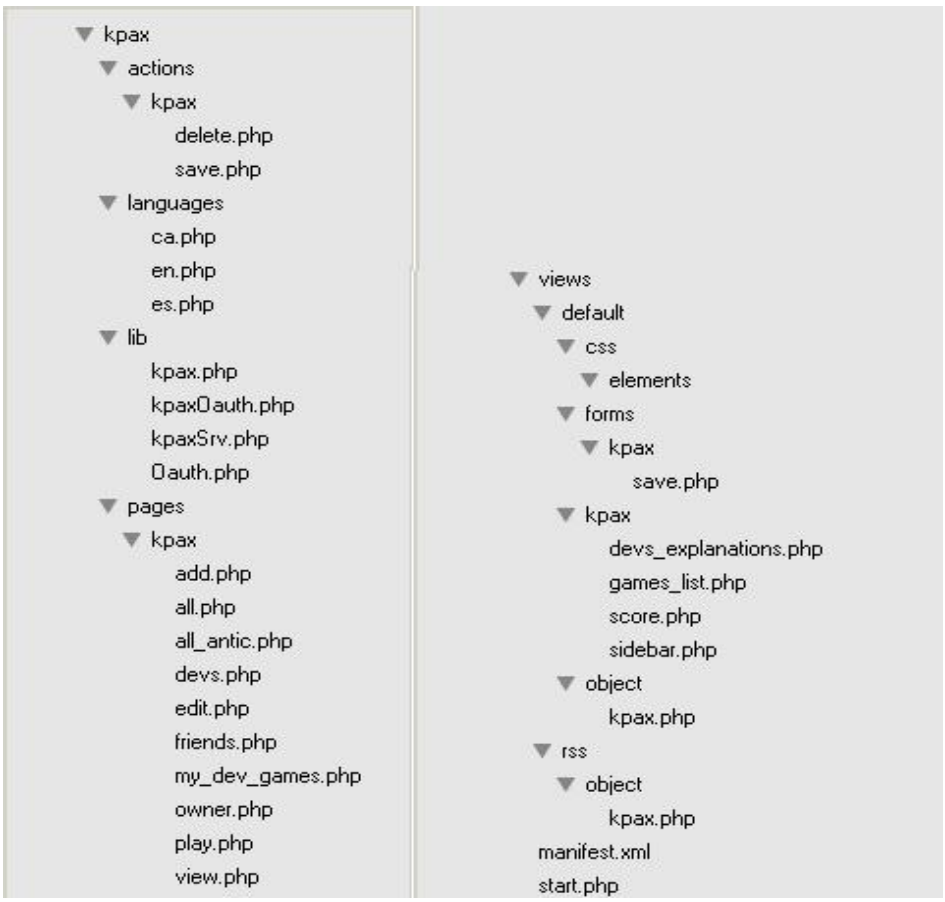

Figura 9 Árbol de directorios y ficheros contenidos en el plugin Kpax

No podemos hacer un recorrido exhaustivo sobre cada fichero, pero trataremos de hacer mención de una serie de puntos que tendrán relación con nuestro propio desarrollo.
El plugin Kpax sigue las directrices de Elgg separando en carpetas distintas las páginas, las vistas, el formulario y las acciones, cada una con su fichero php correspondiente.

La plataforma de servicios Elgg facilita la internacionalización mediante la función elgg_echo() combinado con el uso de ficheros de idioma. Para definir cualquier tipo de frase o etiqueta que deba mostrarse al usuario, se utilizará esta función en lugar de un echo normal, entrecomillando una cadena de texto descriptiva con items separados por dos puntos, de esta forma:

```
elgg_echo('gameserver:gameitem')
```

Los ficheros de idiomas mapean estas cadenas de texto al idioma que se desee, indicando lo con dos caracteres de acuerdo con una tabla de internacionalizacion. Esta es la estructura básica de uno:

```
$mapping = array (
'gameserver:addGame' => 'Add Game to Server',
'gameserver:sidebar:addGame' => 'Add Game to my Server'
);
add_translation('en',$mapping);
```

Cada etiqueta forma parte de un array llamado mapping al que la función add_transation asocia el id de idioma en este caso inglés.
En los settings para cada usuario de Kpax puede modificarse el idioma que se aplicará.

El fichero más importante desde el punto de vista del nuevo desarrollo es la clase php KpaxSrv definida en la ruta: \Kpax\lib\KpaxSrv.php

Esta clase define un objeto KpaxSrv, con unos métodos que producen mensajes http tipo Get o Post principalmente, necesarios para interrogar al servicio web srvKpax. Desde el punto de vista del plugin, este objeto actúa como interface, hacia el resto del sistema.
Para llamar a los métodos dentro del código PHP, es necesario crear un nuevo objeto y llamar al método deseado.

Esta clase posee un constructor, un método privado service, invocado por el resto de métodos, que produce la plantilla del mensaje HTTP:

```
 private function service($action, $type = "GET", $body = "", $header =
"application/json") {
    $url = $this->oauthKpax->getSignature($type, $this->url . $action);
    $options = array('method' => $type,
      'header' => 'Content-type: ' . $header,
      'content' => $body);
    $type_post = array('http' => $options);
    $context = stream_context_create($type_post);
    return file_get_contents($url, false, $context);
  }
```

Como ejemplo de un método ya implementado, tomaremos el siguiente, addGame:

```
  public function addGame($campusSession, $name, $idGame, $idCategory,
$creationDate) {
    $body = 'secretSession=' . $campusSession . '&name=' . $name .
'&idGame=' . $idGame . "&idCategory=" . $idCategory . "&creationDate=" .
```

```
$creationDate;
    return $this->service("game/add", "POST", $body);  }
```

Se define la función addGame que recibirá los cinco parámetros indicados, desde el código PHP. La función construye un cuerpo del mensaje en forma de array e invocará a la función service para que construya un mensaje http con la acción “game/add”, cambiará el método por defecto de service que es get por un post, y finalmente enviará el array de parámetros recibidos por el método.

El mensaje http es recibido en el servidor JBoss que a través de su componente jersey, consultará en el componente games.java del paquete REST, el método que posea la misma ruta en la anotación java @path. A partir de ahí, el servidor dará si procede una respuesta.

El plugin Kpax posee un sistema complejo de autenticación y validación que se apoya en la la clase php KpaxOauth, así como en el plugin apiadmin, y en las clases java situadas bajo el paquete util, como Oauth. Kpax requiere la instalación de una clave RSA propia de la aplicación Kpax que se utiliza para autenticar cada petición HTTP intercambiada y para dar de alta y almacenar las claves RSA correspondientes a cada usuario que desee añadir juegos al sistema. La clave pública que comparten la aplicación Elgg y el servicio web para que sea posible el intercambio de datos debe añadirse al menos a los siguientes ficheros:

- En la clase java ConstantsKpax.java del paquete util como constante ELGG_API_KEY
- En la clase php KpaxSrv.php como constante apiKey
- En el fichero api_key.php del módulo apiadmin

Los ficheros que se han portado al nuevo plugin y en algunos casos realizado pequeñas modificaciones, 0son los siguientes:

Kpax\views\default\Kpax\devs-explanations.php	Página principal de explicaciones para desarrolladores
Kpax\views\default\forms\Kpax\save.php	Formulario para salvar el juego.
Kpax\actions\save.php	Acción que ejecuta el guardado del juego.
Kpax\lib\Kpax.php	Librería auxiliar con funciones accedidas desde otras páginas.

El plugin Kpax define en la base de datos de Elgg un nuevo objeto Kpax, derivado de un objeto blog, y al crear esta nueva entidad[46], define un subtype con valor numérico 6.

Para ello se añade esta línea al fichero start.php del plugin Kpax:

```
elgg_register_entity_type('object', 'Kpax');
```

Desde la acción save, cada juego es creado como un objeto o entidad a partir de sus propiedades, algunas de ellas adquiridas desde el formulario desde el que se da de alta el juego y salvado en la tabla de Elgg elgg_entities con un id, el id del usuario que lo da de alta como propietario y un conjunto de metadatos. A continuación, este juego, es salvado en la tabla de Kpax game, a través del método addGame que invoca al servicio web. El proceso de salvado en Elgg debe ser previo, porque sino, el juego no contaría con un ID de juego.

La tabla que relaciona a los juegos con sus propietarios es elgg_entities y se encuentra en la base de datos Elgg. Existen varias funciones PHP no estándard, propias del framework de Elgg como elgg_get_entity o elgg_get_logged_in_user_guid que permiten obtener datos de esta tabla desde Elgg. Las hemos utilizado como apoyo para obtener el id del desarrollador que desea ver sus juegos, pero no para consultarlas desde el web service, a pesar de que el web service sí hace consultas a Elgg sobre los usuarios logeados.

El plugin Kpax utiliza esta tabla de Elgg para muchas de sus funcionalidades como mostrar todos los juegos, o los juegos de los que un usuario es propietario, en lugar de consultar la tabla games a través del servicio web. Como ya comentamos, los objetos o entidades 'Kpax' de tipo juego, son derivados de las entidades blogs, y las posibilidades que ofrece la adaptación o extensión de las vistas definidas por Elgg para blogs facilita la representación de los datos.

BASE DE DATOS KPAX

Está compuesta por 15 tablas y 3 vistas. Este es un esquema que muestra las tablas con alguna de sus relaciones:

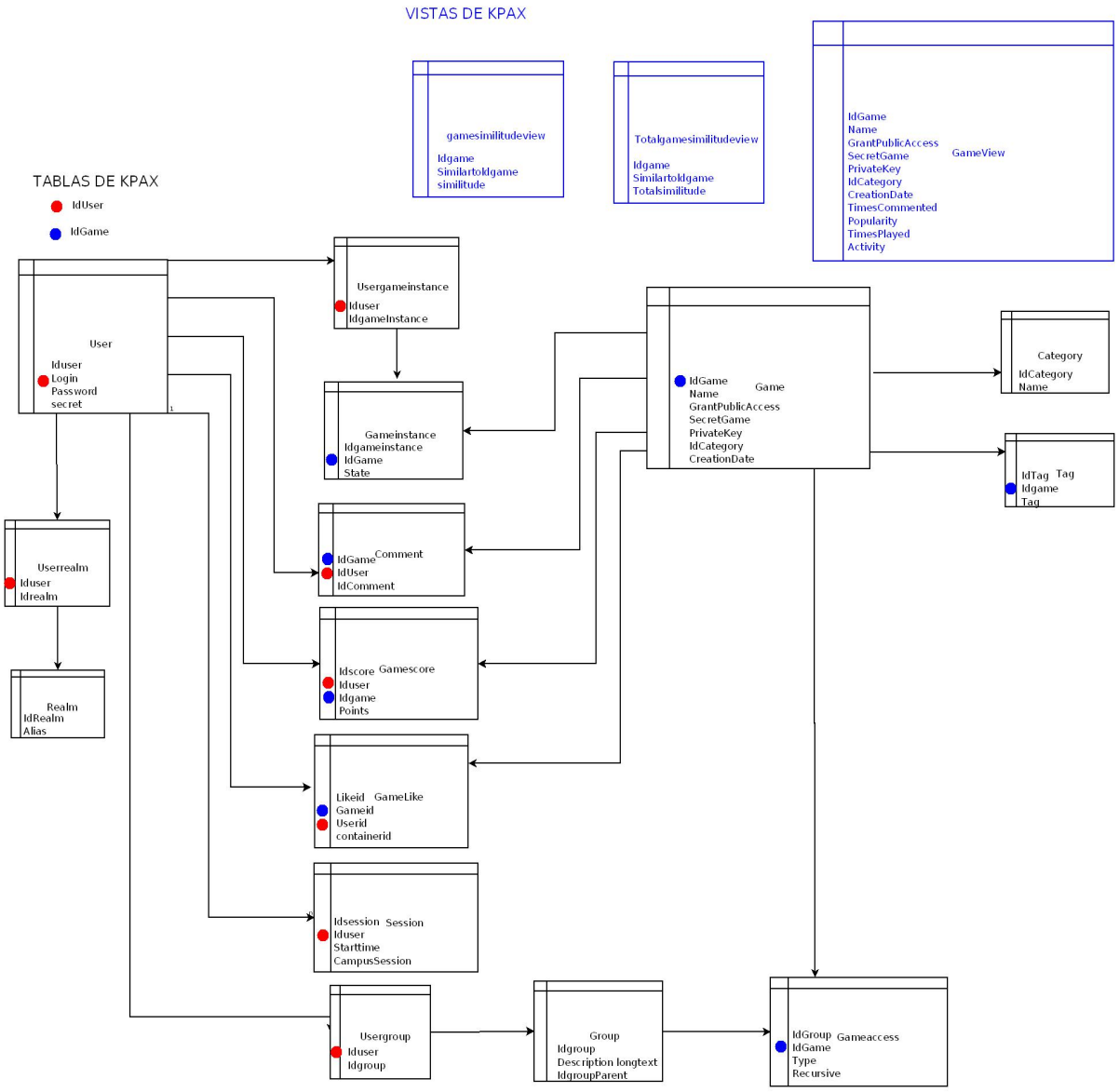

Figura 10 Esquema de tablas para la base de datos Kpax

la tabla User con los campos: Iduser, Login, Password y secret y la tabla Game con los campos IdGame, Name, GrantPublicAccess, SecretGame, PrivateKey, IdCategory, CreationDate y timestamp son las principales, entorno a las que se vertebran las tablas secundarias, completando distintos aspectos y características.

La primera vez que un usuario da de alta un juego, se registra con su fichero de certificado. Se crea una entrada en la tabla user con su login, clave pública y un id de usuario propio, y distinto al asignado en la base de datos de la aplicación Elgg. Esto es así porque el sistema está pensado para admitir login desde otras redes sociales, definiendo para cada tipo de acceso un perfil o Realm. Los distintos realm o ámbitos, se listan en la tabla

correspondiente y la asociación de cada usuario mediante su id con el realm o ámbito que le corresponde, se realiza en la tabla Userrealm.

La tabla Usergameinstance define la asociación entre un usuario y el juego que podría estar ejecutando. Como varios usuarios podrían jugar a la vez un mismo juego, se plantea el concepto de instancia de un juego, como veremos en la siguiente tabla.

La tabla Gameinstance asocia una instancia de un juego con el juego del que proviene, a través del idgame y un estado que indica si la partida se ha comenzado, ha quedado pendiente de acabar o ha finalizado.

El plugin Kpax, como hemos visto, define un nuevo objeto derivado del objeto blog, llamado Kpax en la base de datos Elgg, que es efectivamente el elemento juego con todas sus características. De este objeto se guarda una referencia en la tabla Game.

La tabla Comment relaciona un juego, a través de su id con un comentario, también a través de su id, con el usuario que lo genera. El comentario propiamente dicho se guarda en Elgg.

La tabla Gamescore relaciona a un usuario con el juego que está jugando y con la puntuación obtenida en ese juego.

La tabla Gamelike relaciona a un usuario con los votos de “me gusta” otorgados a un juego

La tabla Usergroup junto con la tabla group relacionan a un usuario con un grupo creado en elgg para cualquier tipo de actividad, competición o debate.

La tabla Gameaccess regula los permisos a grupos sobre juegos.

La tabla Category lista las categorías disponibles para clasificar un juego frente a un id. Este id aparece también como campo en la tabla Game.

La tabla Tag tiene una filosofía distinta, agrupa a la característica Tag o etiqueta asociándola con el correspondiente id y con el juego con el que está relacionada. De esta tabla salen los datos que generan la nube de etiquetas del plugin Kpax.

La tabla Session tiene una relación one to many con la tabla user. Cuando un usario se logea, se crea una sesión asociada. La información de la sesión relativa a inicio de la conexión e id del usuario queda salvada. Sin embargo, no existen campos relativos a fin de sesión o a estado de la misma.

Con esto finaliza la descripción del sistema en su estado inicial.

DISEÑO

Diagrama de Casos de Uso

Este es el diagrama de casos de uso que planteamos inicialmente:

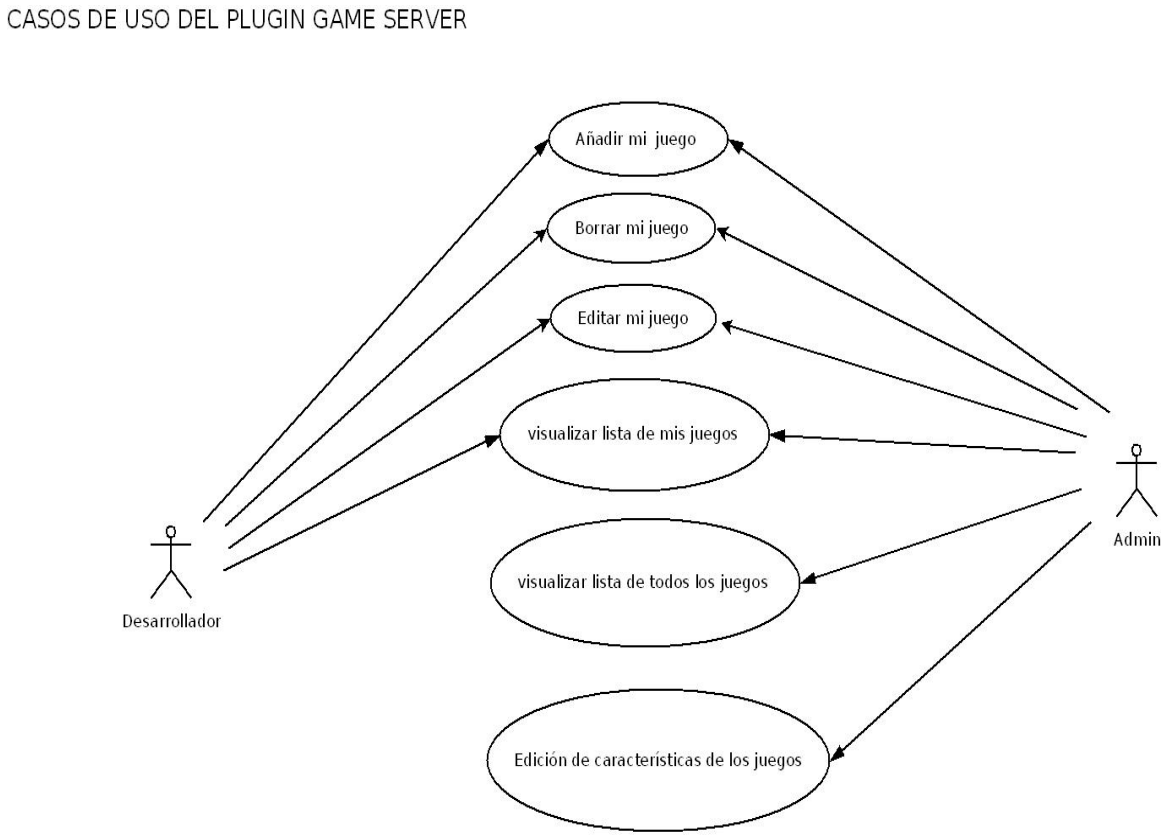

Figura 11 Diagrama de casos de uso para el plugin Gameserver

Casos de uso para el actor Usuario:

Añadir un juego
Produce información nueva en el sistema.
Debe informar de los cambios al sistema.
El sistema debe enviar un mensaje de aceptación de la información o en caso de fallo de un mensaje de error.

Editar un juego
Modifica información existente en el sistema.
Debe informar de los cambios al sistema.
El sistema debe enviar un mensaje de aceptación de la información o en caso de fallo de un mensaje de error.

Borrar mi juego

Adquiere información del sistema
Transforma esta información.
Debe informar de los cambios al sistema.
El sistema debe enviar un mensaje de aceptación de la información o en caso de fallo de un mensaje de error.

Visualizar lista de mis juegos
Adquiere información del sistema.
Necesita una forma de acceder a modificaciones, editar algunos campos de sus juegos.
Ver estado de su juego (aceptado o no).
Debe informar de los cambios al sistema.
El sistema debe enviar un mensaje de aceptación de la información o en caso de fallo de un mensaje de error.

El plugin Gameserver debe ocuparse de altas, bajas y cambios en características.

Casos de uso para el actor Administrador:

Visualizar lista de todos los juegos:
En especial de los nuevos, con sus correspondientes estados pendientes de aceptar, aceptados, etc
Adquiere información del sistema.
Transforma esta información.
Debe informar de los cambios al sistema.
El sistema debe enviar un mensaje de aceptación de la información o en caso de fallo de un mensaje de error.

Editar información de un juego:
Modifica información existente en el sistema.
Debe informar de los cambios al sistema.
El sistema debe enviar un mensaje de aceptación de la información o en caso de fallo de un mensaje de error

Teniendo en cuenta estos casos de uso [48] se ha establecido un esquema básico de páginas del plugin, tratando de seguir el patrón de desarrollo Modelo Vista Controlador[45] [47] que sigue Elgg.

Esquema de las páginas del plugin

Proponemos para el plugin el esquema de páginas mostrado en la figura siguiente. Recordemos que la página debe estar integrada en el sistema de vistas de Elgg. El esquema anterior representa "pantallas" mostradas al usuario y cada una puede corresponderse con varios ficheros para su funcionamiento.

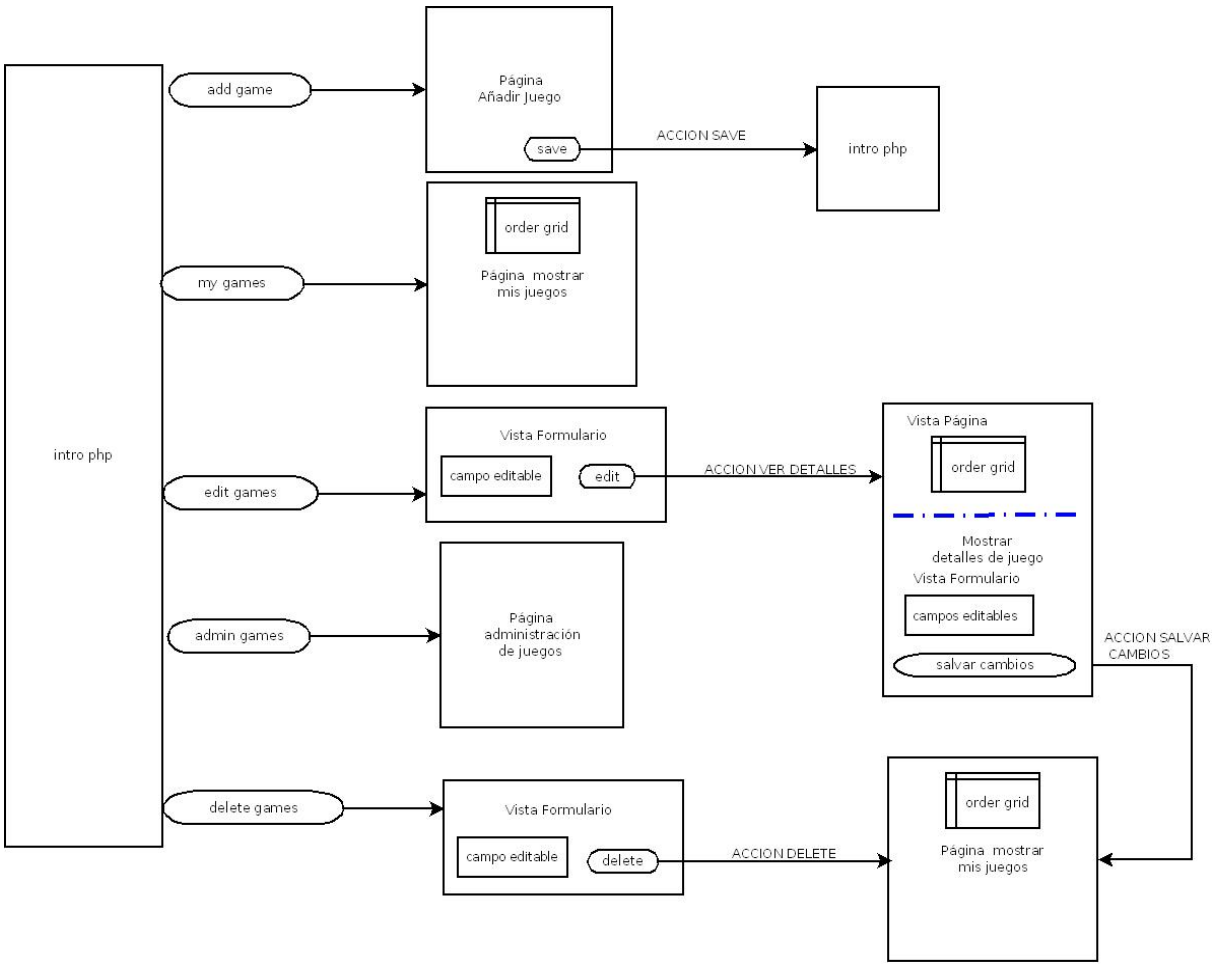

Figura 12 Esquema de las páginas del Plugin Gameserver

El esquema anterior para las páginas del plugin nos servirá de guía y orientación, pero no debemos olvidar como se implementan las páginas en Elgg. Por cada pantalla de presentación al usuario deberían existir como mínimo:

- Un fichero de página que define la disposición de los elementos.
- Un fichero de vista para mostrar datos o de formulario para recogerlos y enviarlos.
- Opcionalmente un fichero con contenido CSS.

Elgg sigue un modelo MVC en el que cada página PHP llama o bien a una vista que muestra resultados obtenidos del sistema, o bien a un formulario. O a ambos, como ocurre en la página en la que se visualizan los detalles de un juego. A través de los botones de submit se desencadenan acciones.

Las acciones no pueden devolver resultados, su función es procesar peticiones, que pueden incluir datos obtenidos de formularios. Con frecuencia se añade a la acción un comando de redirección o forward, para que cuando finalice su proceso, se muestre en pantalla alguna página desde la que poder continuar la interacción con el sistema.

En el caso de las acciones save y delete, no se espera que devuelva más resultados que acción exitosa o acción fallida. Pero en el caso de la acción edit, se espera devolución de resultados, y supone una complicación tener en cuenta que ha de ser la vista la que muestre los resultados al usuario, pero sólo la acción puede recoger los datos enviados por el formulario. En este caso el dato de id de juego la acción lo recoge del formulario, pero ha de transmitírselo a la vista para solicitar al sistema los datos del juego y mostrarlos al usuario. Tras varias pruebas y esquemas, nos quedamos con el propuesto en la figura 12 como definitivo.

Al pulsar la pestaña “Developers” entramos en la página intro, que posee cinco botones en la barra lateral. Cada uno de ellos conduce a las cinco principales páginas del plugin según los requisitos:

- La página “addgame” consta de un formulario en el que se formalizan los datos de un juego y un botón salvar. Finalizada la acción, el usuario es redirigido a la página “mygames.php”.

- La página “mygames” muestra un listado de los juegos en los que el usuario es propietario.

- La pagina “all games” muestra una tabla con un listado de todos los juegos dados de alta en Kpax. La página sólo se muestra para los administradores, en caso de no serlo, el usuario es redirigido a “mis juegos”.

- La página “delete” muestra un formulario sencillo con una caja de texto en la que anotar el ID del juego que se desea borrar y un botón que desencadena la acción delete. Cuando la acción se completa, el usuario es redirigido a la página “mis juegos”, donde puede comprobar si el borrado ha tenido lugar correctamente.

- La página “edit” muestra también un formulario con una caja de texto en la que anotar el ID del juego que se desea editar y un botón. Al pulsar el botón se carga la página gameitem que muestra en la parte superior una tabla con mayor número de detalles del juego solicitado, y en la parte inferior un formulario con cinco cajas de texto correspondientes a los cinco campos que se permite editar, junto con un botón guardar cambios. El botón desencadena la acción que recogerá los datos del formulario y llamará a los métodos necesarios para sobreescribir los datos del juego. Cuando la acción finaliza el usuario es redirigido a la página de detalles del juego, lo que le permite comprobar si los cambios se han producido correctamente o no.

El administrador puede cambiar ciertos campos del juego como la categoría, el estado o si está disponible de forma general o de forma privada, siguiendo un proceso de revisión antes de ofrecer el juego en un listado general de todos los juegos disponibles y accesible para todos los usuarios. Este listado general puede visualizarse desde el plugin Kpax desarrollado con anterioridad.

Nuestro desarrollo sólo incluye, atendiendo a los requisitos, la parte de administración de los objetos, altas, bajas y cambios de campos auxiliares como categoría, plataforma o

nombre. También la disponibilidad o el estado de aceptación. Los campos de ID del juego y del propietario, fecha de alta o claves RSA no pueden ser editados.

Según los casos de uso hemos identificado la necesidad de al menos cuatro métodos para interaccionar con los juegos:

- Añadir juego.
- Listar todos los juegos
- Listar juegos por propietario
- recuperar información de un sólo juego
- Borrar juego.

Para dar una explicación más precisa debemos examinar antes los cambios aplicados en la base de datos Kpax y los correspondientes cambios que se han de realizar en el servicio web tal y como veremos en el apartado siguiente.

DESARROLLO

PRIMER DESARROLLO FALLIDO

Aunque los cambios tienen lugar en varios puntos del proyecto, en el caso de nuestro desarrollo desechado nos vamos a centrar sólo en los cambios aplicados sobre la base de datos Kpax. Hemos creado tres nuevas tablas que se muestran en verde en la figura:

- Nueva tabla Platform
- Nueva tabla State
- Nueva tabla Owner

Añadimos dos campos a la tabla Game:

- idstate el estado del juego (aceptado, pendiente de aceptar)
- idplatform el tipo de plataforma en el que el juego está desarrollado (ios, android....)

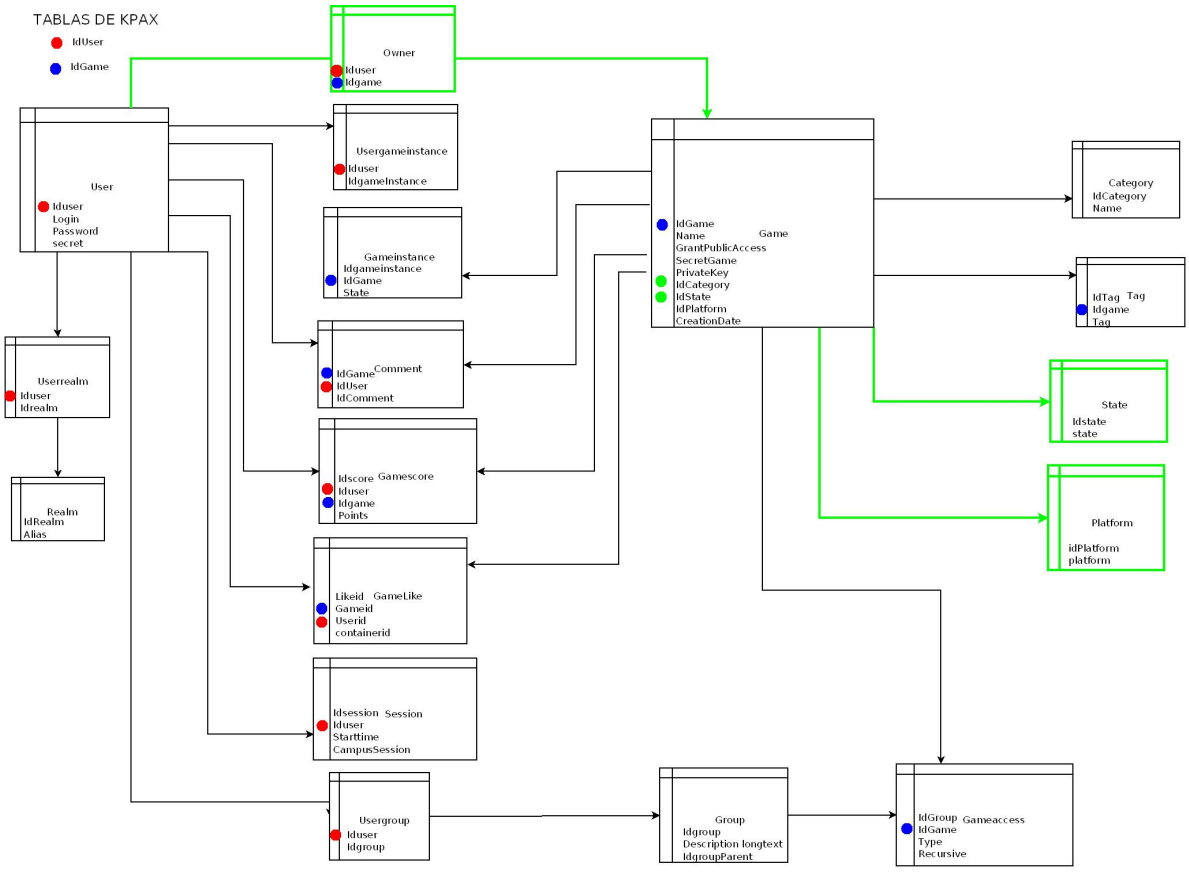

Figura 13 Cambios en la base de datos Kpax durante el diseño fallido

Se añadió una nueva vista AdminGames que muestre todos los campos que necesitamos para administrar los juegos. Creemos que es preferible añadir una nueva vista y no modificar la existente GameView porque ya existen métodos que la utilizan para obtener información.

Obviamos los cambios realizados sobre el proyecto Java para reflejar la existencia de las nuevas tablas con nuevas clases y la modificación de algunas de las existentes, y pasaremos a resumir los problemas encontrados con este diseño.

Es muy problemático que el id de usuario que se pretendía relacionar con el juego en la tabla Owner sea el id almacenado en user. La tabla user se puebla a medida que los usuarios dan de alta un juego por primera vez, pero el id almacenado es autogenerado y no coincide para nada con el almacenado por Elgg en la tabla elgg_entities. Podríamos obtener de Elgg el campo login a través de las funciones del framework, consultar la tabla user para el id y luego consultar la tabla owner, pero sería más farragoso.
En un nuevo diseño nos evitaremos esto creando una tabla developer que guarde el mismo id del usuario que utiliza Elgg frente al id del juego recién dado de alta, y que realice esta operación durante la acción save.

Se ha añadido a la tabla game dos columnas, representando dos características extra de la entidad juego, idplatform e idstate. Esto ha requerido modificar las clases de java relacionadas con game. Además del método addGame perteneciente a la clase games.java del paquete Rest. Sin embargo nos hemos encontrado con la imposibilidad de que el método existente addGame funcionase al añadirle estos dos parámetros. La forma de solucionarlo en el siguiente diseño, añadiendo estos mismos parámetros a la nueva tabla developer no es la más elegante, pero no complica el código y además como veremos más adelante funciona.

DESARROLLO DEFINITIVO

BASE DE DATOS KPAX

Veamos los cambios aplicados sobre la base de datos Kpax, hemos creado tres nuevas tablas: developer, state y platform y dos nuevas vistas adminview y userview, las cinco representadas en verde en el gráfico siguiente:

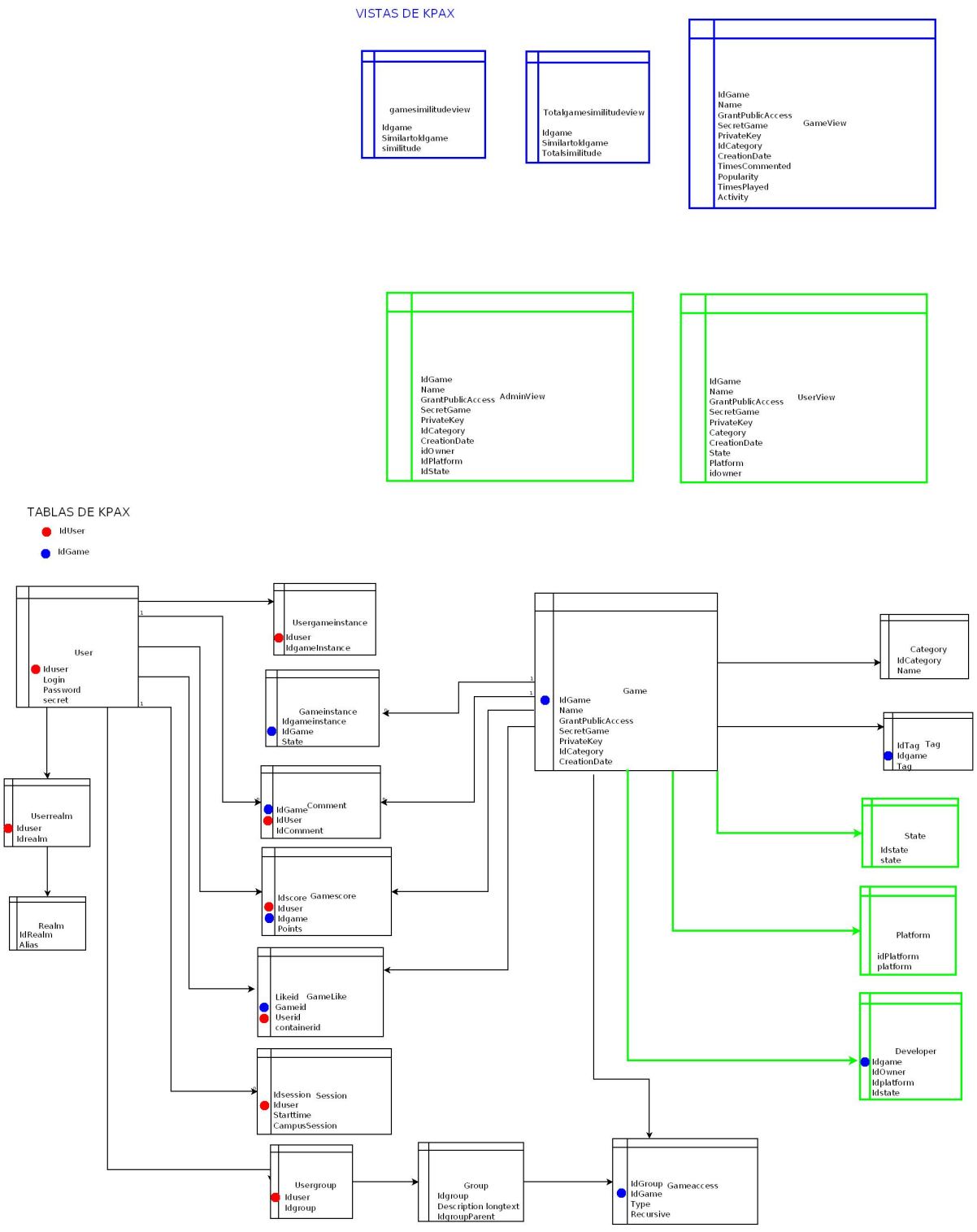

Figura 14 Cambios en la base de datos Kpax en el diseño definitivo

La tabla Developer recoge datos complementarios a la entidad game. La creación de la vista userview nos facilita mostrar la información al usuario de modo más amigable en lugar de utilizar códigos para las características. La creación de la vista adminview, un join de las tablas game y developer no nos obligará a consultar más de una tabla para devolver los datos.

APLICACIÓN JAVA

Para reflejar las modificaciones de BD MySQL Kpax en la aplicación Java, se han realizado los siguiente cambios:

.

Se han creado en total quince nuevos ficheros de clase correspondientes a Developer y a las vistas adminview y userview en los paquetes VO, BO y DAO:

DeveloperVO	userViewVO	adminViewVO
DeveloperBO	userViewBO	adminViewBO
DeveloperBOImp	userViewBOImp	adminViewBOImp
DeveloperDAO	userViewDAO	adminViewDAO
DeveloperDAOImp	userViewDAOImp	adminViewDAOImp

Se ha modificado el fichero AplicationContex.xml, que utiliza Spring para enlazar las clases de acuerdo a como están definidas las depencencias entre ellas.

Se ha modificado la clase Game.REST, dando de alta nuevos métodos que se detallarán a continuación:

1º Metodo addOwner para añadir información a la tabla Developer.
Trabajará complementando la acción de salvado de un juego, que ya añadía un objeto en Elgg y una entrada en la tabla games, con una entrada en la tabla developer. Este método no devuelve nada.

2º Método ListOwnGames para listar los juegos de un usuario dado.
Este método consulta a la vista userview, que combina campos de game, developer, state y platform, presentando la información de forma entendible para el usuario. Devuelve una lista de items en forma de array multidimensional.

3º Método ListAllGames para listar los juegos de todos los usuarios.
También consulta a la vista userview, que combina campos de game, developer, state y platform, presentando la información entendible para el usuario. Esta funcionalidad sólo está disponible para los usuarios administradores. Devuelve una lista de items en forma de array multidimensional.

4º Método getOneGame para listar las características de un sólo juego.
Este método consulta a la vista adminview, que se genera como join de las tablas Game y Developer. La información que ofrece es un poco menos entendible para el usuario, los campos referidos a categorias, plataformas y estado se ofrecen en modo numérico. Devuelve un item en forma de array.

Se ha visto la necesidad de crear algún método más a posteriori:

4º Método delDevelop para borrar de la tabla developer las características

complementarias de un juego que sea borrado. Debe llamarse desde el código junto con el método ya existente delGame. Este método no devuelve nada.

ENTORNO ELGG

Los cambios, como mencionamos con anterioridad, tienen lugar en el plugin Gameserver y en el fichero Kpax\lib\KpaxSrv.php. Es en este fichero en el que se definen los métodos llamados desde el plugin. Se añadieron las siguientes funciones a la clase KpaxSrv.php:

- public function addOwner($campusSession, $idgame, $idplatform, $idowner)
- public function ListOwnGames($campusSession, $idowner)
- public function ListAllGames($campusSession)
- public function getOneGame($campusSession, $idgame)
- public function delDevelop($campusSession, $idgame)

Para una descripción más detallada de la funcionalidad, se puede consultar el apartado anterior referido a las modificaciones en Java.

Página intro

Empezamos a armar el esqueleto del plugin partiendo de los siguientes ficheros:

- manifest.xml
- start.php
- \pages\gameserve\intro.php
- \views\default\gameserver\intro.php

Damos de alta en el fichero start.php los botones de la barra lateral. Por ejemplo el botón add Game:

```
elgg_register_menu_item('page', array( 'name' => 'addgame',
                                       'text' => 'Add Game',
                                       'href' => 'gameserver/addgame',
                                       'context' => array ('gameserver'),
                                ));
```

y la pestaña Developers a través de estas dos líneas:

```
$item = new ElggMenuItem('developers', 'Developers', 'gameserver/intro');
elgg_register_menu_item('site',$item);
```

El fichero Kpax/views/devs_explanations.php se porta desde Kpax como intro.php y se eliminan los botones. El conjunto presenta el aspecto mostrado en la figura siguiente.

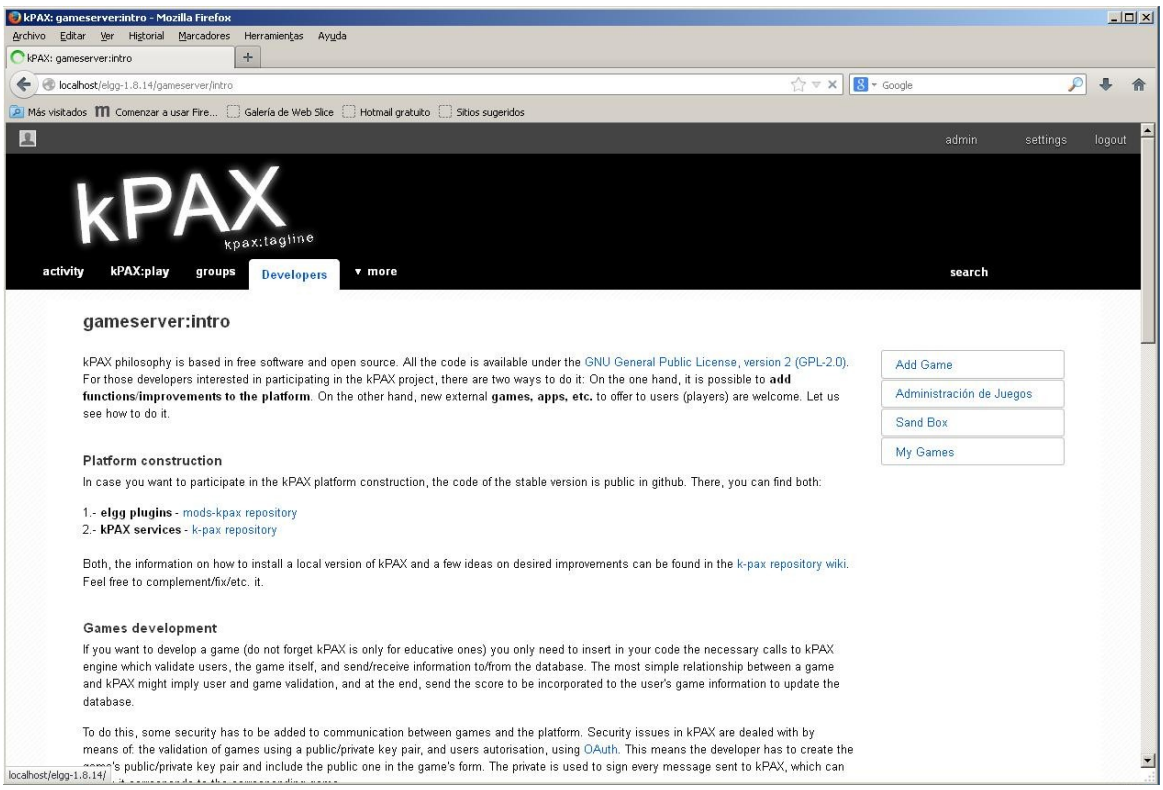

Figura 15 Aspecto inicial de la página principal del plugin Gameserver

El fichero /pages/intro.php define el tipo de vistas y distribución de la página y la vista a la que va a llamar. Elgg deposita el contenido visible en los ficheros de vistas.

Los botones no tienen funcionalidad, así que vamos a añadírsela en los pasos siguientes.

Página add game

Creamos los ficheros:

- \pages\gameserver\addGame.php
- \views\default\forms\gameserve\addGame.php
- \pages\gameserver\save.php
- \lib\gamehelp.php

Referenciamos en start.php la nueva página y la acción que invoca. Como la acción utiliza el método addGame es el momento de registrar en el fichero start la clase Kpaxsrv.php incluyendo la ruta al plugin Kpax. También portamos y registramos la función que reside en \lib\kpax.php.

En este caso se porta desde Kpax el formulario save.php y el fichero con la acción también llamado save.php. La acción utiliza una función de preparación de variables, que se porta desde kpax y se renombra en Gameserver.

La principal modificación del formulario es capturar la variable idplatform, que no se estaba haciendo, y cambiar las checkboxes que recogen el dato de plataforma del juego por un panel de radiobutton. Esto nos asegura la devolución de un valor único y no de un array

como podría ocurrir en el caso anterior si se marcasen varias opciones.

El fichero de acción save.php sufre más modificaciones:
se añade una variable idplatform para recuperar el dato introducido en el formulario y se añade el método addOwner para añadir los datos idgame, idowner, idplatform e idstate a la nueva tabla developer en el momento en que se salva el juego. Cuando la acción finaliza el usuario es devuelto a la página intro.

Esta es una imagen del formulario desde el que se salvan los juegos:

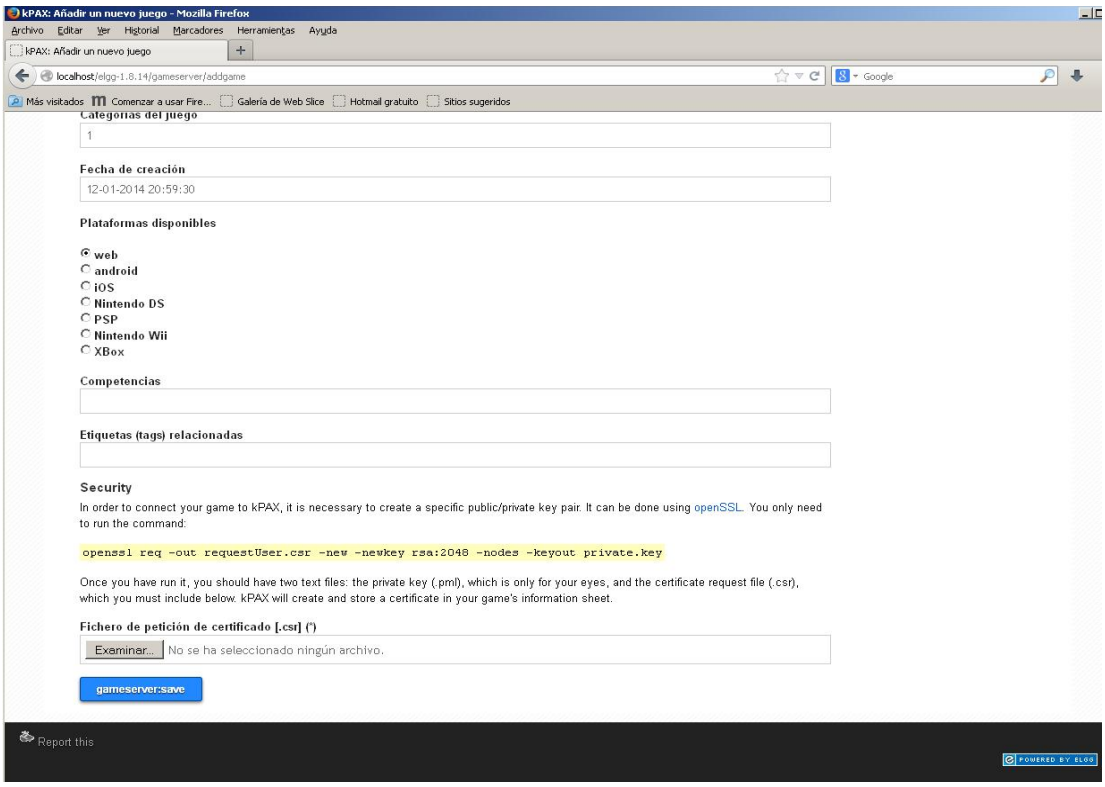

Figura 16 Vista del formulario add game

Página all games

Seguimos con el siguiente botón llamado All games. Añadimos los siguientes ficheros:

- \pages\gameserver\admingames.php
- \views\default\\gameserver\admingames.php

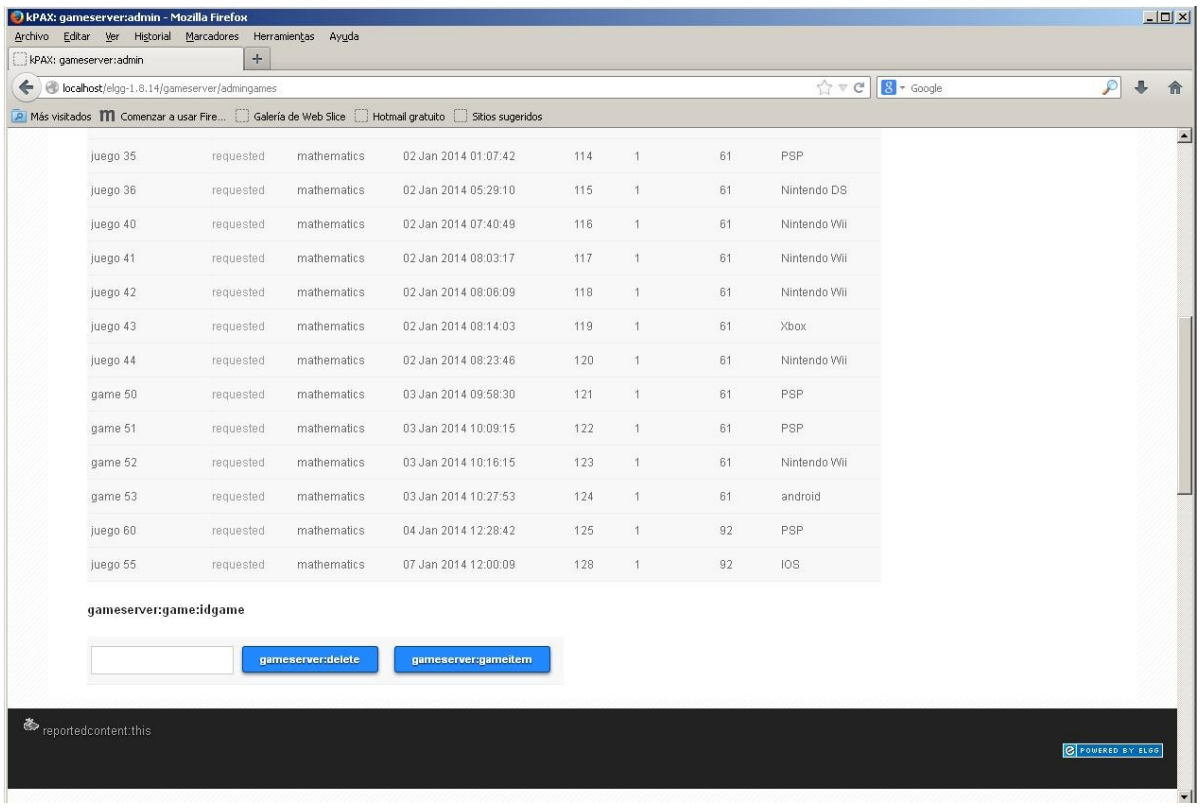

Figura 17 vista de la página all games

Desde el fichero admingames del directorio pages, se llama a la vista admingames, que devuelve una lista de todos los juegos registrados en forma de tabla, consultando la vista userview mediante el método ListAllGames. El formato de la tabla viene dado por un fichero css situado en \views\default\gameserver. Este fichero es un diseño público obtenido en [49] al que se le han modificado los colores. Este estilo se aplicará también a la página producida por mygames.

Página my games

Cambios para el botón my games. Añadimos los siguientes ficheros:

- \pages\gameserver\mygames.php
- \views\default\\gameserver\mygames.php

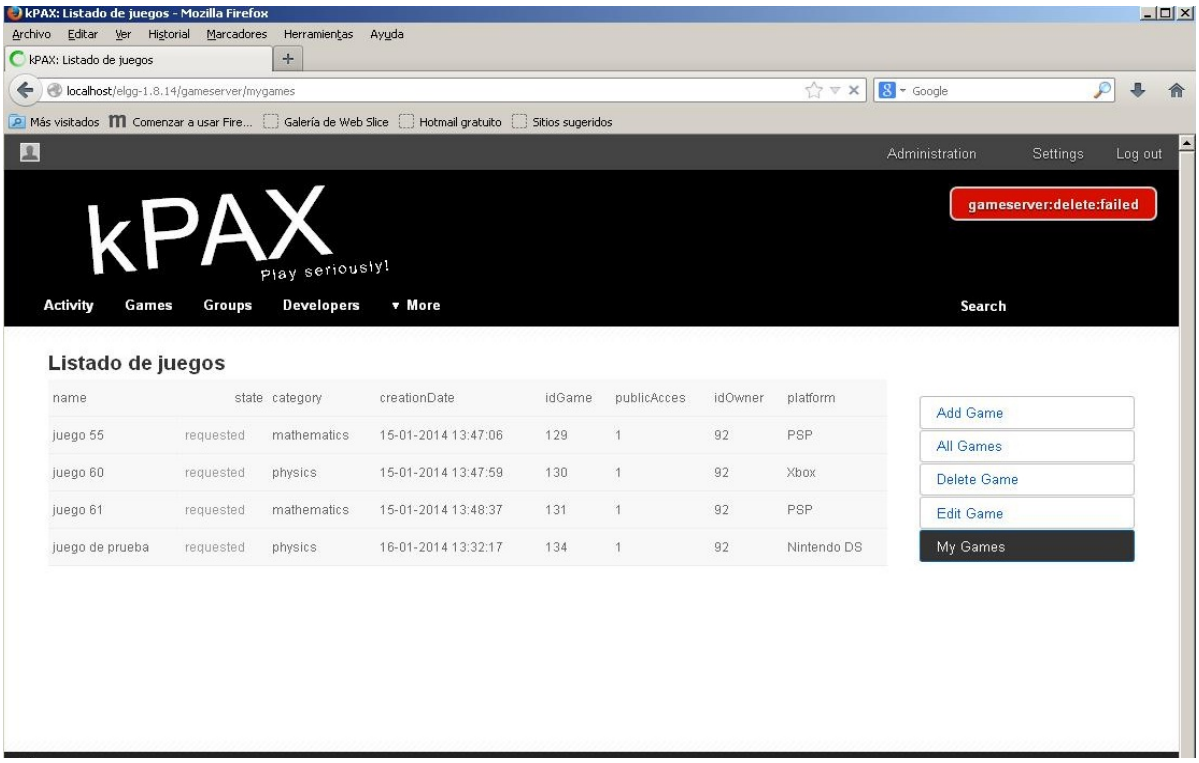

Figura 18 Vista de la página my games

Desde el fichero mygames del directorio pages, se llama a la vista mygames que devuelve una lista de los juegos propiedad del usuario logeado en forma de tabla, consultando la vista userview mediante el método ListOwnGames. El formato de la tabla viene dado por el mismo fichero css situado en \views\default\gameserver que se aplica al caso anterior. El fichero es un diseño público obtenido en [49] al que se le han modificado los colores.

En este momento es necesario añadir dos botones más a la barra lateral derecha, por lo que el aspecto de la página de inicio tras añadir en el fichero start.php los botones y las referencias a las páginas que llaman queda de esta forma:

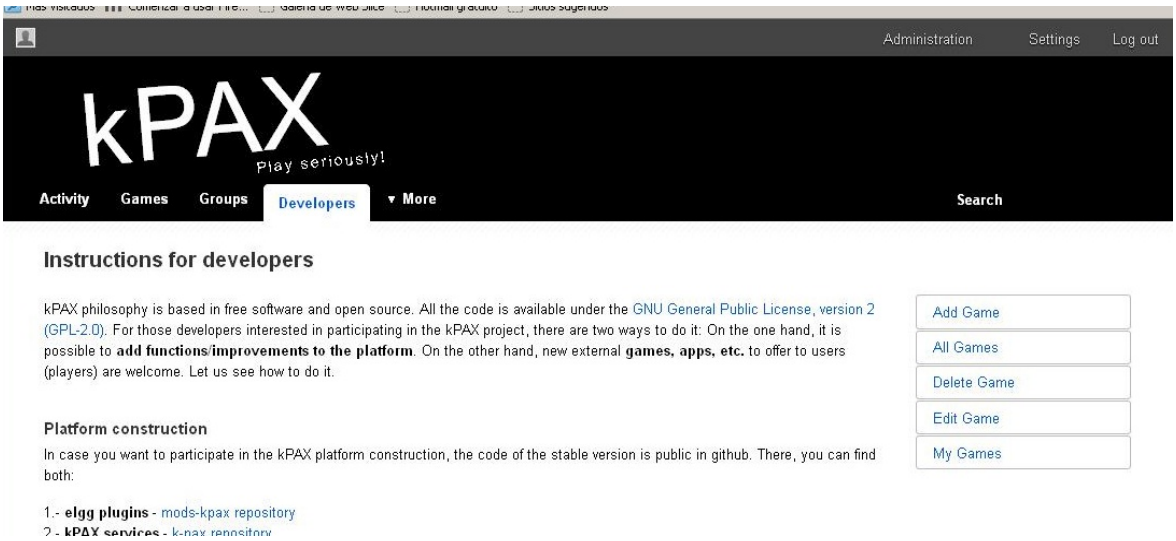

Figura 19 Vista de la página intro tras añadir los botones edit y delete.

Página delete

Cambios para el botón “delete”. Añadimos los siguientes ficheros:

- \pages\gameserver\delete.php
- \views\default\forms\gameserver\delete.php
- \actions\gameserver\delete.php

El formulario delete contiene únicamente una caja de texto, a la que se debe añadir el id de juego que se desea borrar, y un botón delete, que activa la acción delete. La acción revisa que el juego pertenezca al usuario logeado antes de borrarlo. Para ello realiza una petición al servicio web a través del método ListOwnGames para comparar los idgame obtenidos del web service con el obtenido del formulario. Si coinciden, la acción se ejecuta. Desgraciadamente la función delete no ha podido ser completada, falla actualmente en el plugin.

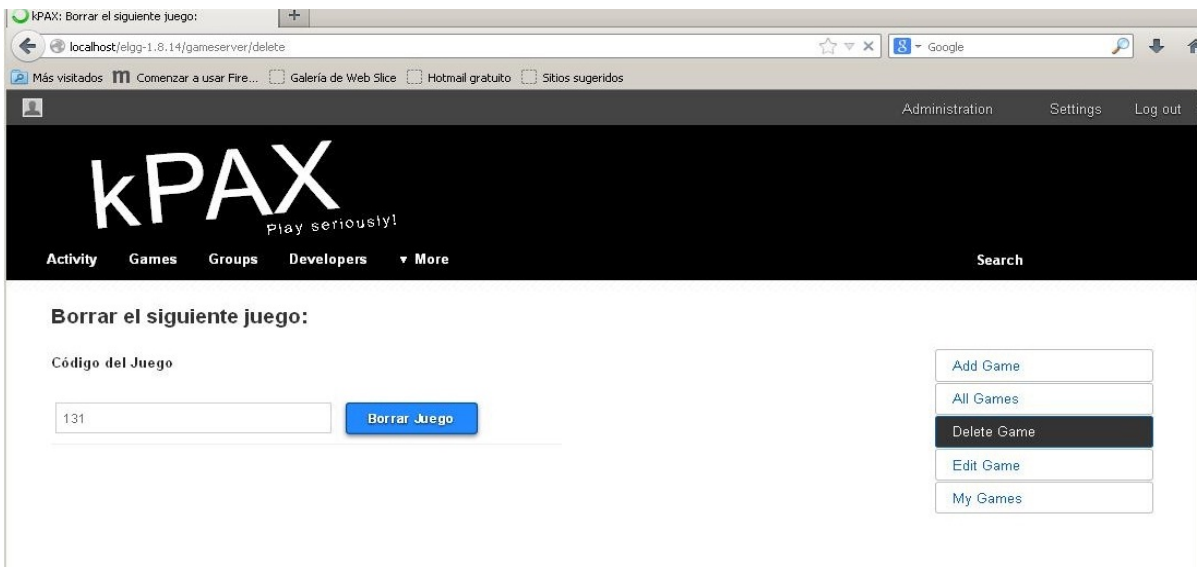

Figura 20 vista de la página delete

Página edit game

Cambios para el botón edit. Añadimos los siguientes ficheros:

Para el formulario Edit games:

- \pages\gameserver\editgame.php
- \views\default\forms\gameserver\gameitem.php
- \actions\gameserver\gameitem.php

El formulario gameitem contiene únicamente una caja de texto, a la que se debe añadir el id de juego que se desea editar, y un botón detalles del juego, que activa la acción gameitem. La acción debería hacer un chequeo similar al que hace la acción delete, pero no es así. La acción se limita a obtener el dato idgame del formulario y dejarlo disponible en la variable global $SESSION para que lo recoja la vista que mostrará los detalles de un juego. Esta acción se ha comprobado que sí funciona.

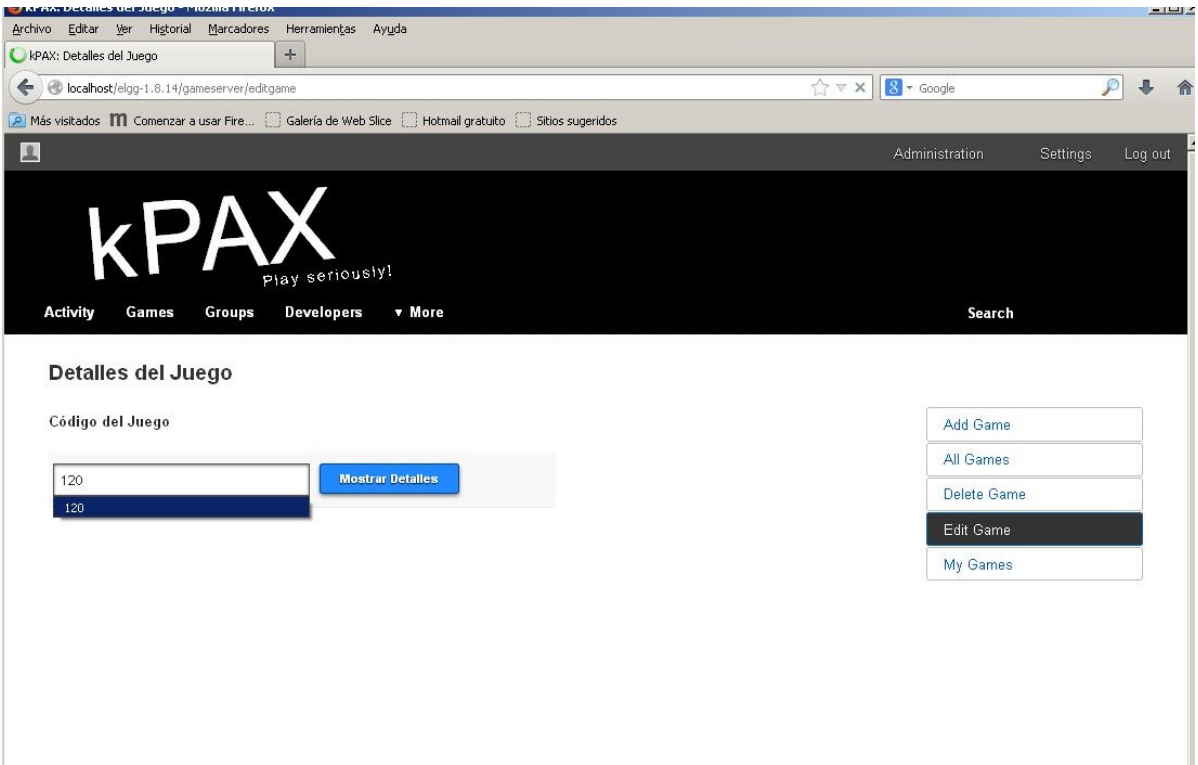

Figura 21 Aspecto del formulario para editar y mostrar detalles de un juego.

Página game item

Finalmente, la acción lanzada por la página editgame envía al usuario a una última página compuesta por una vista en la que se muestran los detalles de un sólo juego en la parte superior mientras que en la inferior observamos un formulario con cinco cajas de texto en las que se añaden los campos que se desea modificar y un botón que lanza la acción de salvar cambios.

Para generar la vista de detalles del juego y la acción que desencadena se requieren los siguientes ficheros :

- \pages\gameserver\gameitem.php
- \views\default\gameserver\gameitem.php
- \views\default\forms\gameserver\savechanges.php
- \actions\gameserver\savechanges.php

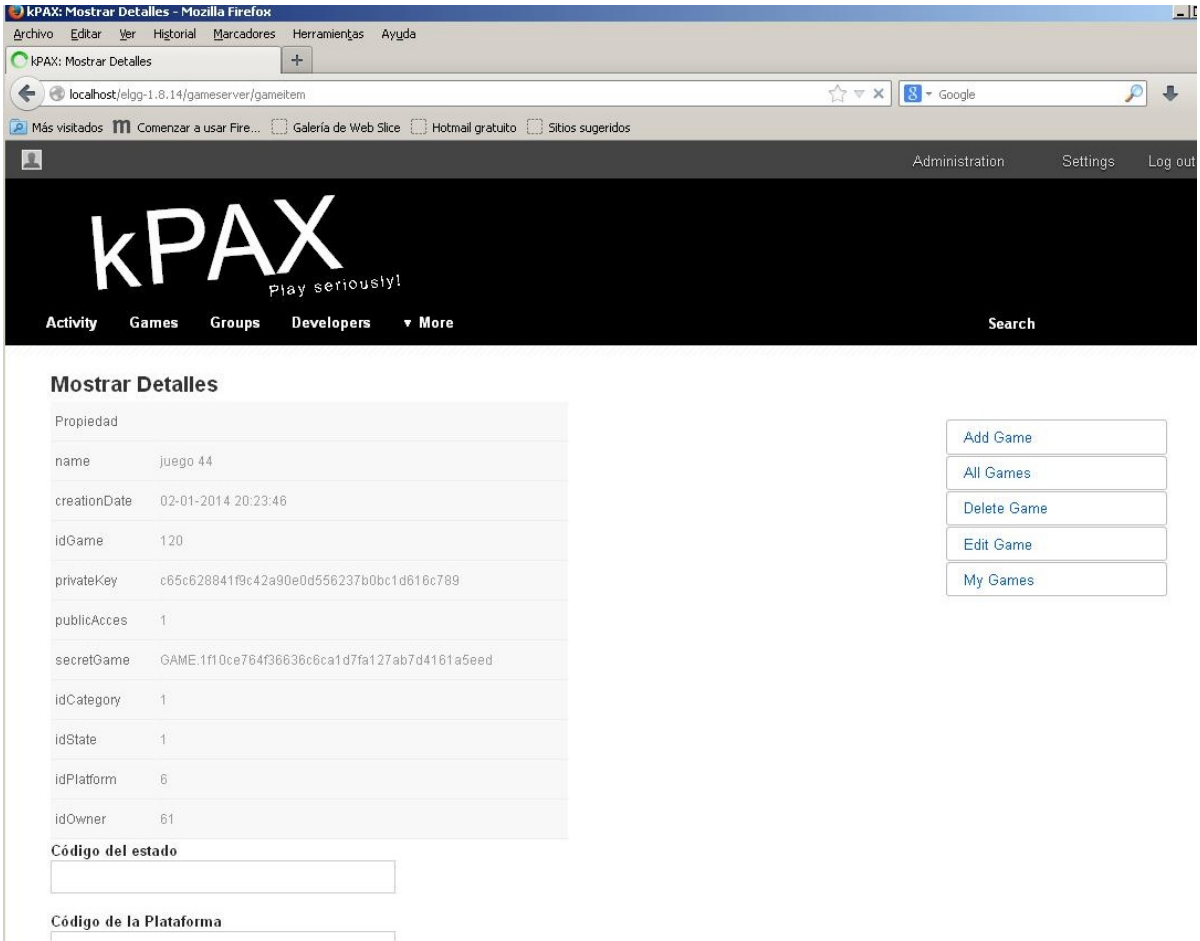

Figura 22 Aspecto de la parte superior de la página. Detalles del juego solicitado.

Los detalles obtenidos por la vista gameitem son solicitados a la tabla adminview de Kpax por el método getOneGame. No todos los campos son editables. Ni el id del juego o del propietario, ni los ficheros de clave, o la fecha de creación. Pueden modificarse estado, plataforma, nombre del juego, categoría y estado de publicación. Falta afinar los permisos para que algunos de los campos, como el estado, no puedan ser modificados ni por el propietario del juego.

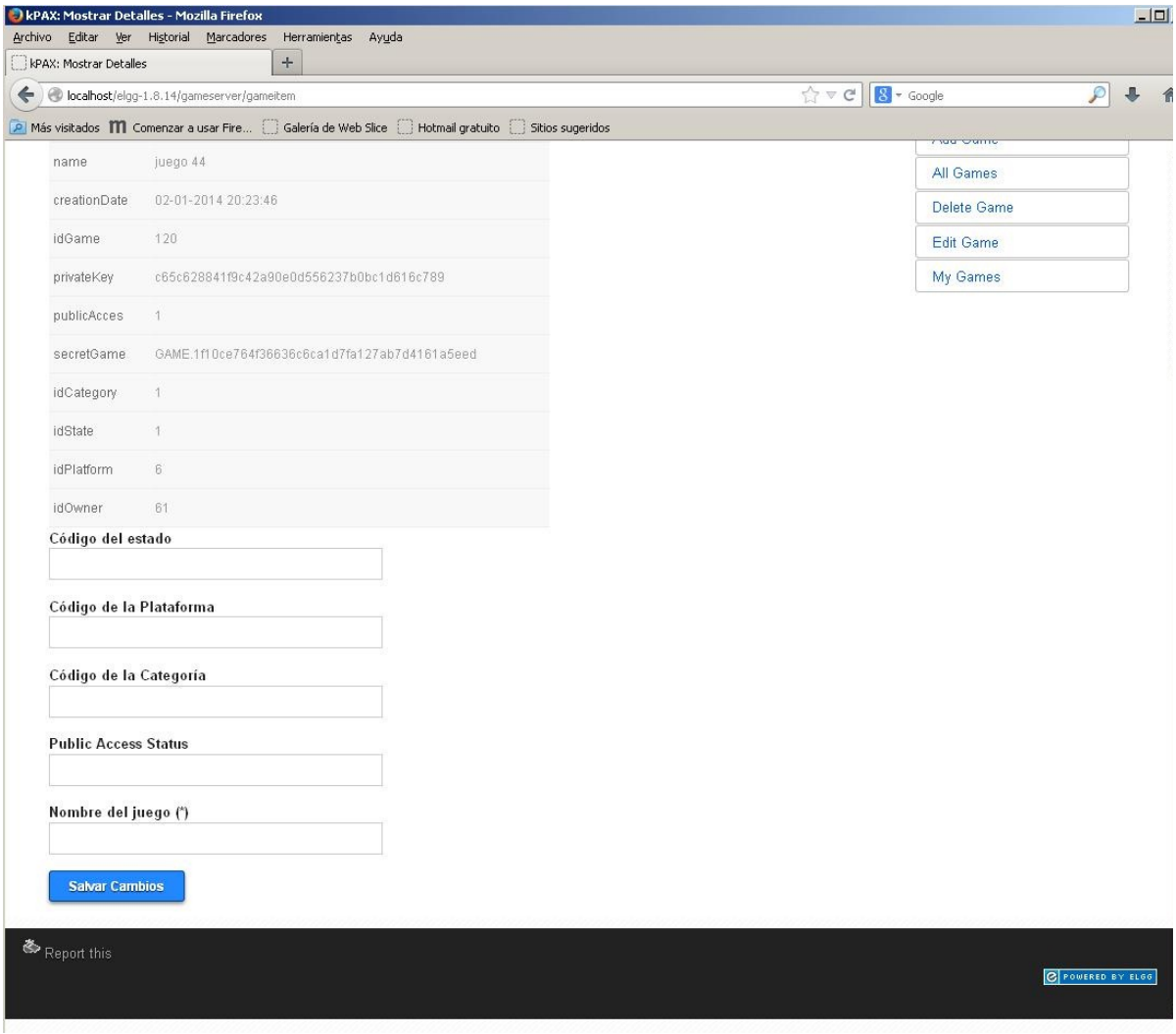

Figura 23 Aspecto de la parte inferior de la página. Formulario de solicitud de cambios.

Si no se cubren todas las cajas de texto del formulario, la acción savegame que se lanza tras pulsar el botón salvar cambios, debería volver a solicitar el juego y completar los valores con los datos obtenidos, pero para que esto sea posible, sería necesario pasar a través de la variable $SESSION, el valor de idgame desde la vista que lo muestra a la acción que lo procesa, de forma similar a lo que se hizo con el formulario editgame.

Descripción de la funcionalidad para ordenar los juegos mostrados en la tabla all games.
Se ha tratado de añadir la posibilidad de ordenar los campos de la tabla all games, para ello se utilizó un plugin de terceros, desarrollado a partir del conjunto de librerías javascript Jquery. El plugin se llama TableSorter [50]. Sus componentes deben copiarse bajo la carpeta \gameserver\vendors. Es necesario registrar en el fichero start.php:

- El componente Jquery-latest.js
- El componente jquery.TableSorter.min.js
- El fichero de estilo asociado, en este caso : mod/gameserver/vendors/jquery-TableSorter/themes/blue/style.css

Además es necesario identificar la tabla, y llamar a la función TableSorter mediante este

código JavaScript:

```
<script language="javascript" type="text/javascript">

$(document).ready(function()
  {
    $("#admintable").TableSorter();
  }
);
</script>
```

En principio, en cualquier punto del documento html, o bien referenciando al script externo en una llamada dentro del head de la página.
Las pruebas que hice añadiendo el script al cuerpo de la vista admingame no fueron exitosas. Elgg permite añadir metatags a las etiquetas head, extendiendo la vista elements/head en el fichero start.php, añadiendo la siguiente línea:

```
elgg_extend_view('page/elements/head','gameserver/metatags');
```

Es necesario crear un fichero metatags.php con el JavaScript comentado más arriba. Ninguna de las opciones que probé, tras haber leído algunos post en los foros de Elgg dió resultado. Sin embargo, los estilos del fichero css asociado a TableSorter sí se cargan correctamente en la página all games y podemos observar en su diseño los iconos que indicarían el sentido de la ordenación de la tabla.

El árbol de directorios para el plugin Gameserver quedaría de la siguiente manera:

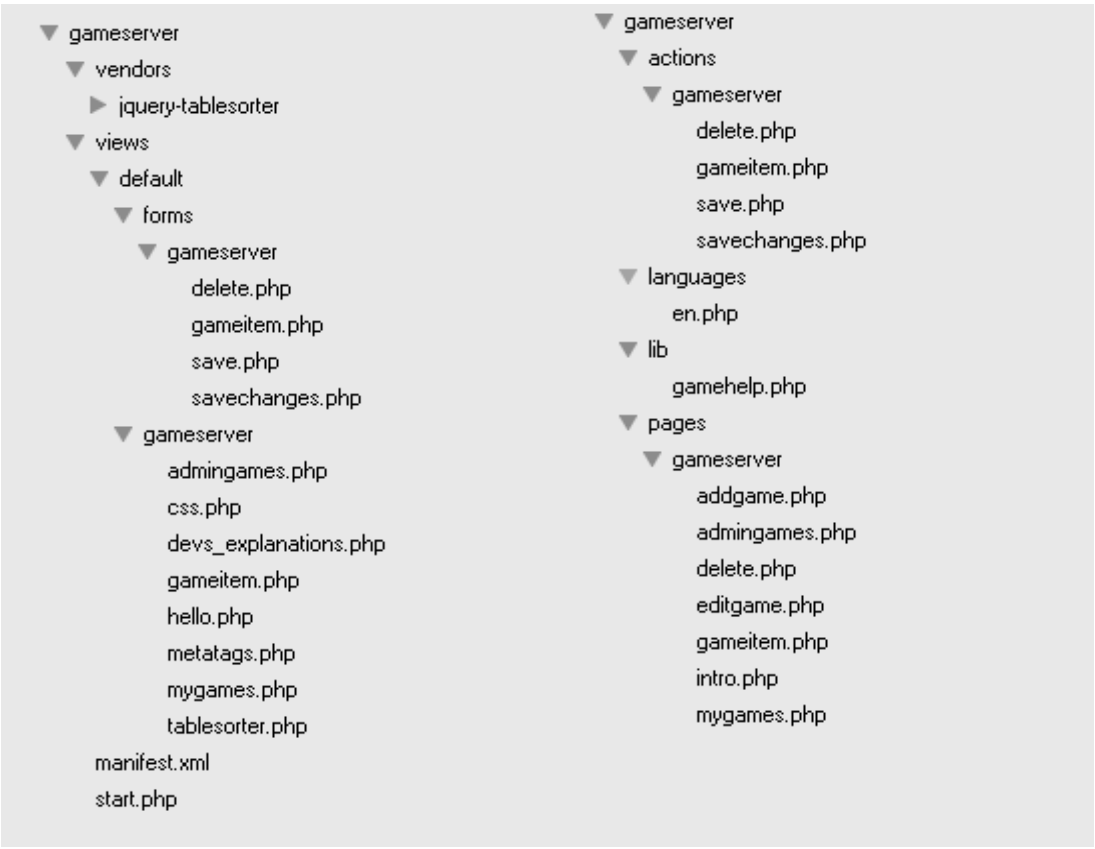

Figura 24 Árbol de directorios para el plugin Gameserver.

IMPLANTACION Y MANTENIMIENTO

la implantación y el mantenimiento correrán a cargo del equipo de desarrollo original de Kpax, que revisarán si el plugin es viable para pasarlo a producción o debe ser adaptado o desechado.

El mantenimiento y soporte a largo plazo requerido sería el necesario para adaptar, si procede, el plugin a las distintas versiones de Elgg que se vayan liberando.

CONCLUSIONES

Se ha realizado un estudio del sistema para obtener una descripción detallada del mismo que nos permita considerar el estado inicial, las posibilidades de mejora y las implicaciones que tendría el desarrollo del nuevo plugin. Los detalles se han recogido en los apartados análisis del sistema y estudio de viabilidad.

Se ha propuesto una serie de modificaciones en el sistema, añadiendo tres tablas: Developer, State y Platform y dos vistas: Userview y Adminview a la base de datos MySQL Kpax. Se han creado también los componentes java necesarios en el código del web service para mapear tres de estos nuevos componentes: Developer, Userview y Adminview. Se han considerado las vistas de MySQL como si fuesen tablas, teniendo en cuenta que es posible que sólo funcionen con los métodos que lean información.

Una vez modificado el sistema, se han creado los métodos necesarios para llevar a cabo las acciones requeridas por el plugin Gameserver de acuerdo a los casos de uso encontrados. Cuando se ha comprobado la funcionalidad de los métodos y se ha completado el desarrollo de las páginas que los utilizan, se han realizado pruebas de utilización del plugin y finalmente se ha dado por finalizado.

El principal objetivo conseguido ha sido la comprensión del sistema formado por la aplicación Elgg, entendiendo el conjunto servidor Apache, MySQL, php, aplicación elgg y plugins relativos a Kpax y el servicio web, tanto en lo que se refiere al código java como a su despliegue en el servidor de aplicaciones JBoss con todos sus componentes.

Otro de los objetivos conseguidos ha sido la creación de un plugin con funcionalidades parciales. Se ha conseguido que el plugin no pierda la funcionalidad de guardar juegos, se ha añadido un método para completar el salvado de datos complementarios para los juegos y se han desarrollado tres métodos que permiten mostrar como un todo la información de los juegos, tanto la que ya se gestionaba hasta el momento como la nueva.

No se ha obtenido un plugin completamente funcional. No se han podido obtener tablas ordenables para los listados de juegos tal y como proponían los requisitos. La solución elegida, implantar un plugin de terceros basado en Javascript llamado TableSorter, no ha podido completarse.

He tenido problemas para implementar las acciones de borrado y editado de los juegos, existen las vistas, pero no poseen la funcionalidad requerida. Los juegos pueden editarse, pero no se salvan los cambios. Las causas del fallo pueden deberse a varios motivos, pero el más probable es que los métodos no reciben correctamente los parámetros, por el tipo de error que devuelve por consola JBoss, un http 404 de página no encontrada. En cualquier caso, sería necesario haber contado con tiempo extra para el desarrollo.

Tampoco se ha encontrado una solución para que los valores sean editables desde la propia tabla y que a continuación los cambios sean transmitidos al servicio web pasando por el código en PHP. Parece más fácil seguir con la estrategia de que el propio usuario cubra formularios con los datos que desea modificar.

Habría sido necesario realizar pruebas más exhaustivas con el software obtenido, por ejemplo probar el sistema desde varios navegadores más aparte de Mozilla y Chrome pero por falta de tiempo no ha sido posible. Tampoco se ha podido completar la documentación, ni el manual de usuario ni notas para desarrolladores.

En mi opinión, el desarrollo se encuentra en un punto en el que es necesario avanzar para mejorar sobre todo los aspectos de seguridad, diferenciando mejor las funcionalidades para administradores y usuarios no privilegiados y mejorando la funcionalidad de acciones y métodos. Si se da por bueno el aspecto general del interface creo que es buena idea continuar, pero si se decide que no es funcional o adecuado, quizá fuese más fácil comenzar con un nuevo diseño.

Las prácticas han supuesto una gran aportación desde el punto de vista de la adquisición de conocimientos, sobre todo en el acercamiento a tecnologías de gran utilidad y aplicabilidad y nociones de integración de dichas tecnologías, gran parte de ellas desconocidas para mí hasta el momento. Respecto a áreas conocidas, como los servicios web, que se han visto con anterioridad en diversas asignaturas, me alegro de haber vuelto sobre ellos. Pienso que su conocimiento es esencial para implementar cualquier desarrollo en la web actual, y era muy necesario refrescarlos.

Las prácticas en mi opinión han sido intensas. El tutor de prácticas no ha puesto ninguna restricción a los aspectos que podía o debía modificar en el entorno. Me ha proporcionado información valiosa para llevar a cabo las modificaciones que considerase necesarias en el servicio web y ha estado disponible para aclarar las dudas.

Me hubiera gustado tener más información del sistema previamente, para avanzar más rápido en la parte de estudio del sistema. En ocasiones me he visto un poco perdida. He pasado mucho tiempo atascada en una modificación en el método addGame que no he sabido resolver más que cambiando de estrategia. Eso me hizo perder muchísimo tiempo.

Reconozco que el tiempo para hacerlo se me ha quedado corto a pesar de que durante este período mi dedicación al proyecto ha sido a tiempo completo. Valoro mucho todo lo que he aprendido y me apena que todo el trabajo dedicado no se haya traducido en un resultado más tangible, es decir, que el código resultante tuviese funcionalidad completa.

Las posibilidades de ampliación son muchas y variadas. Entre las funcionalidades que podrían necesitarse: posibilidad de añadir nuevas plataformas y nuevas categorías.
Añadiría funcionalidades de ordenación y edición de tablas a través de JavaScript, ya que no se ha podido cumplir con este objetivo en el presente proyecto. Esto aligera las consultas a la base de datos, la parte de ordenación tiene lugar en el cliente y no en el servidor. Respecto a la facilidad para seguir en esta línea, existen bastantes consultas sin resolver en los foros de Elgg con integraciones de componentes JavaScript, de lo que deduzco que no debe ser trivial.

Respecto a las funcionalidades de edición desde JavaScript, no lo veo claro en el sentido de que es necesario “volver” a PHP para utilizar los métodos que se comunican con el servicio web Kpax y que efectivamente ejecutan los insert de las modificaciones, pero es posible que un desarrollador con más experiencia en PHP y JavaScript pudiese solventarlo.

Creo que el sistema de estados para los juegos puede ampliarse. En la tabla state se dieron de alta tres estados: 1 para requested, 2 para accepted y 3 para denied, pero se pueden admitir otros casos, como que un juego sea denunciado, que deba ser baneado o borrado tras ser previamente aceptado o simplemente que un usuario se de de baja sin borrar el juego y que éste quede huérfano.

REFERENCIAS BIBLIOGRÁFICAS

[1] C. Costello, Elgg 1.8 Social Networking, Editorial Packt Publishing, 2012

[2] LAMP
http://es.wikipedia.org/wiki/LAMP

[3] Presentación del proyecto Kpax
http://www.innovauoc.org/showcase/?content=load_proyecto&id=112
http://www.innovauoc.org/showcase/uploads/media/in_pid1111_art_cat.pdf

[4] A. Lapedriza, D. Riera, X. Baró, J. Arnedo, C. Córcoler, J. Jorba, D. Masip, JF. Sánchez, F. Santanach. A. Valls. “Kpax Plataforma d'Aprenentatge en Xarxa Juga seriosament”, Universitat Oberta de Catalunya (UOC),

5] Plataformas para redes sociales:
http://vivalogo.com/vl-resources/best-community-software-social-network-scripts-platforms.htm

[6] Elgg.com
http://www.thematic.net/about

[7] Requisitos Elgg:
http://docs.Elgg.org/wiki/Installation/Requirements

[8] Eclipse
http://www.eclipse.org/org/

[9] Licencia Eclipse
http://es.wikipedia.org/wiki/Licencia_Publica_de_Eclipse

[10] Maven Definición
http://di002.edv.uniovi.es/~dflanvin/home/?Docencia:Cursos_Impartidos:Ant%2C_Maven_y_Tomcat
Daniel F. Lanvín

[11] Maven POM
http://Maven.Apache.org/guides/introduction/introduction-to-the-pom.html

[12] Maven Repositorios
http://Maven.Apache.org/guides/introduction/introduction-to-repositories.html

[13] Maven
http://www.genbetadev.com/Java-j2ee/introduccion-a-Maven

[14] Git Hub
http://es.wikipedia.org/wiki/Git

[15] Spring general
http://es.wikipedia.org/wiki/Spring_Framework

[16] Jersey general
http://www.mkyong.com/webservices/jax-rs/jersey-hello-world-example/

[17] Hibernate general
http://es.wikipedia.org/wiki/Hibernate

[18] Java annotations
http://en.wikipedia.org/wiki/Java_annotation

[19] Spring con Hibernate
http://www.mkyong.com/spring/Maven-spring-hibernate-annotation-MySQL-example/

[20] JBoss contenido
http://laurel.datsi.fi.upm.es/~ssoo/DAW/Trabajos/2003-2004/Septiembre/19/Contenido_archivos/resource1/r1conten2-1.htm

[21] JBoss RedHat
http://news.cnet.com/Red-Hat-scoops-up-JBoss/2100-7344_3-6059293.html

[22] JBoss comunnity
http://www.redhat.com/resourcelibrary/videos/JBoss-community-or-enterprise-video

[23] JBoss Admin Guide
http://docs.JBoss.org/JBossas/JBoss4guide/r3/adminguide.pdf

[24] Servicios web
http://searchsoa.techtarget.com/tip/REST-vs-SOAP-How-to-choose-the-best-Web-service

[25] Rest
http://www.ibm.com/developerworks/webservices/library/ws-restful/index.html

[26] Restful
http://www.dosideas.com/noticias/Java/314-introduccion-a-los-servicios-web-restful.html

[27] Json
http://geekytheory.com/json-i-que-es-y-para-que-sirve-json/

[28] Soap
http://www.oracle.com/technetwork/Java/Javase/tech/webservices-jsp-136868.html

[29] JAX-RS
http://stackoverflow.com/questions/15622216/definition-of-jax-ws-and-jax-rs.

[30] Jersey
https://jersey.Java.net/

[31] Jersey modules
https://jersey.Java.net/documentation/latest/modules-and-dependencies.html

[32] Spring
http://es.wikipedia.org/wiki/Spring_Framework

[33] Spring overview
http://www.tutorialspoint.com/spring/spring_overview.htm

[34] Inversión de control
http://es.wikipedia.org/wiki/Inversi%C3%B3n_de_control

[35] Inyección de dependencias
http://es.wikipedia.org/wiki/Inyecci%C3%B3n_de_dependencias

[36] ID en Spring
http://www.tutorialspoint.com/spring/spring_dependency_injection.htm

[37] AOP
http://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_aspectos

[38] AOP estado del arte
http://www.lsi.us.es/~informes/aopv3.pdf .

[39] AOP descripción
http://includeblogh.blogspot.com.es/2011/04/programacion-orientada-aspectos-que-es.html

[40] Spring Source
http://en.wikipedia.org/wiki/SpringSource

[41] Spring y EJB
http://www.onjava.com/pub/a/onjava/2005/06/29/spring-ejb3.html

[42] Hibernate get started
http://docs.JBoss.org/hibernate/orm/4.2/quickstart/en-US/html/

[43] Hibernate presentación
http://es.kioskea.net/faq/3544-hibernate-parte-1-presentacion

[44] Gestión de proyectos
J.R. Rodríguez, J. García Mínguez, I. Lamarca Orozco. Gestión de proyectos informáticos: métodos, herramientas y casos, Editorial UOC, 2007

[45] Modelo vista controlador
http://docs.elgg.org/wiki/Engine/Views

[46] Elgg entities
http://docs.elgg.org/wiki/Engine/DataModel/Entities

[47] Modelo vista controlador general
http://es.wikipedia.org/wiki/Modelo_Vista_Controlador

[48] Puntos de casos de uso
http://es.wikipedia.org/wiki/Puntos_de_caso_de_uso

[49] CSS Viktor Persson
http://www.arcsin.se/external/csstablegallery/prettyinpink.css

[50] TableSorter
http://TableSorter.com/docs/

[51] HTML5
J.D. Gauchat El gran libro de HTML5, CSS3 y JavaScript Ed. Técnicas Marcombo 2012

[52] Licencia CC by-nc-sa
http://creativecommons.org/licenses/by-nc-sa/3.0/deed.es