UNIVERSITAT OBERTA DE CATALUNYA

Estudios de Informática, Multimedia y Telecomunicaciones

Master en Software Libre

Memoria PFM

Mejoras de funcionalidades en la gestión de usuarios de la plataforma K-PAX

Especialidad: Administr web y comercio electrónico

Alumno: José Ramón Santos Dios

Nombre del consultor: Francisco Javier Noguera otero

Nombre del tutor externo: Daniel Riera Terrén

Fecha de entrega: 23/01/13

RESUMEN DEL PROYECTO

xPAX consiste en una plataforma tecnológica cuyo objetivo principal es crear una red de aprendizaje social basada en juegos y simulaciones por medio de la implantación de módulos desarrollados por diferentes usuarios del sistema con código abierto, y que tiene en cuenta los actuales paradigmas de interacción y comunicación de las redes sociales.

Dicha plataforma se basa en tres puntos fundamentales:

1. El apoyo al trabajo y la evaluación de ciertas competencias mediante el aprendizaje basado en JUEGOS.

2. El estado actual de la tecnología móvil, que gracias a los nuevos dispositivos portátiles que incorporan nuevos ámbitos a los lugares donde la comunidad UOC puede estar conectada y aprendiendo.

3. Esta misma tecnología ofrece múltiples herramientas como son las REDES SOCIALES. Estas permiten añadir relaciones virtuales sin precedentes

El objetivo del proyecto es la realización del diseño e implementación de una ampliación de las capacidades soportadas y ofrecidas por la plataforma de gestión de videojuegos educativos K-Pax. Concretamente en el sistema de gestión de usuarios, en diferentes aspectos como los perfiles de usuario, las clasificaciones de los diferentes perfiles, las puntuaciones obtenidas en los diferentes juegos de aprendizaje, un sistema de invitaciones de usuarios a juegos y la búsqueda de información de usuarios por parte de otros.

Table of Contents

INTRODUCCIÓN

Las redes sociales han alcanzado en los últimos años un uso generalizado, y este fenómeno está actualmente cambiando la forma de comunicación entre usuarios así como la manera de desarrollar software. El proyecto del que trata la siguiente memoria propone una forma de utilizar estas redes sociales para ofertar videojuegos educativos como herramientas de formación.

kPax es una plataforma compuesta por un conjunto de servicios creados mediante la interacción entre diferentes aplicaciones. En concreto se trata de la interacción entre Elgg como plataforma de redes sociales y un conjunto de aplicaciones de videojuegos educativos que se apoyan en la infraestructura para gestionar datos como puntuaciones, eventos, ...

KPax es ejecutado desde un servidor de aplicaciones, Jboss en este caso, y ha sido desarrollado usando Java, J2EE y una serie de construcciones como patrones de diseño, arquitectura MVC, bibliotecas de funciones (también llamadas librerías) que permiten agilizar el proceso.

Para añadir funcionalidades a la plataforma ELGG se utilizan los plugins, estos plugins se sitúan en un subdirectorio de la carpeta /mod. Existen multitud de mods o plugins para ELGG publicados en Internet.

Para poder gestionar las características del perfil del jugador dentro de la plataforma k-Pax es necesario extender la función Profil install de Elgg y modificar las capacidades de k-Pax para que sea posible la conexión.

Una parte de estas interacciones ya dada en los Mods Elgg por kPax. Estos mods están compuestos por:

- ***Loginrequired***: esconde todas las páginas de Elgg para los usuarios no registrados, exceptuando las de inicio, registro y olvido de la contraseña al usuario no autenticado.

- ***Kpax***: contiene los webservices necesarios para interactuar desde fuera con el servidor Elgg interno.

- ***Apiadmin***: genera y gestiona los certificados para la autenticación.

- ***LikeKpax***: gestiona las anotaciones "me gusta" en los objetos kPax.

A continuación se detallan los servicios que ofrece actualmente kPax:

Adición de un juego: El fichero ***save.php*** del servidor de elgg construye el formulario de entrada de datos.Al pulsar sobre *añadir* se realiza una llamada a la acción elgg que introduce los datos en la base de datos de elgg y en la de kpax (por medio de una llamada al servicio web de adición de juegos). Desde este segundo fichero se deberá realizar una llamada a la función addGame.

Consulta de la ficha de un juego: El fichero ***kpax.php*** muestra la ficha de un objeto de subtipo kpax, es decir un juego. Desde este fichero se deberá realizar una llamada a la función getGame, para obtener el nombre.

Adición de comentarios: El fichero ***add.php*** añade un comentario en la base de datos de Elgg. El fichero no está en ningún módulo concreto sino que forma parte integra de elgg por lo que será necesario comprobar previamente a la llamada a la función addCommentGame que el objeto sea de subtipo kpax.

Eliminación de comentarios: El fichero ***delete.php*** elimina un comentario concreto. Será necesario realizar una llamada a la función delCommentGame e igualmente comprobar antes de ello que el comentario pertenece a un objeto de subtipo kpax (un juego).

Listado de todos los juegos: El fichero ***all.php*** se corresponde con la pestaña de All del apartado de prototipado en la fase de análisis. Aquí se mostrará como máximo el total de juegos.

Listado de juegos propios: El fichero ***owner.php*** se corresponde con la pestaña de Mine del apartado de prototipado en la fase de análisis. Aquí se mostrarán como máximo los juegos que hayan sido añadidos por el usuario.

Listado de los juegos de amigos: El fichero ***friends.php*** se corresponde con la pestaña de Friends del apartado de prototipado en la fase de análisis. Aquí se mostrarán como máximo los juegos que hayan sido añadidos por los amigos del usuario.

NOTA: la información relativa a los ficheros que controlan los servicios nombrados se encuentra en el Anexo 2.

1. Estudio de viabilidad

Antecedentes

K-Pax es una plataforma que actualmente ofrece unos servicios mínimos, explicados en el anterior apartado, permitiendo la gestión básica de los juegos, usuarios y administración de la plataforma, pero carece actualmente de otros servicios como la gestión de la búsqueda de usuarios, información detallada de los perfiles de los jugadores, así como la mejora de las herramientas de comunicación entre estos, posibilidades de interaccionar entre jugadores por medio de ligas, puntuaciones en los juegos, ... algunos de estos servicio se añadirán con la realización de este proyecto.

K-Pax es una plataforma cliente-servidor de software libre cuyo código es alojado en la web de ***Github***, una herramienta de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones ***Git***, un sistema cuyo objetivo es facilitar la administración de las distintas versiones de cada producto desarrollado, diseñado por Linus Torvalds.

Entre las características de **Github** podemos destacar:

- La posibilidad de decidir si el código alojado será público o limitado a unos usuarios determinados.
- Un visor de código mediante el cual podemos realizar consultas a través del navegador, resaltando la sintaxis correspondiente al lenguaje a consultar.
- Un sistema de seguimiento de problemas mediante tickets
- Una wiki para el mantenimiento de las diferentes versiones de un proyecto
- Una herramienta de revisión de código
- Un visor de ramas para comparar los progresos realizados en las diferentes ramas de nuestro repositorio

El proyecto actual se realizará a partir de una bifurcación (fork) de la rama principal (master) disponible en el repositorio:

https://github.com/jsanchezramos/k-pax

Alcance del sistema

La aplicación será utilizada tanto por usuarios comunes, que podrán ser alumnos o no de la UOC, como por administradores que tendrán permisos especiales para administrar la plataforma de forma más **concreta**

La plataforma deberá cumplir los siguientes requisitos:

- Permitir realizar **búsquedas de usuarios** disponibles en la plataforma según diferentes criterios tales como mejor puntuados, capitanes de equipos, miembros de un grupo, y la ordenación de los resultados de estas búsquedas
- Este diseño deberá permitir la incorporación de nuevos criterios de ordenación y búsqueda de usuarios
- Desarrollar una **documentación** de modo que pueda ser consultada por desarrolladores posteriores, sirviendo de memoria técnica que informe detalladamente de los cambios realizados, de manera que puedan utilizar dicha documentación para la realización de futuros cambios
- Se podrán clasificar usuarios y grupos por medio de un sistema de gestión de etiquetas o tags

Estudio de la situación actual

El sistema actual consiste en un sitio web desarrollado con un Servidor de aplicaciones de Redes Sociales llamado ***ELGG [1]***, implementado sobre una plataforma LAMP (Linux, Apache [7], MySQL [8] y PHP [9]), la plataforma ELGG pertenece a la *Elgg Foundation*, que promueve la implantación de una Red Social de código abierto (http://elgg.org/), y está desarrollada bajo la licencia GNU GPL.

WAMP | LAMP | XAMP (Windows | Linux | OSX, Apache, MySQL, PHP): Pila integrada de aplicaciones para dar servicios web

[7] Apache HTTP servidor web : Servidor web HTTP multiplataforma y de código abierto desarrollado dentro del proyecto HTTP Server de la Apache Software Foundation.

[8] MySQL : Gestor de bases de datos relacionales multiusuario en código abierto, con un esquema de licenciamiento dual, se ofrece bajo GNU GPL para cualquier uso compatible de esta licencia pero para las empresas que quieran incorporarla en productos privativos existe una licencia específica que permita este uso.

[9] PHPMyAdmin : Interfaz web de usuario para la gestión de bases de datos en MySQL.

[10] PHP : Lenguaje de programación de propósito general, de tipo script, especialmente adecuado para entornos web, permite la incrustación de scripts en HTML.

Elgg proporciona una programación orientada a objetos, formalizada mediante las denominaciones de los directorios y ficheros que conforman su estructura, dividida en módulos extensibles. El modelo de datos de Elgg permite la inclusión de nuevos

objetos dado que existe un objeto genérico (ElggObject) que puede ser fácilmente extendido y que se puede relacionar consigo mismo o con otros objetos de kPax, usuarios, grupos, etc.

Es posible utilizar un ElggObject para gestionar los juegos, otro para gestionar los campos del perfil, los likekPax ... es decir es posible utilizar este modelo como el único modelo de datos para múltiples plataformas.

La plataforma implementada por la UOC para desarrollar xPAX por medio de ELGG consiste en una página principal que contiene un menú al que se accede por un sistema de pestañas. Dicha plataforma contiene una serie de módulos desarrollados en el lenguaje de programación JAVA, que contienen los juegos interactivos destinados a crear procesos educativos de aprendizaje, dichos módulos son independientes de la plataforma y desarrollados de forma individual o colectiva por diferentes usuarios de la misma.

Características principales del proyecto:

- Multiplataforma: El proyecto tiene en cuenta los nuevos dispositivos móviles que han proliferado últimamente en el mercado
- Inclusión de las redes sociales, el sistema tiene en cuenta la interacción entre diferentes usuarios/jugadores por medio de este nuevo sistema de comunicación
- Nuevos paradigmas educativos como el aprendizaje basado en juegos y simulaciones
- Acceso universal del sistema por diferentes usuarios y Sistema GNU GPL de acceso al código abierto

Modelo de datos de xPAC - Entidades

Game

Para registrar el juego en la plataforma un usuario debe enviar información a los usuarios y grupos (GUG), la información debe tener un único identificador de usuario, un dominio opcional (o realm) el cual debe ser alguna plataforma social aceptada por

kPAX, el campus UOC o ninguno de ellos, si el usuario quiere usar el mismo kPAX y contraseña. La plataforma devuelve un identificador de seguridad.

GameAccess

El objetivo de esta entidad es restringir el acceso de ciertos usuarios a algunos juegos. La tabla contiene información de acceso a juegos por diferentes grupos. Para hacer la estructura lo más flexible posible contiene no solo los accesos permitidos, y los accesos no permitidos por ciertos grupos. El acceso a los usuarios es concedido o denegado por medio del grupo al que pertenecen.

GameInstance

Para jugar los usuarios necesitan acceder a la instancia de un juego en particular, que contiene la información de estado de el juego. Cada instancia tiene un indicador único llamado **idGameInstance**. La información relativa a un juego es almacenada en un fichero XML, con el formato y los atributos decididos por los desarrolladores.

GameLike

Representa un voto realizado por un usuario sobre un juego. Permite determinar si a un usuario le gusta un juego o no, resultando en algo así como un voto. Permite determinar la puntuación o popularidad de un juego entre los usuarios.

GameScore

Entidad destinada a registrar la puntuación obtenida por los jugadores en un determinado juego.

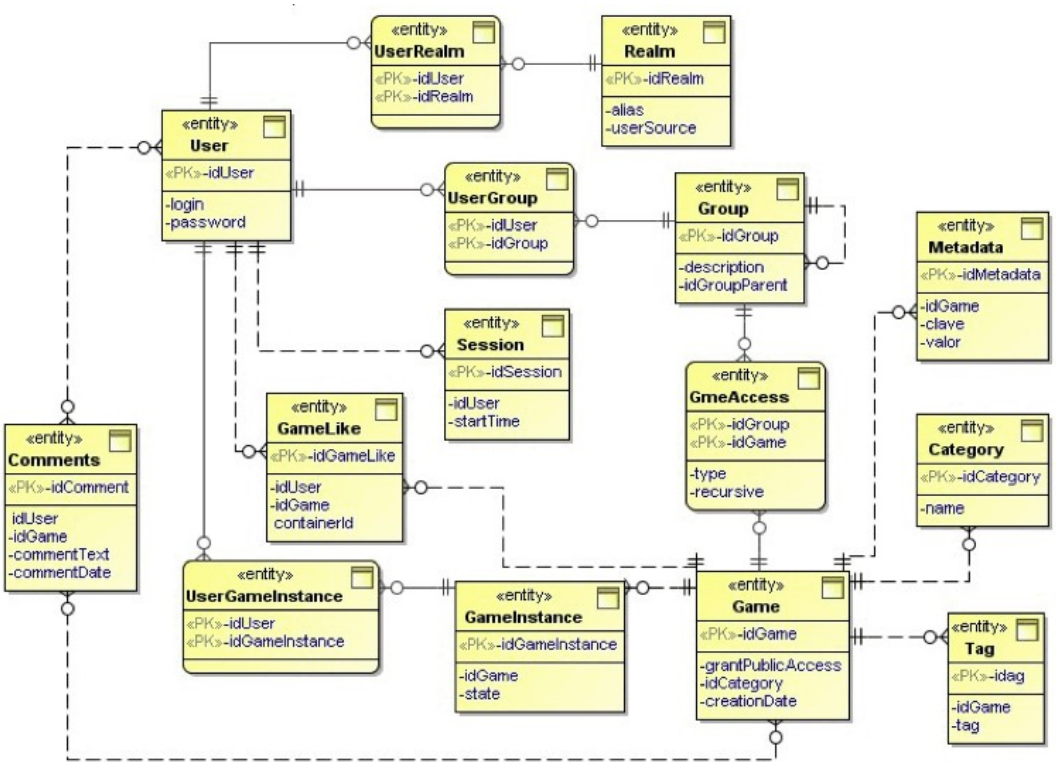
Group

Para gestionar el acceso a juegos, los usuarios se pueden organizar en grupos, un grupo puede ser parte sólo de otro grupo indicado en el campo **idGroupParent**. Es necesario crear un protocolo para gestionar grupos, permitiendo a los usuarios crear grupos, los usuarios pueden crear tantos grupos como sea necesario. La afiliación a un grupo está en la tabla **UserGroup**, los atributos que pueden ser añadidos a esta tabla son tipos específicos de miembros, dando ciertos permisos del grupo sobre algunos usuarios (usuarios de gestión, crear subgrupos,…)

Session

Contienen información sobre el inicio de sesión de un usuario.

Organización del modelo de datos de xPAC, esquema E-R:



Definición de los requisitos del sistema

Requisito nº 1

Para acceder a cualquier funcionalidad del sistema de usuarios se requiere que el usuario haya realizado una sesión y esta continúe abierta en el momento de ejecutar dichas funcionalidades.

Requisito nº 2

La plataforma utilizará un sistema de permisos en el que solamente el administrador pueda modificar funcionalidades de usuarios diferentes a el.

Requisito nº3

El usuario solamente podrá modificar roles de usuario, como capitán del equipo, mientras otro usuario no lo haya realizado antes

Requisito nº4

La búsqueda de información de otros usuarios del sistema será limitada, este no tendrá acceso total a la información de otros usuarios

Estudio de las alternativas de solución

La plataforma ya ha sido creada e incluye numerosas tecnologías ya implementadas, por lo que el desarrollo de estas nuevas funcionalidades en la gestión del sistema de usuarios no deberá afectar a las características y funcionalidades principales de dicha plataforma, solamente se centrará en añadir nuevas funcionalidades.

Por lo tanto no existen muchas opciones dado que la plataforma ya está montada y las tecnologías a utilizar son las existentes, por lo tanto las escasas alternativas estarían encaminadas a buscar algún módulo sustitutivo que implemente la gestión personalizada de los usuarios.

Valoración de las alternativas

Después de revisar la documentación de ELGG respecto a los módulos que implementen alternativas a la gestión de usuarios se ha llegado a la conclusión que ninguno se adapta a los requisitos establecidos en el proyecto por lo que se adoptarán los cambios señalados en los objetivos y requisitos del proyecto con las tecnologías existentes en la plataforma.

A continuación se detalla dos de las alternativas estudiadas con más características:

Alternativa 1: plugin Profile Manager

Aprovechando las capacidades de extensión de Elgg podemos utilizar un plugin ya desarrollado en su versión 7.3 y estable llamado ***Profile Manager*** de Jeroen Pedales, que contiene funcionalidades similares a las descritas en los objetivos del actual proyecto y cuenta con una licencia GPL por lo que se permitiría su inclusión en la distribución de la aplicación de xPAC, esta es la alternativa más adecuada de todas las estudiadas, no obstante no será utilizada en el proyecto ya que uno de los objetivos del proyecto es utilizar la plataforma xPax con su base de datos, y para ello es necesario utilizar los servicios propios de dicha plataforma y de su plugin.

Este plugin permite gestionar usuarios con diferentes perfiles, perteneciente a diferentes grupos. Editar tanto desde el perfil del administrador como del usuario los campos que componen un perfil. Los campos pueden ser de diferentes tipos: texto, calendar, file, multiselect ...

Las principales características de este plugin son:

- Permite el ordenamiento de los campos personalizados (arrastrar y soltar)
- Permite agregar tipos de perfiles así como categorías
- Permite agregar desplegables, radio, archivo, selección múltiple, los tipos de campos de calendario y DatePicker
- Permite campos obligatorios (por formulario de registro, campos de perfil solamente)
- Permite copia de seguridad y restaurar el perfil de configuración de los campos
- Permite exportar perfiles de usuario (meta) datos a csv

Referencia del plugin Profile Manager.

http://community.elgg.org/pg/plugins/project/385114/developer/jdalsem/profile-manager

Alternativa 2: plugin Westors Elgg Manager

Basada en una interfaz RIA (Aplicación de Internet Enriquecida), es una completa interfaz que le permite al administrador borrar, deshabilitar usuarios, editar información, entre otras cosas.

Para un usuario normal de Elgg, con tan solo seleccionar nuestros amigos con un simple click, podremos mandar mensajes de texto SMS, o emails, o simplemente ver su perfil.

También podremos ejecutar acciones a nuestros grupos, con tan solo un click.

Los administradores cuentan con más opciones como eliminar o editar datos de usuario, bloquear o desbloquear usuarios, etc.

Referencia del plugin Westors Elgg Manager.

http://community.elgg.org/plugins/553265/1.8.3%20free/westors-elgg-manager

Riesgos del proyecto

1. Carga de trabajo y aprendizaje del desarrollador del proyecto

   Se tendrá en cuenta la curva de aprendizaje del desarrollador del proyecto, así como el desarrollo paralelo de otras asignaturas de la UOC así como el tiempo dedicado al propio horario laboral, que puedan afectar o incidir en el desarrollo del citado proyecto.

   Plan de contingencia: estimación inicial del ritmo de aprendizaje y desarrollo del proyecto

2. Errores y omisiones: la escasa experiencia del autor del proyecto en cuanto a desarrollo de módulos de la plataforma K-PAX puede llevar a estimaciones de tiempo-esfuerzo incorrectas.

Plan de contingencia: replanteamiento de los objetivos y tiempo dedicado al desarrollo de las nuevas funcionalidades.

3. Pérdida de recursos, como ficheros sobreescritos, modificaciones mal realizadas u otros daños externos

   Plan de contingencia: desarrollo de un sistema de backups detallado, subidas frecuentes de las actualizaciones de los módulos a la plataforma

2. Análisis del sistema

Definición del sistema

El objetivo principal del PFC es la elaboración una serie de mejoras en el sistema de gestión de usuarios de la plataforma K-PAX que permita a los usuarios un sistema de gestión configurable por estos.

El proyecto actual aborda una serie de mejoras en cuanto a la gestión de usuarios del mismo, estás mejoras estarán relacionadas con la gestión de los perfiles de usuario, las clasificaciones de los diferentes perfiles, las puntuaciones obtenidas en los diferentes juegos de aprendizaje, un sistema de invitaciones de usuarios a juegos y la búsqueda de información de usuarios por parte de otros.

Por lo tanto el proyecto deberá implementar un nuevo módulo compatible con k-PAX que permita:

- Realizar búsquedas de diferentes usuarios del sistema
- Crear nuevos perfiles de usuarios
- Crear ligas y sistemas de puntuaciones de jugadores por juego

Establecimiento de requisitos

Las siguientes mejoras propuestas están directamente relacionadas con la interfaz de la plataforma y no con los módulos y juegos que incluye esta. Dado que el proyecto ya está iniciado no se modificaran los aspectos generales de esta.

La aplicación será capaz de:

- Facilitar la creación de subgrupos de usuarios
- Un sistema de invitaciones de usuarios a determinados juegos

- Un sistema de ligas y rankings a través de la puntuación de juegos por usuarios y grupos
- El establecimiento de roles de usuarios, como el capitán, o jugadores en banquillo.

Realizar búsquedas de usuarios por:

- Grupos de usuarios
- Capitanes
- Usuarios más puntuados
- Usuarios que más veces hayan jugado

Adicional y opcionalmente, si el desarrollo del proyecto lo permite, se añadirán algunas otras funcionalidades

Actores del sistema

Distinguiremos en el sistema varios tipos de actores, o usuarios según el rol del perfil que utilice la aplicación. Para controlar el acceso a las distintas partes o funciones de la aplicación se establecerán diferentes privilegios o derechos de usuario de las funcionalidades del sistema.

Administrador

Representa a un usuario que tiene el control total del sistema y acceso a todos los datos críticos o no, que maneja el programa, es decir el usuarios con mayor privilegios en el sistema.

Los creadores o desarrolladores de la plataforma serán los encargados de definir que usuarios realizarán las funciones de usuarios.

El objetivo del perfil de administrador será controlar todo el sistema, gestionar todos los datos de la plataforma, como la gestión de juegos o usuarios, como el alta, baja, modificación de los mismos, permitir su acceso a grupos de usuarios, la realización de estadísticas o informes, conceder permisos a grupos de usuarios para poder acceder a un determinado juego.

Por lo tanto deberá haber al menos un administrador en todo momento

Usuario común

Representa a un usuario no administrador del sistema, el cual puede utilizar la plataforma para jugar, realizar actividades de redes sociales, buscar juegos, participar en ligas, rankings de usuarios, ...

Es el perfil básico de todo usuario del sistema, todos los usuarios por defecto serán usuarios comunes.

Los usuarios comunes podrán realizar búsquedas sobre los juegos a los que tengan acceso así como búsquedas de otros usuarios, ordenar dichas búsquedas, votar un juego de forma positiva o negativa, realizar comentarios, ...

Definición de interfaces de usuario

Como sabemos la interfaz de usuario es un factor muy importante en la determinación del éxito de la aplicación o mejora. En este ámbito crearemos un prototipo que intente ajustarse lo más posible a los requerimientos de la plataforma, utilizando los lenguajes propios de la misma, es decir, HTML y PHP combinado con hojas de estilo CSS.

Así mismo debemos determinar los tipos de usuario que utilizarán el sistema y con qué fines, también los tipos de dispositivos que utilizan para conectarse a la plataforma.

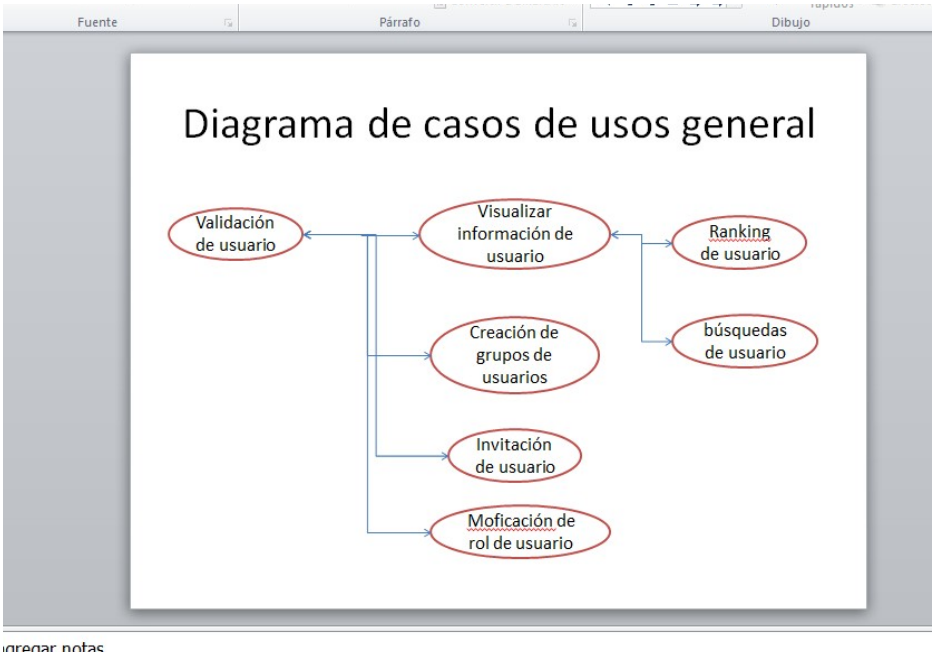
Podemos determinar los siguientes perfiles de usuario según sus conocimientos de la plataforma a utilizar:

- Usuarios sin ninguna experiencia en el manejo de redes sociales
- Usuarios sin ninguna experiencia en redes sociales pero sí en otros ámbitos de Internet
- Usuarios con experiencia en redes sociales pero poco conocimiento de ellas
- Usuarios con problemas de accesibilidad
- Usuarios con gran experiencia en redes sociales

A partir del contexto de uso de cada usuario tendremos que generar los escenarios de uso, herramienta que nos ayudará a realizar hipótesis sobre las diferentes situaciones en las que se encontrarán los usuarios, y que acciones tendrán que llevar a cabo para lograr sus objetivos.

Diagrama de casos de uso

El siguiente sencillo diagrama de casos de uso permite analizar las funcionalidades del actor usuario, se ha omitido el actor administrador ya que las funcionalidades serán exactamente las mismas, pero a diferencia del anterior esté tendrá privilegios para modificar todos los casos de uso de todos los usuarios del sistema.



Modelo de casos de uso general:

Validación de usuario: consiste en identificar y autentificar un usuario del sistema para permitir el acceso a las otras funcionalidades del sistema de gestión de usuarios

Visualizar información de usuario: muestra un listado de características propias del actual usuario: listado de juegos en los que ha participado, puntuación en los mismos, etc

Ranking de usuario: permite acceder a información relacionada con la puntuación del usuario en un juego en relación con el grupo de usuarios que ha participado en le mismo

Búsquedas de usuario: establece un criterio de filtrado para acceder a información de otros usuarios

Creación de grupos de usuarios: permite crear un grupo de usuarios organizados bajo el mismo nombre del grupo

Invitación de usuario: permite crear un sistema de alerta para informar a otro usuario de que ha sido invitado a un determinado juego por este

Modificación de rol de usuario: permite modificar el tipo de usuario para un determinado juego: capitán, usuario en banquillo, etc.

Especificación detallada de cada caso de uso

Validación de usuario:	
Actores	todos
Descripción	Representa el inicio de una nueva sesión en la aplicación, es un mecanismo de seguridad de identificación y autentificación que impide el acceso a la plataforma de usuarios no autorizados. Consiste en la introducción de un identificador y una clave de acceso a la aplicación, en el caso de ser válidos para el sistema se dará acceso al usuario a un panel de control informativo con sus permisos correspondientes al rol que tenga asignado con su identificador. Todo identificador deberá tener asociado un perfil de privilegios, que serán utilizados por este en el caso de ser autentificado. Este caso de uso se considera implementado y por lo tanto no trabajaremos en el en este proyecto.
Secuencia completa:	Condición: el usuario se valida y se muestra un panel con acceso a las partes de la aplicación que le corresponda a su perfil de privilegios. 1. El usuario accede a la página de login de

	la plataforma k-PAX. 2. El sistema solicita el identificador y la clave de acceso. 3. El usuario las facilita. 4. El sistema comprueba la identidad del usuario y su autenticidad. 5. El sistema muestra un panel de control diferente en relacion a los permisos asociados al identificador

Ranking de usuario	
Actores	todos
Descripción	Representa toda la información relacionado con la posición del usuario en relación a su rol de jugador, esta información detallará la puntuación obtenida, el lugar que le corresponde a dicho usuario dentro del ranking de resultados, si el usuario es el mejor puntuado para un juego concreto se señalará explícitamente.
Secuencia completa:	1. El usuario es validado en la plataforma 2. El usuario accede al panel informativo del ranking de resultados 1. El usuario accede a la página de login de la plataforma k-PAX. 2. El sistema solicita el identificador y la clave de acceso. 3. El usuario las facilita. 4. El sistema comprueba la identidad del usuario y su autenticidad. 5. El sistema muestra un panel de control diferente en relacion a los permisos asociados al identificador

Búsquedas de usuario	
Actores	todos
Descripción	El sistema muestra un conjunto de criterios de búsqueda de entre los cuales el usuario deberá escoger uno, y en este caso solicitará los datos necesarios para ejecutar la búsqueda de

	usuarios.
Secuencia completa:	1. El sistema muestra los criterios de búsqueda posibles. 2. El actor indica cual de los criterios de búsqueda desea utilizar. 3. El sistema, dependiendo del criterio de búsqueda escogido, solicita los datos necesarios para realizar la búsqueda con dicho criterio. 4. El actor aporta los datos necesarios. Si no es necesario ningún dato complementario para ejecutar la búsqueda se pasa automáticamente al punto siguiente. 5. El sistema guarda la preferencia

Creación de grupos de usuarios:	
Actores	administrador
Descripción	El sistema permitirá un sistema de creación de grupos de usuarios, estos grupos deberán estar por conformados por usuarios ya existentes en el sistema, un usuario podrá pertenecer a más de un grupo.
Secuencia completa:	1. El usuario es validado en la plataforma como administrador 2. El usuario accede al panel informativo de los grupos de usuarios existentes 3. El sistema presenta una serie de opciones relacionadas con la creación de un grupo nuevo: nombre del grupo, listado de usuarios candidatos al grupo. 4. El usuario selecciona el nombre del grupo y los usuarios participantes del mismo. 5. El panel de grupos de usuarios presentará otras opciones relacionadas con la gestión de los grupos, como la eliminación de un grupo, o modificación de los participantes del mismo

Invitación de usuario	
Actores	todos
Descripción	El sistema presentará un sistema e invitaciones por medio de un panel informativo, estas invitaciones a otros usuarios con el fin de participar en un juego serán enviadas por un sistema de alertas.
Secuencia completa:	1. El usuario es validado en la plataforma 2. El usuario accede al panel de invitaciones a juegos 3. El usuario selecciona uno o varios usuarios para crear una alerta de invitación a un juego 4. La alerta es enviada a los otros usuarios

Modificación de rol de usuario	
Actores	Todos
Descripción	El sistema presentará un panel de modificación de tipos de usuarios agregados a un juego, este sistema de modificación de rol de usuario podrá ser gestionado por cada usuario.
Secuencia completa:	1. El usuario es validado en la plataforma 2. El usuario accede al panel de modificación del rol de usuarios 3. El usuario selecciona uno o varios usuarios para modificar su rol de jugador 4. El sistema presenta un grupo de opciones relacionadas con el rol de cada jugador (capitán, jugador activo, jugador en el banquillo, …)

Especificación del plan de pruebas

Se establece un plan de pruebas que permita averiguar si el proyecto cumple los objetivos señalados anteriormente, el plan de pruebas consistirá pues en:

Pruebas unitarias: se especificarán en cada iteración del proyecto y se realizarán al final de cada una de ellas, el objetivo será comprobar que esta iteración cumple con los requisitos.

Pruebas de integración: se realizarán tras las pruebas unitarias para comprobar que cada nuevo paso del proyecto añadido funciona y se integra correctamente con los anteriores.

Pruebas de sistema: al final del proyecto se realizará con el producto completo, para comprobar que el sistema funciona correctamente

Pruebas de aceptación: la realizará el usuario final del sistema

Definición de las pruebas unitarias:

Validación de usuario: el usuario se identifica correctamente, el usuario accede a toda la información de su perfil de manera adecuada, el usuario accede a la información de su perfil relacionada con los juegos en los que participa. Toda funcionalidad desarrollada en este proyecto requiere que el usuario haya realizado una sesión y que ésta continúe abierta, en el momento de su ejecución.

Ranking de usuario: el usuario accede a la información de su puntuación en los juegos de forma adecuada

Búsquedas de usuario: el usuario accede a información de otros usuarios por medio de búsquedas de forma satisfactoria

Creación de grupos de usuarios: el usuario/s crean un grupo de forma correcta, todos los usuarios forman parte del nuevo grupo creado.

Invitación de usuario: el usuario realiza una invitación a un juego de forma correcta, el usuario invitado es informado por medio de una alerta de la invitación creada.

3. Diseño del sistema

Definición de niveles de arquitectura

Actualmente, la plataforma k-PAX utiliza un patrón de diseño Arquitectura en Capas, estableciendo 3 capas: presentación, lógica de negocio y persistencia, donde la capa de presentación y lógica de negocio utilizan además el patrón de diseño MVC (modelo-vista-controlador), para organizar las responsabilidades de interacción con el usuario.

Definición de capa de persistencia

La aplicación resultante del proyecto utilizará una BD para almacenar los datos necesarios bajo el SGBD MySQL 5.

La capa de persistencia del proyecto debe ser capaz de añadir, modificar, borrar, y consultar datos de la BD, así como de mantener la configuración deseada para la aplicación en cada ordenador en el que se ejecute

Definición de capa de lógica de negocio

Deberá diseñarse una capa que tome los datos necesarios para la realización de los casos de uso definidos en la fase de análisis y que compruebe la validez de los mismos, es decir, que todo aquello que implemente dicha capa deberá estar implementando en la lógica de negocio del sistema y por tanto formar parte de ella.

La capa de lógica del sistema resultante del proyecto se encargará de validar los datos introducidos por el usuario, comprobar que el usuario tiene permiso para realizar las acciones solicitadas, realizar cálculos, comunicarse con servicios externos y de servir de puente entre la capa de presentación y el mecanismo de persistencia, así como de comunicar errores internos a la capa de presentación.

Definición de capa de presentación

La capa de presentación es una capa arquitectónica que se encarga de proveer una interfaz gráfica de usuario que presente los datos al usuario, es decir, que la capa de presentación que se diseñe para el sistema del proyecto deberá servir de interfaz entre

la capa de negocio y el usuario que lo utilice.

Esta capa debe adaptarse lo mejor posible a las necesidades de los usuarios establecidas en la fase de análisis, consiguiendo que esta sea simple, intuitiva y usable.

Resumiendo, el nuevo sistema se descompondrá de la siguiente manera:

1. Capa de persistencia:

a. Sistema Gestor de Base de Datos.
b. Base de Datos de la aplicación.
c. Interfaces de acceso a los datos y sus implementaciones que permiten obtener o modificar los datos almacenados independientemente de como se almacenen o de dónde se haga. Sirven de canal entre la BD y la capa de lógica.
d. Clases de intercambio de información, que sirven para transferir los datos de una entidad entre las distintas capas.
e. En su caso, ficheros de configuración.

2. Capa de lógica:

a. Interfaces BO y sus implementaciones que hacen de canal entre la capa de persistencia y la de presentación.
b. Servicios web o funciones ofrecidas para que sean accedidas desde la capa de presentación con elgg.
c. Resto de clases de soporte para los módulos de Security, Authorization, etc.

3. Capa de presentación:

a. Elgg.
b. Clases y ficheros PHP que hacen de canal entre el usuario y la capa de lógica y que implementan la GUI de la aplicación por medio de llamadas a los servicios web.

NOTA: el Anexo 3 recoge información sobre los paquetes en los que se encuentran almacenados los ficheros de las diferentes capas.

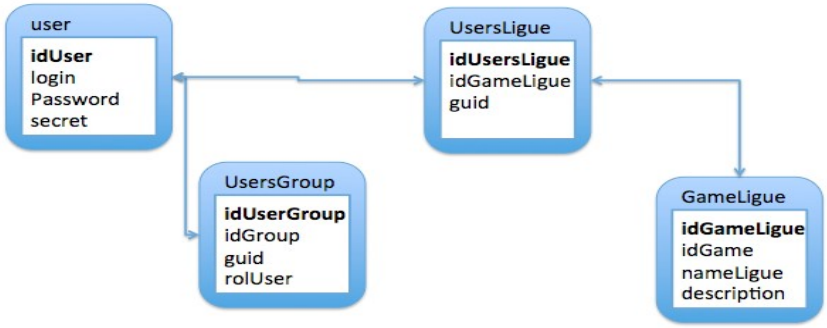
Capa de persistencia. Diseño de base de datos

Primero se diseñó el esquema relacional de la BD por medio de un diagrama de entidad-relación, asignando una tabla a cada entidad, para , después transformar dicho esquema en tablas bien definidas. Posteriormente se aplicarán las técnicas de normalización de tablas que evitarán redundancia de datos y por último se optimizará la BD mediante la definición de índices en algunos de los campos de las tablas más usadas.

Esquema relacional

El diagrama de entidades se ha diseñado en la fase de análisis. La figura siguiente muestra dicho diagrama.

Modificaciones de la BD xPAC



Nueva tabla UsersGroup, que relacione los usuarios con los diferentes grupos

UsersGroup

- **idUserGroup**: clave principal
- idGroup (clave de Groups)
- guid (clave de elgg_users_entity)
- rolUser: tipo de usuario dentro del grupo (jugador, capitán, suplente,…)

Nueva tabla GameLigue donde establecer ligas de jugadores

GameLigue

- **idGameLigue**: clave principal
- idGame (clave de Games)
- nameLigue
- description

La tabla UsersLigue incluye los jugadores de cada liga

UsersLigue

- **idUsersLigue**: clave principal
- idGameLigue (clave de GameLigue)
- guid (clave de elgg_users_entity)

NOTA: El Anexo 4 recoge información sobre las sentencias SQL necesarias para desarrollar las entidades añadidas en el proyecto actual.

Capa de presentación

La capa de la presentación es la capa que se encarga de interactuar con el usuario para servir de puente entre él y el resto de la aplicación. Ofrece una interfaz gráfica de usuario en dónde representar la información que es procesada en la aplicación.

La capa de presentación se descompone en ficheros PHP contenidos en módulos ELGG que realizan comunicaciones con la capa de negocio por medio de los servicios web publicados en Jboss.

Descomposición en ficheros

Durante la fase de análisis se determinaron dos vistas a crear: el listado de jugadores y la ficha de un jugador. En la primera vista, en el listado de jugadores, se ejecutarían

los casos de uso de Ranking de Usuario, invitación de un usuario y búsquedas de usuarios.

En la segunda se realizarían la creación de grupos de usuario y modificación del rol de usuario.

Las dos vistas serán implementadas a través de una nueva pestaña llamada Players, que acompañará a las tres pestañas por defecto de la plataforma: All, Mine, Friends.

Por último las llamadas a las funciones del servidor srvKpax se realizan desde el fichero elgg/mod/kpax/lib. Este fichero es el que se encarga de servir de enlace entre elgg y el servidor de kpax definiendo una librería de funciones, con una función PHP por cada una de los servicios web publicados en el servidor de kpax.

Hibernate

k-PAX utiliza Hibernate que es un ORM (Object / relational mapping), es decir, una librería que permite representar en clases la estructura de una BD del tipo entidad relación, Hibernate guarda en archivos XML información sobre la estructura de la base de datos y se encarga de vincular una clase Java con una tabla de la base de datos y de implementar las típicas operaciones de inserción, eliminación, modificación o selección para la misma tabla, independientemente de las demás tablas con las que se relacionen y encargándose de los problemas de concurrencia.

Es un servicio de alto rendimiento para la persistencia de objetos relacionales y consultas de bases de datos. Su característica más especial es que evita escribir las consultas SQL en código lo que hace a la aplicación que la usa ser más portátil.

Hibernate implementa JPA (Java Persistence API), que es la API de Persistencia para el lenguaje Java. JPA Se trata de una API que marca las pautas para poder representar una BD en clases Java.

La variante de Hibernate denominada Hibernate Annotations, que se encuentra ya integrada en el código actual de k-PAX. Hibernate Annotations utiliza las anotaciones definidas en la JPA y elimina la necesidad de utilizar ficheros XML, para guardar la

información de la estructura de las tablas de la base de datos que contienen en las clases de la persistencia, mediante una serie de anotaciones en su código.

Especificaciones de estándares

Validación de usuario

Para evitar que un usuario sin validar realice cambios en la base de datos, todas las funciones diseñadas deben comprobar que la sesión sea válida, por medio de la función ***validateSession*** de la clase uoc.edu.svrKpax.bussines.SessionBO.java antes de realizar ninguna llamada a las clases de la capa de persistencia. Por supuesto, en caso de que la sesión no sea válida toda respuesta tendrá que ser en forma de error indicando que la sesión no es válida.

4. Desarrollo

Instalación de la plataforma

La instalación se ha realizado en los siguientes equipos:

- Portátil HP con Windows 7, tecnología WAMP
- MacBookAir, tecnología LAMP

A continuación se detallan las notas de la instalación en ambos equipos:

Los pasos de la instalación están basados en las instrucciones que se pueden encontrar en los siguientes recursos pertenecientes al autor de este proyecto:

https://github.com/jsanchezramos/k-pax/wiki

https://github.com/mynt

Registro en **GitHub** y obtención de nueva rama de trabajo vía Git.

1. Registro en GitHub: Se ejecuta la acción y se obtiene un nombre de usuario: *santosdios*.

2. Obtener el código fuente del proyecto kPax de su repositorio, crear una nueva rama bifurcando el proyecto, ésta será la rama de trabajo.

3. Instalar GitHub en el ordenador. Teniendo especial cuidado al añadir la clave SSH en GitHub.

Clonar el repositorio desde el script de git que se acaba de instalar. Con esto ya se tiene el código en local y se puede trabajar en la codificación. Se obtiene el código y se guarda en: / Users / santosdios / Documentos / workspace / kpax donde podrá ser accedido desde Maven y Eclipse.

Plataforma de programación en Java

4. Descargar e instalar el **Java SE Development Kit 6** (JDK 6). Cabe notar que esta no es la última versión.

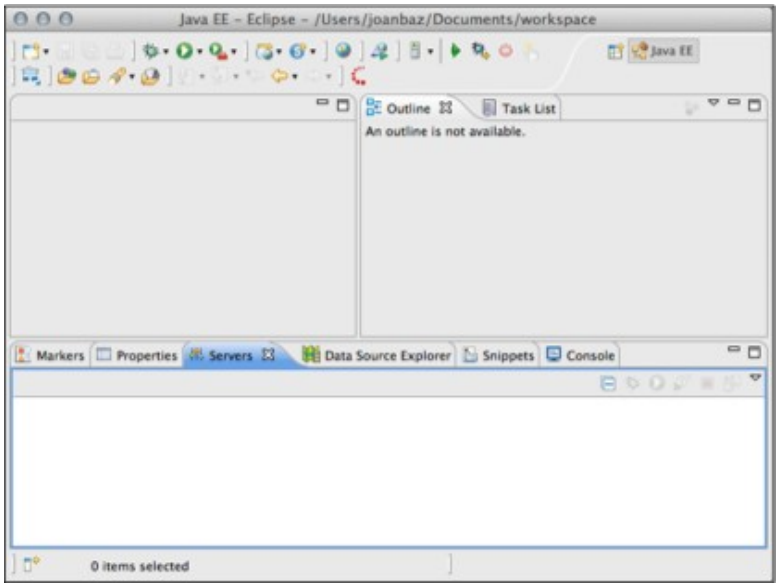
En entornos modernos OS-X, como el que ha utilizado el autor, el JDK 6 está instalado por defecto y están ajustadas las variables de entorno, no hay por tanto efectuar este paso.

Descarga e instalación de **Eclipse**. Para la instalación de Eclipse solamente tenemos que descomprimir dicha aplicación y albergarla en algún directorio del sistema.

Descarga e instalación **JBoss 4.2.3**. Debemos señalar que esta no es la última versión de JBoss. Se descomprime el fichero obtenido en el directorio / Users / santosdios / Documents / workspace / jboss-4.2.3.GA / y se siguieron las instrucciones de configuración de JBoss por Eclipse. Se siguen las instrucciones a partir de la sección "Create a localhost-only server configuration in Eclipse" correspondiente al enlace, la parte anterior de la página no es aplicable. Se instala el servidor para acceso local exclusivamente.

a) Se selecciona la perspectiva J2EE al Eclipse (Window-> Open Perspective-> J2EE)

b) Se abre la pestaña servidores



c) Se hace click en el espacio vacío debajo de "*Servers*" y se selecciona "*New: Server*". Se sigue el asistente de configuración especificando el directorio raíz de JBoss mencionado más arriba, cuando en el resto aceptan los valores por defecto, se añade localhost al nombre del servidor para hacerlo mas descriptivo.

d) En el últimos paso de la configuración se determina que el servidor sólo atienda peticiones locales.

5. Abrir una ventana de command y estando en el directorio raíz del proyecto Java, ejecutar '*MVN install*' para obtener todas las librerías de la plataforma. En este caso el proyecto Java se kPax y se encuentra en: / Users / santosdios / Documentos / workspace / k-pax, donde se había dejado en el apartado anterior correspondiente a Git.

6. Ejecutar MVN eclipse: eclipse para generar el proyecto Eclipse. Esta acción genera los metadatos necesarios para poder importar directamente después el proyecto al eclipse.

7. En Eclipse, crear la variable M2_REPO haciendo click derecho sobre el proyecto y seleccionando: Project properties -> Java build path -> Add variable, con nombre M2_REPO y ruta del repositorio (por ejemplo: / user/nom_de_usuari/.m2/repository /) . Esta acción se lleva a cabo sin complicaciones.

8. Modificar, en el fichero pom.xml, el apartado \ Users \ juanfras \ servidor \ JBoss-4.2.3-1.GA \ server \ default \ deploy con el camino donde esta localizado el servidor.

Se modifica cambiándolo por / Users / santosdios / workspace/jboss-4.2.3.GA/server/default/deploy

9. "Ejecutar MVN -Denver = local clean package. Este paso se debe repetir cada vez que se quiera compilar ".

10. "Añadir la librería mysql conector / j al JBoss. Esto es necesario para conseguir que el servidor tenga acceso a la base de datos después del despliegue.

11. Descargar el fichero de: http://www.mysql.com/downloads/connector/j/ y copiar el jar contenido en el fichero comprimido (por ejemplo en la versión actual: mysql-conector-java-5.1.19-bin . jar) en el directorio $ JBOSS_HOME / server / default / lib ".

Instalación de una infraestructura (X) AMPP

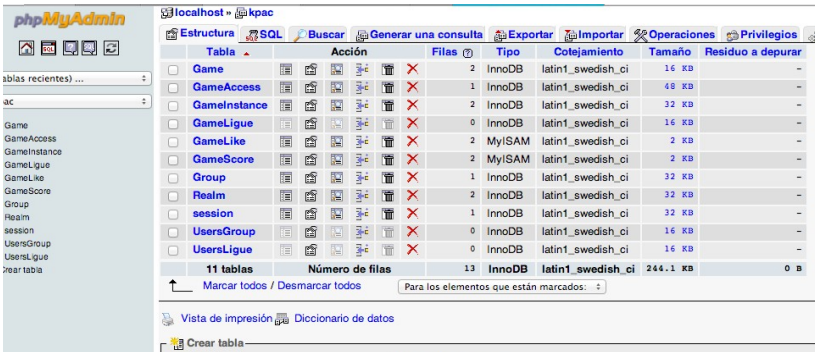
1. Se instala la aplicación MAMP para OS-X y AppServ para Windows, en ambos casos funcionan correctamente.

2. Bases de datos en MySQL para kPax y Elgg

Desde phpMyAdmin se genera el usuario y base de datos Elgg que tiene derechos administrativos en localhost:

También se genera al usuario kpac y la base de datos kpax en las mismas condiciones.

A continuación se generan las tablas de la aplicación usando el script que se puede encontrar en el directorio doc / sql del proyecto k-pax:



3. Se edita el fichero srvKpax-ds.xml para incluir la configuración de la base de datos (host, puerto, user, password), y se guarda en su sitio: / Users / santosdios / workspace/jboss-4.2.3 .GA / server / default / deploy

Código de Elgg

1. Se descarga el código de Elgg (version 1.8.X) de la página de descarga de Elgg y se guarda el contenido en el directorio de páginas del servidor: / Documentos / Elgg

2. Se activa el módulo de reescritura en el servidor web. Se edita / etc/apache2/httpd.conf y se cambia en la sección <Directory> *AllowOverride None* por *AllowOverride ALL*.

3. Se instala Elgg accediendo a: http://localhost/elgg-1.8.3/install.php

En el paso de "Requirements check", se marca la opción que muestra el parámetro del fichero httpd.conf file *AllowOverrride* seleccionado a *ALL*.

4. Se copia el contenido de la carpeta mod del proyecto Elgg a / Documentos / Elgg / mod

Los contenidos mencionados son:

- loginrequired

- kpax
- apiadmin
- likeKpax

5. Se activan los plugins apiadmin 1.8b1, kpax y likeskpax en la página de Administración de Elgg.

6. Finalmente se crean una pareja de claves en la administración de la API. Se incluyen los datos en los archivos:

/ Documentos / Elgg / mod / kpax / lib / kpaxSrv.php

$ apikey = "CLAVE CREADA";

/ Users / santosdios / Documentos / workspace / kpax / src / main / java / uoc / edu / svrKpax / util / ConstantsKPAX.java

public final static String ELGG_API_KEY = "CLAVE CREADA";

7. Por último compilamos de nuevo el código JAVA mediante el comando

"*mvn -Denv=local clean package*".

Incidencias durante la instalación

El desarrollo normal del proyecto se ha visto afectado por una incidencia grave: la imposibilidad de instalar los servicios de la plataforma kPax en un entorno local hasta el último momento.

En los dos equipos la parte de la plataforma ELGG no realizaba correctamente las llamadas a la parte servidora, por lo que no se podía comprobar el funcionamiento real de la aplicación durante las fases de análisis ni el diseño, lo cual ha retrasado considerablemente el proyecto.

Todos los pasos anteriormente descritos durante la instalación se ejecutan correctamente. La parte de los servicios JAVA relacionados con el módulo kPax parece que funcionan correctamente y de echo cuando cargamos el servidor Jboss mediante Eclipse la consola indica que las clases correspondientes a los servicios se han cargado correctamente:

```
INFO: Root resource classes found:
  class uoc.edu.svrKpax.rest.User
  class uoc.edu.svrKpax.rest.Games
  class uoc.edu.svrKpax.rest.Jsonp
```

Sin embargo cuando accedemos a la pestaña Games de nuestra plataforma ELGG-xPax la parte correspondiente a la información sobre los juegos no aparece y la consola de Eclipse muestra los siguientes mensajes de error:

```
12:57:07,693 INFO  [STDOUT] URL PETICION USER SIGN :
http://kpax.uoc.es/elgg/services/api/rest/xml/?
method=auth.sign&api_key=f9c64ef8683cf61f68f7266a4a459d098f3432a4&username=admin
12:57:07,861 ERROR [STDERR] 10-ene-2013 12:57:07
com.sun.jersey.api.container.filter.LoggingFilter$Adapter finish
INFO: 15 * Server out-bound response
15 < 401
15 <
12:57:07,870 ERROR [STDERR] 10-ene-2013 12:57:07
com.sun.jersey.api.container.filter.LoggingFilter$Adapter finish
INFO: 16 * Server out-bound response
16 < 404
16 <
```

Modificaciones realizadas en el proceso de instalación de la plataforma

Se realizan los siguientes cambios para solucionar los errores del servidor Jboss:

A. En el fichero ***kpaxSrv.php***, se tuvo que cambiar el valor de la variable $url para que apuntara al servidor JBoss en local que recibe las peticiones HTTP en el puerto 8080, es decir que se tuvo que cambiar por:

protected $url = "http://localhost:8080/webapps/svrKpax/";

B. En el mismo directorio que kpaxSrv.php, se tuvo que cambiar en el fichero ***kpaxOauth.php*** las variable URL y API_URL por las rutas al servidor Elgg y de Jboss (aunque parece que este paso no es relevante para la resolución del problema):

URL = 'http://localhost/elgg/';

API_URL = 'http://localhost:8080/webapps/srvKpax';

C. En la base de datos de kpax ha sido necesario insertar un usuario con el nombre del usuario administrador de Elgg (***admin***), con privilegios para la tabla kpac.

D. En el fichero ConstantsKPAX.java,a demás de la clave principal se ha modificado la siguiente línea de código:

public final static String URL_ELGG = "http://localhost:8888/elgg/"

A continuación se ejecuta el último paso: compilamos de nuevo el código JAVA mediante el comando "*mvn -Denv=local clean package*".

El servidor JBoss muestra un nuevo error:

13:10:55,028 ERROR [JDBCExceptionReporter] Table 'kpac.User' doesn't exist

Se detectan varias tablas que faltan en la BD Kpac:

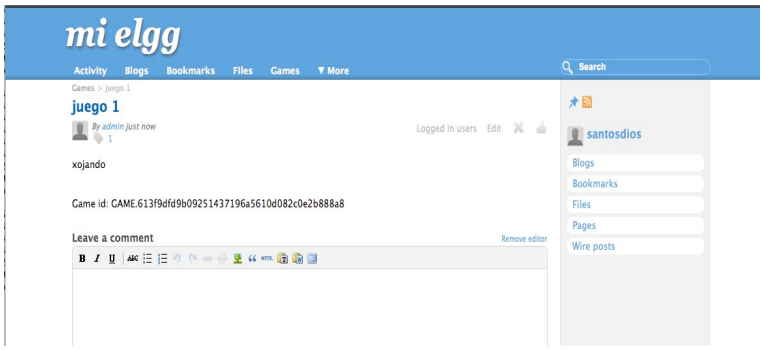
- User

- Session (con mayúscula la primera letra, ya que la tabla session ya existía)

Se procede a crear las tablas que faltan

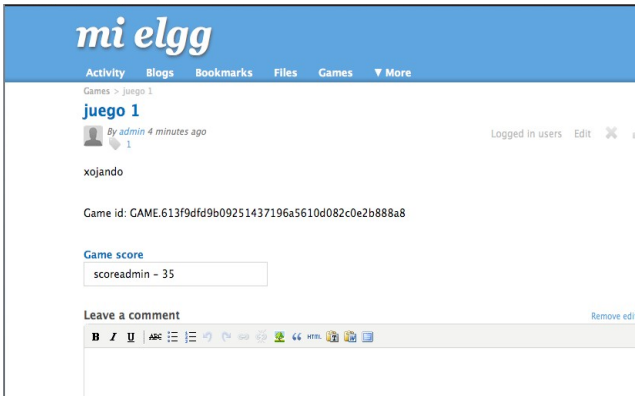
Después de realizar todos estos cambios por fin concluimos la instalación de los servicios de la plataforma kPax en un entorno local.

Una vez realizados estos pasos podemos ver la información adicional sobre un juego como el campo IdGame:



Si añadimos registros a la table GameScore con la id de los usuarios, la id de los juegos:

Podemos ver los cambios reflejados en la plataforma:



Implementación

Debido a la grave incidencia descrita anteriormente, la mayor parte del proceso de implementación no se ha podido desarrollar correctamente, de todos modos a continuación se describe la parte de código desarrollada así como los pasos necesarios para concluir definitivamente el proyecto.

Modificar las pestañas All Mine Friends

Debemos copiar en el directorio del plugin, en este caso Kpac, el fichero: views/default/layouts/content/filter.php

NOTA: Durante el desarrollo de plugins es muy importante actualizar o habilitar y des-habilitar el plugin cada vez que una nueva vista se adicione

Añadimos las siguientes líneas de código al fichero para crear la pestaña Players:

```
'players' => array(
                'text' => elgg_echo('players'),
                'href' => (isset($vars['player_link'])) ? $vars['player_link'] :
"$context/players/$username",
                'selected' => ($filter_context == 'players'),
                'priority' => 400,
```

),

Modificaciones y nuevas funciones añadidas

Durante la fase de diseño no se han tenido en cuenta algunas consideraciones que han propiciado la modificación de funciones Java o la creación de nuevas, el desarrollo inicial solamente contemplaba la modificación del código PHP de la plataforma Elgg y el acceso a los datos mediante la utilización de las funciones del fichero kpaxSrv.php.

Esto ha sido debido a mi inexperiencia con JAX-RS o el funcionamiento de Elgg, así como por no haber podido instalar correctamente la plataforma en local hasta un fecha muy cercana al límite de entrega del proyecto.

Las modificaciones son las siguientes:

Capa de persistencia:

A. Interfaces definidas en el paquete *uoc.edu.svrKpax.dao*:

i. *UsersGroupDao*, con las funciones siguientes:

a) *getUsersGroup*: devuelve el listado de usuarios de grupos.

b) *getUsersGroup*: devuelve la información del usuario de grupo con id *idUser*.

c) *addUsersGroup*: Almacena la información de un usuario dentro de un grupo.

d) *delTagsGame*: Elimina todos los tag de un juego con id *idGame*

ii. *GameLigueDao*, con las funciones siguientes:

a) *getAllGameLigues*: devuelve el listado de todas las categorías de juegos posibles.

b) *getGameLigue*: Devuelve la información de la categoría con id *idGameLigue*.

iii. *UsersLigueDao*, con las funciones siguientes:

a) *getAllUsersLigue*: Devuelve el listado de todos los usuarios de una liga con id *idGameLigue*

b) *getLigue*: Devuelve la información de una liga con id *idGameLigue*

c) *addUserLigue*: Almacena la información de un usuario de una liga.

d) del*UserLigue*: Elimina un usuario de una liga.

iv. Sus respectivas implementaciones: *UsersGroupDaoImpl*, *UsersLigueDaoImpl* y *GameLigueDaoImpl*.

B. En el paquete *uoc.edu.srvKpax.vo*: En este paquete se han definido las **clases que permitirán realizar el intercambio de la información de cada entidad** entre el resto de interfaces y clases:

i. Clase *UsersGroup*: Que incluye un campo *idUsersGroup* para establecer su identificador, *así como el resto de los campos de la tabla correspondiente UsersGroup*., y los campos *idGroup y* guid para relacionar el grupo con el identificador de un usuario.

ii. Clase *GameLigue*: Que incluye un campo *idGameLigue* para establecer su identificador.

iii. Clase *UsersLigue*: Que incluye un campo *idUsersLigue* para establecer el identificador de un usuario de una liga.

iiii. Clase *User*: No se ha modificado. Sólo se muestra en el diagrama para representar la relación entre la entidad de grupos y la de usuario.

Capa de lógica:

A. En el paquete uoc.edu.srvKpax.bussines.

i. Interfaz *UsersGroupBO*:

a) *listUsersGroup* : devuelve el listado de usuarios de un grupo con identificador *idGroup* si la sesión *campusSession* es válida.

b) *addUsersGroup* : Añade un listado de Users al grupo con identificador *idGroup*, si la sesión *campusSession* es válida.

c) *delUsersGroup* : Elimina todos los Users de un juego con identificador *idGroup*, si la sesión *campusSession* es válida.

ii. Interfaz *UsersLigueBO*:

a) *listUsersLigue* : devuelve el listado de usuarios de una liga con identificador *idLigue* si la sesión *campusSession* es válida.

b) addUsersLigue : Añade un listado de Users a la liga con identificador *idLigue*, si la sesión *campusSession* es válida.

c) *delUsersLigue* : Elimina todos los Users de una liga con identificador *idLigue*, si la sesión *campusSession* es válida.

iii. Interfaz *GameLigueBO*:

a) *listGameLigue*: devuelve el listado de usuarios de una liga con identificador *idUser* si la sesión *campusSession* es válida.

b) *addGameLigue*: Añade un usuario a la liga con identificador *idLigue*, si la sesión *campusSession* es válida.

c) *delGameLigue*: Elimina un usuario con identificador *idLigue*, si la sesión *campusSession* es válida.

Sus respectivas implementaciones:

a) GameLigueBOImpl

b) *UsersLigueBOImpl*

c) UsersGroupBOImpl

Ficheros de configuración

Como indicamos anteriormente, la capa de persistencia cuenta con una serie de ficheros de configuración:

1. Hibernate.cfg.xml: Contiene los datos de acceso a la base de datos para facilitar la portabilidad de las aplicaciones independientemente de la base de datos a la que acceda.
2. ApplicationContent: L.as variables de las clases del paquete uoc.edu.srvKpax.bussines deben ser inicializadas con objetos de sus clases implementadoras. El fichero ApplicationContent permite inicializar las variables de forma automática mediante sus funciones get y set.

Implementación para el acceso a las nuevas tablas creadas en la Base de Datos kPax

A continuación relatamos un ejemplo de implementación para el acceso a los datos de una de las nuevas tablas creadas en kPax, para ello debemos crear y modificar varias clases de la parte de los servicios java, el siguiente ejemplo contempla el acceso a la tabla ***GameLigue*** de la Base de Datos kPax:

Se deben incluir las siguientes clases:

- Un fichero GamesligueBO.java en la carpeta business
- Otro fichero en la misma carpeta que se llame GamesligueBOImp.java
- Dos ficheros: GamesligueDao.java y GamesligueDaoImpl en la carpeta dao
- Un fichero Gamesligue.java en la carpeta vo

En el fichero **Games**, de la carpeta rest, añadiremos el tratamiento de la llamada desde PHP:

```
import uoc.edu.svrKpax.bussines.GameLiguesBO;
import uoc.edu.svrKpax.vo.GameLigues;
```

Donde están las diferentes llamadas:

```
/* gameligues */
@GET
@Path("/Gameligue/{param}/list")
@Produces( { MediaType.APPLICATION_JSON ,MediaType.APPLICATION_XML})
public List<Gameligue> getGameligues (@PathParam("param") String campusSession){
return LigueBo.listGameligues(campusSession);
}
```

y al final de todo:

```
public void setLigueBo( GameLiguesBO LigueBo) {
this.ligBo = ligBo;
}
```

Implementación en los ficheros de configuración

El fichero **applicationContent.xml** que está en src/main/resources se añadirá el siguiente código:

```
<bean id="ligDao" class="uoc.edu.svrKpax.dao.GamesligueDaoImpl">
<property name="sessionFactory" ref="sessionFactory" />
</bean>

<bean id="ligBo" class="uoc.edu.svrKpax.bussines.GamesligueBOImp">
<property name="sBo" ref="sBo" />
<property name="ligDao" ref="ligDao" />
```

```
</bean>
```

(entre <!-- GAMES --> y <!-- END GAME --><!-- END BUSSINES →)

El fichero Hibernate.cfg.xml contiene los datos de acceso a la BD, los datos de configuración del pool de conexiones a la BD y muchas más propiedades para configurar las conexiones que realiza Hibernate. No se han realizado cambios en este fichero.

Modificaciones de la plataforma Elgg

Como indicamos anteriormente, cada una de las funciones debían añadirse una a una en el fichero kpaxSrv.php. La clase dispone de la función service que permite realizar una llamada vía GET o POST, a una URL concreta indicando los parámetros a enviar. Dependiendo de si el servicio recibe la petición por POST o por GET, se deben incluir las llamadas siguientes:

- GET: this->service($url);
- POST: $this->service($url, “POST”, $body);

Donde $url es una url que concuerda con el solicitado en la anotación @Path -los valores incluidos son recogidos por los parámetros con @PathParam- de la clase Games o Jsonp y $body sigue las reglas HTML para enviar parámetros por Get o Post -con Get, el primer parámetro debe venir seguido de un símbolo de interrogación final '?' y el resto de parámetros del ampersand '&', por ejemplo: ?field=name&values=a; y en el POST el inicial no viene seguido de nada: field=name&values=a -.

Finalmente en kpaxSrv.php deberíamos tener:

```
public function getGameLigues($campusSession) {
$listGameligues = json_decode($this->service("game/gameligue/" . $campusSession . "/list/"));
return $listGameligues;
}
```

Una vez implementado todo el código en las clases nombradas sólo tendríamos que declarar una variable de la clase y hacer una llamada a la función getGameLigues para acceder a los datos de la tabla.

```
$objKpax = new kpaxSrv(elgg_get_logged_in_user_entity()->username)
```

La clase kpaxSrv se importa a elgg desde el fichero ***start.php*** de la raíz del módulo. En este fichero se añaden las librerías de la carpeta lib o las acciones de la carpeta actions entre otras cosas y permite a elgg conocer la ruta de todos los ficheros PHP que incluye el módulo.

```
$listGameligues = $this->getGameLigues($campusSession);
```

Para codificar la función me base en la función getGames del fichero kpaxSrv.php. En esta función, se devolvía el valor por medio de la función var_dump, que al ejecutarla en elgg se mostraba el array de juegos como una cadena de texto por lo que su contenido no era accesible, para solucionarlo utilicé la función json_decode.

Elgg pone a disposición del desarrollador una diversa cantidad de funciones que permiten interaccionar con elgg y su base de datos. Las funciones utilizadas más importantes son las siguientes:

- **register_error**(elgg_echo("mensaje de error")): Muestra un mensaje de error en color rojo por medio de un div en la esquina superior derecha del navegador tras terminar la carga de la pantalla. Este mensaje se desvanece al cabo de un tiempo. Se muestra un mensaje de error por cada register_error ejecutado.
- **system_message**(elgg_echo("mensaje")): Muestra un mensaje informativo en color negro por medio de un div en la esquina superior derecha del navegador tras terminar la carga de la pantalla. Este mensaje se desvanece al cabo de un tiempo. El mensaje no se muestra si se ha registrado algún mensaje de error.

- **elgg_get_logged_in_user_entity()**: recupera un objeto de la clase ElggEntity que contienen la información del usuario. Su variable username contienen el nombre del usuario. Dato requerido para construir un objeto de la clase kpaxSrv.
- **get_subtype_from_id**: Esta función se ha utilizado en el fichero add.php y delete.php del directorio elgg/actions/comments, para determinar si el comentario a añadir/eliminar es sobre un juego, de subtipo kpax. Así se evita intentar añadir/eliminar un comentario en la base de datos de kpax cuando el comentario no está asociado realmente a ningún juego.
- **elgg_list_entities**: Esta función es utilizada en los ficheros all.php, owner.php y friends.php de la carpeta elgg/mod/kpax/pages/kpax/ para listar en elgg de forma formateada (tal y como se hace actualmente) los juegos recuperados con el servicio getGames.

  Requiere de un array con las opciones de búsqueda que permite filtrar, ordenar y paginar el conjunto de entidades a listar. El conjunto de opciones por defecto utilizado para listar los juegos en los tres ficheros es el utilizado en el fichero all.php hasta el momento:

```
$options = array(
'types' => 'object',
'subtypes' => 'kpax',
'limit' => 10,
'full_view' => false,
);
```

- El campo 'types' y 'subtypes' restringe la búsqueda a los objetos de subtipo kpax, es decir, los juegos.
- Con el campo 'limit' se establece un límite de 10 juegos por página.
- La opción 'full_view' indica que no se muestren ni la descripción, ni la categoría ni la fecha de creación del juego, que son mostradas con full_view = true. Los campos a mostrar con full_view = false y full view = true se definen en el fichero

object/kpax.php. Este fichero es el que se tendría que haber modificado si se hubiera incluido el cambio de aspecto previsto en el apartado de prototipado en la fase de análisis.

El listado de juegos propios y de juegos amigos requerían además de otros dos campos:

- El de juegos propios requería el campo '*container_guid*' indicando como valor el identificador del usuario propietario.
- Para el listado de juegos de amigos se ha utilizado el campo 'o*wner_guids*' indicando los identificadores de los usuarios propietarios. Estos valores se obtienen por medio de la función *elgg get_user_friends_of,* que devuelve un array de objetos de la clase ElggUser hasta el limite establecido como parámetro.

5. Conclusiones

Software de desarrollo

Sistemas operativos

Para el desarrollo del proyecto se ha utilizado la distribución OSX 10.8 sobre un MacBook Air.

Accesoriamente se ha utilizado una distribución de Windows 7 sobre un portátil HP.

Edición de código

Para la implementación del código de la parte servidora (lenguaje Java), se ha utilizado el IDE *Eclipse*. Para el desarrollo de la parte cliente, en PHP, se ha utilizado la aplicación *Macromedia Dreamweaver CS6* en algunos casos y accesoriamente un editor de texto plano, además de los navegadores web *Firefox* y *Safari*.

Otras herramientas utilizadas

Además se ha utilizado la herramienta de edición de código *Xcode* integrada por defecto en el Sistema Operativo OS X, para realizar modificaciones rápidas de código.

Para la elaboración de la documentación del proyecto, además de las presentaciones realizadas se ha utilizado el paquete *OpenOffice 3.4.1.*

Para la realización del video explicativo del proyecto se ha utilizado el *Quick Time Player*, integrado en el Sistema Operativo OS X.

Hitos conseguidos y no conseguidos

Retraso del proyecto y problemas para instalar Kpax en entorno local

El proyecto se ha visto afectado por varios problemas graves que han retrasado el proyecto e impedido la correcta realización de las fases en sus plazos previstos. Como se ha indicado anteriormente, se detecta que la documentación sobre la instalación de la plataforma kPax en un entorno local está incompleta, por lo tanto la parte Elgg no realizaba correctamente llamadas a la parte servidora y por eso no se podía comprobar el funcionamiento real de la aplicación durante la realización de las fases de análisis ni diseño. Esto ha propiciado que una parte importante de la realización del proyecto ha estado dedicada a solucionar los problemas de instalación de la misma y a documentarlos, este hecho retrasa gravemente el desarrollo del proyecto de mejoras de la plataforma kPax.

Durante las últimas semanas del proyecto se consigue solucionar los problemas de instalación y es cuando comienza realmente la fase de implementación y mejoras del mismo.

En esta fase, en la cual se ha podido por fin acceder realmente a todos los servicios de kPax, y una vez estudiado el funcionamiento de la plataforma, se llega a la conclusión de que para implementar las mejoras de la gestión de usuarios que propone dicho proyecto no es suficiente con trabajar con el código PHP de la plataforma ELGG, que era la idea inicial del proyectante, si no que también es necesario modificar las clases .java así como crear nuevas clases ya que los servicios

existentes en el módulo kPax no son suficientes para realizar las llamadas necesarias a la Base de Datos, lo cual añade una nueva dificultad al proyecto.

Además, parte de las funciones ya existentes en el fichero kpaxSrv.php estaban incompletas o en fase de desarrollo, por lo que algunas de ellas carecían de la clausula *return*, y devolvían error a la hora de llamarlas, y otras contenían la función *var_dump*, que no era adecuada para manejar los arrays de datos ya que los imprimía directamente en pantalla.

Hasta este momento sólo se había podido codificar los servicios basándose en el código existente y según el diseño del proyectante, en un principio decidí entregar únicamente la implementación algunos servicios web. La mayoría de estos servicios no se han podido probar en la plataforma debido a la falta de tiempo, así como tampoco se han podido realizar modificaciones considerables en la interface de la plataforma ELGG.

Hitos no conseguidos

Finalmente, se ha conseguido incluir el uso de los servicios web en Elgg a costa de eliminar otras funcionalidades.

La capa de la presentación, encargada de interactuar con el usuario no ha sido totalmente desarrollada. Por lo que algunos ficheros PHP contenidos en módulos que realizan comunicaciones con la capa de negocio por medio de los servicios web no están finalizados.

De forma resumida, las funcionalidades más importantes eliminadas son:

- No se ha estudiado la posibilidad de invitar a usuarios a juegos
- No se han implementado los servicios de búsquedas de usuarios
- No se ha implementado el sistema de ligas y ranking
- El resto de los servicios carecen de interface en la plataforma, solamente se ha codificado el acceso a los datos mediante los servicios web

Hitos conseguidos

Las funcionalidades implementadas son.

A. En la parte de los servicios web en Java:

- Se ha desarrollado íntegramente la Capa de Persistencia y sus correspondientes cambios en la Base de Datos kpax, relacionadas con la mejora en la gestión de usuarios.
- Se ha desarrollado parcialmente la Capa Lógica, concretamente:

  a. Interfaces BO y sus implementaciones:

  UsersGroupDao, GameLigueDao, UsersLigueDao

  UsersGroupDaoImpl, UsersLigueDaoImpl y GameLigueDaoImpl.

  UsersGroupBO, UsersLigueBO, GameLigueBO

  GameLigueBOImpl, UsersLigueBOImpl, UsersGroupBOImpl

  b. Servicios web o funciones ofrecidas para que sean accedidas desde la capa de presentación con elgg.

  UsersGroup, GameLigue, UsersLigue, User

  c. Ficheros de configuración

  ApplicationContent.xml

  d. Creación de nuevas funciones en las clase *Games* (getUsersLigue, getUsersGroup, getGameLigue, etc) y *Jsonp* del paquete uoc.edu.svrKpax.rest que obtuvieran un objeto UsersGroup, GameLigue o UsersLigue, con la información del objeto correspondiente.

B. En la parte de la plataforma Elgg en PHP:

- Modificación del fichero kpaxSrv.php, donde se añaden las funciones de acceso a los servicios web anteriormente descritos.
- Modificación del objeto kpax.php, que realiza la llamada a las funciones establecidas en el anterior fichero kpaxSrv.php.

– Creación de una nueva pestaña llamada “Players”, mediante la modificación del fichero filter.php.

Valoración personal del proyecto

Escasa documentación del proyecto

La escasa documentación de la plataforma kPax, sobre el que se realiza el actual proyecto de mejoras, ha sido uno de los principales problemas que ha retrasado el mismo, para paliar estos problemas en el futuro se han generado documentos nuevos, como el manual de instalación actualizado y otra documentación que se adjunta en el anexo.

JAX-RS: Java API for RESTful Web Services es una API Java que provee soporte a la creación de servicios web acordes con el estilo arquitectónico Representational State Transfer, o REST, utilizando anotaciones, lo que simplifica el desarrollo y despliegue de clientes web service. Debido a lo relativamente novedoso de esta tecnología existe escasa documentación sobre su uso en la red.

También faltan descripciones de los requerimientos por los que se ha hecho el desarrollo anterior de la plataforma, así como una descripción detallada de las funcionalidades que se habían logrado hasta la versión actual, de las que se piden en un futuro y del funcionamiento interno de los distintos elementos que la componen.

Valoración de la plataforma kPax y el actual proyecto

Después de analizar a lo largo de este semestre los módulos de la plataforma xPax implementados sobre Elgg, podemos realizar las siguientes valoraciones:

Escasa implementación del proyecto ya existente

La plataforma xPax sobre el que se sustentaban las mejoras a implementar en el actual proyecto se encuentra muy poco desarrollada, a opinión del relator del proyecto, en la fecha de la realización del actual proyecto la plataforma sólo contaba con una pestaña llamada “Games” en la que se mostraba una serie de entradas a la entidad Blog, propias de la plataforma Elgg, y con un módulo kPax que lo único que mostraba

era la puntuación obtenida por un juego, además de la carencia casi absoluta de estos juegos educativos que serían la base que sustentaban dicha plataforma.

En concreto se ha detectado las siguientes carencias:

- Ninguna información sobre los juegos, ficha de un juego, autor, desarrollo del mismo, normas, etc.
- Ninguna información sobre el modo de añadir juegos, puesto que el botón Add realiza una entrada a la entidad Blog propia de Elgg, pero sin especificar como vincular un juego a dicha entrada al blog.
- Tampoco existe información sobre como modificar o eliminar los juegos.
- No están establecidos los idiomas de la plataforma.
- Las herramientas de la plataforma Elgg relacionadas con las redes sociales no están orientadas a los juegos educativos, no existen foros temáticos, ni sistema de comunicación entre los usuarios de dichos juegos. Por lo tanto tampoco se están aprovechando plenamente las funcionalidades propias de la plataforma Elgg.

Aprovechamiento de la plataforma Elgg y de los módulos ya existentes

Otro problema detectado luego de estudiar el funcionamiento de Elgg y los diferentes módulos o plugins existentes en la red para añadir funcionalidades, es que no se están aprovechando correctamente tanto las propias herramientas de la plataforma como aquellas que proveen dichos módulos.

Se ha creado una nueva Base de Datos, llamada kpax, añadida a la propia Base de Datos que maneja la plataforma Elgg, que complica el desarrollo del proyecto, cuando se podrían aprovechar módulos ya existentes que trabajan sobre la propia Base de Datos del sistema.

Por ejemplo la creación de unas tablas: Users, Realm o Sessions, que realizan anotaciones que se podrían hacer sobre la propia Base de Datos Elgg, y ampliar por medio de módulos descritos en el capítulo de Alternativas de esta propia memoria.

Esta duplicidad de tablas en dos Bases de Datos distintas, además de complicar el desarrollo de la plataforma kPax, impiden que se pueda adecuar la implementación de estos módulos ya existentes, que simplificaría enormemente el trabajo sobre la misma.

Futuro

Futuras líneas de trabajo propuestas para el actual proyecto

Como se ha explicado anteriormente gran parte del actual proyecto se ha quedado sin desarrollar por lo tanto describimos a continuación las futuras líneas de actuación necesarias para desarrollar definitivamente el proyecto.

A. Modificaciones en la plataforma Elgg:

- Desarrollo de una nueva pestaña (Players) con los accesos a las nuevas funcionalidades relacionadas con la gestión de usuarios. Estableciendo un menú principal de acceso a la búsquedas de jugadores, creación/gestión de grupos, sistema de invitaciones a juegos, etc
- Desarrollo de las funciones del fichero kpaxSrv.php con llamadas a los nuevos servicios web.
- Modificaciones en el objeto kpax.php con llamadas a estas funciones.

B. Modificaciones en los servicios web de kPax:

- Creación de nuevas clases para implementar los servicios de búsquedas de jugadores e invitaciones de jugadores a un determinado juego.

Futuras líneas de trabajo propuestas para la plataforma kPax

Se propone mejorar la plataforma Elgg-kPax con:

- Estudio detallado de la propia plataforma Elgg y de la gran variedad de módulos existentes en la red para añadir funcionalidades concretas a la misma.

- Generación de documentación de todo el proceso de desarrollo, desde la recogida de requisitos del sistema hasta las líneas de código final. Especialmente la documentación referente a los módulos desarrollados para la plataforma.
- Mejorar la estructura del código, sobre todo, añadiendo comentarios a las partes relevantes, describiendo los requerimientos y funcionalidades, etc.
- Crear una API en Elgg – kPax.
    - Exponer los métodos
    - Implementar la autenticación de la API
    - Implementar la autenticación del usuario. Crear la llamada desde kPax
    - Exponer la invocación a la función kPax
- Simplificación del modelo con las partes ya realizadas, abandonar la vía de añadir nuevas tablas y utilizar el modelo de datos de Elgg aprovechando tanto éste como su capacidad de extensión para dar cabida a los nuevos datos y funciones. Intentando utilizar el modelo de datos propio de Elgg, aprovechando su capacidad de extensión para añadir nuevos datos, en lugar de crear nuevas tablas.

Bibliografía

Documentación sobre la plataforma ELGG

[1] **Elgg.org**, *Reference Manual*. [Online]. Available: http://docs.elgg.org/wiki/Web_Services

[2] Wiki Oficial de ELGG http://docs.elgg.org/wiki/Main_Page

[3] Documentación de la comunidad de ELGG http://community.elgg.org/groups/profile/20578/elgg-documentation

[4] ELGG Documentation in Spanish http://docs.elgg.org/wiki/Documentation_in_Spanish

[5] Github https://**github**.com

[6] Git http://git-scm.com

[7] Apache http://httpd.apache.org/

[8] Mysql http://www.mysql.com/

[9] phpMyAdmin http://www.phpmyadmin.net/

[10] PHP http://www.php.net/

[11] Redes Sociales en Educación Juan José de Haro http://danzanet.org/data/2011/10/21/35/file/1319411880redes_sociales_educacion.pdf

[12] Moodle y Elgg en Educación

http://community.elgg.org/groups/profile/1057/moodle-and-elgg-in-education

Anexos

ANEXO 1: DEFINICIÓN DE LAS PRUEBAS UNITARIAS

✔ PRUEBA UNITARIA 1 Validación de usuario
Descripción: Representa el inicio de una nueva sesión en la aplicación. Es un mecanismo de seguridad de identificación y autenticidad que impide el acceso a la aplicación a usuarios no autorizados. Consiste en la introducción de un identificador y una clave de acceso, que en caso de ser validos para el sistema, dará acceso al usuario a un panel de control con los permisos correspondientes al rol que tenga relacionado con su identificador. Todo identificador ha de tener asociado un perfil de privilegios. La identificación correcta del usuario permite a la aplicación conocer sus privilegios y autorizarle el uso de todas o parte de las funciones de la aplicación.
Secuencia:
1. El usuario accede a la página de login de la plataforma k-PAX. 2. El sistema solicita el identificador y la clave de acceso. 3. El usuario las facilita. 4. El sistema comprueba la identidad del usuario y su autenticidad. 5. El sistema muestra un panel de control diferente en relación a los permisos asociados al identificador.
RESULTADO ☐ **válida** ☐ **no válida**

✔ PRUEBA UNITARIA 2 Ranking de usuario
Descripción: acceso validado a el panel de información de ranking de usuario, el usuario modifica su puntuación en alguno de los juegos, a continuación accede al panel de información del ranking de usuario para visualizar la información modificada
Secuencia:
1. El usuario accede a la página de login y se valida correctamente 2. El usuario juega a algún juego y modifica la puntuación del mismo 3. El usuario accede al panel de información del ranking y visualiza los cambios realizados en la puntuación del juego
RESULTADO ☐ **válida** ☐ **no válida**

✔ PRUEBA UNITARIA 3 Creación de grupos de usuarios
Descripción: validación del usuario de forma correcta a través del formulario de login, el usuario accede a la información de los grupos a los que pertenece, el usuario crea un nuevo grupo de usuarios a través del correspondiente formulario

RESULTADO ☐ **válida** ☐ **no válida**

✔ PRUEBA UNITARIA 4 Invitación de usuario
Descripción: el usuario accede al correspondiente formulario donde selecciona los usuarios a los que invitar….
RESULTADO ☐ **válida** ☐ **no válida**

✔ PRUEBA UNITARIA 5 Búsquedas de usuario
Descripción: el usuario accede al formulario de búsquedas de usuarios y realiza una consulta de búsqueda. La información del usuario buscado aparece como resultado en la plataforma.
RESULTADO ☐ **válida** ☐ **no válida**

ANEXO 2: DEFINICIÓN DE LOS FICHEROS QUE CONTROLAN LOS SERVICIOS DE LA PLATAFORMA kPAX

Fichero **save.php**: adicción de un juego por medio de la llamada addGame

elgg/mod/kpax/views/default/forms/kpax/save.php

Fichero **kpax.php**: muestra un objeto de tipo kpack (ficha del juego)

elgg/mod/kpax/views/default/object/kpax.php

Fichero **add.php**: añade un comentario en la Base de Datos de ELGG

elgg/actions/comments/add.php

Fichero **delete.php**: elimina un comentario de la Base de Datos ELGG

elgg/actions/comments/delete.php

Fichero **all.php**: muestra el total de juegos de la plataforma

elgg/mod/kpax/pages/kpax/all.php

Fichero **owner.php**: muestra los juegos añadidos por el usuario

elgg/mod/kpax/pages/kpax/owner.php

Fichero **friends.php**: muestra los juegos añadidos por los amigos del ususario

elgg/mod/kpax/pages/kpax/friends.php

ANEXO 3: DEFINICIÓN DE LOS FICHEROS QUE COMPONEN LAS DIFERENCTES CAPAS DE LA PLATAFORMA

Capa de persistencia

Interfaces de acceso a los datos
uoc.edu.svrKpax.dao

Clases de intercambio de información
uoc.edu.srvKpax.vo

Ficheros de configuración
main/resources.

2. Capa de lógica

Interfaces BO y sus implementaciones
uoc.edu.srvKpax.bussines

Servicios web
uoc.edu.srvKpax.rest

Resto de clases de soporte para los módulos de Security, Authorization, etc.
uoc.edu.srvKpax.util.

ANEXO 4: Modificaciones de la BD xPAC: código SQL desarrollado

Nueva tabla UsersGroup, que relacione los usuarios con los diferentes grupos

UsersGroup

- **idUserGroup**: clave principal
- idGroup (clave de Groups)
- guid (clave de elgg_users_entity)
- rolUser: tipo de usuario dentro del grupo (jugador, capitán, suplente,…)

Sentencia SQL

```
-- Estructura de tabla para la tabla `UsersGroup`

--

CREATE TABLE `UsersGroup` (

  `idUserGroup` int(11) NOT NULL AUTO_INCREMENT,

  `idGroup` int(11) NOT NULL,

  `guid` int(11) NOT NULL,

  `rolUser` varchar(25) NOT NULL DEFAULT 'player',

  PRIMARY KEY (`idUserGroup`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
```

Nueva tabla GameLigue donde establecer ligas de jugadores

GameLigue

- **idGameLigue**: clave principal
- idGame (clave de Games)
- nameLigue
- description

Sentencia SQL

```
-- Estructura de tabla para la tabla `GameLigue`

--

CREATE TABLE `GameLigue` (

  `idGameLigue` int(11) NOT NULL AUTO_INCREMENT,

  `idGame` int(11) NOT NULL,

  `nameLigue` varchar(25) NOT NULL,

  `description` text NOT NULL,

  PRIMARY KEY (`idGameLigue`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
```

La tabla UsersLigue incluye los jugadores de cada liga

UsersLigue

idUsersLigue: clave principal

idGameLigue (clave de GameLigue)

guid (clave de elgg_users_entity)

Sentencia SQL

```
-- Estructura de tabla para la tabla `UsersLigue`

--

CREATE TABLE `UsersLigue` (

  `idUsersLigue` int(11) NOT NULL DEFAULT '0',

  `idGameLigue` int(11) NOT NULL,



`guid` int(11) NOT NULL,

  PRIMARY KEY (`idUsersLigue`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```