UNIVERSITAT OBERTA DE CATALUNYA

Estudios de Informática, Multimedia y Telecomunicaciones

Master en Software Libre

Incorporación de funcionalidades a la red social kPAX

Especialidad: Administración web y comercio electrónico
Nombre de la autora: Elena Sanchis Sanchez
Nombre del consultor: Manel Zaera Idiarte
Nombre del tutor externo: Daniel Riera Terren
Fecha de entrega: 1 de junio de 2013

Licencia de publicación del documento

Resumen

K-Pax es una plataforma de aprendizaje basada en juegos, transversal a todos los estudios de la UOC, que permite la incorporación de recursos de aprendizaje en línea o fuera de línea y que es accesible desde múltiples plataformas.

El objetivo de este proyecto es la implementación de la ampliación de las funcionalidades incluidas en la plataforma k-Pax, mas específicamente el desarrollo de un plugin que permita validar usuarios en kPAX a partir de sus usuarios de la red social twitter. Como plataforma se ha utilizado el motor libre para redes sociales elgg.

La naturaleza del proyecto implica su constante ampliación con aportaciones de diversas fuentes por lo que si el plugin cumple los requerimientos del sistema será incluido en su estructura.

Índex de continguts

Capítulo I. Introducción

1. Objetivos

El proyecto a desarrollar se incluye en el proyecto de innovación kPAX realizado en la UOC cuyo objetivo es la implementación de una plataforma tecnológica que soporte la inclusión de módulos independientes para el aprendizaje basado en juegos. El resultado es la base de una red de aprendizaje social que facilita la inclusión de juegos y módulos heterogéneos.

El objetivo específico del proyecto que la estudiante tiene que realizar es la creación de un plugin independiente que permitirá delegar la autenticación de usuarios de la plataforma Kpax a la través de la red social Twitter, sin tener que darse de alta en kPAX.

2. Estado de la técnica

El término juegos serios fue acuñado por Clark Abt en su libro “Serious Games” [Abt70] en el que se investigaba la utilización de juegos tradicionales para mejorar los procesos de aprendizaje. Con el tiempo, la masiva incorporación de las tecnologías de la información y la comunicación (TIC) propició la construcción de juegos de ordenador y simuladores basados en las ideas de Clark Abt. Aunque con un claro corte educativo, estos productos se caracterizan por incorporar elementos de entretenimiento tradicionalmente ligados a los juegos. Fue en 2002 cuando el Woodrow Wilson International Center for Scholars inWashington DC puso en marcha la "Iniciativa de Juegos serios” con el objetivo de fomentar el desarrollo de los juegos de ordenador con fines educativos.

Los avances en los juegos serios no se han dado únicamente en el campo del software sino que el desarrollo de nuevos periféricos, tecnologías GPS, equipos sensoriales o tecnologías móviles han facilitado su utilización desde múltiples contextos. Por otra parte la proliferación de redes sociales ha puesto de manifiesto la conveniencia de integrarlas con los juegos serios dando lugar a espacios en los que se puedan mejorar los procesos de aprendizaje de forma colaborativa. El proyecto kPax lleva a cabo esta idea mediante una plataforma tecnológica que facilita la incorporación de juegos heterogéneos disponibles en cualquier lugar y situación.

Una de las redes sociales más utilizadas es Twitter, cuya utilización se ha incrementado notablemente desde su creación en el año 2006 hasta el punto de que mas de quinientos millones de usuarios tienen una cuenta en este servicio de microblogging.

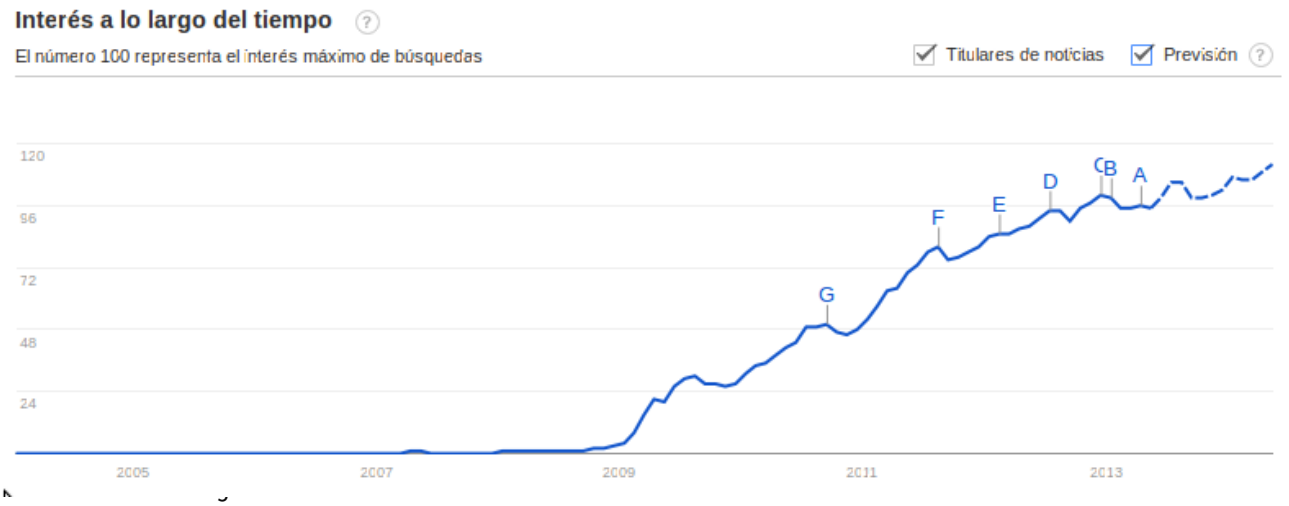


Figura 1. Evolución de tendencias de búsqueda de Twitter. Fuente: Google Trends.

Uno de los objetivos de la plataforma kPax es facilitar la incorporación de usuarios provenientes de diferentes ámbitos no necesariamente pertenecientes a la comunidad universitaria de la UOC por lo que se ha considerado oportuno permitir la autenticación en la red social kPax de usuarios de otras redes sociales, entre ellas twitter.

3. Estructura de la memoria

La memoria tiene la siguiente estructura: En el **capítulo I** se han incluido los objetivos, el estado de la técnica y la estructura de la memoria. El **capítulo II** trata sobre el establecimiento del alcance del sistema, el estudio de la situación, la definición de requisitos del sistema, estudio de las alternativas de solución, valoración de las alternativas y selección de la solución. El **capítulo III**, dedicado al análisis del sistema, consta de la especificación de la solución con la definición del sistema, de las interfaces y la especificación del plan de pruebas. En el **capítulo IV**, diseño del sistema, se incluye la arquitectura, la revisión de los casos de uso y la elección de alternativas de componentes. El **capítulo V** trata sobre el desarrollo y en él se explica la preparación del entorno de generación y desarrollo, la implementación, el trabajo desarrollado por el estudiante así como los principales problemas y soluciones aportadas. El capítulo V cierra con las conclusiones que incluyen los objetivos conseguidos y no conseguidos, las posibilidades de ampliación, las decisiones tomadas que ahora cambiaría la estudiante y la experiencia personal con el software libre.
En la bibliografía se cita la documentación utilizada para la elaboración del proyecto y por último están el Anexo I con la planificación del proyecto, Anexo II con la estructura de tablas y el Anexo III con la instalación de la plataforma.

Capítulo II. Estudio de viabilidad

1. Establecimiento del alcance del sistema.

kPAX es un proyecto de innovación docente incluido en el proyecto APLICA y desarrollado por profesorado de la UOC. Se trata de una plataforma de aprendizaje basada en juegos, transversal a todos los estudios de la UOC, que permite la incorporación de recursos de aprendizaje en línea o fuera de línea y que es accesible desde múltiples plataformas. El proyecto se encuentra finalizado aunque, al tratarse de una plataforma abierta, permite su ampliación mediante módulos heterogéneos independientes. En el caso que nos ocupa se trata de la incorporación de un plugin que permita la validación de usuarios a partir de los usuarios de Twitter.

La aportación económica a este proyecto de investigación proviene del Vicerectorado de Recerca i Innovacio de la UOC.

El código de la plataforma se ha publicado con licencia abierta (GNU GPL V2) para que programadores externos puedan revisarlo, mejorarlo y completarlo si lo desean.

A nivel operativo, la incorporación del nuevo módulo no modificará en ningún caso la funcionalidad que en estos momentos tiene la plataforma sino que, de llevarse a buen término, la ampliará.

2. Estudio de la situación actual.

La plataforma kPAX está implementada sobre el motor libre para redes sociales elgg y en estos momentos se encuentra en producción la primera versión. El código, abierto, se encuentra en github y está a disposición de la comunidad de programadores que irá añadiendo nuevas funcionalidades a partir de requerimientos todavía no incorporados.

Se utiliza la base de datos mysql, integrada en la plataforma, para crear y validar usuarios, dar de alta juegos y puntuaciones o listar rankings. También se está trabajando el diseño y codificación de juegos serios para añadir a la red social. Asimismo se están incorporando cambios en el diseño de la red social mediante el diseño artístico de las diferentes vistas del PAX.

Ya que el proyecto consiste en la aportación de un módulo y aunque este no afecte al funcionamiento global del sistema, deberemos conocer el diseño general de la plataforma y específicamente los módulos relacionados con el registro de usuarios. Para tener un

conocimiento lo más completo posible de la plataforma, se dispone del apoyo del tutor de prácticas externas. También es necesaria la participación del consultor de la asignatura para aportar soluciones a la parte técnica del problema.

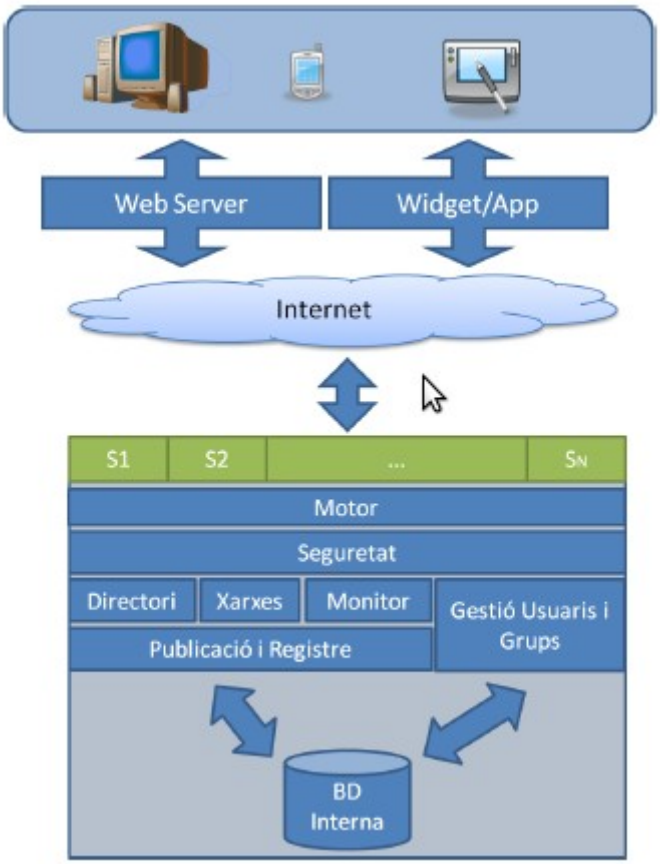


Figura 2. Esquema de la plataforma

3. Definición de requisitos del sistema.

A continuación se definen los requisitos del sistema de los que, en esta primera fase, se ofrece una visión global.

Requisitos técnicos

- Arquitectura
    - El usuario/a podrá validarse correctamente en la kPAX a partir de la contraseña de la red social Twitter.
    - kPAX tiene una arquitectura "núcleo" basada en servicios que a su vez tiene una librería que implementa estos servicios y permite desarrollar juegos y aplicaciones externas que actúan con el núcleo.

- El núcleo o core tiene un conjunto de bases de datos a mantener que en nuestro caso será la de usuarios/as y su relación con las redes sociales de las que son miembro.
- El core tiene un motor de servicios internos basado en Apache / Tomcat que permite mecanismos de comunicación con Sockets i/o XML y con canales que permitan transmisiones seguras cifradas, sobre todo lo relacionado con usuarios y redes sociales.
- Los servicios estructurados en módulos independientes que afectan a nuestro desarrollo son los siguientes.
- **GUG**: Módulo de Gestión de Usuarios /as y grupos
- **S:** Módulo de seguridad. Gestiona la seguridad en las comunicaciones y la gestión de la información.

- Normativa: La implementación está basada en estándares W3C multiplataforma (basados en CCS3 y Javascript) que son exportables a las diferentes plataformas.
- Seguridad: el sitio web solo podrá ser accedido por usuarios validados desde la red social Twitter.

Requisitos legales

- El plugin desarrollado deberá ser compatible con la licencia abierta GPL V2.

Requisitos operativos

- El plugin debe facilitar el acceso a la red social.
- La plataforma debe ser accedida desde cualquier

Requisitos económicos

- No existen requisitos económicos.

4. Estudio de las alternativas de solución.

No existen muchas alternativas para la creación del plugin ya que los aspectos técnicos vienen dados por la plataforma y el proyecto.

5. Valoración de las alternativas.

Dado que no intervienen factores económicos y que las únicas alternativas vienen dadas por el desarrollo de la programación en sí, la valoración es que tenemos que acogernos a las especificaciones dadas por las características del proyecto a medida que vayamos programando veremos las posibles alternativas que vayan surgiendo.

6. Selección de la solución.

Se instalará una plataforma para desarrollar la aplicación consistente en:

- Sistema de control de versiones Github
- Entorno de programación en Java
- Infraestructura WAMP
- Base de datos MySQL
- Código Elgg

El **procedimiento para la conexión** de los usuarios será la siguiente:

Habrá dos casos diferentes según sea la primera vez que se conecte el usuario/a o no: "alias" es el alias asignado al dominio (Realm) de twitter.

+newRealmUser(alias): idUser

+addUserRealm(idUser, alias): boolean

+login(alias):idSession

La **base de datos** contendrá toda la información relativa a los usuarios y dado que algunos provendrán de otra red social es posible que no tengan cuenta dentro de kPAX por lo tanto es posible que los campos login y password de la tabla User estén vacios.

La autenticación de los usuarios, en nuestro caso, será delegada. Los usuarios utilizaran un sistema externo para autenticarse. En este caso el usuario solicita ser autenticado por un servicio externo, twitter, que deberán ser dados de alta en la tabla Realm. Una vez autenticado el usuario en el sistema externo, se utiliza la información disponible del sistema externo para conseguir el usuario local utilizando las vinculaciones declaradas en la tabla UserRealm. En caso que el sistema externo autentique correctamente al usuario pero que el usuario del sistema externo no haya estado registrado en la table UserRealm se dará al

usuario como no autenticado.

El plugin que vamos a desarrollar permitirá delegar la autenticación a sistemas externos, realizando las acciones específicas de twitter y añadiendo la información necesaria en el vínculo entre usuarios y sistemas externos, traducido en algunos campos de información adicional a la tabla UserRealm.

Capítulo III. Análisis del sistema

1.- Introducción

El objetivo del análisis del sistema es establecer los objetivos del sistema así como determinar su funcionamiento y marcar las directrices que hacen posible alcanzar los objetivos propuestos.

2.- Especificación de la solución escogida en el estudio de viabilidad

En este apartado concretaremos la solución escogida en el estudio de viabilidad en el que se han tenido en cuenta las restricciones económicas, técnicas, legales y operativas.

2.1. Definición del sistema

Requisitos exactos del sistema

El sistema que vamos a desarrollar en este proyecto deberá cumplir una serie de requisitos:

- **Requisito 1.** Se creará un botón de twitter en el portal kPAX que haga posible la validación a los usuarios que no tengan cuenta en la base de datos de kPAX.
- **Requisito 2.** Los usuarios utilizaran un sistema externo para autenticarse de forma que la autenticación será delegada a la red social twitter que solo permitirá el acceso en caso de que se posea una cuenta.
- **Requisito 3.** En este caso el usuario solicitará ser autenticado por un servicio externo, twitter, y deberá ser dado de alta en la tabla Realm. Para ello se añadiran campos de información adicional a la tabla UserRealm con el objetivo de realizar la vinculación entre el usuario y el sistema externo.
- **Requisito 4**. En caso de que el usuario sea autenticado en la red social twitter, se utilizará su información para utilizar un usuario local a kPAX utilizando las vinculaciones que se han declarado en la tabla UserRealm. Aunque twitter realice correctamente la autenticación, ésta no será válida y el usuario no ha sido registrado en la tabla UserRealm.

Entorno tecnológico del sistema

- Correcto funcionamiento de la plataforma consistente en los siguientes elementos:
  - Sistema de control de versiones Github
  - Entorno de programación en Java
  - Infraestructura WAMP
  - Base de datos MySQL
  - Código elgg

Normas a seguir en el sistema

- La implementación del plugin estará basada en estándares W3C multiplataforma.
- El plugin desarrollado deberá ser compatible con la licencia abierta GPL V2.
- La plataforma deberá ser accesible tanto desde aplicaciones clientes de escritorio como mediante la web.

Identificación de usuarios

- Los usuarios validados en el sistema serán usuarios normales y podrán utilizar la plataforma para participar en juegos, hacer búsquedas de juegos, relacionarse con los otros miembros de la red social u otros, pero en ningún caso podrán realizar funciones de administración.

2.2. Definición de interfaces de usuario

Perfiles de usuarios

La plataforma social en red kPAX será utilizada por usuarios procedentes de diferentes ámbitos cuyo objetivo común es el aprendizaje mediante los juegos serios. En este sentido el plugin que vamos a desarrollar debe ajustarse a los requerimientos del entorno, utilizando, sobre PHP, MySql y algo de JAVA. Dado que lo único que hará el plugin es permitir el acceso a la plataforma a través de twitter tampoco es necesario acotar excesivamente el perfil de los usuarios potenciales del sistema, que podríamos diferenciar en dos grupos:

- Usuarios con experiencia en redes sociales ya que, al menos, tienen una cuenta en twitter.
- Usuarios con problemas de accesibilidad

Principios generales de la interfaz de usuario

El acceso a la red social kPAX tendrá las siguientes características:

- La validación se realizará mediante un botón con el logotipo de twitter. Estará en un lugar visible y cercano al botón de login.
- En caso de que la validación sea errónea se mostrará un mensaje de error en el que se indique que el usuario no tiene cuenta en twitter.
- El botón de acceso se hará mediante los principios de accesibilidad para que pueda ser visualizado con problemas de vista.

2.3. Especificación del plan de pruebas

La especificación del plan de pruebas nos permitirá decidir si el sistema cumple con los requerimientos planteados por los usuarios.

- **Pruebas unitarias.** Se hacen con el objetivo de comprobar el correcto funcionamiento por separado de cada uno de los módulos que forman el sistema. Estas pruebas se realizarán en la plataforma local.
- **Pruebas de integración.** Se realizan para comprobar que los módulos que forman un programa no fallen cuando trabajan de forma conjunta. En nuestro proyecto las pruebas unitarias y las de integración se hacen de forma simultánea ya que no tiene sentido probar el módulo sin estar integrado en el sistema.
- **Pruebas de sistema.** Sirven para de testar el funcionamiento de los módulos actuando de forma coordinada. Estas pruebas se harán en la plataforma local.
- **Pruebas de implantación.** Se hacen con el objetivo de comprobar el funcionamiento del sistema en su entorno de operación. La realización de pruebas de implantación en nuestro proyecto implica la comprobación de su correcto funcionamiento en la plataforma remota.
- **Pruebas de aceptación.** Las realizarán los usuarios finales del sistema para que comprueben su correcto funcionamiento.

Definición de las pruebas unitarias

- El portal kPAX será accedido correctamente mediante un botón de twitter que permita validarse a partir de la otra red social a los usuarios que no tengan cuenta.
- La validación en la kPAX a partir del servicio externo twitter se realizará correctamente.
- Cuando los usuarios que utilizan el sistema externo twitter para autenticarse no posean cuenta en esta red, la validación no podrá realizarse.

- La vinculación a la base de datos del sistema kPAX de los usuarios que entren por primera vez mediante una cuenta de twitter se hará de forma correcta.
- Cuando los usuarios que utilizan el sistema externo twitter para autenticarse no hayan sido dados de alta en el sistema kPAX, no se podrá realizar la validación.

Capítulo IV. Diseño del sistema

La especificación del diseño del sistema nos permitirá identificar los componentes del mismo y proporcionará una base para comenzar las siguientes fases del diseño.

1. Arquitectura

La arquitectura de la plataforma kPAX, ejecutada en un entorno web cliente-servidor, está estructurada en un modelo de tres capas donde el objetivo principal es separar los distintos aspectos del desarrollo, como los relacionados con la presentación, lógica de negocio o mecanismos de almacenamiento (persistencia). La aplicación debe conectarse al SGBD MySQL5 mediante el mapeador objeto-relacional (ORM) Hibernate.

- Persistencia o acceso a datos: En esta capa se realizará la conexión, mediante el SGBD MySQL5, a la base de datos en la que se almacenan todos los datos del sistema, así como todas las operaciones necesarias para su mantenimiento: acceso, modificación, inserción, eliminación, etc. En algunos modelos existe una separación entre la BD y el acceso a los datos pero en nuestro caso la base de datos está incluida en la capa de persistencia.

- Lógica de negocio: En esta capa se deben implementar los requisitos definidos en la fase de análisis. En la capa lógica se encuentran las funciones que se ejecutan, se reciben las peticiones de usuario, se procesa la información y se envían las respuestas. Esta capa se comunica con la capa de persistencia para solicitar o almacenar los datos y con la capa de preentación para recibir las peticiones y presentar los resultados.

- Presentacion: La capa de presentación es la encargada de presentar la interfaz gráfica al usuario final. Su función es la de lograr la interacción usuario-sistema comunicándose con la capa lógica.

Las capas lógica de negocio y persistencia utilizan el patrón de diseño modelo-vista-controlador (MVC) en las que se muestran las interacciones con el usuario.

- En el modelo (M) se representa la información de la lógica del negocio sobre el que opera la aplicación. A pesar de utilizar una base de datos para guardar la información, no se representa en este modelo ya que se encuentra encapsulada.

- La vista (V) proporciona un modelo de interacción con el usuario. En un modelo pueden existir varias vistas.

- El controlador (C) responde a las peticiones del usuario generando la vista adecuada en función de dichas peticiones.

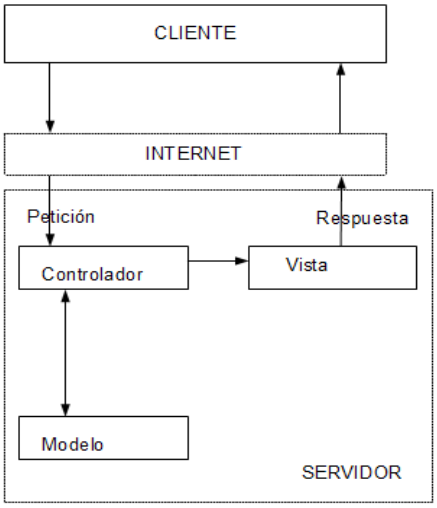


Figura 3. Arquitectura MVC

En el caso de nuestra plataforma kPAX la arquitectura en capas se concreta en el siguiente modelo:

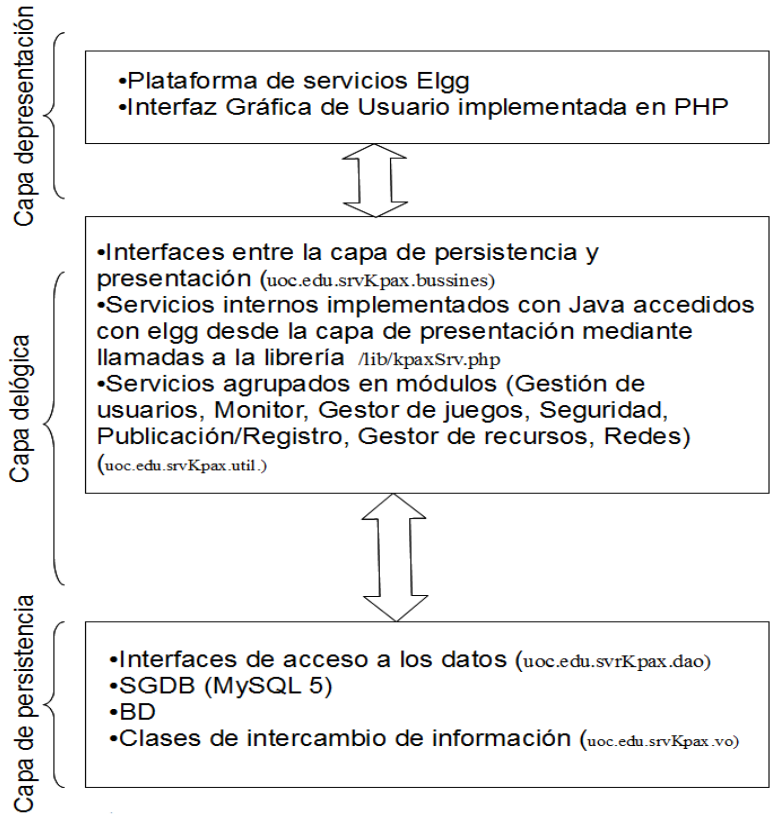


Figura 4. Arquitectura kPax

2. Revisión de los casos de uso

En esta fase se hará una revisión de los casos de usos realizados en la fase de análisis y se determinarán las operaciones que deberán implementar las interfaces de cada uno de ellos.

Dado que el sistema que estamos diseñando está centrado en el desarrollo, a continuación pasaremos a incorporar la definición de las clases (con sus atributos, operaciones y relaciones) así como los diagramas que representarán la interacción entre ellos.

2.1. Revisión de los subsistemas según los casos de uso.

Para cada uno de los casos de uso definidos anteriormente concretaremos cuales serán los objetos que intervienen y qué mensajes se intercambian entre ellos. Dado que en este proyecto hay un único subsistema se procederá a la descripción de las interfaces que intervienen teniendo en cuenta todos los casos de uso en que intervienen.

Vamos a introducir un cambio que afectará al Requisito 4 y que se refleja en el diagrama de casos de uso. Consideraremos que cuando un usuario está validado por twitter pero no existe en la BBDD de kPax se creará el nuevo usuario con un identificador relacionado con el dominio además de los datos propios del usuario.

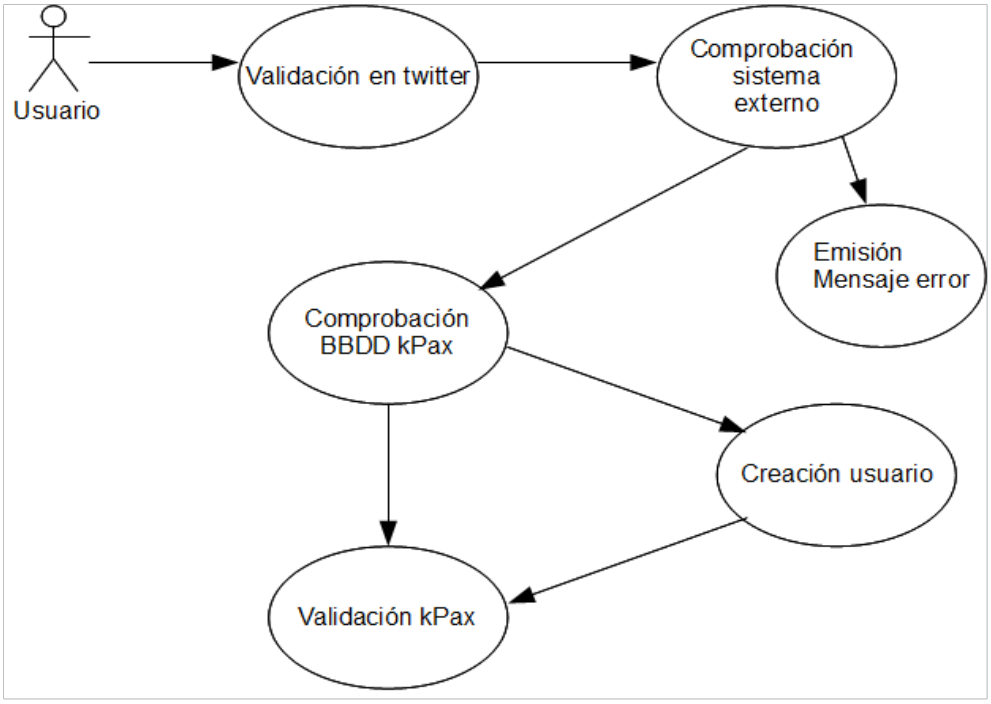


Figura 5. Diagrama de casos de uso

Caso de uso número 1: Validación en twitter

Resumen	El usuario de kPAX utiliza el botón de validación de

	twitter para validarse
Actores	Usuario
Casos de uso relacionados	
Precondición	El usuario no ha entrado en kPAX
Postcondición	El usuario ha utilizado el botón de twitter para validarse en kPAX.

Caso de uso número 2: Comprobación sistema externo

Resumen	El sistema externo comprueba si el usuario está dado de alta en su sistema.
Actores	Sistema externo
Casos de uso relacionados	Validación en twitter
Precondición	El usuario ha solicitado la validación mediante twitter.
Postcondición	El sistema externo ha comprobado la existencia del usuario en twitter y proporciona el identificador del usuario

Caso de uso número 3: Emisión mensaje de error

Resumen	Si el usuario ha sido validado
Actores	Sistema externo
Casos de uso relacionados	Comprobación sistema externo
Precondición	El sistema externo ha comprobado que el usuario no existe.
Postcondición	Twitter proporciona un mensaje de error

Caso de uso número 4: Comprobación Base de Datos kPAX

Resumen	Con los datos proporcionados por el sistema externo, comprobación de la existencia del usuario en la BD kPAX.
Actores	kPAX
Casos de uso relacionados	Comprobación sistema externo
Precondición	El sistema externo proporciona el identificador de dominio así como los datos del usuario de twitter.
Postcondición	El sistema ha comprobado si el usuario existe en la BD kPAX

Caso de uso número 5: Creación de usuario en kPAX

Resumen	Si el usuario ha sido validado por twitter pero no existe en la base de datos de kPAX se creará un nuevo usuario con el identificador de dominio proporcionado por el sistema externo.
Actores	kPAX
Casos de uso relacionados	Comprobación BD kPAX
Precondición	El usuario existe en twitter pero no existe en la BD kPAX
Postcondición	El usuario ha sido dado de alta en la BD kPAX

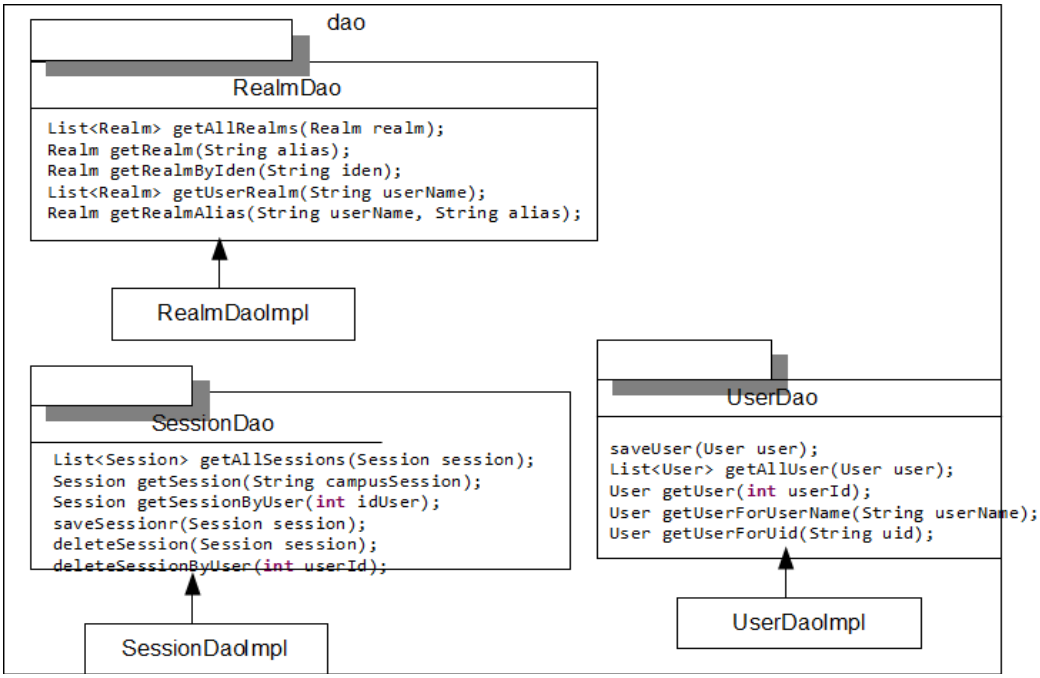
Caso de uso número 6: Validación en kPAX

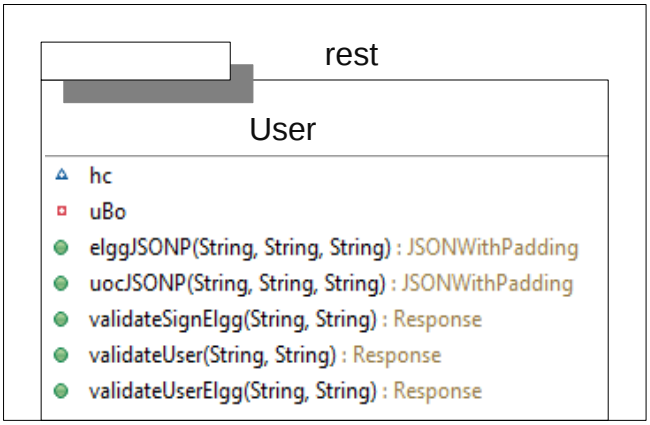
Resumen	Si el usuario ha sido validado por twitter y existe en la base de datos se procederá a su validación y se permitirá su acceso al sistema kPAX
Actores	kPAX
Casos de uso relacionados	Comprobación BD kPAX Creación de usuarios
Precondición	El usuario existe en la BD kPAX.
Postcondición	El usuario está validado en el sistema kPAX.

La capa de acceso a datos está implementada mediante el ORM Hibernate que conecta la Base de Datos, formada por un conjunto de tablas que corresponden a distintas entidades, y controlada por el SGBD MySQL 5. Hibernate facilita la independencia entre

la capa de presentación y la lógica. Cada entidad corresponde a una clase -que pertenece a la capa de lógica- y a una tabla de la base de datos.

La interfaz definirá los servicios web a los que elgg accederá para facilitar a la capa de presentación y que utilizarán las clases de la capa de acceso de datos. El diagrama de clases mostrado a continuación contiene únicamente las clases que intervienen en el proyecto y que se han modificado para implementar las nuevas funcionalidades.





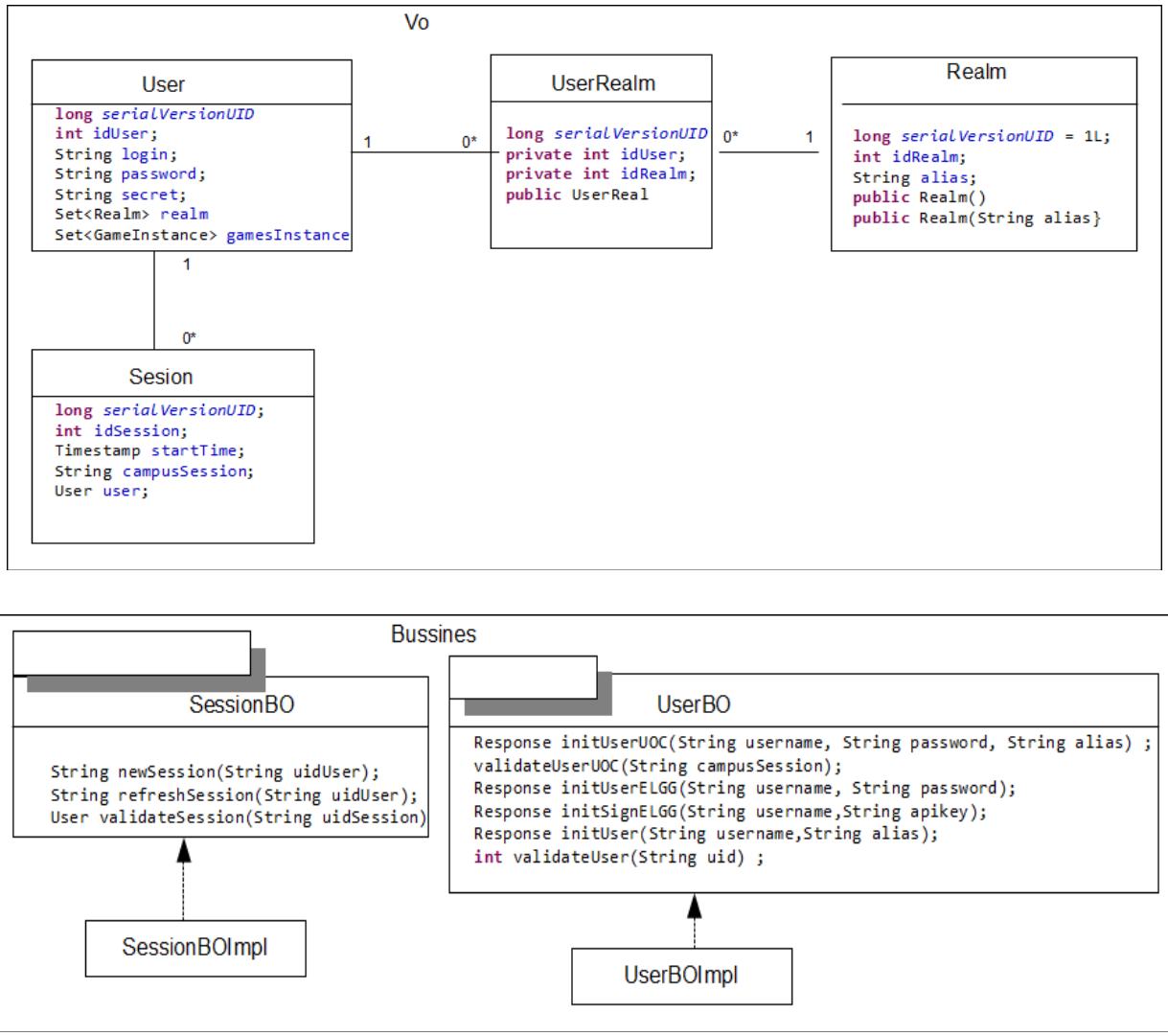


Figura 6. Diagrama de clases

2.2. Diseño de la base de datos

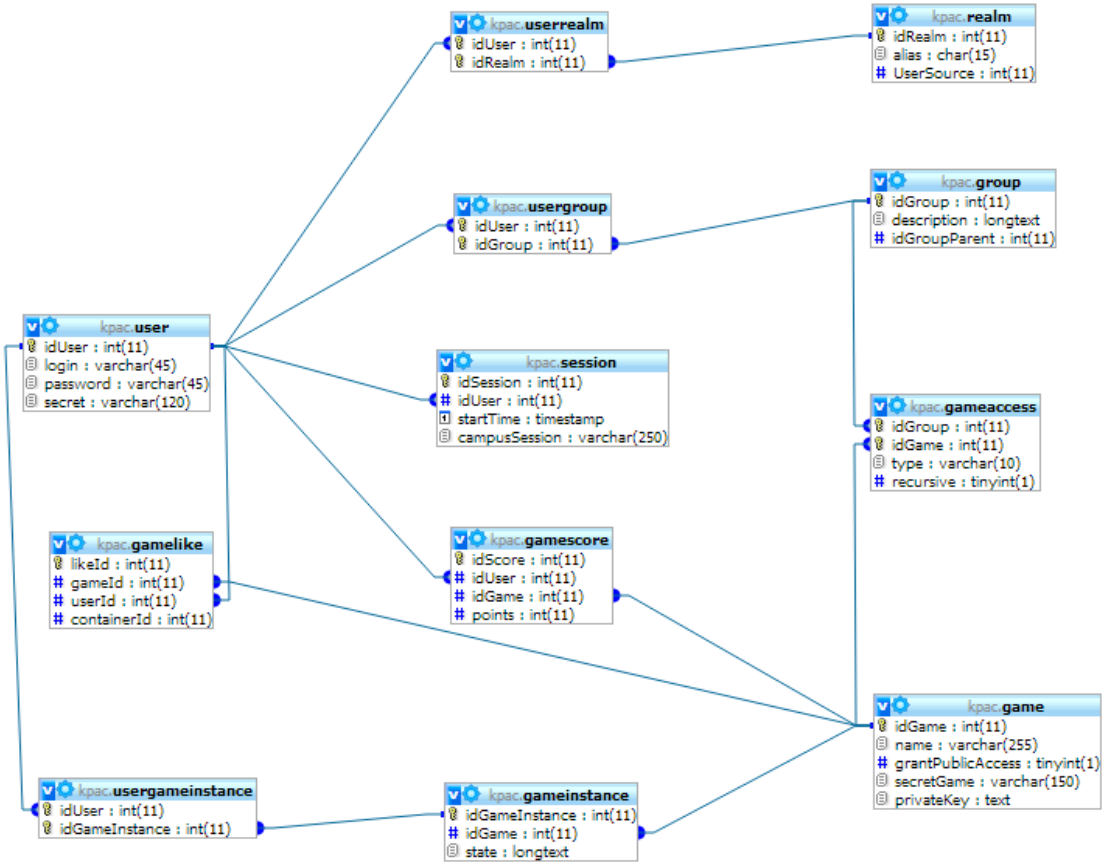


Figura 7. Esquema de la base de datos

Como se puede observar en el diagrama anterior, se añade la tabla userrealm y se modifica la tabla realm. El sql de creación de tablas se refleja en el **Anexo II**.

2.3. Elección de alternativas de componentes y licencias

Tabla 1. Principales componentes de la fase de desarrollo			
Componente	**Paquete**	**Versión prevista**	**Licencia**
Base de datos	MySql 5.0	5.0.7	GNU / GPL
Sistema operativo	Multiplataforma		
Servidor Web	Apache / Jboss	Apache 2.2.17 / 4.2.3.GA	Apache License Version 2.0 /LGPL
Interprete de scripts	PHP	5 3 5	PHP License
Implementación código	Eclipse	Indigo Service Release 2	Eclipse Public License
Desarrollo	Dev PHP	2 6 0	GPL 2.0

2.4. Especificaciones de desarrollo y pruebas

Las condiciones y características del entorno de desarrollo están descritas en el apartado anterior.

Las pruebas unitarias que se realizaran para asegurar el funcionamiento del sistema son las siguientes:

Validación de usuario mediante autenticación delegada
Es el inicio de una sesión en la aplicación que facilita el acceso al sistema kpax a usuarios que pueden no tener cuenta en el mismo pero sí en un sistema externo autorizado.
Secuencia:
1. El usuario accede a la pagina de login de la plataforma k-PAX y pulsa la opción Login with Twitter. 2.-El sistema delega la autenticación a la red social twitter. 3.-Twitter solicita el usuario y la contraseña. El usuario las facilita. 4.-El sistema externo comprueba que el usuario tiene cuenta y facilita los datos a kpax. 4.-El sistema comprueba que twitter está autorizado a autenticar la identidad del usuario. 5.-El sistema muestra la pantalla de correspondiente al usuario registrado.
RESULTADO ☐ válida ☐ no válida

Capítulo V. Desarrollo

1. Preparación del entorno de generación y desarrollo

Es necesaria la instalación de una plataforma local para llevar a cabo el desarrollo del proyecto que constará de los siguientes elementos:

- Registro en Github y creación de un nuevo fork.
- Entorno de programación Java.
- Infraestructura en XAMPP
- Bases de datos en MySql para kpax y elgg
- Código Elgg

En el **Anexo III** se detalla el proceso seguido para la instalación de la plataforma según las instrucciones de https://github.com/jsanchezramos/k-pax/wiki

2. Implementación

Una vez instalada la plataforma el siguiente paso consiste en lograr la autenticación en Elgg mediante la plataforma twitter. Para ello utilizaremos el api de twitter de los desarrolladores de elgg que se encuentra en elgg-1.8.14\mod\twitter_api. Este módulo deberá ser activado desde el panel de administración de elgg y necesita el plugin oauth_api para realizar la autenticación de los usuarios.

Estructura del plugin twitter_api para elgg.

Se encuentra en la carpeta **Elgg-1.8.14/mod/twitter_api.** A continuación se describen los archivos más relevantes.

- **manifest.xml:** Metadatos del plugin en que se citan las dependencias y la licencia.
- **Start.php**: Inicializaciones para elgg en que se incluyen los comportamientos, scripts
- **/actions/twitter_api/intersitial_settings.php**: Guarda la configuración del inicio de sesión la primera vez que se conecta a twitter.
- **/graphics/login_twitter.png**: Archivo gráfico del botón de conexión mediante twitter.
- **/lib/twitter_api.php:** Librerías de funciones utilizadas por los servicios de twitter.
- **/vendors/twitteroauth/**: Carpeta para ubicar los paquetes de terceros. En este caso,

la librería de PHP desarrollada por Abraham Williams trabajar con la API OAuth de Twitter cuyo proceso de autenticación se describe en el apartado siguiente.

- **Views:** En esta carpeta se organizan los archivos de estilo de otros plugins.

2.1.-Proceso de autenticación con twitter

Para interactuar con la API de twitter se utilizará la clase http://github.com/abraham/twitteroauth que ya está implementada y que se encargará de realizar la conexión con twitter. Se encuentra en la carpeta twitter_api/vendors.

Para autenticar con twitter llamaremos a la url https://api.twitter.com/oauth/authenticate y le pasaremos los parámetros necesarios para que kpax envie la petición a twitter. El proceso se describe a continuación teniendo en cuenta que kpax es el consumer y twitter el service provider.

1. kpax pide un **token** a twitter
2. kpax redirige al usuario a una página segura de twitter pasandole el **token** como parámetro
3. El usuario se autentica en la página de twitter validando el **token**
4. Twitter envía al usuario de vuelta a la página de kpax especificada en el parámetro **oauth_callback**.
5. kpax recoge al usuario en la **callback URL** junto con el **token** de confirmación de identidad.

Todo este proceso de comunicación se realiza mediante las librerías contenidas en elgg-1.8.14\mod\oauth_api\vendors\oauth\library.

- OAuthStore: Contenedor de almacenamiento para las credenciales oauth, el servidor y el lado del consumidor.
- OAuthServer: Capa de servidor en el controlador de OAuthRequest
- OAuthRequestVerifier: Verifica la solicitud actual. Comprueba si está firmado y si la firma es correcta. Cuando es correcta, también anota el nombre del usuario que se está haciendo esta solicitud.
- OAuthRequestSigner: Registra peticiones antes de realizar la solicitud.
- OAuthRequestLoger: Loguea solicitudes Oauth
- OAuthRequester: Realiza una petición OAuth firmado con un GET, POST, PUT o DELETE operación
- OAuthRequest: Solicita clase contenedora. Prepara una solicitud para el consumo de

las rutinas de OAuth.

- OAuthException: Envoltorio de excepción simple para OAuth.
- OAuthDiscovery: Maneja el descubrimiento de las entidades que se comunican mediante OAuth.

Por otra parte se utiliza el fichero **OAuth** para manejar los datos entre el plugin **twitter_api** y la librería **oauth**. En este fichero se definen las CLASES OAuthToken, twitterOAuthRequest y funciones necesarias para intercambiar información entre OAuth y Twitter_api.

Antes de poder realizar el primer paso (Request token) hay que registrarse como kpax en twitter . Conectamos con https://dev.twitter.com/apps/new la cuenta de twitter.com que queremos que sea la propietaria de la aplicación. Encontramos este formulario que rellenamos.

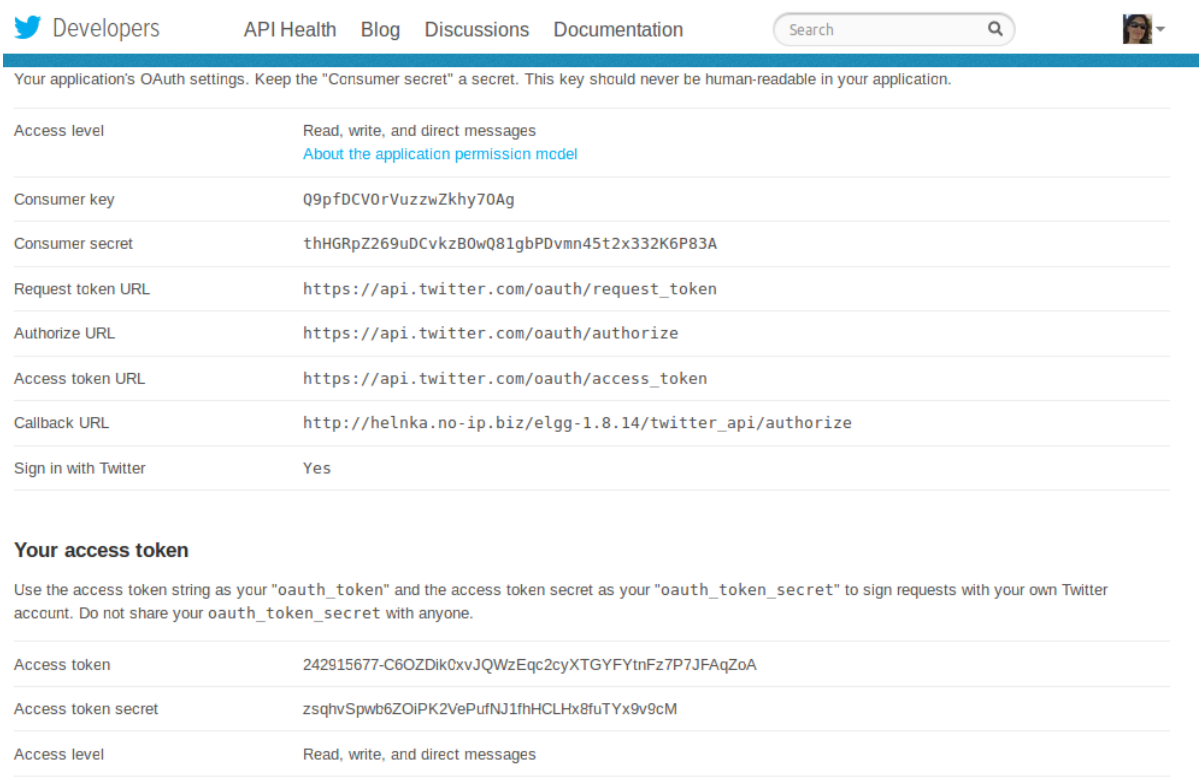


Figura 8. Parámetros de la aplicación

Tras registrar la aplicación, Twitter nos proporciona los datos necesarios para crear nuestra firma digital y las URLs donde se encuentran los servicios.

También debemos tener en cuenta de que la opción "Allow this application to be used to Sign in with Twitter? está activada y que el nivel de acceso es, como mínimo, de lectura y escritura.

En la Callback URL hemos puesto la dirección de http://helnka.no-ip.biz/elgg-1.8.14/autenthicate para lo cual hemos necesitado una IP Pública obtenida en

en http://www.noip.com.Se ha tenido que configurar el router para asignar el puerto 80 y que el pc se viese desde fuera y en el archivo windows\system32\drivers\etc\hosts se ha cambiado el host que siempre resolviese mi dirección localmente.

Una vez obtenidos los parámetros de configuración copiaremos el "Consumer key" y el "Consumer secret" en la configuración del plugin. Para ello iremos a la página de Administración -> Administración de Herramientas y localizaremos el plugin "twitterservice" donde copiaremos ambas claves.

En el fichero OauthDiscovery.php pondremos los parámetros proporcionados por twitter para validarnos.

Por último en el fichero http://localhost/elgg-1.8.14/services/api/rest/xml/?method=system.api.list podemos comprobar que tenemos el método auth.gettoken que nos sirve para obtener un token de autenticación.

Para validarnos en elgg/kpax con twitter creamos un gráfico que hará las funciones de botón de validación y lo guardamos en elgg-1.8.14\mod\twitter_api\graphics y cuya definición está en elgg-1.8.14\mod\twitter_api\views\default\twitter_api\login.php.

A continuación pasamos a modificar el archivo twitter_api/lib/twitter_api.php. Nos fijamos en la función **twitter_api_get_authorize_url** que se encarga de obtener la url para autorizar al usuario creando una instancia de Twitteroauth pasando como parámetros la consumer key y la consumer secret. Obtendremos el oauth_token y el oauth_secret que guardaremos en las variables de sesión y que servirán para redirigir al usuario hacia Twitter para su autorización. La función **twitter_api_get_access_token** devuelve el token de acceso para utilizar en llamadas de Twitter.

La pantalla de acceso a kPax tiene el siguiente aspecto.

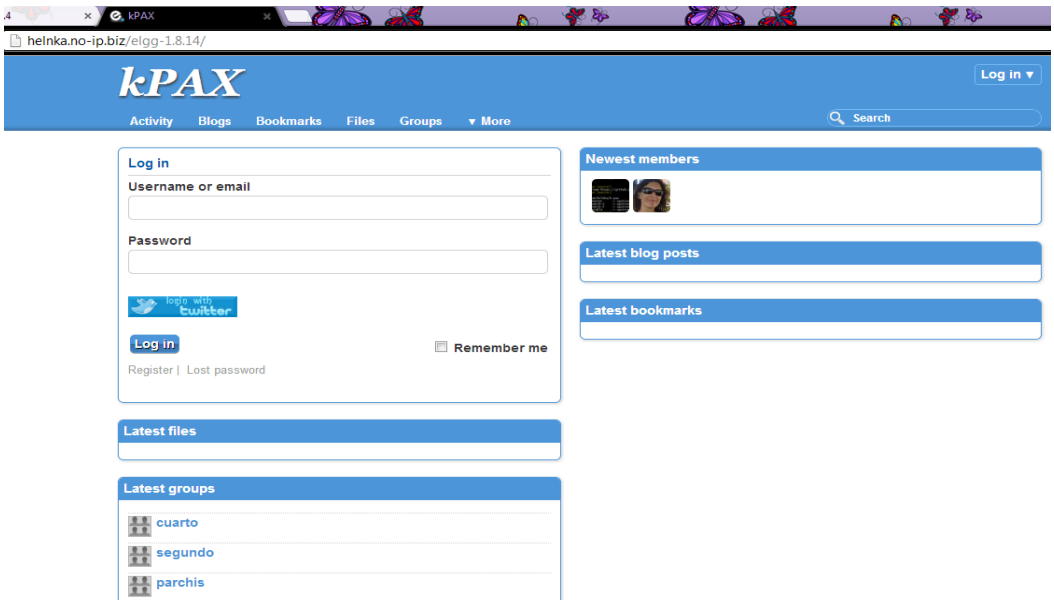


Figura 9. Pantalla de acceso a kPax

En el siguiente paso el usuario se redirige a la página de autenticación de Twitter, pasando el token como parámetro con la siguiente URL.

Figura 10. Pantalla de autenticación de twitter

Después de validar el token, Twitter redirige al usuario a la url que establecimos en la página de configuración: http://helnka.no-ip.biz/elgg-1.8.14/. El token que devuelve Twitter es el que tenemos almacenado en nuestra sesión de usuario, por lo que comparando ambos valores se permite el acceso a la aplicación.

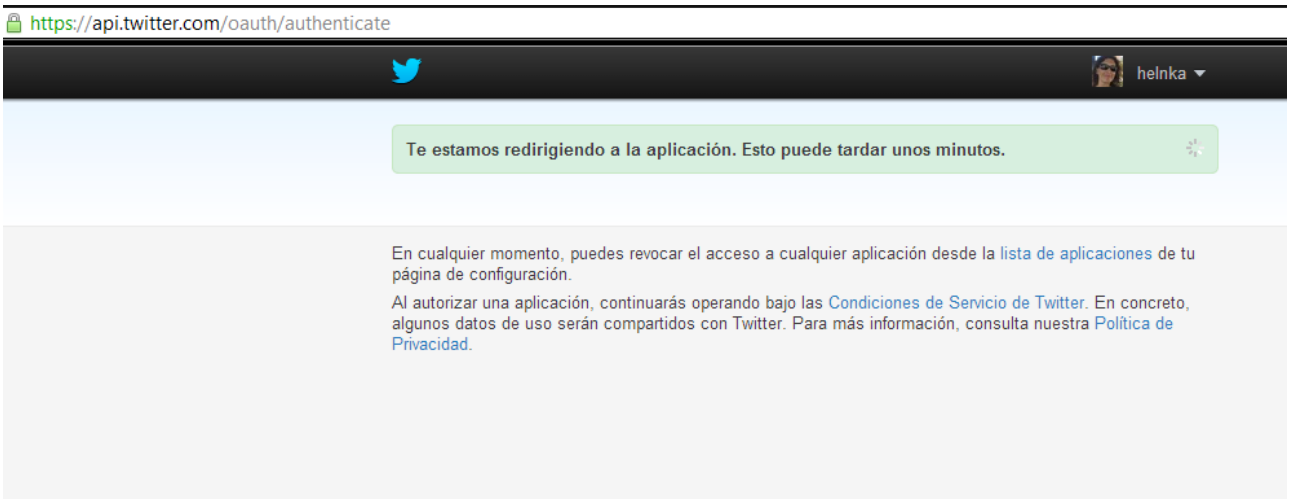


Figura 11. Pantalla de redirección a kPax

Por último se permite el acceso a la plataforma.

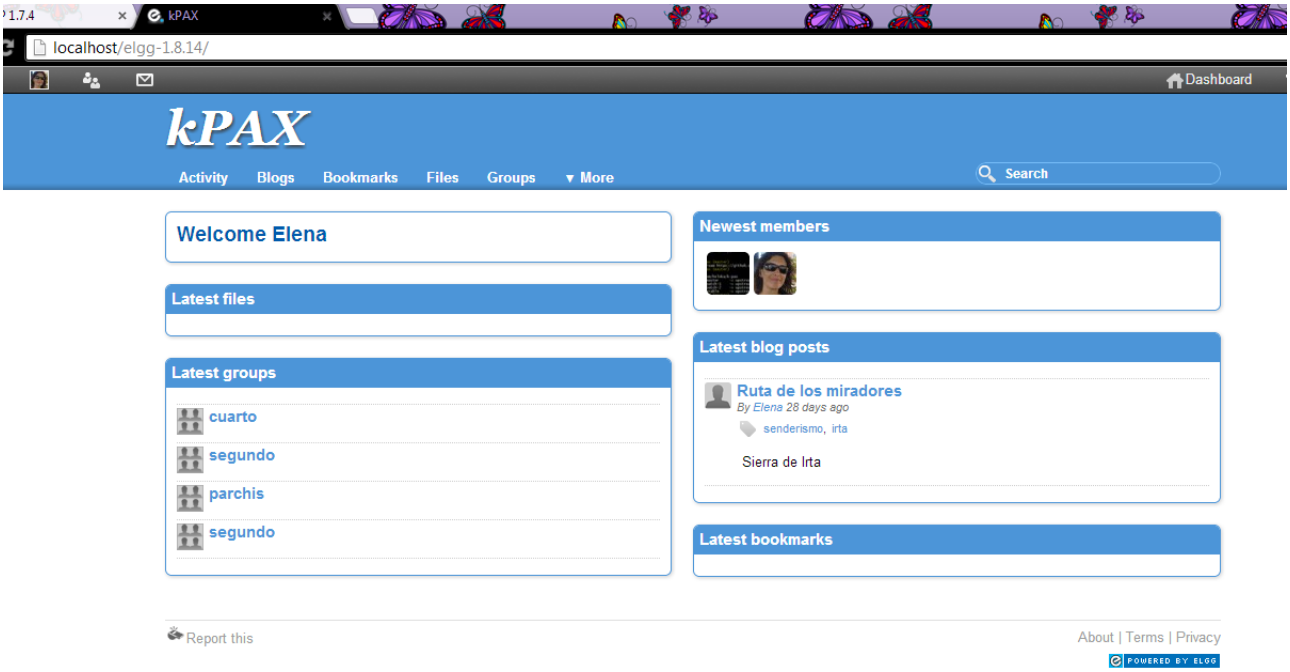


Figura 13. Usuario twitter validado en la plataforma

2.2. Funciones añadidas o modificadas

Llegados a este punto se ha logrado el objetivo de validar usuarios en la plataforma kPax con usuarios twitter, el siguiente paso será conseguir que se almacenen en la base de datos kpac. Para ello, en primer lugar se deben crear o modificar varias clases de la parte de los servicios java. A continuación mostramos los diferentes ficheros pertenecientes a cada una de las capas.

Capa de persistencia:

1.- En el paquete **uoc.edu.svrKpax.dao** se han definido las siguientes interfaces:

a.- **UserDao y UserDaoImpl**

b.- **UserRealmDao,** con las funciones siguientes y su implementación **UserRealmDaoImpl**

- addUserRealm: almacena la información de un UserRealm
- getAllUserRealm: devuelve el listado de tags del juego con id userrealm
- getUserRealm: devuelve la información del UserRealm con id UserId
- getUserForUid: devuelve la informacion del UserRealm con id uid.

c- Se ha modificado **RealmDao y RealmDaoImpl** añadiendo:

- getRealmByUserSource: devuelve la información del realm con usersource user.
- addUserRealm(UserRealm objUserRealm)
- List<UserRealm> getAllUserRealm()

2.- En el paquete **uoc.edu.svrKpax.vo** se han definido las siguientes clases:

a.-Clase **UserRealm:** Contiene los campos idUser que contiene el identificador de usuario y userRealm que contiene el identificador de realm. Se han definido las siguientes funciones:

- getIdUser y setIdUser:
- getIdRealm y setIdRealm
- GetUserSource:

b.- Clase **Realm:** Se ha añadido el campo UserSource que contiene la autorización para crear nuevos usuarios para ese Realm.

Capa lógica:

1.- En el paquete **uoc.edu.svrKpax.bussines** se ha definido:

a.- La interfaz **UserRealmBO** con las funciones siguientes y su implementación **UserRealmBOImpl**.

- AddUserRealm: Añade un registro a la tabla UserRealm y crea el usuario en la tabla User si no existe el identificador de usuario y si el campo Realm.UserSource=1.
- getUserRealm: Devuelve el valor de realm en UserRealm.
- ListRealms: Devuelve el listado de los realms.
- GetRealm: Devuelve un registro por identificador de realm.
- GetRealmByIden: Devuelve un registro por alias.
- GetByUserSource: Devuelve un registro por UserSource.

b.- La interfaz **UserBO** con las funciones siguientes y su implementación **UserBOImpl**.

- AddUser: Añade un usuario a la tabla User

- GetUser: Obtiene un usuario de la tabla User

2.-**Clase Jsonp**: Para que elgg pueda realizar las llamadas por medio de ajax, se han modificado las funciones descritas para la clase User anterior. Todas las funciones necesitan el parámetro callback para funcionar bajo Json. Tambien se han añadido variables para hacer llamadas a las funciones de la capa bussines desde las implementaciones de las funciones anteriores.

Ficheros de configuración

1. En el fichero **ApplicationContent** se inicializan las variables de las clases declaradas de los siguientes paquetes:

   - uoc.edu.srvKpax.bussines
   - uoc.edu.srvKpax.dao

2. En el fichero **Hibernate.cfg.xml** se almacenan los datos relativos al acceso a la base de datos.

2.3.-Estructura del plugin kPax para elgg

En este plugin, ubicado en elggdocs/mods se lleva a cabo la ampliación de la funcionalidad de kPax que viene dada en los modulos Elgg. Estos módulos están formados 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 servicios web 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 describen los archivos más relevantes contenidos en este módulo.

- **Start.php**: Se encuentra en la raíz del módulo y se encarga de realizar funciones como registrar acciones, controladores de página, controladores de entidades url, menús o librerías. Contiene una serie de funciones elgg del siguiente tipo:
  - elgg_register_event_handler: Registra la callback como un controlador de eventos Elgg.
  - elgg_register_action: Registra una acción.
  - elgg_register_page_handler: Establece un controlador de página para un identificador.

- elgg_register_entity_url_handler: Establece el controlador de URL para una entidad.
- elgg_register_entity_type: Registra un tipo de entidad como una entidad de cara al público que debe mostrarse en la búsqueda.
- elgg_register_menu_item: Registra un item para un menu elgg.
- elgg_register_library: Registra la librería.
- elgg_load_librarym. Carga la librería.

- Las **ACCIONES** (elgg_register_action) se guardan en la carpeta mods\kpax\actions\kpax y hay dos:
  - save.php: Construye un objeto kpax. Carga las peticiones en un formulario en la cache.
  - delete.php: Elimina un objeto kpax.
- **/lib/kpax.php:** En este archivo está la función kpax_prepare_form_vars que se encarga de preparar las variables para añadir o editar en el formulario.
- **/lib/kpaxOauth.php:** En este archivo se define la clase kpaxOauth que construye el objeto kpaxOauth. Está formado por las siguientes funciones:
  - setKeySecret, getSignature y test que construye un objeto **OAuthConsumer.** y obtiene los datos con **OAuthRequest**
- **/lib/Oauth.php:** Integra la clase **kpax** en las peticiones de servicios **oauth y** crea las clases OAuthConsumer, OAuthToken, OAuthSignatureMethod, OAuthRequest, OAuthServer, OAuthDataStore, OAuthUtil.
- **/lib/kpaxSrv.php:** Declara la clase kpaxSrv y deben declararse todas las funciones utilizadas en las clases VO y REST, en este caso las relacionadas con la creación de un usuario y su almacenamiento. Las llamadas a la URL se hacen via GET o POST mediante una URL en la que se envían determinados parámetros. En este archivo se han añadido las siguientes funciones:

```
public function addUser($campusSession, $password, $login, $idUser) {

        $body = 'secretSession=' . $campusSession . '&password' . $password . '&login=' . $login . '&idUser=' . $idUser;

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

public function getUser($idUser, $campusSession) {

        return json_decode($this->service("user/" . $campusSession . "/get/" . $idUser));

 public function addUserRealm($campusSession, $idUser, $idRealm) {
```

```
        $body = 'secretSession=' . $campusSession . '&idUser' . $idUser . '&idRealm=' .
$idRealm;

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

3. Trabajo realizado por la estudiante

El trabajo realizado ha consistido, en primer lugar, en hacer una aproximación al entorno de trabajo con la plataforma elgg, completamente nuevo para la estudiante. En segundo lugar se ha instalado la plataforma completa descrita en el Anexo II. Simultáneamente se ha hecho un análisis del sistema para obtener una visión global del mismo así como un diseño del sistema. Cuando se ha instalado la plataforma se ha procedido a la instalación y desarrollo del plugin de twitter y a la adición y modificación de funcionalidades en los ficheros del entorno java. Finalmente se han modificado los ficheros de configuración y de llamadas a los servicios web.

Principales problemas y soluciones aportadas

A destacar la complejidad en la instalación de la plataforma. El manual de instalación es muy esquemático y ha sido necesaria una búsqueda exhaustiva de información así como sucesivas consultas al profesor de prácticas externas para solucionar los problemas que han ido surgiendo.

1. El primer inconveniente fue la necesidad de utilizar Windows XP ya que el manual ha sido elaborado para este sistema operativo. En un primer momento intenté instalarlo en mi distribución linux pero en vista de los problemas que iban surgiendo y la escasez de información sobre esta plataforma preferí instalar Windows XP en una nueva partición de mi disco duro.

2. Una vez añadidos los módulos en la carpeta correspondiente deben activarse. Para ello hay que acceder a la administración de la plataforma elgg y activar los diferentes plugins, pero era imposible activar el plugin **Twitter API 1.8 .** Se solucionó descomentando la línea php_curl del fichero php.ini.

3. Cuando se registra la aplicación en twitter, hay que proporcionar una callback url a la que volver tras la autenticación. Dado que nuestro servicio estaba en un servidor local y twitter no podía acceder ha sido necesario, tras una sugerencia del consultor de la asignatura, lograr una IP pública obtenida en http://www.noip.com. Se ha tenido que configurar el router para asignar el puerto 80 y que el pc se viese desde fuera y en el archivo windows\system32\drivers\etc\hosts se ha cambiado el host para que resolviese la dirección localmente.

4. Se registraba un error que ha sido solucionado incluyendo en la base de datos de kpax un usuario con privilegios de administrador.

5. En el fichero kpaxOauth.php de la carpeta elggdocs se han cambiado las siguientes variables que apuntan al servidor local:

    - URL = 'http://localhost/elgg-1.8.14/';
    - API_URL = 'http://localhost:8080/webapps/srvKpax'

Capítulo VI. Conclusiones

Para finalizar este trabajo a continuación se exponen los objetivos conseguidos y los no conseguidos, se estudian posibles ampliaciones al proyecto inicial y se detallan las decisiones tomadas que ahora cambiaría la estudiantes. Por último se ofrece una perspectiva personal de la experiencia de la autora con el software libre.

1. Objetivos conseguidos y no conseguidos.

El proyecto se ha desarrollado conforme a la planificación previamente establecida, sin apenas modificación de plazos.

Objetivos conseguidos

- Instalación local de la plataforma
- Integración del plugin de twitter para realizar la autenticación delegada.
- Implementación y modificación de las funciones de los servicios web en Java.
- Modificación de los ficheros de configuración.
- Modificación de los ficheros kpax.php y kpaxSrv.php

Objetivos no conseguidos

- El nuevo usuario procedente de la red social twitter se almacena en la base de datos elgg pero no he logrado que se guarde en la base de datos kpax.

2. Posibilidades de ampliación.

Ya que no se ha logrado el objetivo de insertar el nuevo usuario en la tabla kpax, una posibilidad sería la finalización de la implementación del almacenamiento del nuevo usuario. Aunque, por otra parte, también se podría replantear la estructura de la plataforma de forma que se almacenaran los datos en la propia base de datos elgg.

Desarrollar una línea de trabajo consistente en la creación de documentación exhaustiva acerca de la arquitectura de la plataforma.

3. Decisiones tomadas que ahora cambiaría la estudiante.

A pesar de no haber obtenido los resultados esperados, me ha resultado gratificante la participación en este proyecto aunque me ha supuesto bastante esfuerzo la comprensión de la estructura de la aplicación. Tal vez cambiaría la elección del tipo de proyecto, si tuviera que volver a escoger seguramente me inclinaría por un proyecto menos técnico y más teórico.

4. Experiencia personal con el software libre.

Mi experiencia personal con el software libre es muy positiva. De hecho, la decisión de cursar este máster de software libre estuvo motivada por el deseo de profundizar en su conocimiento, ya que profesionalmente venía utilizando diferentes distribuciones de linux.

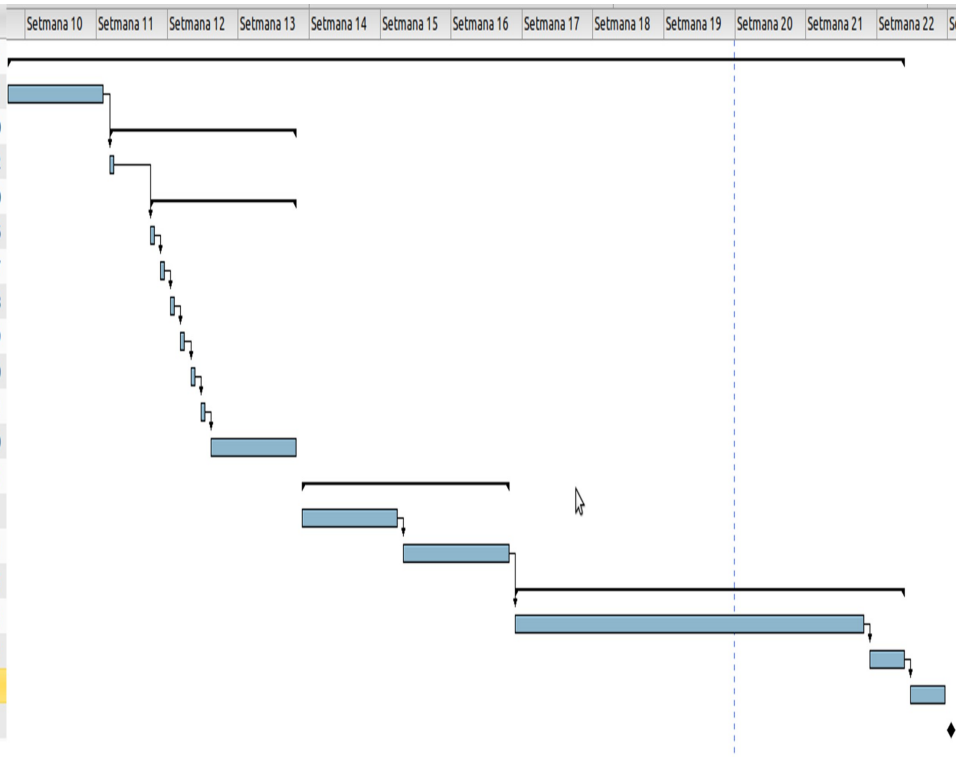
Constato que la evolución de la calidad software libre ha ido aumentando paulatinamente de forma que, muchos productos ofrecen iguales prestaciones, sino superiores, a las del software propietario. De hecho, creo que la mayor utilización del software propietario viene dada, en gran medida, por el desconocimiento que de este tipo de productos tiene gran parte del público mayoritario.

Bibliografía

- Elgg.org, "*Reference Manual*." [Online]. http://docs.elgg.org/wiki/Web_Services
- Twitter developers, "*Documentation"* [Online]. https://dev.twitter.com/docs
- Github, "*Reference Manual".* [Online]. http://git-scm.com/documentation
- Oauth, "*Documentation".* [Online] http://oauth.net/documentation/
- Higueras, N; Vidal, R "*Redacción de textos científico-técnicos*" UOC
- Otero, A. "*Proyecto Web*" UOC
- Beneito, R. "*Presentación de documentos y elaboración de presentaciones".* UOC

Anexo I. Planificación del proyecto

WBS	Nom	Comença	Acaba
1	**PFM. Creación de un plugin para kPAX**	**mar 2**	**mai 29**
1.1	Realización del estudio de viabilidad del proyecto	mar 2	mar 11
1.2	**Instalación y configuración de la plataforma**	**mar 12**	**mar 30**
1.2.1	Registro y configuración en Github	mar 12	mar 12
1.2.2	**Instalación del entorno de programación JAVA**	**mar 16**	**mar 30**
1.2.2.1	Instalación de java 6 y jdk	mar 16	mar 16
1.2.2.2	Instalación de eclipse	mar 17	mar 17
1.2.2.3	Instalación de maven 3.0.5	mar 18	mar 18
1.2.2.4	Instalación de jboss 4.3.2	mar 19	mar 19
1.2.2.5	Instalación de XAMP	mar 20	mar 20
1.2.2.6	Instalación de MySQL	mar 21	mar 21
1.2.2.7	Instalación y configuración del código Elgg	mar 22	mar 30
1.3	**Diseño del plugin de kPax**	**mar 31**	**abr 20**
1.3.1	Diseño conceptual del plugin	mar 31	abr 9
1.3.2	Diseño lógico	abr 10	abr 20
1.4	**Construcción y pruebas del plugin**	**abr 21**	**mai 29**
1.4.1	Construcción del plugin	abr 21	mai 25
1.4.2	Pruebas	mai 26	mai 29
2	Revisión del código y de la documentación	mai 30	jun 2
3	Entrega trabajo final prácticas externas	jun 3	jun 3



Anexo II. Estructura de las tablas

En este apartado se incluirán únicamente las tablas añadidas o modificadas de la estructura original.

Estructura de la tabla `Realm`

```
CREATE TABLE `Realm` (
  `idRealm` int(11) NOT NULL AUTO_INCREMENT,
  `alias` char(15) NOT NULL,
`UserSource` int(11) NOT NULL,
  PRIMARY KEY (`idRealm`),
  UNIQUE KEY `alias_UNIQUE` (`alias`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=5 ;
```

Estructura de la tabla `UserRealm`

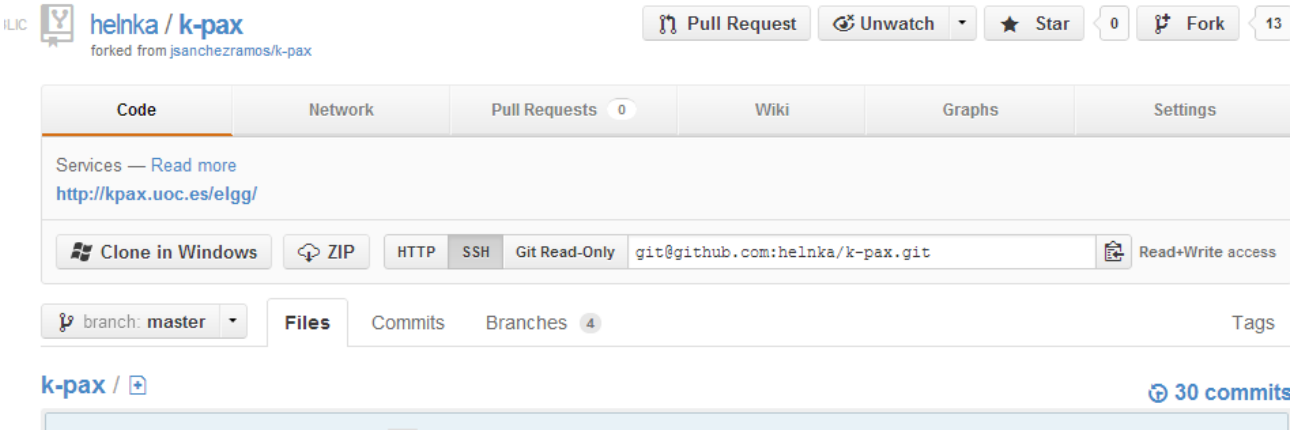
```
CREATE TABLE `UserRealm` (
  `idUser` int(11) NOT NULL,
  `idRealm` int(11) NOT NULL,
  PRIMARY KEY (`idUser`,`idRealm`),
  KEY `UserRealm_User` (`idUser`),
  KEY `UserRealm_Realm` (`idRealm`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Anexo III – Instalación de la plataforma

A.- Registro en Github y creación de un nuevo fork.

1. Alta en Github (https://github.com)

2. Creación de una nueva rama en el repositorio Kpax (https://github.com/jsanchezramos/k-pax)



3. Instalación de github en local siguiendo las instrucciones de https://help.github.com/articles/set-up-git

4. Generación de las claves SSH para establecer una conexión segura entre nuestro equipo y GitHub. Los siguientes pasos generaran la clave SSH y añadiran la clave pública a la cuenta GitHub.

   - Comprobación de la existencia de claves ssh en el ordenador. Abrimos Git Bash y ejecutamos:

```
elena@SOBREMESA ~
$ cd ~/.ssh
```

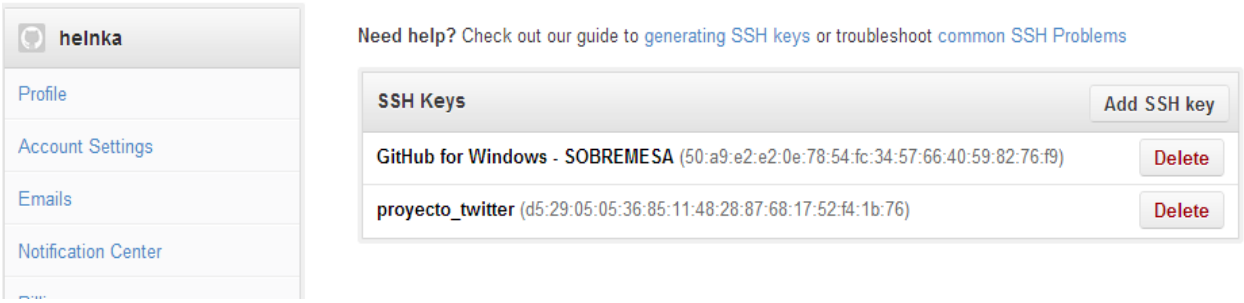
   - Creación de una copia de seguridad de las claves existentes.

```
$ ls
github_rsa  github_rsa.pub
elena@SOBREMESA ~/.ssh
$ mkdir key_backup
elena@SOBREMESA ~/.ssh
$ cp github_rsa key_backup
elena@SOBREMESA ~/.ssh
$ rm github_rsa
```

   - Generación de una nueva clave SSH.

```
elena@SOBREMESA ~/.ssh
$ ssh-keygen -t rsa -C "elenasanchis@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/elena/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/elena/.ssh/id_rsa.
Your public key has been saved in /c/Users/elena/.ssh/id_rsa.pub.
The key fingerprint is:
d5:29:05:05:36:85:11:48:28:87:68:17:52:f4:1b:76 elenasanchis@gmail.com
elena@SOBREMESA ~/.ssh
$ clip < ~/.ssh/id_rsa.pub
elena@SOBREMESA ~/.ssh
$ clip < ~/.ssh/id_rsa.pub
elena@SOBREMESA ~/.ssh
$
```

- Adición de la clave SSH a GitHub.



- Conexión a GitHub para comprobar la correcta instalación de la clave.

```
elena@SOBREMESA ~/.ssh
$ ssh -T git@github.com
The authenticity of host 'github.com (207.97.227.239)' can't be established.
RSA key fingerprint is 16:27:ac:a5:76:28:2d:36:63:1b:56:4d:eb:df:a6:48.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,207.97.227.239' (RSA) to the list of know
n hosts.
Enter passphrase for key '/c/Users/elena/.ssh/id_rsa':
Hi helnka! You've successfully authenticated, but GitHub does not provide shell
access.
```

- Clonado del proyecto en nuestro repositorio local

```
$ git clone https://github.com/helnka/k-pax.git
Cloning into 'k-pax'...
remote: Counting objects: 356, done.
remote: Compressing objects: 100% (252/252), done.
Receiving objects: 100% (356/356), 17.40 MiB | 1.36 MiB/s, done.
emote: Total 356 (delta 117), reused 309 (delta 70)R
Resolving deltas: 100% (117/117), done.
elena@SOBREMESA ~
```

- Configuración del repositorio remoto para hacer un seguimiento del repositorio original.

```
elena@SOBREMESA ~
$ cd k-pax
elena@SOBREMESA ~/k-pax (master)
$ git remote add upstream https://github.com/helnka/k-pax.git
elena@SOBREMESA ~/k-pax (master)
$ git fetch upstream
From https://github.com/helnka/k-pax
 * [new branch]      master     -> upstream/master
 * [new branch]      patch-1    -> upstream/patch-1
 * [new branch]      patch-2    -> upstream/patch-2
 * [new branch]      stable     -> upstream/stable
```

B.- Instalación del entorno de programación JAVA

1. Descarga de java 6 y jdk http://www.oracle.com/technetwork/java/javase/downloads/jdk-6u31-download-1501634.html). Para instalar seguimos las instrucciones de http://www.oracle.com/technetwork/java/javase/documentation/install-windows-152927.html

2. Descarga de eclipse desde la web oficial (http://www.eclipse.org/). La instalación no presenta

ninguna peculiaridad.

3. Instalación de maven 3.0.5 de la web oficial (http://maven.apache.org/) para cuya configuración debemos añadir las siguientes variables:

    - Adición de la variable de entorno M2_HOME y de la variable de entorno M2 en las variables de usuario con el valor M2_HOME%% \ bin.
    - Adición de la variable de entorno MAVEN_OPTS para especificar las propiedades de JVM, por ejemplo, el valor Xms256m-Xmx512m.
    - Creación de la variable de entorno Path en las variables de usuario y anteposición del valor %% M2 para añadir Maven-
    - Creación de JAVA_HOME en las variables de usuario o en las variables del sistema y establecimiento de la ubicación del JDK C: \ Archivos de programa \ Java \ jdk1.5.0_02 . Comprobación de que % JAVA_HOME% \ bin está en su variable de entorno Path.
    - Ejecución de mvn - version para comprobar que está correctamente instalado.

```
C:\Users\elena>mvn --version
Apache Maven 3.0.5 (r01de14724cdef164cd33c7c8c2fe155faf9602da; 2013-02-19 14:51:
28+0100)
Maven home: C:\Program Files (x86)\Apache Software Foundation\apache-maven-3.0.5

Java version: 1.6.0_43, vendor: Sun Microsystems Inc.
Java home: C:\Program Files (x86)\Java\jdk1.6.0_43\jre
Default locale: es_ES, platform encoding: Cp1252
OS name: "windows 7", version: "6.1", arch: "x86", family: "windows"
```

4. Instalación de jboss desde la web oficial (http://www.jboss.org/)

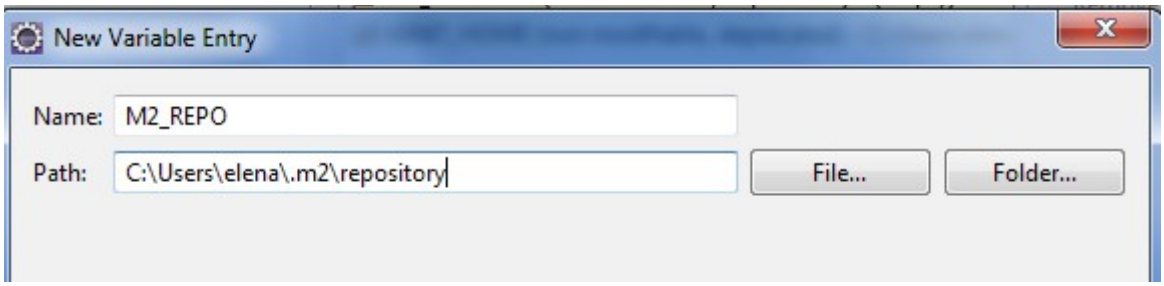
    - Comprobación del funcionamiento del servidor

5. Nos situamos en el directorio del proyecto C:/Users/elena/k-pax y ejecutamos el comando "mvn install". Cuando comprobamos que se ha instalado correctamente, ejecutamos “mvn

eclipse:eclipse" para hacer que eclipse detecte correctamente todas las variables.

```
      Additional settings will be preserved, run mvn eclipse:clean if you wan
old settings to be removed.
[INFO] Wrote Eclipse project for "svrKpax" to C:\Users\elena\k-pax.
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 30.570s
[INFO] Finished at: Tue Mar 19 15:01:54 CET 2013
[INFO] Final Memory: 6M/16M
[INFO] ------------------------------------------------------------------------
C:\Users\elena\k-pax>
```

6. En Eclipse creamos la variable M2_REPO en Project properties --> Java build path --> Add variable --> Create a new variable with name M2_REPO with the repository route



7. Modificación de \Users\juanfrasr\servidor\jboss-4.2.3-1.GA\server\default\deploy indicando la ruta donde está nuestro servidor local. El fichero a modificar es pom.xml.

C:\workspace\jboss-4.2.3.GA\server\default\deploy

8. Ejecutamos "mvn -Denv=local clean package". Este paso se repetirá cada vez que queramos compilar.

```
1/ant-launcher-1.7.1.jar (12 KB at 57.6 KB/sec)
Downloaded: http://repo.maven.apache.org/maven2/org/apache/ant/ant/1.7.1/ant-1.7
.1.jar (1292 KB at 742.5 KB/sec)
[INFO] Executing tasks
     [copy] Copying 1 file to C:\workspace\jboss-4.2.3.GA\server\default\deploy
[INFO] Executed tasks
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 9.880s
[INFO] Finished at: Tue Mar 19 15:28:27 CET 2013
[INFO] Final Memory: 12M/29M
[INFO] ------------------------------------------------------------------------
C:\Users\elena\k-pax>
```

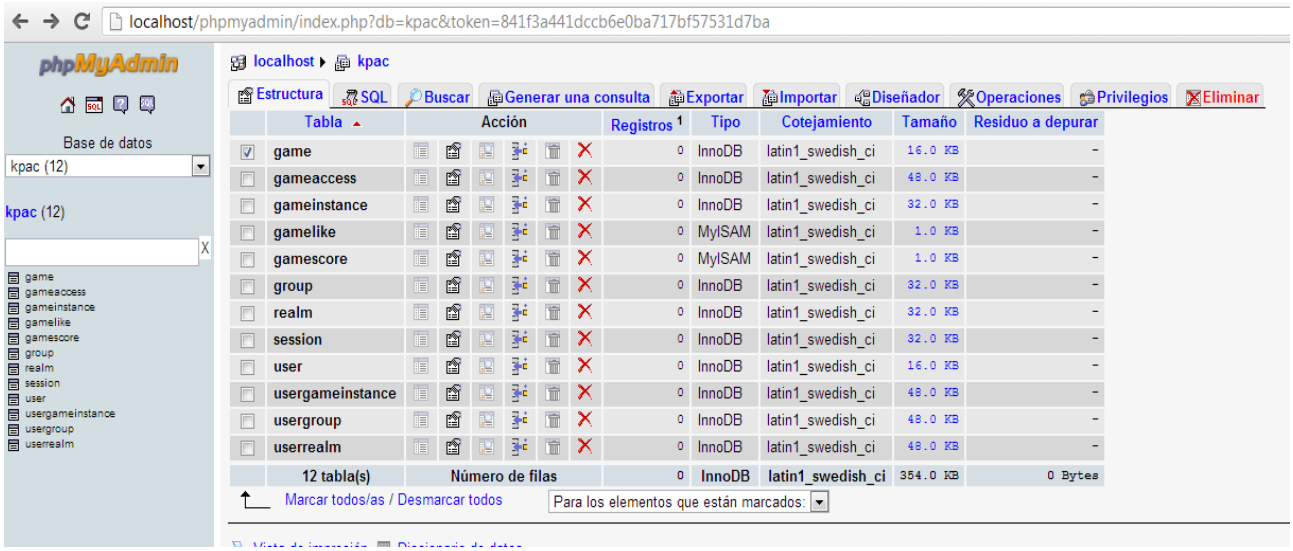
9. Añadimos mysql connector/j library a Jboss y copiamos el fichero comprimido jar (i.e.: mysql-connector-java-5.1.19-bin.jar) al directorio$JBOSS_HOME/server/default/lib

C.- Instalación de XAMP.

1. Descarga de la web oficial http://sourceforge.net/projects/xampp/files/XAMPP%20Windows/1.8.0/xampp-win32-1.8.0-VC9-installer.exe/download . La instalación es automática.

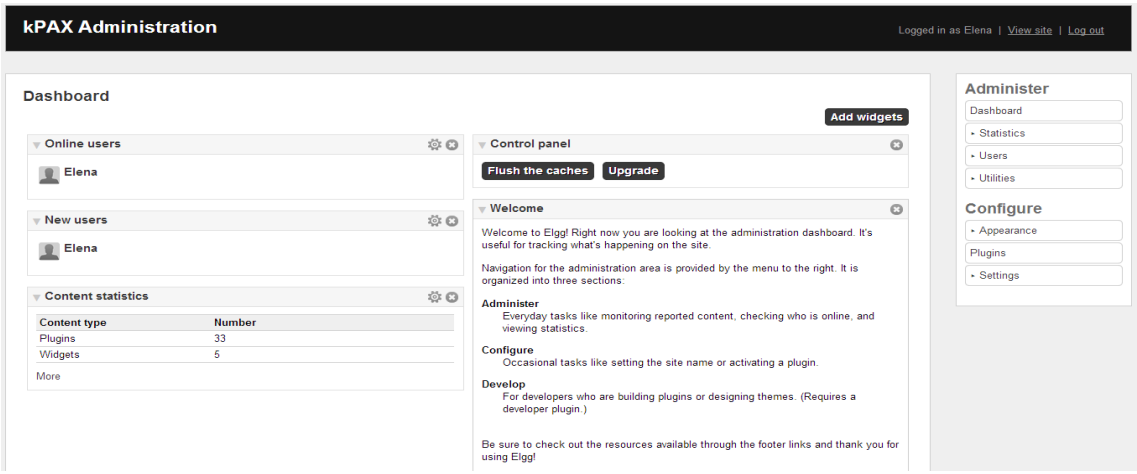
D. Bases de datos MySQL para elgg y kPAX

1. Creación de una base de datos KPAX y un usuario con todos los privilegios
2. Utilización de la secuencia de comandos de la carpeta doc / sql en el proyecto k-pax para crear una base de datos vacía.
3. Una vez que la base de datos ha sido creada, edición del archivo srvKpax-ds.xml e inclusión de la configuración de base de datos (host, puerto, usuario, contraseña). Guardar este archivo en / home / server / jboss / server / default / deploy / *



E.- Código de elgg

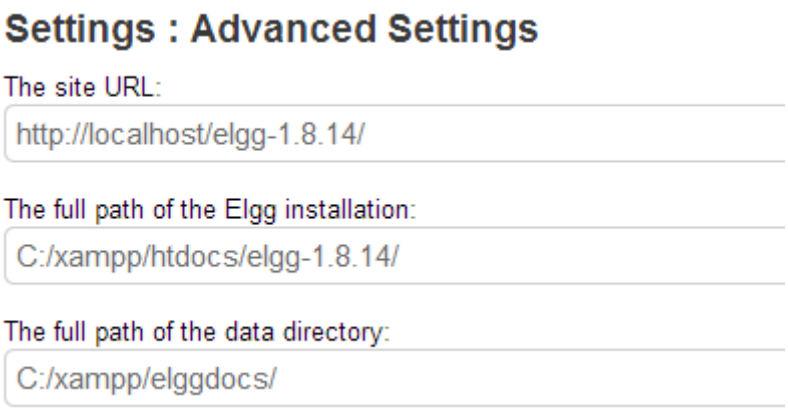
1. Descargar elgg código (versión 1.8.x) desde la página de descarga Elgg.
2. Guardar el contenido de la carpeta www del servidor wamp / xammp.
3. Iniciar el servidor wamp / xammp y todos los servicios.
4. Crear una base de datos vacía para elgg en http://localhost/phpmyadmin).
5. Activar el "rewrite_module" (Apache) y el "php_openssl" (PHP) en WampServer.
6. Instalar en una carpeta temporal. Por otra parte, crear el "elgg" base de datos y un usuario con todos los privilegios.
7. Instalar elgg de http://localhost/elgg-1.8.X/install.php . En la sección "Requisitos de verificación" paso, comprobar que el archivo httpd.conf tiene el conjunto AllowOverrride .
8. Por motivos de seguridad la instalación de elgg se crea en una carpeta de datos fuera del directorio htdocs. C:\xampp\elggdocs



Instalación

9. Descargar los módulos git checkout. Tienen la estructura de ficheros /elgg/etc... es decir que deben ser copiados en la carpeta mod de la plataforma elgg, copiar todo el contenido a la carpeta mod de elgg

10. Una vez puesto el contenido en la carpeta mod, se deben activar. Para ello hay que acceder a la administración de la plataforma elgg y activar los diferentes plugins. Para activar el pluggin **Twitter API 1.8** que se tiene que descomentar la línea php_curl del fichero php.ini



11. Activar los módulos apiadmin 1.8b1, kpax, login required y likeskpax en elgg Administration. En primer lugar descargamos y copiamos la carpeta apiadmin en C:\xampp\htdocs\elgg-1.8.14\mod

12. y generamos el par de claves

List of existing Keys

No	API Key
1	e23df504329dbf8d8644207eb8600253dcf2ab53

katalina21 Revoke key
Public: e23df504329dbf8d8644207eb8600253dcf2ab53
Private: e7d0b0e33e17632f60aa7f85127816953da1aa98

13. Compilamos el código Java de nuevo ejecutando "mvn -Denv=local clean package".