Nuevas funcionalidades para la red social Kpax

Trabajo Final de Master

Eugenia Carrera Formoso

Motivación de Kpax

- La red social Kpax ha sido desarrollada e implementada sobre la plataforma Open Source Elgg por un grupo de la escuela de Estudios de Informática, Multimedia y Telecomunicaciones de la propia UOC.
- La motivación del proyecto es conseguir una plataforma de aprendizaje mediante juegos serios, pensados para entrenar ciertas capacidades desde un punto de vista práctico.

Requerimientos de Kpax

Kpax implementará estas funcionalidades:

- Conexión on-line y off-line desde una variedad de dispositivos.
- Aprendizaje y entrenamiento en distintas áreas.
- Facilidad para que tanto la comunidad UOC como desarrolladores externos publiquen sus juegos.
- Flexibilidad para constituir comunidades participativas tanto de aprendizaje como de desarrollo de juegos orientados a la enseñanza.

Kpax se construye sobre Elgg

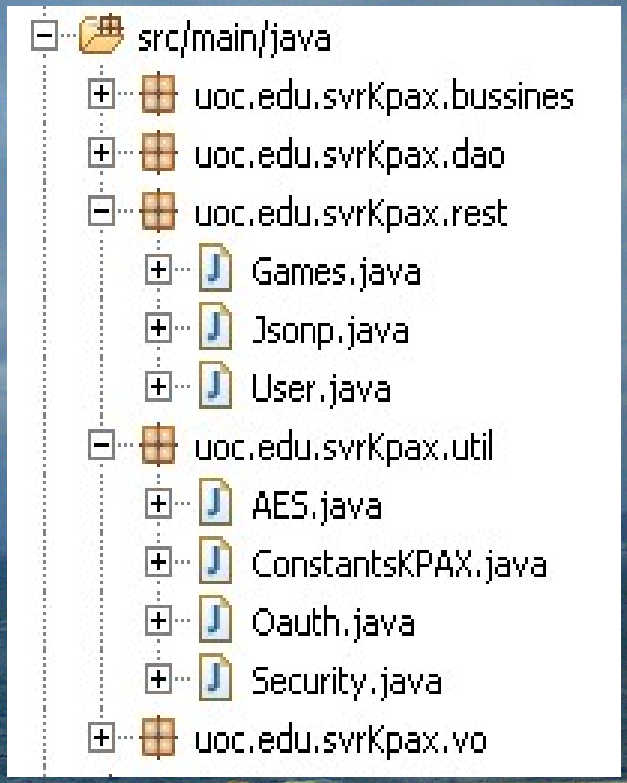
- Kpax ha sido creada a partir de la plataforma Elgg, Open Source, respaldada por una comunidad muy activa y desarrollada en PHP.
- Esta plataforma funciona sobre un entorno LAMP, que abarca un servidor Web Apache con PHP y una base de datos MySQL.
- Las funcionalidades extras deben ser implementadas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

Servicio Web: aplicación srvKpax

- La plataforma de servicios para red social se completa con una aplicación Java publicada como servicio web de tipo REST, que ofrece operaciones CRUD (commit, read, update, delete) sobre una base de datos MySQL propia también llamada Kpax.
- Sobre este sistema se implementará la lógica de la aplicación para poder crear funcionalidades de interacción con los juegos más allá de las ofrecidas por defecto por Elgg.

Servicio Web srvKpax

- La aplicación web srvKpax fue desarrollada en Java con una arquitectura en capas:
  - Capa de acceso a datos DAO.
  - Representación de tablas como objetos VO.
  - Capa de lógica de negocio BO.
  - Capa de servicios web o Rest.



Servicio Web srvKpax

- La aplicación web utiliza la tecnología Hibernate para el mapeo y el aceso a la BD mysql Kpax. Para cada tabla se crea una clase en VO.

- Por cada clase VO se crean dos clases en DAO: la clase de implementación de métodos y su interface de definición de métodos.

- Estos objetos utilizan para el acceso a datos el lenguaje HQL implementado por Hibernate.

Servicio Web srvKpax

- La capa de servicios Rest utiliza Jersey como implementación de servicios tipo JAX-RS.
- Define en los métodos que son ofrecidos por el servicio web características como:
  - Que parámetros recibe o devuelve.
  - Que tipo de mensaje http se envía.
  - Que clase de BO debe ejecutar el método solicitado.

Servicio Web srvKpax

- La capa de lógica de negocio BO también posee por cada tabla representada como objeto una clase y su interface.
- Procesa los parámetros enviados por Rest
- Invoca a los métodos de las clases DAO de su mismo objeto para operaciones sobre la base de datos.

Servicio Web srvKpax

- El contenedor de beans Spring mantiene en tiempo de ejecución instancias de cada clase BO, DAO y VO.
- Spring gestiona la comunicación y las relaciones entre ellas mediante varios mecanismos. Este conjunto de funcionalidades se conoce como inyección de dependencias.

Requisitos del plugin Gameserver

- Crear una vista desarrolladores donde los usuarios tendrán accesibles los botones añadir juegos y mis juegos.
- Desde el botón “añadir juegos” accederán a un formulario que les permitirá dar de alta su desarrollo tras la creación de una clave RSA.
- Desde el botón “mis juegos” podrán acceder a una vista con todas sus publicaciones.

Requisitos del plugin Gameserver

- Crear una vista Gestión de juegos para los administradores, desde la que podrán consultar a la base de datos Kpax sobre el estado de los juegos añadidos.
- La consulta devolverá una tabla ordenable con toda la información de estado de los juegos, y un botón para editar su información relevante.

Metodología

- Ciclo de vida del plugin
  - Primera etapa de planificación y análisis del estado previo de la aplicación:
    - Estudio del plugin Kpax ya desarrollado.
    - Estudio del servicio web y de la base de datos que mapea.
  - Segunda etapa de diseño del plugin:
    - Que cambios se necesitan en la base de datos.
    - Análisis de casos de uso.
    - Resumen de métodos necesarios.

Metodología

- Construcción y Desarrollo de acuerdo al diseño planteado modificando los siguientes aspectos:
  - Cambios en la base de datos Kpax.
  - Reflejo de estos cambios en la aplicación Java.
  - Construcción o modificación de métodos.
  - Desarrollo en PHP de las vistas del plugin.

Desarrollo

- Cambios llevados a cabo sobre la base de datos Kpax:
  - Creación de tablas Developer, Platform y State.
  - Creación de vistas adminView y userView.
- Cambios llevados a cabo sobre el servicio Web Java:
  - Creación de clases Developer:
    - DeveloperVO
    - DeveloperBO
    - DeveloperBOimp
    - DeveloperDAO
    - DeveloperDAOimp

Desarrollo

- Cambios llevados a cabo sobre el servicio Web Java:
  - Creación de clases userView:
    - UserViewVO
    - UserViewBO
    - UserViewBOimp
    - UserViewDAO
    - UserViewDAOimp
  - Las mismas en el caso de adminView.
  - Modificación de ficheros de configuración.

Desarrollo del plugin

- Siguiendo las directrices de Elgg para el desarrollo de componentes en PHP.
- Todos los plugins deben situarse en la carpeta /mod con una estructura de carpetas y ficheros obligatoria:
  - Carpeta Actions
  - Carpeta Languages
  - Carpeta Pages
  - Carpeta Views con vistas y formularios separados
  - Fichero manifest.xml
  - Fichero start.php

Desarrollo del plugin

- Los plugins para Elgg deben respetar la arquitectura Modelo Vista Controlador:
  - Los ficheros de página sólo se ocupan del layout o disposición de elementos, vistas y formularios.
  - Las vistas sólo muestran datos al usuario.
  - Los formularios sólo recogen datos.
  - Las acciones son las encargadas del procesamiento y no devuelven ni muestran resultados.

Coste y horas de trabajo

- El proyecto va a ser llevado a cabo por una sola persona que invertirá 96 jornadas en las fases de desarrollo y pruebas, repartidas de la siguiente forma:

Nombre	Fecha de inicio	Fecha de fin	Duración
Preparar Entorno de desarrollo	01/10/2013	15/10/2013	10
Estudiar el desarrollo ya hecho	14/10/2013	01/11/2013	14
Iniciar el desarrollo de la 1º entrega	25/10/2013	26/11/2013	22
Iniciar el desarrollo de la 2º entrega	25/11/2013	04/01/2014	30
pruebas	09/12/2013	04/01/2014	20
TOTAL jornadas			96

Conclusiones

- El desarrollo del plugin Gameserver es parte de un proyecto más amplio, el desarrollo de una plataforma de red social Kpax que permita el intercambio de experiencias y conocimientos entorno a los juegos serios.
- La estructura de la red Elgg, motor de Kpax, propicia el desarrollo de los componentes añadidos en forma de plugins.

Conclusiones

- Se ha desarrollado el plugin de acuerdo a las directrices de la comunidad Elgg, haciendo las correspondientes pruebas en un entorno LAMP +Elgg +Web Service .
- El plugin utiliza parte de las funcionalidades desarrolladas anteriormente en Kpax. Es requisito previo para su uso que el plugin Kpax esté presente y habilitado en el sistema.
- El plugin Gameserver debe registrar uno de los componentes de Kpax para su funcionamiento.

Conclusiones

- El plugin Gameserver no se ajusta completamente a los requerimientos establecidos por el grupo de desarrollo de Kpax.
- Los plazos de entrega propuestos en las fases iniciales del proyecto no se han cumplido.
- El tiempo de estudio del entorno se ha alargado. Esto, junto con algunos problemas encontrados durante el desarrollo han agotado el tiempo programado para el trabajo.