Máster en Software Libre

Trabajo Final de Máster

Mejora en el listado de juegos de kPax

Especialidad: Administración web y comercio electrónico

Institución: FUOC – EIMT

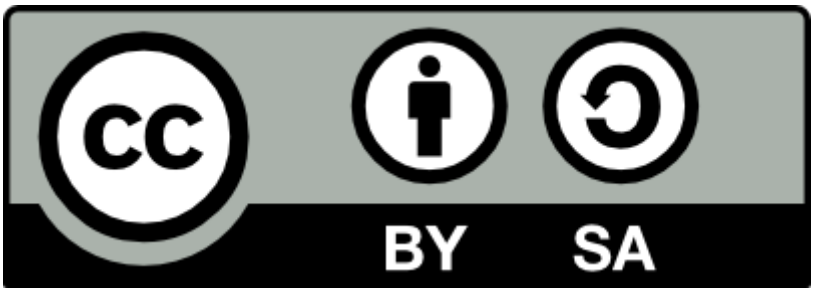
Alumno: Javier Ramírez Sánchez

Consultor: Francisco Javier Noguera Otero

Tutor externo: Daniel Riera Terren

Fecha: 15 de junio de 2017

Licencia



Usted es libre de:

- Compartir — copiar y redistribuir el material en cualquier medio o formato
- Adaptar — remezclar, transformar y crear a partir del material para cualquier finalidad, incluso comercial.

Bajo las condiciones siguientes:

- Reconocimiento — Debe reconocer adecuadamente la autoría, proporcionar un enlace a la licencia e indicar si se han realizado cambios. Puede hacerlo de cualquier manera razonable, pero no de una manera que sugiera que tiene el apoyo del licenciador o lo recibe por el uso que hace.
- CompartirIgual — Si remezcla, transforma o crea a partir del material, deberá difundir sus contribuciones bajo la misma licencia que el original.

Resumen del proyecto

kPax es una plataforma de juegos serios el cual basa el aprendizaje con el uso de juegos y así poder multiplicar las oportunidades de divulgación. Facilitando la participación de los usuarios con interfaces accesibles independientemente de la plataforma o sistema operativo del mismo. El proyecto de kPax está basado como plugin dentro de la plataforma libre Elgg.

kPax incentiva el uso por parte de los usuarios gracias a que tiene incorporada una parte de gamificación en la cual los usuarios pueden establecerse puntuaciones por el buen uso de los juegos y por tanto competir unos usuarios con otros.

Este proyecto se basa en la creación de una interfaz gráfica para poder facilitar a los usuarios la localización de juegos y poder encontrarlos dentro del mismo. Definiéndose por diversos criterios de búsqueda. El objetivo es que sea similar a la “play store” de Google.

En trabajos anteriores ya se había realizado trabajos para mejorar el diseño gráfico de la interfaz de búsqueda, el problema es que el proyecto de kPax continuó su desarrollo sin tener en cuenta los mismos. Por lo que la actividad se basará en encontrar y actualizar los desarrollos para que sean compatibles con la versión actual de kPax.

Índice del proyecto

Índice de ilustraciones

Introducción

kPax es una plataforma de juegos serios el cual basa el aprendizaje con el uso de juegos y así poder multiplicar las oportunidades de divulgación. Facilitando la participación de los usuarios con interfaces accesibles independientemente de la plataforma o sistema operativo del mismo. El proyecto de kPax está basado como plugin dentro de la plataforma libre ELGG.

Hemos realizado un desarrollo para una infraestructura para gestión de juegos educativos el cual se encargaba de mostrar en una interfaz más visual que la anterior.

El proyecto kPax está desarrollado en varias partes:

- Entorno servidor.
- Parte cliente basado en Elgg.

Para el proyecto usaremos como referencia el desarrollo de un anterior alumno, Marc Barceló Martín (http://hdl.handle.net/10609/43063) el problema es que su proyecto no se incorporó la rama central (master o devel) de kPax_core. El proyecto de kPax siguió con las modificaciones pertinentes de otros alumnos pero sin tener en cuenta el proyecto de Marc. Por lo que su puesta en marcha no ha sido nada trivial. Ha sido necesario realizar modificaciones en la librería principal para que funcionase el conjunto.

Hace un año se hizo un gran cambio estructural de la aplicación cambiando la base de datos a una base de datos no relacional, esta modificación fue realizada tanto en el entorno servidor como en el entorno cliente. Muchas de las llamadas en que existían en kPax no fueron migradas al 100% y sumando que la en la evolución de la herramienta ha sido considerable. Una de las primeras tareas que se ha tenido que realizar es realizar llamadas no migradas.

El software que se ha realizado ha sido publicado con una licencia libre, para la publicación del código se ha usado la forja Github. Github se ha convertido en un estándar de facto en cuanto a proyectos de software libre se refiere. Los proyectos donde se han publicado las modificaciones son:

- kPax server, donde hemos incorporado ciertas llamadas, aprovechando las modificaciones hemos limpiado cierto código no usado. https://github.com/javierrami/kPax2_server
- kPax Core, hemos realizado modificaciones principalmente a la librería principal en el fichero kPaxserv.php adicionalmente también he limpiado funciones que no tenían uso y hemos añadido algunas funciones necesarias. https://github.com/javierrami/kPax2_elgg_frontend_core
- kPaxList, es donde está publicado mi proyecto. El cual puede ser ejecutado como proyecto. https://github.com/javierrami/kPaxList

Objetivos

A nivel personal, para mi el principal objeto es que mis modificaciones se integren en dentro de la rama devel del proyecto principal de Daniel. https://github.com/drierat/kPax2_server o https://github.com/drierat/kPax2_elgg_frontend_core

Para lograr llevar a cabo el proyecto será necesario:

- Comprender la arquitectura de kPax y los componentes que la forman.
- Definir la metodología de desarrollo del proyecto a utilizar.
- Recopilar y analizar los diferentes plugins ya implementados y la versión inicial de kPax.
- Incorporar los plugins de manera incremental para obtener una nueva versión estable con las nuevas funcionalidades.
- Realizar un entorno vistoso para los usuarios y visual.
- Permitir a los usuarios encontrar Juegos de una forma sencilla.

Estado del arte

kPax es una plataforma de juegos serios, en la cual se centralizan juegos con una temática educativa, basado en la gamificación con el objetivo de motivar a los usuarios para mejorar sus habilidades durante el aprendizaje.

El proyecto de kPax ha contado con la participación de numerosos estudiantes, gracias a la aportación de cada uno con sus proyectos se está consiguiendo llegar a construir el desarrollo necesario para la puesta en producción del sistema.

En la web de http://openaccess.uoc.edu podemos encontrar los siguientes proyectos de alumnos que han participado en el proyecto:

- Migracion a MongoDB, Junio 2016 http://hdl.handle.net/10609/52741
- Incorporacion de funcionalidades sociales, Junio 2015 http://hdl.handle.net/10609/42772
- Migración a ELGG 2.1, Junio 2016 http://hdl.handle.net/10609/52702
- Mercado de aplicaciones, Junio 2015 http://hdl.handle.net/10609/43063
- Integración de modulos a kPax, Junio 2015 http://hdl.handle.net/10609/42744 Castellano.
- Incorporación de funcionaliades, Junio 2015 http://hdl.handle.net/10609/42726 Castellano.
- Incorporación de login desde Facebook, Junio 2015: http://hdl.handle.net/10609/42772
- La gestión de los juegos, Junio 2014 http://hdl.handle.net/10609/32981
- Incorporacion de funcionalidades, Julio 2014 http://hdl.handle.net/10609/35341
- Incorporacion del perfil de usuario, enero 2014 http://hdl.handle.net/10609/29902

- Ampliación de funcionalidades, enero 2014 http://hdl.handle.net/10609/29821
- Ampliación de funcionalidades, Junio 2013 http://hdl.handle.net/10609/23533
- Modulo de juegos, Junio 2013 http://hdl.handle.net/10609/22443

De todos estos proyectos podemos encontrar tres que son similares al que yo he realizado y que se versan sobre la creación de una interfaz gráfica que permita encontrar juegos dentro del sistema. Voy a proceder a realizar un análisis de cada proyecto de cara a poder obtener una base en desarrollo, aprovechar sus aportaciones y partir de una base. Justo al ser un Master de Software Libre considero un gran valor ese reaprovechamiento del código como uno de los objetivos del mismo.

Mercado de aplicaciones
Junio 2015 Realizado por Barceló Martí, Marc http://hdl.handle.net/10609/43063

El proyecto de Marc, hizo un gran progreso en cuanto a la interfaz gráfica dando como resultado la imagen vemos a continuación

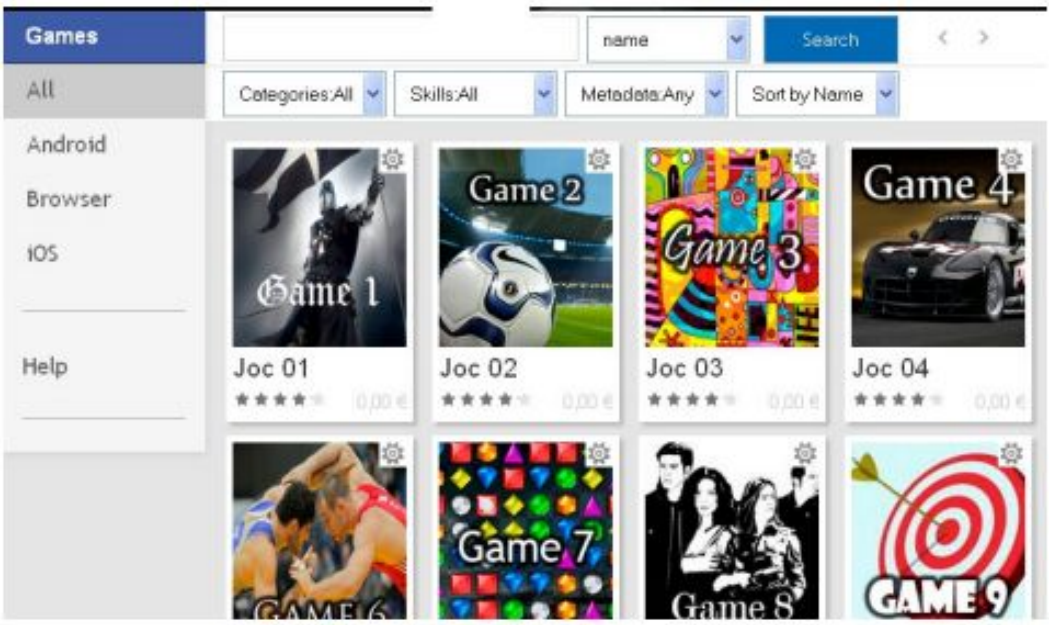


Imagen 1: Resultado del proyecto de Marc, por Barceló Martí, Marc

Como resultado e interfaz gráfica el proyecto de Marc es realmente bueno, ya que muestra una interfaz gráfica realmente muy buena y se asemeja a los software actuales de referencia, reduciendo por tanto la curva de aprendizaje al máximo.

En la actualidad el proyecto no es funcional, ya que kPax ha evolucionado sin tenerlo presente, un gran fallo fue no incorporarlo a la rama central de desarrollo. Esto ha provocado que el desarrollo de kPax se realizará de forma tangencial a este desarrollo y no se tuviese en cuenta en las evoluciones del mismo.

Incorporación de funcionalidades
Julio 2014 Realizado por Corral Blanch, Víctor http://hdl.handle.net/10609/35341

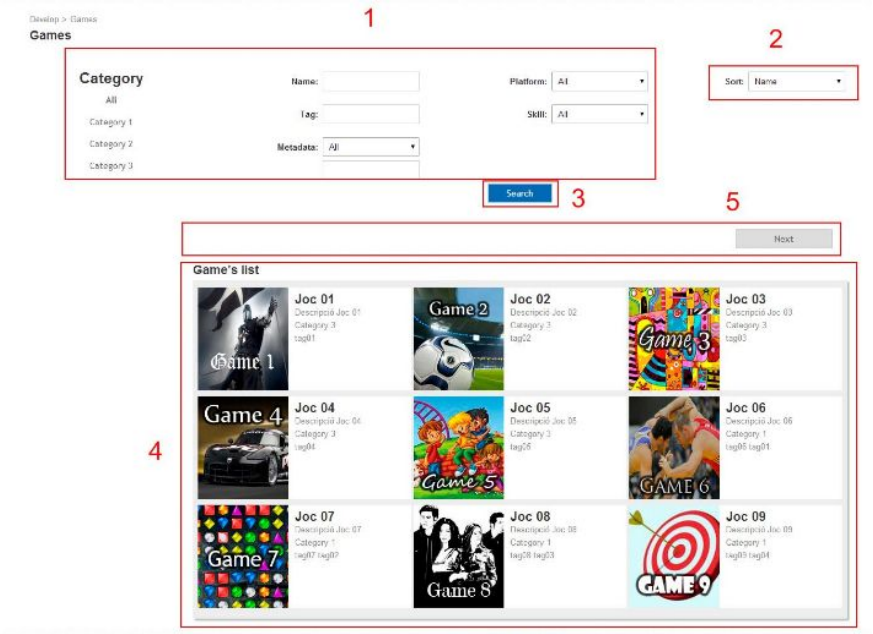


Imagen 2: Búsqueda de juegos por Corral Blanch, Víctor

El proyecto de Víctor no tiene una interfaz gráfica destacable pero si es funcional que por un lado es realmente interesante. Muestra imágenes de los juegos y permite una búsqueda, la cual la podríamos considerar avanzada aunque a lo mejor demasiado compleja para usuarios amateur.

Módulo de juegos

Junio 2013 Realizado por Giró Oriol, Albert http://hdl.handle.net/10609/22443

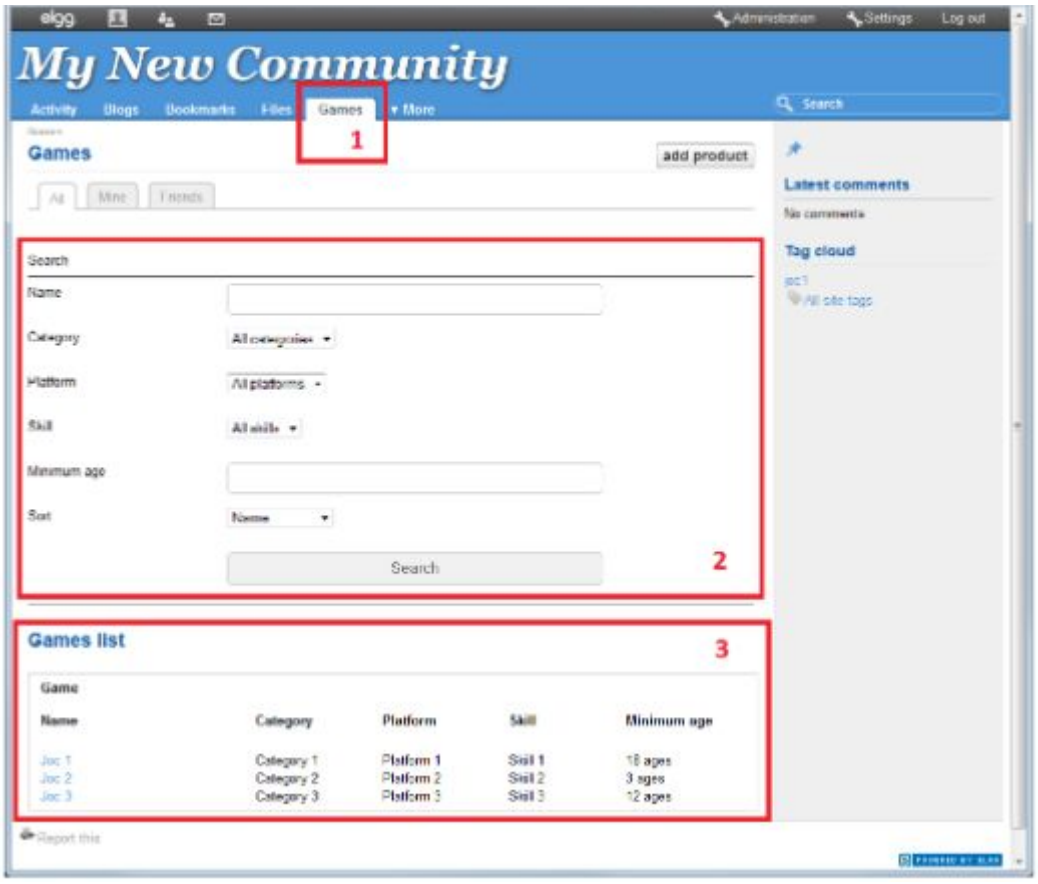


Imagen 3: Listado de los juegos, por Giró Oriol, Albert

Por lo que comenta Barceló Martí, Marc (la persona que realizó el primer proyecto descrito) el desarrollo que se realizó no era funcional, por lo tanto pudo cogerlo como punto de partida para realizar su desarrollo ya que le suponía más esfuerzo revisar el mismo que empezar de cero.

Elgg

La red social Elgg[1] – sobre la que está implementada kPax – es un sistema de gestión de contenidos de código abierto compatible con una serie de estándares y que corre en entornos del tipo XAMP[2], permitiendo crear y administrar redes sociales a la vez que ofrece un alto grado de control sobre el acceso a los contenidos.

En la documentación de Elgg se recomienda enfáticamente a modificar su código, esgrimiendo como razones las posteriores dificultades de actualización, agujeros de seguridad o roturas de funcionalidades existentes. Propone que los cambios y nuevas características se implementen mediante plugins. Más información en http://learn.elgg.org/en/2.3.3/guides/dont-modify-core.html

Se puede consultar la api de Elgg desde esta dirección (http://learn.elgg.org/en/2.3.3/guides/plugins.html) desde ahí se puede encontrar como interactuar con el propio sistema.

Para el uso de kPax es necesario los siguientes plugin de Elgg:

- Elgg Developer Tools: Es un set de herramientas para desarrolladores de los plugins o temas para Elgg.
- Web Services: permite un framework para construir servicios web RPC.
- Api Admin: Es el encargado de los certificados de autenticación.
- Login Required: Obliga a los visitantes a logearse para ver el site.
- HTML5: Tiene cantidad de librerías javascript y permite al desarrollador el uso de etiquetas HTML5 en Elgg.

Es importante tener los plugins correctamente ordenados, ya que en función del orden carga las librerías y si por ejemplo un módulo carga antes de cargar la librería este dará error. Y para poder desactivarlo es necesario crear un fichero para desactivar todos los plugin.

Requisitos de la solución

Para la creación del plugin hemos definido los siguientes requisitos, los cuales por diversa índole es necesario cumplirlos:

- El software debe de cumplir la licencia original o mantener compatibilidad. Por tanto publicamos con licencia Creative Commons Attribution-ShareAlike 4.0
- El código debe ser publicado en una forja, en este caso Github.
- Integración con los plugin ya existentes.

Riesgos del proyecto

Se estima que existe una serie de riesgos que pueden atentar contra el desarrollo del proyecto:

- Curva de aprendizaje, tardar mucho tiempo en comprender cómo funciona Elgg y conocer la estructura interna de K-PAX.
- Falta de experiencia desarrollando, la programación no es una de mis mejores habilidades principalmente por falta de experiencia en grandes proyectos.
- Que la evolución de K-PAX haya sido tal que no sea posible incorporar como base el proyecto de referencia.
- Problema idiomático, la mayoría de los trabajos realizados por alumnos anteriores han sido en catalán; idioma que desconozco. Es cierto que se puede llegar a entender con paciencia y cuidado pero son muchos proyectos lo que hay que revisar y entender para llegar a conclusiones.

Herramientas usadas

Las siguientes herramientas han sido utilizadas para el desarrollo del proyecto, he procurado utilizar herramientas que sean accesibles desde el navegador web. Por

comodidad y por los recursos que tengo he preferido usar un entorno web lo cual me ha permitido aislarme del equipo informático que he usado en cada momento, teniendo los mismos recursos independientemente de donde me encuentre:

- **Codiad**, (codiad.com) es un editor de código online, para la ejecución del código solo era necesario guardar y actualizar la página php que he editado. Esto ha resultado verdaderamente útil ya que la edición la he realizado directamente desde un servidor externo y no en local. Tiene licencia MIT.
- **Robomongo**, proporciona una interfaz gráfica desde la cual se puede hacer una gestión de las colecciones, facilitando la edición de la misma. También permite la edición de colecciones
- **Google Docs**, procesador de textos online propiedad de Google. Nos permitirá la posibilidad de colaborar de forma simultánea en el documento y la posibilidad de revisión del documento sin necesidad de estar pendiente de cuál es la última versión del mismo.
- **Advanced REST Client**, extensión de Google Chrome la cual permite hacer consultas al servidor por JSON y poder recibir las respuestas en el mismo formato.
- **Trello,** usada para definir la acciones a realizar en el desarrollo y poder enseñar al equipo el estado de cada acción y poder aportar ideas para futuras mejoras.
- **Github,** mediante este servicio de alojamiento repositorio Git basado en la web, es posible el control de revisión y gestión de código fuente. Sistema cuyo objetivo es facilitar la administración de las distintas versiones de cada producto, gracias a los comentarios realizados en los commits los siguientes desarrolladores que recojan el proyecto podrán comprender las modificaciones y el histórico del proyecto.
- **appear.in**, herramienta para reuniones la cual permite juntarnos a Roger (otro estudiante del Master) y a mi con el cliente (Daniel Riera) y el tutor del proyecto (Francisco Javier Nogera). Gracias a lo cual podemos realizar videollamadas siendo por tanto las reuniones mucho más efectivas.

Análisis del sistema

En el presente capítulo analizaremos el sistema actual, incluyendo algunas modificaciones del mismo que hemos planteado realizar.

En este capítulo trataremos:

- Definición del modelo de datos, en el cual reflejaremos la base de datos usada y una definición del porqué de la organización.
- Arquitectura del sistema, en el cual realizaremos un análisis de la infraestructura del sistema,
- Instalación del sistema, realizaremos una guía paso a paso de como dejar el proyecto instalado.

Definición del modelo de datos

En este apartado definiremos los datos y la definición de los mismos que he ideado para el funcionamiento de los datos. Definiremos el motor de base de datos usado.

kPax usa como motor de base de datos MongoDB[3] una base de datos NoSQL. En lugar de guardar los datos en tablas como se hace en las base de datos relacionales, MongoDB guarda estructuras de datos en documentos similares a JSON. Lo que permite tener una especie de polimorfismo en los datos, ya que no es obligatorio que todos los datos de dentro de la colección sean iguales. Por otro lado mejora considerablemente la escalabilidad

Al no tener una definición de los datos obligatoria de cumplir es importante que en la parte de diseño definir los tipos de colecciones existentes, sobre el papel para que no existan equívocos a la hora de realizar consultas o inserciones. Es necesario realizar una gestión más exhaustiva en el código (programa) a la hora de consultar o insertar datos.

El entorno de los datos está formado por tres colecciones, lo que en MySQL se llamaría tablas:

- Games: contiene el listado de los juegos.
- gamesCategories: Contiene el listado de categorías por tipo, adicionalmente almacena el listado de tags que lo forma.
- users: Almacena los usuarios, con ello puede guardar la documentación de cada usuarios.

Games

```
{
    "_id": "585a1b278af0a338431a3f03", Identificador único generado automáticamente.
```

"guid": "66", Identificador único

"name": "Prova sense likeskPax activat", Nombre del juego

"description": " Prova sense likeskPax activat ", Descripción del juego en formato HTML

"urlImage": "URL", Url de la imagen de la carátula/portada del juego.

"category": "1", Categoría a la que pertenece

"tags": { Contiene referencias a: Similar Games, Skills, Platforms.

"similar_games": {

"a": 1, -Se identificara el nombre del juego y el identificador del mismo.

"D": 1,

"d": 1,

"c": 1

},

"owner": 36, El identificador del propietario del juego.

"updated_at": "2016-12-21T06:03:16.918Z", última actualización.

"status": 1, El estado del juego 1 es activo.

"nlikes": 0, Número de likes que tiene la aplicación.

"created_at": "2016-12-21T06:03:16.918Z"

}

gamesCategories

{

"_id": "592aebb746be1c482b598011", Identificador único.

"guid": "4", Identificar que le damos al crearse-

"name": "skill chula", Nombre de la categoría.

"type": "skill", El tipo de tag que es

}

Los type de tags pueden ser los siguientes, aunque podrían ser más:

- Similar Games, Contiene el _id de los juegos que son similares, esto será un array.
- Skills, Contiene los skills del juego, es un array.
- Platforms: Listado de plataformas disponibles.

Aun así esta colección está creada para facilitar las llamadas y podría no existir, ya que se podría obtener de la colección games, realizando consultas para que muestre las categorías extraídas de la colección Games. Pero lo dejamos así para facilitar el desarrollo y tener la posibilidad de tener categorías ocultas.

Arquitectura del sistema.

El sistema kPax, está formado a grandes rasgos por dos partes claramente diferenciadas:

- Entorno cliente, está formado por un entorno en php usando Elgg como base que proporciona un entorno social.
- Entorno servidor, se compone de la base de datos y una interfaz realizada con NodeJS la cual permite extraer datos en formato JSON.

El entorno cliente está formado por un servidor LAMP el cual está compuesto por:

- Sistema operativo Linux, Ubuntu 16.04 LTS.
- Servidor web Apache2, es un servidor web HTTP de código abierto, con un uso muy extendido.
- PHP 7.0, se instala como módulo dentro de Apache2 para que lo utilice como intérprete del lenguaje usado en Ellg y en kPax.
- Servidor de bases de datos Mysql el cual será usado por Elgg y sus componentes internos. Ya que los juegos se encuentran en otra infraestructura.
- Elgg, es una plataforma de red social de código abierto, que se usa como base de kPax.
- Plugin kPax

Entorno servidor, realmente sería el entorno donde están almacenados los datos de la base de datos. En este caso está compuesto por:

- Servidor NodeJs, es el intérprete del código del servidor, facilita la iteración
- MongoDB, base datos no relacional.
- kPaxServer

La comunicación entre ambos servicios se realiza mediante una api REST intercambiando archivos en formato JSON (JavaScript Object Notation).

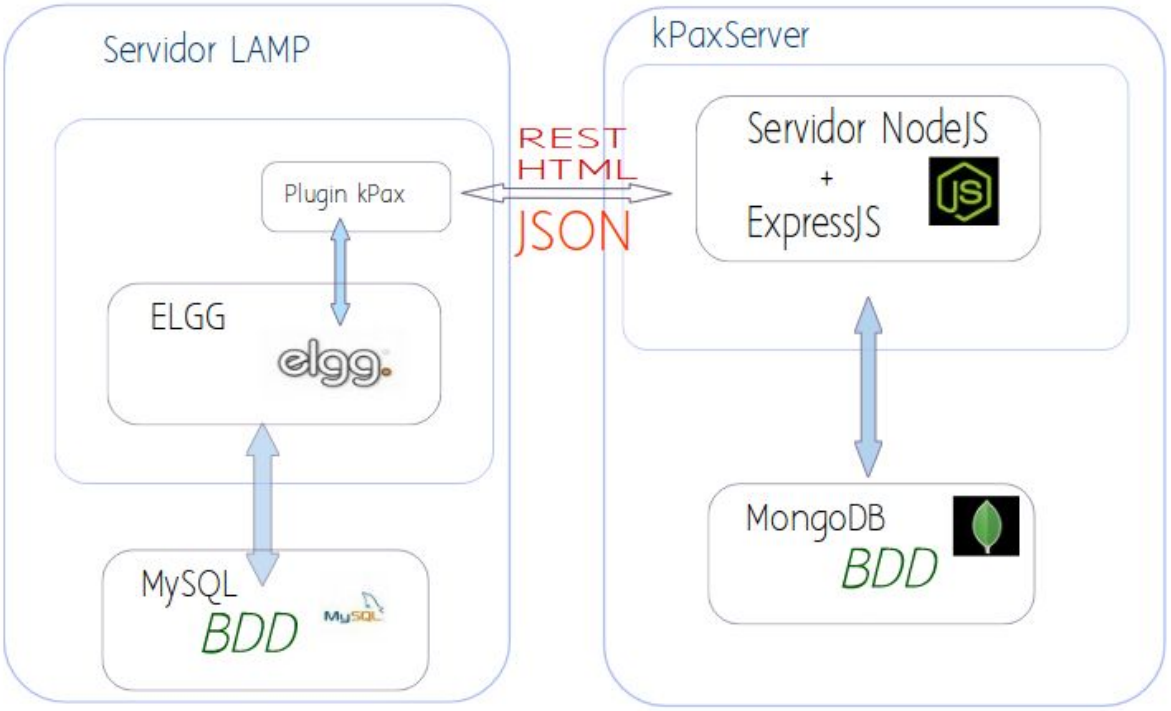


Imagen 4: Esquema de kPax Server 02 y NodeJS sobre MongoBD por Miquel A. Muntaner

Instalación del sistema

Para la ejecución del sistema y para las pruebas hemos realizado una instalación independiente de kPax, con ello contábamos con un entorno exclusivo para las pruebas dentro de nuestro entorno desarrollo. Como llevamos hablando el sistema está basado en dos partes claramente diferenciadas, cliente y servidor.

He aprovechado la realización de esta sección para actualizar la wiki del proyecto en Github y así hacer más fácil la instalación de Kpax y evitar la confusión a la hora de instalarlo. Ya que en mi experiencia tuve un problemas debido principalmente a una mala instalación del entorno servidor debido a las versiones del software instaladas.

Instalación cliente

Para la instalación del entorno cliente, servidor LAMP y Elgg existe un archivo de instalación el cual se encarga de realizar todos los pasos. Antes de realizar la ejecución del script de instalación es necesario instalar unzip.

```
sudo apt-get install unzip
```

Para la instalación realizamos los siguientes pasos:

1. Clonamos el repositorio de github, en este caso usaremos la rama de desarrollo que contiene la versión más reciente.

```
git clone https://github.com/drierat/kPax2_server.git
git checkout devel
```

2. Accedemos a la carpeta kPax2_elgg_frontend_core/install/ y ejecutamos el archivo de instalación (InstallElgg1604LTS.sh) el cual se encargará de instalar todos los complementos necesarios.
3. Una vez completado accedemos con el navegador web a http://[ip-servidor]/kPax2/install.php
4. Seguimos el instalador automático, con esto ya estaría elgg instalado.
5. Accedemos al portal de administración y activamos los siguientes módulos ordenándolos en este orden:
    a. Elgg developer tools, lo ponemos en primer lugar. Una vez activado accedemos y desactivamos la caché y permitimos mostrar los mensajes de error.
    b. Web services
    c. Api_admin
    d. loginrequired
    e. HTML5
    f. kPax_core, accedemos al archivo “mod/kPax_core/lib/kPaxSrv.php” y en el archivo copiamos en la url el $apikey generado con el plugin API admin. http://[ip-servidor]/kPax2/admin/administer_utilities/apiadmin . En el mismo fichero indicamos el servidor de kPax que realizará las llamadas.
    g. kPax_Theme, en este caso lo dejamos al final del listado.

Nota: En el caso que nos equivoquemos al activar algún plugin al activarse produzca un error la configuración de los plugin se vuelve inaccesible. La solución es desactivar todos los plugin para ello es necesario crear el archivo (sin ningún contenido) disabled dentro de la carpeta mod.

Instalación servidor

La instalación del entorno servidor se ha realizado sobre Ubuntu 16.04 LTS siguiendo los siguientes pasos:

1. Instalación de npm (Node Package Manager),

```
sudo apt-get install npm
```

2. Instalación de NodeJS en su versión 6.0 o superior

```
curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash -
sudo apt-get install -y nodejs
sudo apt-get install -y build-essential
```

3. Instalación de MongoDB en su versión v.2.6.10 para ello lo primero que debemos realizar es importar la clave del nuevo repositorio que contiene la versión de MongoDB requerida. Posteriormente ya podemos instalarlo.

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv EA312927
```

```
echo "deb http://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.2 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.2.list
sudo apt-get update
sudo apt-get install -y mongodb-org
```

4. Clonamos el repositorio de github, en este caso usaremos la rama de desarrollo que contiene la versión más reciente.

```
git clone https://github.com/drierat/kPax2_server.git
git checkout devel
```

5. Podemos iniciar el servidor, para ello accedemos

```
MONGODB_URL="mongodb://readwrite:1234@ds021462.mlab.com:21462/kPax2" bin/www
```

En modo debug.

```
DEBUG=*
MONGODB_URL="mongodb://readwrite:1234@ds021462.mlab.com:21462/kPax2" bin/www
```

Diseño de la aplicación

En el presente capítulo primeramente se realiza un análisis de los todas las modificaciones que se han realizado y así quede documentado y reflejado.

Planificación de las tareas

Las tareas a realizar durante la ejecución del proyecto las podemos dividir en las siguientes, estas tareas se han definido los siguientes objetivos, los cuales pueden variarse en función del progreso de cada uno. Sin necesidad de estar anclados al principio del proyecto:

- Recoger el trabajo anterior y ponerlo en funcionamiento. Usaremos como base un proyecto anteriormente realizado por un alumno. Partiendo así de una base bastante buena podemos alcanzar un mejor resultado; es en parte un objetivo que tiene el master de forma intrínseca.
- Analizar el trabajo anterior y funcionalidades. Una vez puesto en marcha el trabajo anterior es necesario revisar en que podemos realizar mejoras y montar un plan de trabajo.
- Estudiar las mejoras del mismo y futuras necesidades.

kPax Core

- Instalador para que se instale todos los requisitos que faltaban como el unzip que no se instalaba.
- Se han añadidos las siguientes funciones a kPaxserv.php

La interfaz de kPax Core es la que verán los usuarios y con la realizará iteraciones, es por ello que debe ser una interfaz visualmente muy atractiva y además dar una funcionalidad en la cual el usuario lo considere útil.

Para empezar hemos realizado una serie de mejoras técnicas que facilitaran el uso a futuro. Hemos modificado el manifest.xml, este archivo se encarga de definir los requisitos (dependencias, versiones) a la hora de instalar un plugin así como el nombre del plugin y más datos descriptivos.

Para la realización de las descripciones nos hemos fijado en la guía explicada por Elgg http://learn.elgg.org/en/2.3.3/guides/plugins.html

Kpax 2

Kpax integration

Información | Dependencias

Tipo	Nombre	Valor de Test	Valor Actual	Comentario
Requiere	Versión Elgg	>= 2.0	2.2.2	OK
Requiere	Prioridad	Luego html5	Luego html5	OK
Requiere	Prioridad	Luego loginrequired	Luego loginrequired	OK
Requiere	Prioridad	Luego apiadmin	Luego apiadmin	OK
Requiere	Prioridad	Luego web_services	Luego web_services	OK
Requiere	Prioridad	Luego developers	Luego developers	OK
Provee	Plugin: kPAX_core	1.5	--	OK

Adicionalmente hemos modificado la wiki del proyecto indicando la necesidad de tener instalada en el servidor la herramienta unzip, necesaria durante la instalación para poder descomprimir el archivo zip con el Elgg.

Se ha realizado modificaciones en la librería central de kpax (kpaxSrv.php) en el cual se ha realizado una serie de limpieza eliminando funciones que no son necesarias en la actualidad o que no aportaban un valor.

getTags($tag)

$tag tag que es necesario que devuelva.

Devuelve un listado con los tags que se ha especificado en un primer momento puede ser: similarGames, skill, platform y categories.

getCategories($type)

Esta función es exactamente igual a la anterior, pero la mantenemos por el tema de retrocompatibilidad.

kPax Server

En esta parte describiremos las modificaciones que hemos realizado en la interfaz de servidor kPax, la cual es accedida desde la interfaz del servidor mediante el uso de llamadas JSON. Esas misas llamadas pueden ser realizadas desde una aplicación en un teléfono o cualquier interfaz con conexión a internet. Por ejemplo nosotros hemos realizado pruebas con Advanced REST Client.

En la actualidad kPax server tenia referencias a games y user, hemos creado la capacidad de poder realizar consultas en función de los tipos de tags o categorías creando por tanto la interfaz de categorías. El cual responde a las siguientes funciones:

/category/list
Devuelve el listado de todas las categorías almacenadas en la colección Categories con el siguiente formato.

```
{
    "_id": "592af12f46be1c482b598013", -> es id único que le da la base de datos.
    "guid": "6", -> identificador único que se mantiene como histórico, realmente podría ser eliminado.
    "name": "skill 3", -> Nombre del tag.
    "type": "skill" ->  Tipo de tag.
}
```

Los tipos de tags pueden ser los siguientes:

- Categories, será la categoría disponibles de los juegos.
- Skills, el aprendizaje que se puede obtener de ese videojuego.
- Platforms, plataformas en las cuales el juego puede ser ejecutado.

En el caso que se requiera se pueden realizar consultas dentro de la propia llamada, por ejemplo se puede ejecutar una llamada para que solo muestre las categorías de Skills, la llamada sería de la siguiente forma:

http://[dirección del servidor]:8081/category/list?q={"type":"skill"}

Devolviendo el listado de los skils.

```
    {
        "_id": "592aebb746be1c482b598011",
        "guid": "4",
        "name": "skill chula",
        "type": "skill"
    },
    {
        "_id": "592af11846be1c482b598012",
        "guid": "5",
        "name": "skill 2",
        "type": "skill"
    },
```

```
{
    "_id": "592af12f46be1c482b598013",
    "guid": "6",
    "name": "skill 3",
    "type": "skill"
}
```


Uno de los mayores problemas que he tenido en la realización del proyecto ha sido tener la versión de node js incorrecta. Ha supuesto un gran fallo ya que la propia aplicación no hacia una correcta gestión de errores y se paraba cuando se realizaban peticiones incorrectas. Es decir tenía un comportamiento muy anómalo. Para que a mis sucesores no les vuelva a ocurrir he realizado modificaciones en la wiki del proyecto.

Conclusiones

En primer lugar creo que el tiempo de desarrollo del proyecto ha sido muy corto, hay que tener en cuenta que las personas que decidimos cursar un máster o carrera a distancia es porque nuestro tiempo o situación geográfica no nos permiten adaptarnos a estándares. Por lo que tener que realizar un proyecto en unos pocos meses es una tarea complicada de conseguir. Afortunadamente tanto Javier como Daniel son conscientes del problema.

Esta ha sido mi primera experiencia realizando un desarrollo a gran escala y el primer desarrollo en software libre que he participado. Con ello me he dado cuenta de la importancia de la documentación de forma correcta y entendible.

En un entorno de colaboración distribuida en el cual muy posiblemente los colaboradores del proyecto no los conozcas personalmente en la vida, es muy importante tener una comunicación fluida para que el flujo de las decisiones de cada uno estén complementadas y consensuadas.

Durante el desarrollo del proyecto he tenido dificultades idiomáticas ya que la mayoría de trabajos realizados con anterioridad estaban realizados en Catalán y desafortunadamente no es un idioma que conozca, es cierto que su similitud con el castellano he llegado a superar algunas barreras pero no ha sido fácil.

La necesidad de hacer un merge de tu proyecto en la rama principal, se convierte en algo esencial; así el siguiente podrá partir de la base con el proyecto incorporado. Si el desarrollo se realiza como un módulo especial para el proyecto ten como premisa que esté referenciado; aún así es muy probable que se pierda con el paso del tiempo y la evolución del software.

Como posibles soluciones se podría estudiar en volver a la versión anterior, ya que después de haber analizado todos los proyectos realizados la verdad que el grado de avance en cada parte individual era muy bueno. Aunque después de haberlo comentado, los desarrollos son completamente independientes y aislados entre sí, debido a que las modificaciones en la base de datos se han realizado de forma aislada sin tener presente el resto. Es por ello que se decidió usar MongoDB ya que el entorno de base de datos es más flexible y no dificulta las modificaciones.

Sobre el proyecto a nivel personal no considero haber llegado al objetivo esperado y planteado al inicio del mismo, tenia pensado que la puesta en marcha del proyecto marcado como referencia del mismo fuese mucho más sencilla.
A nivel de proyecto he podido colaborar en un proyecto de software libre y poder comprobar como es la colaboración en un proyecto a largo plazo, conociendo las dificultades y ventajas.
Adicionalmente he descubierto cómo trabajar con base de datos noSQL y la simplicidad y facilidad del desarrollo con NodeJS. El uso de trello, me ha permitido ver la simplicidad que

tiene como herramienta de gestión de tareas y la he empezado a aplicar dicha tecnología con mi equipo de trabajo.

Mantenimiento y posibilidades de mejora

De cara a pensar en un futuro y para que la aplicación fuese fácilmente escalable se podría Dockerizar la aplicación, esto es una evolución de la virtualización en la cual se aísla el sistema operativo de la capa de aplicación. Dentro del contenedor se configura todo lo necesario para que la aplicación pueda funcionar, por ejemplo en este caso en la parte de Elgg con el plugin de kPax se configuraría el contenedor con el servidor Apache, el intérprete de PHP y el código de kPax. La base de datos se deja como servicio aparte para que sea atacada desde todas las instancias.

En un futuro se puede eliminar la colección gamesCategories y sustituirlo por una función que recoja el valor único que esté dentro de la colección games.

Al ser software libre, estar publicado y accesible cualquier persona podría recoger el código e implementar mejoras sobre este, solo que las modificaciones del código deberían estar publicadas con la misma licencia. Por lo que el mantenimiento no supone ningún coste quitando el coste de la mano de obra, sería necesario realizar tareas de backup

A futuro creo que se debería centrar y recopilar el trabajo que se ha realizado hasta para crear un producto final, por lo menos aprovechar por un lado las interfaces gráficas disponibles aunque el código interno sea necesario reescribirlo.

De cara a la interfaz gráfica sería necesario rehacer el diseño de la visualización de los datos, he procurado dejar bien hecho el backend para que luego un diseñador pueda centrarse únicamente en la parte gráfica del proyecto y no en el backend.

Referencias

Elgg, Elgg es una plataforma de Servicios de red social de código abierto que ofrece Blogueo, trabajo en red, comunidad, recolección de noticias vía feeds e intercambio de archivos. Todo puede ser compartido entre los usuarios, utilizando los controles de acceso y puede ser catalogado mediante tags.
XAMP, XAMPP es un servidor independiente de plataforma, software libre, que consiste principalmente en la base de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y Perl.
MongoDB, es un sistema de base de datos NoSQL orientado a documentos, desarrollado bajo el concepto de código abierto.

Anexo: Imágenes de la interfaz