Kpax Plataforma d'aprenentatge en Xarxa: Migració de kPaxServer a NodeJs amb MongoDB

Miquel A. Muntaner Morey
Administració web i comerç electrònic
Màster en Programari Lliure

Daniel Riera Terrén
Francisco Javier Noguera Otero

19 de juny de 2016

FITXA DEL TREBALL FINAL

Títol del treball:	*Kpax Plataforma Aprenentatge en Xarxa*
Nom de l'autor:	*Miquel A. Muntaner Morey*
Nom del consultor/a:	*Daniel Riera Terrén*
Nom del PRA:	*Francisco Javier Noguera Otero*
Data de lliurament (mm/aaaa):	*06/2016*
Titulació o programa:	*Màster en Programari Lliure*
Àrea del Treball Final:	*Administració web i comerç electrònic*
Idioma del treball:	*Català*
Paraules clau	*KpaxServer, Servidor, Nodejs*

Resum del Treball :

L'objectiu aquest projecte de final de màster és la migració del servidor de kPax a una nova arquitectura basada en NodeJs amb Express com framework usant la base de dades no relacional MongoDB, amb la finalitat d'aconseguir un entorn més eficient en el seu funcionament i especialment apte per incorporar les noves funcionalitats que es vagin desenvolupant de manera simple.

La primera versió del servidor desenvolupada es va mostrar cada cop més complexa a l'hora d'integrar-li noves ampliacions, per la qual cosa s'ha decidit des del principi l'ús de NodeJs per evitar que es reprodueixi la situació.

El projecte s'ha desenvolupat fent ús de metodologies 'àgils' per coordinar tots els aspectes d'aquest TFM. Les eines col·laboratives, com ara gitHub, Trello reforçades per reunions virtuals setmanals han estat fonamentals per l'evolució correcta del projecte.

El resultat del projecte ha estat la realització d'un servidor completament funcional, capaç de donar servei a la capa d'ELGG, que s'ha actualitzat paral·lelament a aquest projecte, de manera coordinada i en el context d'un altre projecte de final de Màster.

El servidor realitzat és plenament funcional i robust. L'arquitectura en la qual s'ha desenvolupat permet incorporar noves funcionalitats o modificacions de manera simple tant al codi del servidor com a l'estructura de la base de dades.

Abstract :

The aim of this project is the migration of the kPax server to a new architecture based on NodeJs with Express framework, using non-relational database MongoDB, in order to achieve a more efficient in its operation and particularly suitable to incorporate new features.

The first server version has shown increasingly complex when you integrate new extensions, so it was decided from the outset to use NodeJs to simplify the process.

The project was developed using methodologies 'agile' to coordinate all aspects of TFM. Collaborative tools like GitHub, Trello reinforced by weekly virtual meetings are essential to proper development of the project.

The result of the project was the realization of a fully functional server, able to service layer Elgg, which was updated in parallel to this project in the context of another project.

The server made is fully functional and robust.The architecture in which has been developed allows to incorporate new features or changes in simple way, both in the server code as to the structure of the database.

RESUM DEL PROJECTE

kPax és una xarxa social destinada a l'aprenentatge basat en jocs seriosos, que neix de l'esforç per conciliar les xarxes socials i el joc com compromís i repte. kPax uneix les característiques de les xarxes socials i els jocs en una plataforma tecnològica de gran disponibilitat; usa interfícies web per facilitar-ne la màxima accessibilitat, aconseguint que kPax sigui clarament multiplataforma.

La gran difusió de dispositius tecnològics, portables i no portables, que trobem en l'entorn tecnològic actual, en faciliten la connexió i amplien notablement els canals de transmissió de la xarxa, fent-la accessible des de qualsevol plataforma.

kPax té una estructura de nucli de serveis i està construïda sobre el motor de programari lliure per a la construcció de xarxes socials ELGG, que és àmpliament configurable i personalitzable.

En l'actualitat encara no s'han desenvolupat totes les funcionalitats necessàries, però s'han posat de manifest les dificultats que l'actual arquitectura[1] comporta a l'hora d'afegir aquestes noves funcionalitats. Així que s'ha proposat la migració del servidor a una plataforma de desenvolupament que ho simplifiqui, a l'hora que permeti augmentar el rendiment i les prestacions de l'actual entorn.

El projecte que es desenvoluparà en aquest treball de Final de Màster és la migració del *core* de kPax, denominat kPaxServer, a una altra arquitectura, més moderna i eficient a l'hora que més fàcilment ampliable: Node.js[i] amb MongoDB[ii] com base de dades no relacional, en substitució de l'actual nucli implementat sobre Java amb MySQL.

1 Java amb JBoss + Hibernate contra BDD MySQL

INDEX DE CONTINGUTS

1 Introducció

1.1.- Introducció

kPax uneix les característiques dels jocs seriosos com eina d'aprenentatge i la potència de les xarxes socials, alhora que aprofita l'alta disponibilitat de dispositius electrònics actual per multiplicar les oportunitats de divulgació. Amb la finalitat d'aconseguir una gran difusió i participació dels usuaris, pretén fer-se àmpliament accessible usant interfícies web, que faciliten la seva accessibilitat des de qualsevol plataforma o sistema operatiu.

La xarxa social de KPax està programada sobre la plataforma oberta ELGG[iii] per a la construcció de xarxes socials, que és àmpliament configurable i personalitzable.

L'arquitectura de kPax és un nucli o '*core*' basat en serveis API REST, que s'encarrega de mantenir una base de dades amb informacions relatives als jocs, usuaris, desenvolupadors i les seves interrelacions.

El projecte ja està desenvolupat fins a cert nivell, però s'ha observat una dificultat creixent d'integració cada cop que s'han introduït noves funcionalitats al nucli, que han suposat modificacions substancials de la base de dades i del mateix nucli desenvolupats en un entorn relativament rígid i complex.

Per tal de facilitar-ne les ampliacions de les funcionalitats s'ha optat per canviar cap a un entorn amb una arquitectura diferent i més fàcilment ampliable: s'ha decidit realitzar una migració completa a NodeJs del nucli de kPax. La base de dades es migrarà també a MongoDB, que s'integra bé amb NodeJS. Es tracta d'una base de dades no-relacional (no-SQL), que per estar característicament d'orientada a documents i d'esquema lliure, és fàcilment ampliable i molt flexible.

1.2.- Estat de l'art. Objectius.

La part de la xarxa social de kPax està configurada amb ELGG sobre PHP amb MySQL. El Nucli de kPax està sobre un servidor en JBoss programat en Java amb una altra base de dades MySQL.

Les diferents ampliacions de funcionalitats que s'han anat desenvolupant han posat en evidència les mancances d'aquest model, especialment pel que suposa el constant redisseny de la base de dades amb cada evolució, ja que implica constants modificacions i la incorporació de noves taules per adaptar-se a les noves funcionalitats.

Aquests fets han forçat l'evolució del nucli de kPax-Server 01 cap al que serà la seva versió 02, completament renovada, ja que suposa la total reescriptura del codi en un nou paradigma de programació: NodeJS, orientat a esdeveniments i amb un rendiment molt alt per aquest tipus d'aplicacions API REST i MongoDB com base de dades, No-SQL, d'esquema lliure i orientada a documents, que absorbeix molt fàcilment els canvis d'estructura sense haver de repercutir fortament sobre l'estructura ja definida.

Paral·lelament al canvi de kPax-Server, s'aprofita per migrar ELGG a la seva darrera versió i adaptar els *plugins* que puguin haver quedat afectats per la migració.

El producte final serà una plataforma internament renovada per enter i actualitzada al màxim pel que fa a ELGG, que serà el punt de partida de futures incorporacions de noves funcionalitats.

Objectius i preparació del projecte

Prèviament, per poder portar a terme el projecte ha estat necessari:

- Estudiar i comprendre l'estructura actual del nucli de kPax per poder dimensionar la seva migració.
- Estudiar i comprendre la base de dades MySQL del servidor per poder migrar-la.
- Familiaritzar-se en l'entorn de programació de NodeJS + Express.js i amb les tècniques de programació orientades a esdeveniments.

- Familiaritzar-se amb un entorn de base de dades no relacional i comprendre les diferències i equivalències amb l'entorn SQL clàssic.

Així la realització del projecte suposa l'execució dels següents objectius genèrics:

- S'ha aprofitat per adaptar les crides als *endpoints* a la norma REST.
- S'ha reescrit completament el nucli de kPax en NodeJs.
- S'ha reescrit l'estructura de la base de dades adaptant-la a l'estructura d'una base de dades no relacional com MongoDB.
- S'ha migrat la base de dades a MongoDB.
- Finalment, es comptarà amb una serie de proves de validació per confirmar el funcionament idoni del desenvolupament realitzat per si mateix i en el seu entorn operatiu.

1.3.- Estructura de la memòria

L'estructura d'aquesta memòria és la següent:

Introducció. En aquest apartat es realitza una breu introducció del projecte; s'especifiquen breument quins són els objectius a assolir així com es revisa l'estat actual del sistema. Finalment s'inclou aquesta estructura del document.

Estudi de la **viabilitat**. S'especifica quina és la millor de les solucions possibles i perquè s'ha escollit. Es fa una visió de l'abast del sistema així com un repàs de la situació actual, el punt de partida del projecte. Es defineixen els requisits del sistema i valora la solució proposada i les alternatives.

Anàlisi del sistema. En aquest capítol es detalla específicament la solució escollida, definint el sistema, el seu entorn tecnològic, i com es portarà a terme la solució escollida, ja orientada al seu desenvolupament.

Disseny. Presenta els models arquitectònics del sistema i les especificacions i estàndards que es faran servir. S'identifiquen els subsistemes i els seus requisits de llicència o funcionalitat. S'explica el disseny de la base de dades i del servidor kPax i la seva arquitectura.

Desenvolupament. Explica el procés de construcció i desenvolupament del servidor kPax 2.0 i de l'estructura de la nova base de dades en MongoDB.

Implantació. Explica el pas de l'aplicació al seu entorn de producció i de la migració dels sistemes actuals al nou servidor i base de dades.

Manteniment. S'apunta el manteniment i millores a realitzar a la plataforma.

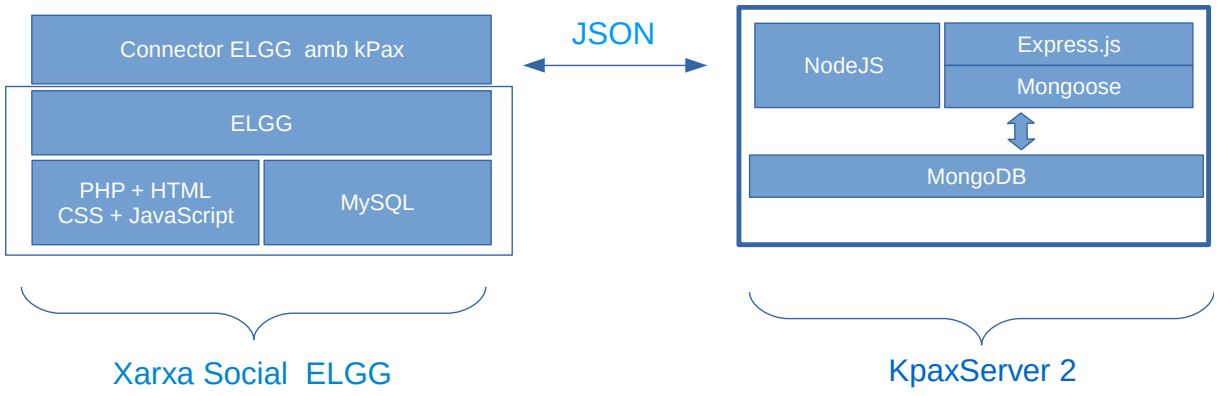
Conclusions. S'exposen en aquest capítol els objectius assolits i aquells que han quedat pendents, així com aquelles propostes de millora o ampliació del projecte.

2 Estudi de la Viabilitat

La plataforma kPax està implementada sobre el motor lliure per a xarxes socials ELGG, que interactua amb un servidor o nucli que treballa sobre JBoss i està programat en Java treballant contra una base de dades MySQL.

Les tecnologies en què està construït kPax dificulten la incorporació de noves funcionalitats sense patir un gran impacte sobre el codi existent; fàcilment s'han de reestructurar les taules de la capa de persistència, aquelles que contenen la informació, o afegir-ne de noves, a més a més d'haver de modificar el model de dades *d'hibernate* per connectar en Java.

A fi de fer més senzilla qualsevol futura ampliació s'ha decidit renovar completament el servidor kPax, creant una nova versió a l'hora que es migrarà a la versió més avançada d'ELGG i s'adaptaran els *plugins* existents que faci falta.



Imatge 2.1: Esquema connexio ELGG amb kPaxServer 02

Es vol crear un servidor de gran rendiment, que sigui fàcilment implementable i ampliable.

Es vol canviar completament el model de base de dades del servidor per tal que les modificacions que comportin les futures ampliacions puguin ser assolides sense un cost inassumible o molt treballós que suposi canvis estructurals.

El nou nucli ha d'interactuar amb la base de dades de la manera més natural possible (per això s'ha pensat en MongoDB, que s'integra de manera molt natural amb NodeJs).

Com que la implantació del sistema kPax Server 01 està encara en una fase alfa, sense implantació, la migració al nou sistema es veu simplificada, perquè tot i que es tracta de mantenir les cridades als *endpoints* en la mesura del possible, també és cert que poden ser modificades a conveniència i aprofitar per adaptar-les al màxim als estàndards REST.

La migració de la base de dades, pel mateix motiu, queda simplificada: no hi ha dades en explotació, per la qual cosa la informació de la nova base de dades pot estructurar-se amb major llibertat i segons les necessitats que vagin sorgint, fins i tot permeten explorar noves propostes estructurals per representar-les.

2.1.- Requisits Econòmics.

Estrictament, no hi ha requisits econòmics que limitin el projecte. Solament els requisits temporals en què s'emmarca el treball de Final de Màster poden ser un requisit limitant: El projecte ha de ser viable dins la durada del quadrimestre que dura el TFM, la qual cosa obliga a ser moderadament ambiciosos en els objectius i deixar deliberadament parts sense desenvolupar, pendents de futurs projectes.

2.2.- Requisits tècnics.

S'ha optat decididament per una tecnologia concreta per realitzar la migració de la versió del servidor kPax 01 a la 02: NodeJS amb Express.js contra MongoDB. Altres tecnologies que podrien quedar-se a mig camí pel que fa a altres objectius, com ara l'ús de NodeJs amb MySQL, tot i ser tècnicament possible, i presentar alguna avantatja, com evitar la migració de la base de dades ja escrita en MySQL i en funcionament, s'ha descartat pel fet que incompliria el requisit de flexibilitat buscat en pensar en una solució de tipus MongoDB, lliure d'esquema i orientada a documents. Tot i l'interès d'evitar una migració de les dades i de l'estructura de la informació de MySQL, que seria bastant interessant, es descarta pel fet que, en gran part, la motivació principal de la decisió del canvi d'arquitectura ve donada per les limitacions estructurals que es deriven quan s'usen models de bases de dades relacionals tradicionals, SQL, que no permeten aquest creixement heterogeni que esperem de MongoDB.

D'aquesta manera, MongoDB es presenta com una solució natural quan es parla d'estructures de dades d'aquest tipus, i a més a més, NodeJS és un llenguatge idoni per a l'escriptura de servidors, altament integrat amb MongoDB; ambdós treballen en JavaScript de manera 'natural' i s'entenen bé amb estructures de la informació (dades) en JSON[iv] . NodeJs ha mostrat un alt grau d'eficiència en servidors d'aquesta mena, especialment per la seva 'orientació a esdeveniments' que li permet processar moltes de peticions al servidor sense els problemes d'escalabilitat que podrien produir-se ràpidament en un servidor tradicional Apache amb PHP.

NodeJs es caracteritza per ser programació asíncrona, específicament dissenyat per estar orientat a 'esdeveniments'; molt indicat pel disseny d'aplicacions no bloquejants d'alta concurrència i disponibilitat. NodeJs treballa sobre un sol fil i s'estima que fàcilment pot mantenir fins a 20000 clients concurrents sobre un servidor tipus, sense que hi hagi gaire retard en la resposta[v].

2.3.- Requisits legals.

No hi ha problemes legals amb les llicències en la tecnologia escollida per la realització del projecte: tant NodeJs com MongoDB són de codi obert, compatible amb les llicències de la resta de components de kPax. El producte final es llicenciarà sota una llicència lliure compatible.

2.4.- Requisits operatius.

El servidor kPax ha d'estar accessible des de qualsevol client que usi estàndards web. Les respostes s'atendran a una sèrie de *endpoints* (diferents URL) que sol·liciten diferents serveis (sovint denominats recursos) del servidor; habitualment operacions del tipus CRUD (Create Read Update Delete) sobre la base de dades, en funció dels diferents paràmetres i de la URL usada.

La resposta a qualsevol petició de tipus GET serà en forma d'objecte JSON cap al client. Les altres accions: POST, DELETE, UPDATE interactuen de la manera adequada amb la base de dades. Totes les interaccions amb el servidor generen la resposta d'estatus HTTP adequada per comunicar al client l'estat de la petició.

S'usen estàndards oberts i àmpliament difosos (JSON) per a la comunicació de dades entre components.

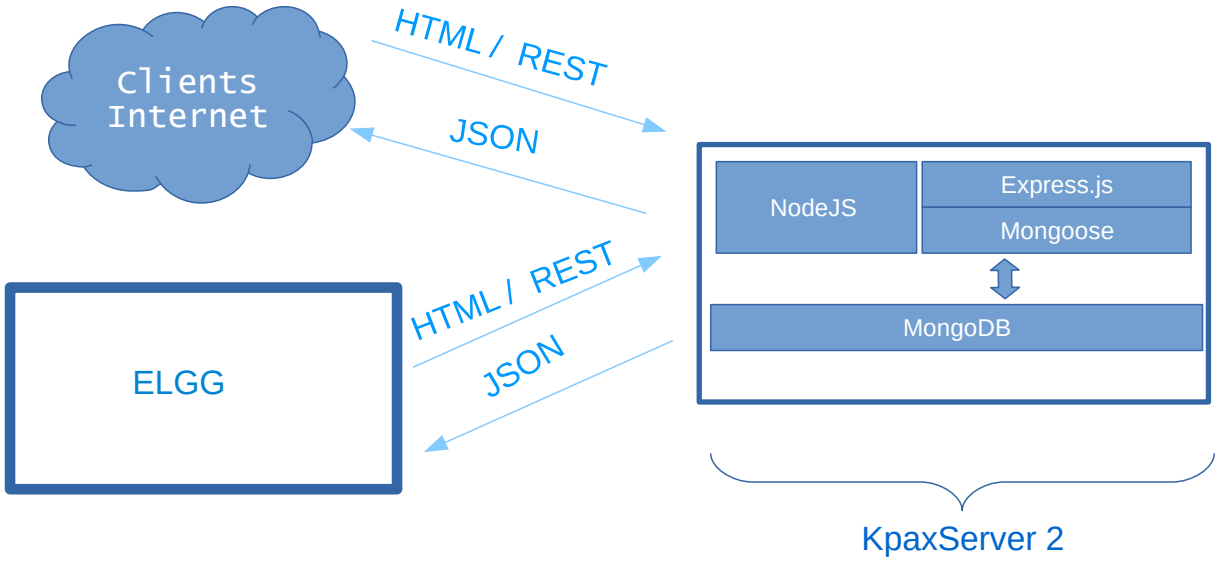
3 Anàlisi

En aquest capítol s'explica el plantejament tecnològic detallat per portar a terme la solució escollida.

3.1.- Definició del sistema

Es tracta de reescriure de nou el nucli del servidor kPax com un servidor API REST usant NodeJS amb Express.js sobre una base de dades MongoDB. El nucli ha de donar servei a clients externs que ho requereixin i s'encarregarà dels aspectes dels usuaris, els jocs els desenvolupadors i les seves interrelacions.

Aquest servidor serà principalment requerit per interactuar amb ELGG, que és qui implementa la part de la xarxa social de kPax. Així i tot, està previst que pugui rebre sol·licituds de qualsevol altre client; en aquest sentit, no està limitat. Aquestes peticions es fan seguint l'estàndard REST, que ens permet crear serveis o aplicacions que poden ser usades per clients o dispositius usant HTTP.



Imatge 3.1: KpapxServer 2 connexions a traves de peticions REST amb clients

3.2.- Requisits

Els principals requisits del servidor kPax s'enumeren a continuació:

R1. El servidor kPax atendrà cridades de tipus REST, és a dir, es crearà una API amb arquitectura REST per donar serveis orientats a internet de manera natural i estàndard.

R2. Seguint l'estàndard totes les peticions es fan a través d'una URL (URI única) que identifica unívocament un recurs del servidor al qual volem accedir per crear-lo, modificar-lo, esborrar-lo o obtenir-lo (CRUD)

R3. Segons quines siguin les peticions, i en funció dels paràmetres i la URL es desencadenaran accions al servidor que interactuaran conseqüentment amb la base de dades.

R4. El servidor atendrà peticions de tipus GET per servir informacions dels usuaris, els desenvolupadors i les seves interrelacions.

R5. El servidor atendrà cridades de tipus POST amb paràmetres per crear o modificar alguna informació dels usuaris, els jocs, els desenvolupadors i les seves interrelacions.

R6. El servidor atendrà cridades de tipus DELETE[2] amb paràmetres per esborrar alguna informació dels usuaris, els jocs, els desenvolupadors i les seves interrelacions.

R7. Les peticions als *endpoints* seran reescrites segons els estàndards REST, canviant les crides antigues quan sigui necessari per adaptar-les a les normes. Es cuidarà especialment que les accions a fer estiguin sempre implícitament expressades amb el tipus de petició (POST, GET , DELETE, ...) evitant-se els verbs a les URI per expressar una acció ja ben definida dins el protocol HTTP.

R8. Les crides GET generaran una recerca i una composició d'una resposta que preferentment es servirà en format estàndard JSON des de el servidor.

2 Finalment s'ha preferit realitzar els recursos d'esborrat usant POST en comptes de DELETE per simplicitat en les peticions HTTP, que en entorns Apache requereixen configuracions especials; en Node.js les peticions DELETE es codifiquen de manera similar a com ho fan les POST

R9. El servidor es programarà fent ús de NodeJs

R10. La base de dades serà MongoDB, que té com principals característiques que es tracta d'una base de dades no relacional, d'esquema lliure , orientada a documents, que permetrà integrar canvis amb facilitat sense necessitat de comprometre l'estructura global de les dades.

R11. Les operacions rebran resposta usant els codis d'estat HTTP[vi] per respondre adequadament a les accions, és a dir, per exemple respondrà amb un «Status Code[3] 201» a la creació correcta d'un nou recurs al servidor. Si a més a més es requereix d'alguna resposta per part del recurs, s'enviarà en format JSON.

R12. Es migrarà la informació de la base de dades MySQL a una estructura convenientment adaptada en MongoDB, de manera que sigui capaç d'atendre les mateixes peticions que ja s'havien configurat pel servidor kPax-Server 01.

R13. Econòmicament no ha de tenir cap cost. Temporalment, el projecte s'emmarcarà en el període semestral de la durada del treball de final de màster. Es desenvoluparan només els endpoints més significatius; aquells que fan referencia als usuaris , jocs , desenvolupadors i relacions bàsiques1.

R14. La llicència del producte final enllestit serà una llicència lliure compatible amb la resta de components de kPax. La compatibilitat de llicències està garantida; cap component de kPax ens força a haver d'usar alguna llicència que pugui originar incompatibilitats.

R15. Operativament, el fet que kPax no estigui desenvolupat més que en fase alfa, no ens obliga a haver de garantir el funcionament en paral·lel dels components que seran reescrits. Tot i així, el desenvolupament del nucli de kPaxServer es prou independent d'altres components que no s'impedirà l'actualització de la plataforma ELGG i els seus plugins simultàniament sense interferir sobre aquella part del projecte, fins i tot en cas que aquesta no es pogués portar a terme exitosament.

3 Veure també annex Error: no se encontró el origen de la referencia

R16. Operativament, Kpax serà accessible des de qualsevol client d'Internet com un recurs REST, realitzant crides als endpoints a través de diferents URL per sol·licitar algun recurs del servidor.

3.2.1 Entorn tecnològic

L'entorn tecnològic del sistema consisteix en:

- Linux, distribució Ubuntu Gnome 15.10. Sistema operatiu usat per programar i on corren NodeJs i MongoDB.[4]
- Entorn amb NodeJS v.6.0.0 instal·lat.
- npm v.3.8.6 (*node package manager*) instal·lat i en funcionament per poder instal·lar components de Node fàcilment
- MongoDB (base de dades): mongod v2.6.10 amb mongoDB shell versió 2.6.10 .
- Components de Nodejs accessibles o instal·lats:
  - express.js : framework de node que incorpora facilitats per programar servidors WEB de manera ràpida.
  - mongoose: mòdul de connexió a mongoDB.
  - Altres components que s'especifiquen a package.json de l'aplicació instal·lables a través de **npm:**
    - body-parser : És el mòdul de nodejs especialitzat en analitzar el cos de les peticions i de les URL, especialment útil per detectar els paràmetres.
    - Jade , actualitzat a Pug. Intèrpret de plantilles HTML que permet usar variables.
    - morgan: mòdul de node per fer un seguiment (log9 de les peticions HTTP al servidor.
    - Debug: Utilitat de depuració senzilla per node.js
    - cookie-parser: mòdul de nodeJS analitzador i manipulador de les cookies de la capçalera HTTP

4 Aquest entorn és l'escollit per portar a terme el projecte, però és igualment possible acomodar un entorn similar en altres sistemes operatius o distribucions de Linux

- Un editor de text per programació, ATOM, per exemple. Està altament integrat amb Git i GitHub.
- GitHub instal·lat per gestionar el control de versions del codi , a través d'un repositori remot sincronitzat adequadament amb el local.
- Eines com ara *curl* per testejar el comportament dels *endpoints*
- Un navegador instal·lat, per exemple Firefox, amb els plugins adequats
  - un *plugin* instal·lat poder realitzar amb comoditat crides tipus POST, GET ... amb paràmetres inclosos, per exemple, POSTMAN .
- Com eina de coordinació '*agile*' usem Trello, que amb el seu sistema de maneig de targetes ajuda a coordinar les tasques de cadascú dins l'equip i a conèixer quines parts del projecte estan en desenvolupament en cada moment, ajudant a mantenir-se informats de l'evolució global del projecte.

3.3.- Pla de proves

Amb el pla de proves podrem establir si el servidor funciona com esperem amb els requisits definits.

3.3.1 Tipus de proves

Proves unàries

Les proves unàries de cadascun dels components funcionant per separat es realitzaran sobre l'entorn de desenvolupament, per verificar que cada mòdul realitza correctament les seves funcions.

Proves d'integració.

Serveixen per testar el funcionament dels components coordinadament. Les realitzarem a l'entorn de desenvolupament, conjuntament amb les proves unàries dels mòduls desenvolupats, que necessàriament han de funcionar coordinats.

Proves de Sistema.

Quan tots els mòduls estiguin integrats es provaran a l'entorn de desenvolupament per verificar el funcionament correcte del sistema com a tal.

Proves d'implantació

Les proves d'implantació es realitzen per testar el sistema al seu entorn d'operació; es realitzaran un cop el sistema estigui funcionant a la plataforma. En aquest moment s'han de realitzar igualment proves d'integració amb el client ELGG..

3.3.2 Definició de les proves.

Proves d'integració. Un cop realitzat cada mòdul i integrat dins el servidor, es realitzarà una prova mitjançant *curl* o el navegador sol·licitant un recurs i es comprovarà que la resposta del servidor és l'adequada i es verificarà que l'acció sol·licitada s'ha realitzat correctament.

Igualment es forçarà a la petició d'un recurs deliberadament inexistent per comprovar que la resposta del servidor és l'adequada.

Proves de sistema. Se sol·licitaran al servidor kPax diferents *endpoints* variant els paràmetres i s'examinarà la resposta. Igualment es testarà una resposta a un conjunt de paràmetres deliberadament erronis per comprovar que el sistema respon correctament sense 'blocar-se'.

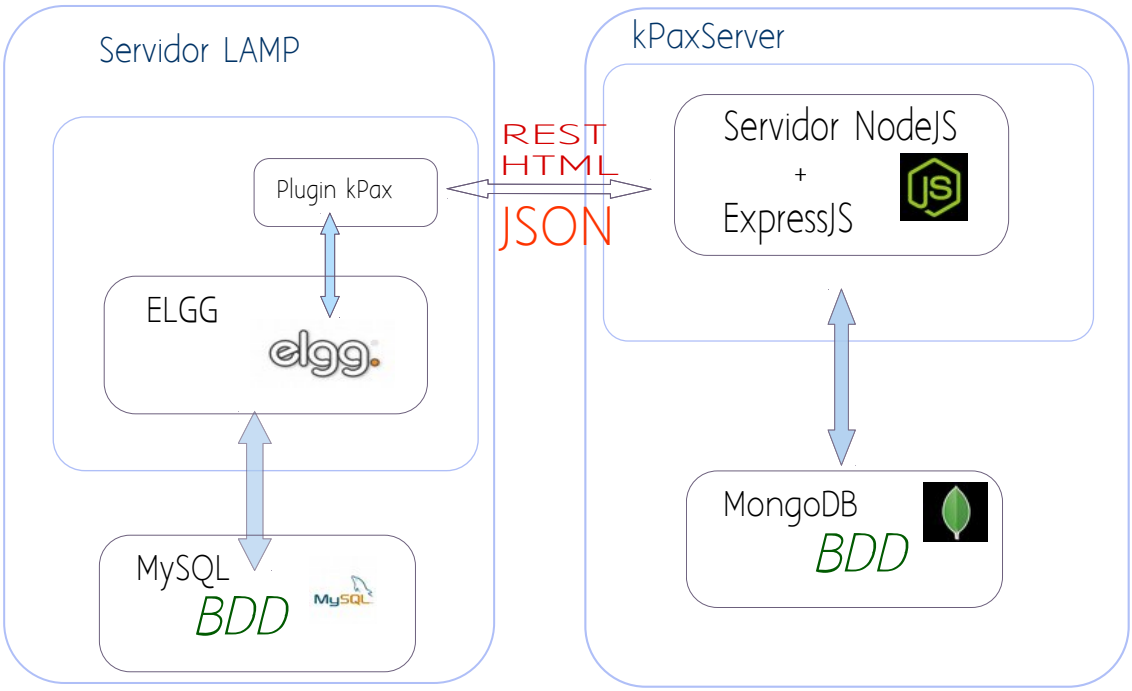
Proves d'implantació. Quan el sistema estigui instal·lat a la plataforma ELGG requerirà els diferents recursos programats i es comprovarà que la resposta és correcta. En cas d'enviament de dades en format JSON, es verificarà que el format de la resposta és l'esperat i que s'acobla perfectament a ELGG.

4 Disseny del sistema

L'objectiu de la fase de disseny és obtenir el model arquitectònic del sistema i les especificacions i els estàndards que es faran servir. Identificarem els components del sistema i els seus requisits d'integració de llicències o funcionalitats.

4.1.- Arquitectura

Identificarem els components del sistema, tant des del punt de vista conceptual com des de la seva perspectiva lògica.



Imatge 4.1: Esquema de Kpax Server 02 amb NodeJS sobre MongoBD i la seva interacció amb ELGG a traves de protocols REST

Pensant en l'escalabilitat i l'accessibilitat del servidor kPax i de la plataforma s'ha desenvolupat una arquitectura basada en NodeJS contra una base de dades no relacional

MongoDB, de manera que totes les peticions de recursos del servidor es faran a través del protocol REST (RESTFul) ; es tracta d'un protocol client/servidor sense estat, amb operacions ben definides que s'apliquen a tots els recursos i que es recolzen en el protocol HTTP. Els recursos estan programats en forma d'endpoints sobre NodeJs; és a dir, cada petició a un recurs es fa a través d'una operació HTTP de tipus PUT, POST, DELETE o GET a una URI (URL) concreta, que el servidor processa i tradueix en l'acció pertinent sobre la base de dades, la capa de persistència.

Totes aquestes peticions s'adrecen a la base de dades en MongoDB per realitzar una intervenció sobre les dades de tipus CRUD, és a dir, afegir, modificar, eliminar o consultar alguna informació continguda en la BDD.

Així, en el servidor s'hi programen el conjunt *d'endpoints* que executen diferents accions sobre la base de dades, tots ells amb una URI única definida atenent a les accions que es volen realitzar, en general posant l'èmfasi en definir-les a través de substantius que millor les representen.

La base de dades MongoDB conté tota la informació de la plataforma kPax: usuaris, desenvolupadors, jocs i les seves interrelacions. En tractar-se d'una base de dades sense esquema definit, afegir un canvi d'estructura a les dades o fins i tot afegir informació en una nova 'col·lecció' [e.g. taula] es fa de manera immediata, sense haver de redefinir o reestructurar la informació ja continguda o haver de detallar-ne la nova. Les modificacions es realitzen al programar els *endpoints* sobre NodeJs, i la base de dades les absorbeix de manera immediata.

4.2.- Components del sistema.

Servidor: aplicació del servidor, programada en NodeJS amb el framework Express, que facilita gran part de les eines per la construcció del servidor.

Base de dades: Capa de dades, sustentada per MongoDB. És d'esquema lliure, orientada a document, que gestiona les col·leccions de manera similar a com ho fa JSON.

4.2.1 Estructura del servidor

La senzillesa conceptual del desenvolupament del servidor es fonamenta en l'ús del framework 'express' amb tota la seva potència. NodeJS es caracteritza pels seus desenvolupaments ben compartimentats en blocs que interactuen. És important tenir en compte la qualitat de NodeJs d'estar orientat a esdeveniments; això s'allunya de conceptes tradicionals de programació de servidors i afegeix un punt de dificultat a l'hora de programar diverses funcionalitats, ja que en tot moment s'haurà de tenir present el fet que en general les respostes són asíncrones. Tot això determina en quin punt del servidor, i en quin bloc s'ha de realitzar determinada acció.

L'estructura del servidor és la bàsica estàndard d'un servidor NodeJS amb Express, creada a partir d'**express-generator** i completada amb **express init,** per acabar de configurar 'package.json', a la que s'ha afegit un mòdul específic per facilitar a connexió a la base de dades, per així incloure les intervencions sobre la 'capa de persistència'.

Ilustración 1: Estructura de directoris d'uns típic servidor NodeJS amb Express

Tot el desenvolupament del servidor es fa dins la carpeta 'server' de la que pengen 4 directoris principals i dins la que hi ha dos fitxers vitals per la configuració de l'aplicació: app.js i package.json.

Dins la carpeta '**bin**' hi ha el fitxer **www** que és el 'main' de l'aplicació que està configurat per executar el servidor. No obstant això, en aquest fitxer no s'hi treballa per configurar el servidor, la configuració de l'aplicació resideix a l'arrel, dins de la carpeta server, en el fitxer **app.js.**

El fitxer **package.json** conté en format json les principals característiques de l'aplicació: nom, versió, fitxer d'inici i en especial la col·lecció de les diferents dependències i les seves versions per poder instal·lar-la correctament a través de **npm** (node package manager), incloent-hi les versions correctes de les dependències escollides.

El fitxer **app.js** es configura especialment a través de funcions de middleware, una de les principals característiques de NodeJS, és a dir, es tracta de funcions que accedeixen directament a l'objecte sol·licitud, a l'objecte resposta i a la següent funció de middleware en el cicle de sol·licitud/resposta de l'aplicació. Aquest fitxer

- Connecta amb la base de dades.
- Carrega els diferents complements a través d'instruccions de tipus 'require'.
- Configura el directori 'views' per indicar que contendrà les vistes o pàgines html, codificades en 'jade' i que en aquest projecte no farem servir.
- Afegeix la configuració de les 'routes' que servirà per definir els endpoints i el seu comportament.

La carpeta '**routes'** és una de les més importants pel desenvolupament del servidor: en ella es desenvoluparan cadascun dels *endpoints:* l'accés als recusos del servidor, les intervencions amb les dades i les respostes esperades.

Els recursos del servidor son requerits per realitzar intervencions sobre els usuaris o sobre els jocs del sistema. Així es distingeixen dos grans blocs en les 'routes' per crear els endpoints: 'games' i 'user'.

Dins 'games.js', inclòs a la carpeta 'routes' hi programarem tots els endpoints referents a accions sobre els jocs.

Paral·lelament, dins 'user.js' hi programem tots aquells endpoints les accions dels quals tenen a veure amb els usuaris.

Aquests endpoints en node es defineixen com a elements de l'objecte Router d'express, expressant per cadascun quina és la URI i el 'mètode d'accés' HTTP (GET, POST, ...) que el

desencadena. Cadascun realitza les accions requerides i retorna la resposta a través de l'objecte response del middelware.

4.2.2 La Base de Dades no relacional

Anteriorment ja hem comentat que la base de dades és d'esquema lliure i orientada a documents. Aquestes característiques permeten que ens puguem desentendre d'haver de realitzar cap intervenció directa sobre la base de dades, ja que serà el servidor interactuant amb la base de dades qui de manera dinàmica anirà creant, modelant i gestionant les dades. No és necessari crear cap estructura per poder manipular dades, el gestor de base de dades incorpora qualsevol document a la col·lecció apuntada, encara que la seva estructura no s'hagi definit o difereixi de la resta de documents d'aquesta col·lecció; dit d'una altra manera, cada document d'una col·lecció internament té la seva pròpia estructura, i no és necessari predefinir-la, ni mantenir una homogeneïtat estructural entre els diferents documents d'una col·lecció. És més, si una col·lecció no existeix, es crea en la seva primera *instanciació*.

Les col·leccions de documents mixts, amb diferent estructura, poden conviure sense error perfectament dins una mateixa col·lecció. Si es creu necessari, es pot pensar en una intervenció manual que ens porti a homogeneïtzar els documents d'una col·lecció però en general no és necessari, i es pot funcionar amb documents mixts sense error per part del gestor de base de dades.

4.2.2.i El disseny de la BDD

El disseny de les bases de dades de MongoDB es basa en col·leccions de documents. En el nostre cas definirem dues grans col·leccions: Usuaris (users) i Jocs (games).

El disseny. Recordem que el disseny no s'ha de fer de manera explícita, ja que es va formant a mesura que es realitzen 'inserts' dins la col·lecció. En aquest sentit, es delega als 'endpoints' del servidor, que realitzen les funcions d'agregar documents a les col·leccions a través dels 'inserts'; en fer-ho amb determinada estructura i tindrem la seguretat que aquesta estructura es trasllada automàticament a la base de dades.

Seguidament es detalla l'estructura de les dades de les dues col·leccions que s'ha formalitzat per aquesta fase del projecte. Canviar-la, si fos necessari, és molt senzill, sense comprometre seriosament cap part del desenvolupament realitzat fins al moment.

Estructura de **games**

```
GAMES
{
// basic information
_id     :       {ObjectId}
name    :       {String}              this is the name of the game
owner   :       {Objectid}            ID of the owner (users._id)
status :        {Integer}             0: unavailable,
                                      1: available,
                                      3:deleted
public : {Boolean}                    true if the game is public

// likes / dislikes
nlikes :        {Int}                 number of likes
ulikes : [                            list of users that like (user._id)
        {
                userId : {ObjectId}
                date : {Date}
        }
        ]

// scores
uscores : [                           number of scores (user._id)
        {
                userId :        {ObjectId}
                score :         {Int}
                date :  {Date}
        }
        ]

created_at : {Date}           when game was created
```

```
updated_at : {Date}          last time game was updated
}
```

Estructura d' **user**:

```
USER
Main document

{
        // basic information
        _id     : {ObjectId}          id of the user in MONGO
        login   : {String}            user login (possible id of the user in elgg)
        name    : {String}            this is the full name of the user
        games   : [                   list of own games (games._id)
{ObjectId}
        ]

        created_at     : {Date}       when user was created
        updated_at     : {Date}       last time user was updated
        status :       {Integer}      0: unavailable,
                                      1: available,
                                      3:deleted
}
```

4.3.- Normes i notacions de la documentació

Per realitzar la documentació de les API de jocs i d'usuaris s'han fet servir documents de Google Docs amb permisos de lectura i d'edició per part de tots els membres de l'equip, per fer possible la col·laboració.

Aquests documents tracten sobre les funcionalitats dels 'endpoints' ja existents a la versió 1 de kPaxServer i quina és la seva transformació a kPaxServer 2.

Cada document té el següent format:

- Mètode HTTP per accedir al recurs (GET, POST...)
- URI (URL)

    - L'antiga de KpaxServer1
    - La proposta de kPaxServer 2, adaptada a REST
- Descripció del propòsit de l'endpoint; què fa.
- Nom, tipus i descripció dels diferents paràmetres per cridar el recurs.
- Sortida (si n'hi ha)
    - Explicitació d'un exemple de sortida JSON esperada per aquests recursos
- Vincle amb Kpax1
    - URL de crida dels recursos
    - Mètode: GET, POST ....
    - Codi Java referenciat pel recurs.

4.4.- Casos d'ús.

Els casos d'ús del servidor estan íntimament lligat a les diferents crides als recursos del servidor i a les diferents respostes que serveix. Així els casos d'ús més comuns coincideixen amb cadascun dels '*endpoints*' definits al servidor:

- Alta d'un joc, amb totes les seves característiques.
- Alta d'un usuari, amb totes les seves propietats.
- Esborrat d'un joc.
- Esborrat d'un usuari.
- Llistar informació de tots els usuaris
- Llistar informació de tots els jocs.
- Llistar informació d'un joc determinat, identificat per codi del joc.
- Llistar informació d'un usuari determinat, identificat per codi de l'usuari.
- Llistar informació dels usuaris a partir d'un paràmetre lliure, configurable.
    - Llistar informació d'un usuari determinat, identificat pel nom de l'usuari. (resolt amb l'anterior

  - Llistar informació d'un usuari determinat, identificat per l'e-mail, 'correu electrònic', de l'usuari. (resolt amb el paràmetre lliure)
- Llistar informació dels jocs a partir d'un paràmetre lliure, configurable. Permet la realització de consultes obertes, en format JSON de mongoDB.
- Marcar un joc amb *like*: incrementar el comptador de *likes* i enregistrar la informació de l'usuari i la data i hora del *like*. Només el primer cop que es fa el like, si ja hi ha registre, no es repeteix i no s'incrementa el marcador.
- Desmarcar el joc si estava marcat com '*like*', descomptar un al comptador de *likes* i esborrar la informació de l'usuari relativa al *like* d'aquest joc. Només actua si hi havia un *like* marcat anteriorment per aquest usuari sobre ell joc.

4.5.- Recursos (endpoints) definits.

Al servidor s'han desenvolupat dues APIs fonamentalment: l'API d'usuaris i l'API de jocs. A continuació s'expliquen breument els recursos (endpoints) definits en cadascuna d'elles, els seus paràmetres i mètode HTTP per realitzar les crides:

usuaris: API user

- Afegir nou usuari al sistema

URL	**/user**
METHOD	POST
PARAMS	• name (en body) Nom usuari Required • login (en body) login email required
OUT	Cap

- Llistar tots els usuaris del sistema

URL	**/user/listall**
METHOD	GET
PARAMS	Cap
OUT	JSON registres d'usuari

- Llistar usuaris, paràmetre configurable

URL	**user/list**
METHOD	GET
PARAMS	• q — Parametre obert en format filtre JSON de MongoDB — Optional
OUT	JSON registres d'usuari que compleixen condició q

- Llistat d'informació d'un usuari , per codi d'usuari

URL	**/user/{user}**
METHOD	GET
PARAMS	• user (en URL) — ID usuari — Required
OUT	JSON registres d'usuari que té _id usuari sol·licitada.

- Esborrar un usuari del sistema

URL	**/user/del**
METHOD	POST
PARAMS	• user (en body) — ID usuari — Required
OUT	Cap

Jocs: API game

- Afegir nou joc al sistema

URL	**/game**
METHOD	POST
PARAMS	• name (en body) — Nom joc — required • owner (en body) — propietari del joc — required
OUT	Cap

- Llistar tots els jocs del sistema

URL	**/game/listall**
METHOD	GET
PARAMS	Cap
OUT	JSON registres d'usuari

- Llistar jocs, paràmetre per condició configurable

URL	**game/list**

METHOD	GET
PARAMS	• q Parametre obert en format filtre JSON de MongoDB Optional
OUT	JSON registres dels jocs que compleixen condició q

- Llistat d'informació d'un joc , per codi de joc.

URL	**/game/{game}**
METHOD	GET
PARAMS	• game (en URL) ID joc Required
OUT	JSON registres dels jocs que té la _id usuari sol·licitada.

- Esborrar un joc del sistema

URL	**/game/del**
METHOD	POST
PARAMS	• game (en body) ID joc Required
OUT	Cap

- Afegir LIKE a un joc per part d'un usuari

URL	**/game/{:game}/like**
METHOD	POST
PARAMS	• game (en URL) ID joc required • user (en body) ID Usuari required
OUT	Cap
Notes	El primer cop incrementa el comptador de likes en un, i enregistra id de l'usuari i de la data i hora del registre. Si l'usuari ja havia enregistrat, el LIKE no fa res.

- Demarcar LIKE a un joc per part d'un usuari

URL	**/game/{:game}/unlike**
METHOD	POST
PARAMS	• game (en URL) ID joc required • user (en body) ID Usuari required
OUT	Cap

Notes	Si l'usuari no havia marcat el LIKE, no fa res. Si l'havia marcat, descompta un al comptador de likes i esborra la informació de l'usuari de uLIKE

5 Desenvolupament

5.1.- L'entorn de desenvolupament

En aquest punt explicitem quines són les característiques de l'entorn de desenvolupament escollit per portar a terme el projecte de construcció del servidor kPaxServer2.

5.1.1 Sistema operatiu . Entorn de treball.

Per realitzar el desenvolupament s'ha escollit un entorn amb sistema operatiu Linux, concretament Gnome Ubuntu 15.04, que ja compta amb moltes facilitats per instal·lar posteriorment la resta d'eines de programació.

5.1.2 Eines de programació

```
miquel@ubuntu1510:~$ mongod --version
db version v2.6.10
2016-05-29T22:59:03.068+0200 git version: nogitversion
2016-05-29T22:59:03.068+0200 OpenSSL version: OpenSSL 1.0.2d 9 Jul 2015
miquel@ubuntu1510:~$ mongo --version
MongoDB shell version: 2.6.10
miquel@ubuntu1510:~$ npm --version
3.8.6
miquel@ubuntu1510:~$ node --version
v6.0.0
miquel@ubuntu1510:~$ npm --version
3.8.6
miquel@ubuntu1510:~$
```

Ilustración 2: Versions d'algunes eines de programació i treball

Enumerem a continuació les principals eines de programació, la versió usada i el seu ús, així com la manera d'instal·lar-les en l'entorn de treball escollit.

Npm

Instal·lem npm versió 3.8.6 (Node Package Manager) com eina de maneig de complements de node. A través de npm es poden instal·lar gran nombre dels complements de NodeJS i a més a més es poden realitzar algunes altres funcionalitats interessants, com ara, inicialitzar el fitxer package.json (npm init) o iniciatlitzar l'aplicació (npm start) si està ben configurat el fitxer package.json.

És molt útil per a la instal·lació de components i a través del fitxer package.json amb npm install app permet la instal·lació correcta de l'aplicació amb totes les seves dependències ben configurades.

Per instal·lar-lo en un entorn de treball Ubuntu es suficient usar el següent comandament des de la terminal

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

NodeJS

Llenguatge de programació escollit per desenvolupar el servidor. És una biblioteca i un entorn d'execució orientat a esdeveniments (asíncron) que s'executa sobre l'interpret de Javascript creat per Google v8.

Instal·lem la darrera versió de NodeJS: v6.0.0 . En l'entorn descrit la instal·lació es pot fer des de la terminal amb

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

Sovint és recomanable afegir-hi le eines de compilació d'Ubuntu per poder compilar alguns *addons* de npm :

des de la terminal ho fem amb

```
sudo apt-get install -y build-essential
```

En distribucions com Ubuntu és recomanable instal·lar també el denominat *nodejs-legacy package* que instal·la un vincle simbòlic que molts de mòduls necessiten per poder-se compilar i funcionar correctament.

MongoDB

Base de dades no relacional d'esquema lliure orientada a documents, que es guarden en BSON (representació binària de JSON) No necessita tenir un mateix esquema per a tots els documents d'una mateixa col·lecció .

Per instal·lar-la en Ubuntu 15.10 de 64bits, des de la terminal es poden seguir les passes següents:

- Importar la clau publica:

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

- crear un fitxer llista per MongoDB i actualitzar

```
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
```

- Instalar els paquets de MongoBd

```
sudo apt-get install -y mongodb-org
```

La versió instal·lada és la v.2.6.10 tant per mongod (el motor de la base de dades) com per mongodb shell (la interfície de mongoDB pel terminal)

Imatge 5.1: Versions de mongod i el shell de mongoDb usades

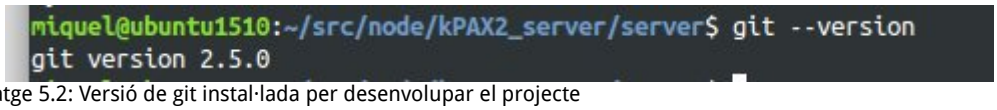
Git

Eina de control de versions, que es coordina bé amb el repositori gitHub usat per salvaguardar còpies del projecte i per posar-lo a disposició de la comunitat a l'hora que a la resta dels desenvolupadors de l'equip per poder treballar de manera coordinada.

Per instal·lar-ho en Ubuntu basta executar des de la terminal

```
sudo apt-get install git
```

La versió instal·lada és la 2.5.0. Amb aquesta eina podrem treballar fàcilment amb els repositoris remots de GitHub i sincronitzar-los amb els repositoris locals.



Imatge 5.2: Versió de git instal·lada per desenvolupar el projecte

La manera de treballar escollida és realitzar un fork del repositori principal del projecte i crear-ne una rèplica «fork» al repositori remot propi. Sincronitzem el repositori «remot propi» amb el «local de treball» a través d'un `git clone url/repositori/propi/remot` i ja el tenim disponible per desenvolupar. Els arxius a controlar s'afegeixen amb `git add «fitxer»` i finalment es poden anar realitzant commits de diferents punts del desenvolupament del codi font. És recomanable afegir un comentari explicatiu als commits aclarint quines noves funcionalitats s'han afegit.

És comú dividir el projecte en branques, sobre les que es treballen algunes característiques o novetats mantenint intacta la branca mestra (`master`) i que posteriorment es pot fusionar si les modificacions es consoliden o en cas contrari es pot abandonar la branca i seguir el desenvolupament per una altra.

Totes aquestes modificacions i branques es poden sincronitzar amb el repositori remot propi a través de `git push ...` cap al repositori remot de la branca o branques que es desitgi sincronitzar.

Un cop el repositori remot propi ja ha assolit un cert grau de desenvolupament i solidesa es pot sol·licitar incorporar les modificacions al repositori remot principal a través d'un `pull request`, que necessitarà ser acceptat per l'administrador del repositori principal.

En el desenvolupament del projecte l'ús de git amb connexió als repositoris de gitHub ha estat intensa i ha contribuït a fer possible la col·laboració entre els diferents components de l'equip i la coordinació final amb la part de la xarxa social de kPax amb ELGG.

El treball individual, l'he coordinat amb el meu repositori de gitHub, denominat 'origin' i que parteix d'un fork inicial del projecte global de kPax , que s'ha denominat 'upstream'.

Així, la mecànica de treball ha consistit en treballar en varies branques: la *màster* pel contingut més estable (definitiu) i una altra branca *devel* pensada pel nou desenvolupament. Un cop s'ha desenvolupat qualsevol petit mòdul (en general cadascun dels endpoints o altres modificacions) s'han realitzat commits locals per desar els canvis. Regularment aquests commits s'han pujat al meu repositori remot 'origin' amb un '`git push origin devel`' quedant una còpia del projecte (en el seu estat de desenvolupament) a gitHub, a disposició de qualsevol que hi pugui estar interessat i funcionant a l'hora com còpia de seguretat.

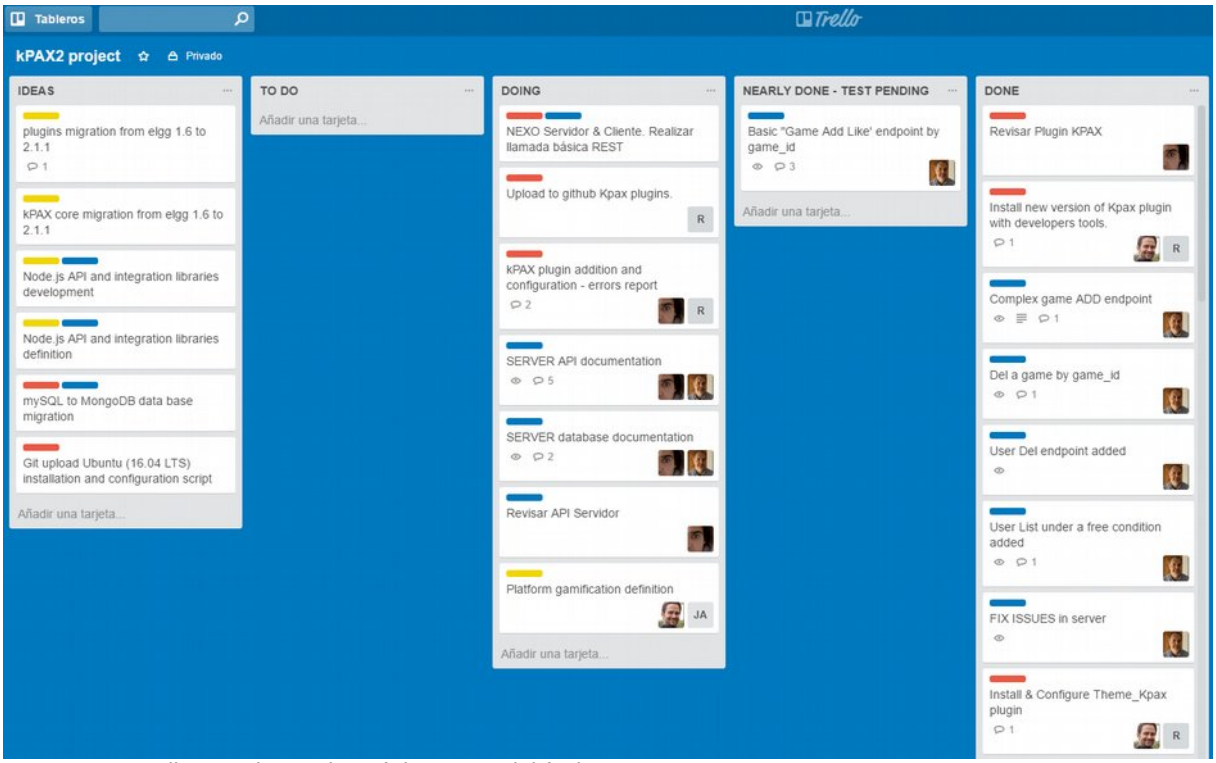
Quan el contingut del repositori origin ha tingut suficients novetats i ha començat a ser operatiu, l'hem anat sincronitzant (barrejant) amb el repositori inicial 'upstream' a través de *pull requests*, que necessiten ser confirmats pel propietari del repositori. El resultat és que un cop confirmat el *pull request* i resoltes les inconsistències detectades, el repositori 'upstream' conté la versió més avançada del projecte amb totes les contribucions de tots els desenvolupadors. Si hi ha hagut altres contribucions al repositori 'upstream' és convenient i necessari realitzar un '`git rebase .. `' o bé '`git merge upstream/devel`' per incorporar-les i seguir desenvolupant coordinadament.

Trello

Eina d'internet que permet la coordinació d'equips de treball mitjançant l'ús de targetes. Un dels membres de l'equip convida als altres a formar part d'un projecte comú. En el taulell del projecte es disposen per columnes targetes indicant tasques; cada columna simbolitza un estat de la idea, concretament n'hem usat els estats:

- idea,
- pendent,

- en marxa,
- fet pendent de revisió final i acceptació
- fet



Imatge 5.3: Trello: Eina de coordinació d'equips amb l'ús de targetes.

5.2.- Entorn de treball

L'entorn de treball pel desenvolupament és un ordinador en Linux i amb les eines explicades en el punt anterior instal·lades:

- **npm** : Manejador de paquets de nodeJs i altres utilitats menors.
- **NodeJS amb express**: Llenguatge de programació i l'entorn d'execució de programes
- **MongoDB**: BDD i client de MongoBD accessible des de la terminal per gestionar la base de dades.

- **Git** : eina de control de versions.
- **Atom** : editor de text per desenvolupar en el client
- **curl:** Permet connectar amb servidors a través de la seva URL; suporta la majoria dels protocols més comuns: http, https, ftp , sftp, imap ... De gran utilitat per comprovar que les peticions als recursos del servidor realitzen de la manera esperada, analitzant les seves capçaleres i continguts, així com les accions realitzades a la base de dades.
- **Robomongo.** Proporciona un entorn de caràcter gràfic, més potent i còmode que la terminal per accedir a les bases de dades de MongoDB; és especialment interessant per poder visualitzar còmodament els resultats de les consultes en diferents formats de sortida més fàcilment llegibles.
- **Poster.** Plugin de Firefox que proporciona un entorn gràfic per realitzar operacions semblants a les de curl sobre el servidor i comprovar les respostes.
- **Postman**. Plugin de chrome per realitzar tasques semblants a les de Poster.

6 Implantació

La implantació del servidor és gairebé la mateixa que es realitzaria en l'àmbit local, però es realitza sobre l'entorn de producció.

L'entorn de producció té instal·lades les versions apropiades de nodeJS , mongoDB:

- versió v.6.0.0 per NodeJS
- versió v2.6.10 de mongod

La instal·lació es realitza a partir del paquet contingut al gitHUB a la branca *master*, https://github.com/drierat/kPAX2_server que conté la versió consolidada, que es desempaqueta a la carpeta que contindrà el servidor.

Amb **npm install** s'instal·len automàticament els components i dependències necessàries per poder executar el servidor. Aquests components són els que s'especifiquen al fitxer package.json :

```
...
  "dependencies": {
    "body-parser": "^1.13.3",
    "cookie-parser": "^1.3.5",
    "debug": "^2.2.0",
    "express": "~4.13.1",
    "jade": "^1.11.0",
    "mongodb": "^2.1.18",
    "morgan": "^1.6.1",
    "pug": "*",
    "serve-favicon": "^2.3.0"


  }
}
```

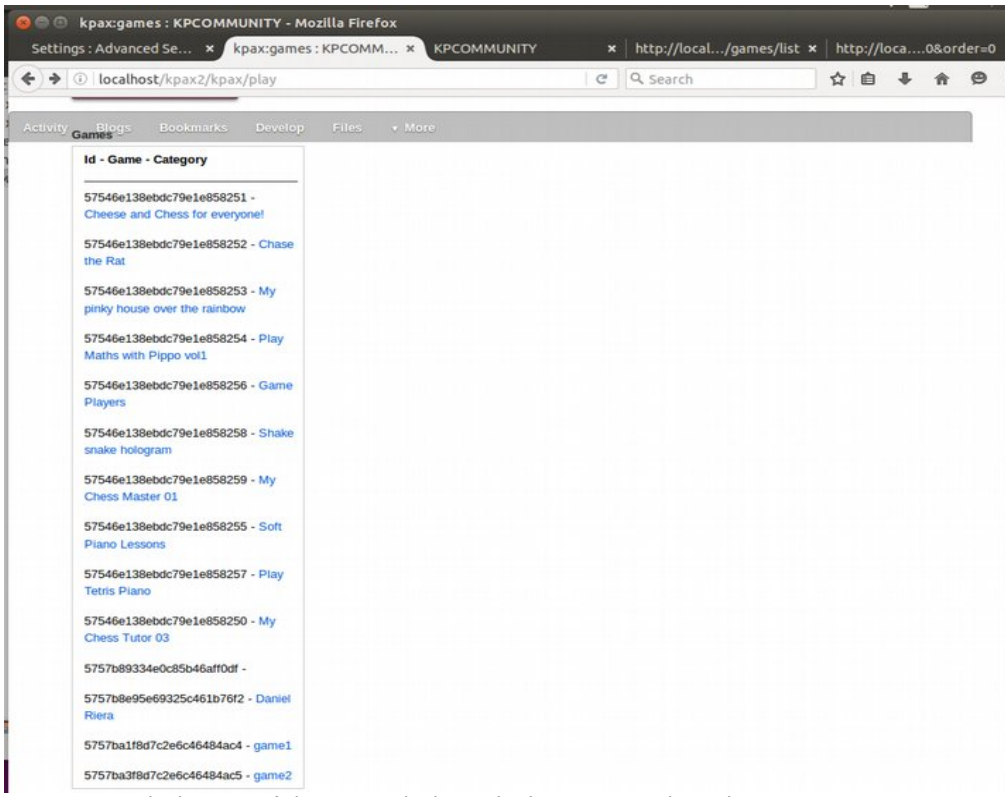
Per iniciar el servidor ho podem fer amb

```
MONGODB_URL="mongodb://readwrite:xxxx@ds021462.mlab.com:21462/kpax2" bin/www
```

on la variable d'entorn MONGODB_URL conté la cadena de connexió a la base de dades; si no s'especifica aquesta variable d'entorn, el servidor es connecta a la base de dades local.

Un cop arrancat el servidor, ja està disposat per atendre peticions als recursos.

A l'entorn de producció també s'ha configurat el servidor de ELGG i està configurat per enviar peticions a kPaxServer2, que d'acord a les sol·licituds rebudes interactua amb la base de dades i envia la resposta adequada si s'escau.



Imatge 6.1: Exemple d'interacció de ELGG amb el servidor kPaxServer2. Llistat de Jocs.

A la imatge adjunta podem veure un exemple de com el plugin d'ELGG crida al servidor de Kpax i com processa la informació rebuda.

7 Manteniment. Possibilitats de millora.

Amb el projecte actual s'ha aconseguit desenvolupar un servidor completament funcional pel nucli de kPax, reescrit des de zero usant una nova arquitectura per a la seva programació, NodeJs, així com la completa migració de la base de dades a MongoDB.

Tot i així, per les limitacions temporals que emmarquen el TFM, aquest projecte representa només l'embrió del nou servidor. S'han desenvolupat les principals funcionalitats, els recursos bàsics i s'han assentat les bases pel desenvolupament de nous, que simplement, inspirant-se en el treball realitzat poden avançar sobre les fites marcades per aquest projecte.

Les possibilitats de millora, pel fet d'haver desenvolupat el nucli completament des de zero, són múltiples: queden moltes tasques per realitzar i moltes vies per explorar i completar el servidor actual.

La més immediata pot ser dotar d'una capa de seguretat el servidor actual, de manera que es permeti realitzar determinades accions sols a aquells usuaris autoritzats.

Per altra banda el desenvolupament de nous endpoints pot continuar de manera que cada cop es doni resposta a aspectes més específics per millorar la interacció amb ELGG, per exemple, qüestions sobre els usuaris desenvolupadors i els seus jocs i les valoracions dels jocs; introduir els conceptes de partides entre usuaris i lligues de jocs; explorar el concepte d'equips de jugadors, de reptes a altres jugadors o equips... en definitiva s'obre per davant tot un ampli ventall de possibilitats, tant de migració de les antigues funcionalitats com del desenvolupament de noves, de les que el servidor desenvolupat en aquest projecte n'és el punt de partida.

La nova arquitectura del servidor i de la base de dades ha mostrat la seva flexibilitat i adaptabilitat per incorporar canvis durant el desenvolupament del projecte actual: alguns endpoints s'han modificat i conseqüentment la informació base de dades associada, de manera gairebé indolora. Aquesta és, sens dubte, la gran avantatja d'haver migrat a l'entorn actual, que posa de manifest que s'ha assolit l'objectiu de simplificar la incorporació de

millores i predisposa a una evolució constant de les prestacions del servidor, fins a arribar a una fase madura.

8 Conclusions

Els objectius del projecte, centrats en la implantació d'un servidor reescrit per complet en la nova arquitectura Node.js i la migració de la seva base de dades a un entorn No-SQL, representat per MongoDB, ha estat assolit. Al final del projecte es té un servidor completament funcional, connectat a una BDD MongoDB amb informació bàsica dels usuaris i jocs i algunes interaccions definides. S'han desenvolupat les APIs de jocs i d'usuaris amb recursos per gestionar-ne tant uns com els altres i per realitzar consultes diverses d'ambdues entitats (col·leccions). Tot i que queda bastant de camí per recórrer, aquest TFM ha posat les bases del servidor kPaxServer2, desenvolupant una estructura d'endpoints bàsics, sobre els que més endavant es poden desenvolupar noves funcionalitats.

La migració del servidor i de la base de dades no ha estat realment tan impactant com si s'hagués trobat el sistema en producció; el fet de refer el servidor i la base de dades sense haver d'atendre un sistema funcionant en paral·lel, que no pot ser interromput i del qual s'han de respectar les seves dades, ha estat una avantatja significativa. Per exemple, això ha permès modificar l'estructura de les dades per adaptar-la a MongoDB i les crides als recursos s'han pogut adaptar millor als estàndards REST sense que hagi suposat un problema d'incompatibilitats amb les peticions anteriors.

Pel que fa a la base de dades, cal dir que no s'ha hagut de realitzar una migració efectiva de les dades, sinó que s'ha començat a modelar l'estructura sobre MongoDB segons les conveniències actuals, usant l'antiga estructura de les dades com referència més que com imposició. Al llarg del projecte aquesta estructura ha patit algun canvi, i s'ha demostrat l'avantatja d'usar una base de dades com MongoDB per la seva flexibilitat, ja que els canvis han estat absorbits directament per la base de dades, sense cap incompatibilitat i simplement s'han reflectit a la part del codi que les ha de tractar.

El projecte es tanca en aquest punt pel fet que el temps del TFM s'ha esgotat i el projecte s'ha de finalitzar. Arribats en aquest punt, seria senzill seguir desenvolupant alguns endpoints

més, com hem descrit anteriorment, simplement basant-nos en els que ja hi ha fets, en els que es resolen gran part dels problemes que pot presentar.

Durant la realització d'aquest TFM s'ha estat en permanent contacte a través de reunions virtuals setmanals, el Trello i el repositori gitHub amb la factoria d'un altre projecte de TFM relacionat amb kPax i que suposa la migració d'ELGG a la seva versió més moderna i l'adaptació dels pluguins per finalitzar amb la interconnexió d'ambdós projectes: el mòdul ELGG contra el servidor desenvolupat en aquest projecte. Aquest objectiu s'ha assolit també amb èxit: el mòdul d'ELGG interactua amb el servidor kPaxServer 2 de manera satisfactòria.

En conclusió, els objectius marcats pel projecte: desenvolupar el nou servidor en NodeJs contra una base de dades en MongoDB, fàcilment adaptable a noves funcionalitats, i capaç de donar servei al mòdul de la xarxa social de kPax, el mòdul d'ELGG han estat assolits amb èxit.

El projecte es deixa en un estat idoni per madurar, apte per anar incorporant noves funcionalitats sense grans reestructuracions que ho facin inviable però a l'hora el que s'ha desenvolupat és perfectament funcional i suficientment desenvolupat per posar-se en explotació sense més intervencions immediates que la incorporació d'una capa de seguretat per controlar els accessos i canvis a la informació que proporciona el servidor.

Glossari

i **Node.js** és un entorn de programació dissenyat per escriure aplicacions d'Internet escalables, notablement servidors web.[1] Els programes estan escrits en JavaScript, utilitzant una arquitectura orientada a esdeveniments, i entrada/sortida asíncrona per tal de minimitzar el temps de sistema i maximitzar l'escalabilitat.[2] Node.js consisteix en el motor de JavaScript V8 de Google i de diverses llibreries incloses. (Font Viquipèdia: https://ca.wikipedia.org/wiki/Node.js)

ii **MongoDB** és un programari de codi obert, per a la creació i gestió de base de dades orientada a documents, escalable, d'alt rendiment i lliure d'esquema programada en C++. La base de dades és orientada a document d'aquesta manera gestiona col·leccions de documents similars al format de dades JSON. (Font: Viquipèdia: https://ca.wikipedia.org/wiki/MongoDB)

iii **ELGG** és una plataforma de serveis de xarxa social de codi obertque ofereix la creació de blogs, treball en xarxa, comunitat, recol·lecció de notícies via feeds i intercanvi d'arxius. Tot pot ser compartit entre els usuaris, utilitzant els controls d'accés i pot ser catalogat mitjançant etiquetes (Font: Viquipèdia: https://ca.wikipedia.org/wiki/Elgg, veure també : https://elgg.org/features.php)

iv **JSON: J**ava**S**cript **O**bject **N**otation. JSON és un format d'intercanvi de dades lleuger, independent del llenguatge de programació i auto-descriptiu, bo d'entendre. (sic. JSON is a lightweight data-interchange format, language independent and "self-describing" and easy to understand.)(http://www.w3schools.com/json/) També http://json-schema.org/latest/json-schema-core.html.

v Óscar Ray - conceptos que deberías conocer sobre Node.js .
http://unadocenade.com/una-docena-de-conceptos-que-deberias-conocer-node-js/

vi **Codis d'estat HTTP**: 1xx per respostes informatives; 2xx per peticions correctes; 3xx per redireccionaments; 4xx per errors del client; 5xx per errors del servidor; font Wikipedia: https://es.wikipedia.org/wiki/Anexo:CC3%B3digos_de_estado_HTTP , veure també https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html