Treball Final de Màster

Especialitat: web i comerç electrònic
Consultor: Francisco Javier Noguera Otero
Tutor extern: Daniel Riera i Terrén

Alumne: Sergio Morente Fernández
Data: 22/01/13

Resum

En aquest treball es proposa l'elaboració d'un tema adaptat especialment per a la plataforma de jocs seriosos *kPAX*[1]*.* S'estudiarà breument el funcionament del seu gestor de continguts, el CMS *Elgg*[2]*,* amb l'objectiu de modificar el codi CSS generat pel seu nucli i altres connectors *(plugins)* per incloure millores en el rendiment i a la vegada mostrar un nou aspecte al portal seguint criteris de bones pràctiques d'usabilitat.

Quan es vol construir un espai web amb continguts dinàmics el més habitual és recórrer a programes gestors de continguts o *CMS* (de l'anglès *Content Management System)* que s'encarreguin de generar el codi que s'enviarà al client. La raó del seu èxit es deu a que aquests programés simplifiquen considerablement la gestió i l'actualització de la informació, i a més aporten mecanismes de control d'accés. Però aquests programes no són perfectes. Aporten una solució genèrica i sovint excessiva en el consum de recursos, o bé no fan ús de les tecnologies més recents per por a perdre la compatibilitat cap enrere.

En aquest treball estudiarem el CMS Elgg, el qual es basa en un conjunt d'scripts PHP que s'executen en el servidor i s'encarreguen de generar tot el codi amb el contingut i que posteriorment s'enviarà a l'usuari. D'aquest codi ens centratem especialment en CSS (Cascading Style Sheets), el codi que permet controlar la presentació i que per tant, ha de ser la part central d'un tema. Veurem com generar un tema com si fos un connector més per Elgg, una capa de programari capaç de modificar les normes d'estil del nucli i d'altres plugins actius. I tot sota criteris estrictes de qualitat, eficiència i compatibilitat amb tota mena de dispositius susceptibles de connectar-se a Internet.

1 http://kpax.uoc.edu/elgg/
2 http://elgg.org

Índex

1. Introducció

kPAX és un portal de jocs seriosos, una eina oberta que té com a principal objectiu facilitar l'aprenentatge a través dels jocs. S'enfoca com un portal social on els usuaris podran jugar, competir entre ells, col·laborar-hi i fins i tot crear nous jocs.

El portal ja disposa d'un tema propi que es vol actualitzar per fer-lo compatible amb les darreres tecnologies web amb l'objectiu de millorar-ne l'eficiència i qualitat en el codi. Amb tot es pretén millorar i a la vegada facilitar-ne la interacció amb l'usuari. Aquest tema ens servirà com a guia per desenvolupar un tema propi.

En aquest treball aprendrem a elaborar un tema per Elgg, el programa gestor de continguts utilitzat per implementar el portal. L'elaboració del tema es farà seguint criteris fàcilment exportables a altres portals: fer servir bones pràctiques en el codi, aportar solucions alternatives per a dispositius amb recursos limitats, disminuir el temps de descàrrega i visualització de continguts, i obrir la porta a les novetats i millores en el rendiment que ens aporten les darreres versions de les tecnologies web.

Elgg

Elgg és un programa gestor de continguts (*CMS*) força popular que permet crear i administrar portals a Internet de caire social, on els usuaris poden interactuar entre ells. Es tracta d'una eina molt potent que pot encarregar-se de tota la maquetació del portal, de la navegació i de la gestió d'usuaris i grups. A tota aquesta funcionalitat, que és la que propiament fa el nucli o motor d'Elgg, s'hi pot anar afegint més funcionalitat mitjançant connectors o plugins, com per exemple afegir perfils personals, blogs, visors de Twitter o fins i tot un tema.

Elgg ha estat implementat en el llenguatge de programació PHP (acrònim recursiu de *PHP Hypertext Pre-processor*). Aquest llenguatge de guions interpretats és molt comú en la programació al costat del servidor degut a que que posa a disposició dels programadors de *back-end* (capa lògica de programari no visible per l'usuari) una API[1] molt extensa i modular que permet, entre altre coses, connectar-se a una base de dades, gestionar arxius al servidor o establir un flux privat de comunicacions per usuari i sessió[2].

El codi d'aquest gestor de continguts s'organitza segons el patró model-vista-controlador o MVC. Es divideixen els components de l'aplicació en tres capes independents, de manera que un canvi en una d'elles no afecti les altres. Aquestes capes són les següents:

- **Model**: són les dades i la seva manipulació. Elgg fa servir una base de dades mySQL on emmagatzema el contingut, les dades dels usuaris, dels grups, missatges entre usuaris, la configuració del propi portal, etc.

1 Application Programming Interface o interfície de programació d'aplicacions és el conjunt de funcions a l'abast del programador per realitzar una tasca concreta, com connectar-se a una base de dades o escriure en un fitxer.

2 Per més informació, consultar http://php.net/

- **Vista**: capa de programari que s'enviarà a l'usuari. Habitualment es correspon amb la interfície gràfica que es veu i amb la que s'hi ineractua. *Elgg* permet generar vistes en HTML (la vista per defecte), RSS, XML, JSON, etc.

- **Controlador**: capa lògica que interacciona amb les altres dues. Quan rep la petició d'un usuari, consulta si cal la base de dades i genera codi en conseqüència, el que posteriorment serà enviat a l'usuari.

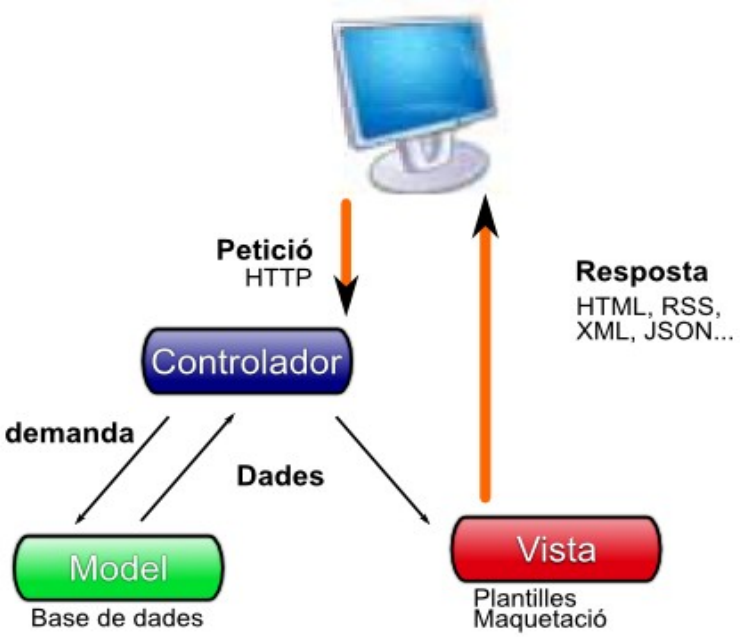

Fig 1. Patró Model-vista-controlador.

El gestor de continguts Elgg és completament modular. Es pot estendre la funcionalitat del seu nucli afegint porcions de codi amb funcionalitats independents, com els blogs, els perfils o l'enviament de missatges entre usuaris. Cada un d'aquests components o connectors (i el propi nucli) segueix el model vista-controlador i genera la seva pròpia sortida, la seva vista. El motor d'Elgg s'encarregarà posteriorment d'unificar la sortida de tots els components i enviar al client una sola vista global.

Cal tenir present que en el moment en què s'unifiquen totes aquestes vistes pot donar-se el cas en que un d'aquests connectors sobreescrigui la sortida d'altres. És el que es coneix com “sobrecàrrega de vistes”, i aprofitarem aquest fet per elaborar el nostre **tema** (com veurem tot seguit en l'apartat d'introducció a CSS). Pot semblar poc eficient el fet de generar primer un munt de vistes independents per després sobreescriure el seu codi, però de fet ens aporta un grau extra de control en el codi en actuar de filtre i en unificar les normes d'estil en un sol punt (recordem que es vol controlar la sortida de tots els altres connectors). A més, com que aquesta barreja es fa en el costat del servidor, Elgg pot emmagatzemar el fitxer resultant en la seva memòria cau i actualitzar-lo només quan consideri oportú. Elgg ja ho fa amb arxius de contingut estàtic, com els fulls d'estil en CSS. A la figura següent podem veure un diagrama d'aquest procés.

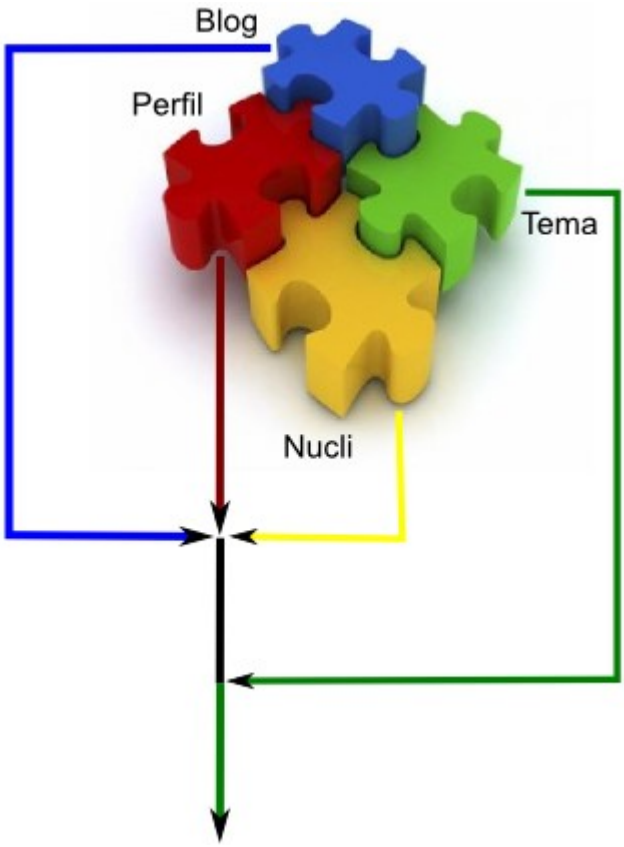

Fig 2. Agrupació de vistes i sobrecàrrega

Com s'ha dit, a l'usuari s'enviarà un document web complet amb el contingut generat pel gestor. Seguint criteris de separació de capes, el client rebrà quatre tipus d'arxius diferents:

- Codi HTML per la maquetació i el contingut, pròpiament dit. És la base per construir un document web i, en el nostre cas, l'únic arxiu realment dinàmic (pot variar entre connexions consecutives).

- Codi CSS per controlar la presentació i en menor grau la interacció.

- Codi JavaScript per controlar-ne el comportament i afegir un cert dinamisme en la interacció amb l'usuari (permet la programació dirigida per esdeveniments).

- Altres: imatges, pdf, vídeos, arxius d'àudio...

De moment ens quedarem aquesta breu introducció d'HTML i JavaScript i ens centrarem en l'ús de CSS ja que serà el codi central del connector.

CSS (Cascading Style Sheets o Fulls d'estil en cascada)

CSS és una tecnologia web creada a mitjans dels anys 90 amb l'objectiu de separar el contingut de la presentació en els documents d'hipertext. Actualment s'està treballant en la versió CSS3, la qual, malgrat que encara no és un estàndar al 100%, és suportada en major o menor grau per tots els navegadors.

El principal avantatge de separar l'estructura de la presentació és que permet adaptar el contingut al dispositiu on es mostra la pàgina, ja sigui la pantalla d'un ordinador, una impressora, un projector, un dispositiu braille o un sintetitzador de veu. La nova versió permet, a més, modificar la presentació segons determinats esdeveniments produïts al navegador, com girar un tablet, tal com veurem més endanvant quan parlem de les “media queries”.

Els elements en un document HTML s'organitzen de manera jeràrquica segons el model DOM (Document Object Model), amb l'element html com a arrel de la jerarquia. CSS permet aplicar normes d'estil als elements inclosos en aquesta jerarquia i, a més, introdueix herència d'estil entre elements pares i fills. L'estructura d'una norma ha de seguir aquesta sintaxi:

```
selector1,..., selectorN  {norma1: valor1; ...; normaN: valorN}
```

Un selector pot ser el nom d'un element html, com per exemple “p” per als parágrafs, pot ser un identificador d'un element concret (precedit pel símbol '#'), o bé el nom d'una classe (precedit per un punt '.') si es vol compartir diverses normes d'estil en un conjunt determinat d'elements.

Per exemple, si en l'HTML tenim `<p id=”hola-mon” class=”vermell”>Hola Món</p>`

podrem canviar el color del text de diverses maneres:

```
p {color:red}
#hola-mon {color:red}
.vermell {color:red}     = *[class='vermell']
body {color:red}
```

Amb la primera regla canviarem el color de tots els paràgrafs del document, amb la segona només el paràgraf amb l'identificador donat; i amb la tercera tots els elements que a l'HTML s'hagi establert l'atribut class=”vermell”. La darrera norma també canviarà el color del parágraf donat que la propietat color s'hereta.

Quan es parla de cascada es refereix al fet que CSS fon informació d'estil d'origens diversos i per l'herència de propietats d'elements pares a fills (bàsicament el color i la tipografia). El navegador obtindrà les normes d'estil per aquest ordre:

1. Full d'estil per defecte del navegador (tots els elements tenen un estil per defecte)
2. Full d'estil en un arxiu extern enllaçat amb l'element <link> a la capçalera del document HTML.
3. Contingut de l'element <style> al document HTML actual.
4. Contingut de l'atribut style d'un element HTML donat.

En cas que s'indiqui una mateixa norma d'estil en diversos punts, CSS donarà prioritat a les normes indicades de manera més específica (més avall en la llista anterior, o bé s'utilitzi un selector amb més pes). En cas d'empat aplicarà la regla que aparegui després. De fet, seguint criteris de bones pràctiques, es desaconsella rotundament l'ús de l'atribut i de l'element style (punts 3 i 4) i fer servir només fulls d'estil externs per separar completament el contingut de la presentació.

Donat que Elgg només generarà un full d'estil extern per al tema, per sobreescriure les normes que afegeixen els altres connectors i el nucli caldrà que el nostre plugin sigui l'últim en afegir normes al full d'estil i evitar que aquestes siguin sobreescrites. També caldrà estudiar el codi CSS que hem de modificar i fer servir selectors d'igual pes o superior per modificar les propietats existents (en cas contrari el navegador les ignorarà).

El pes del selector es calcula segons el nombre d'etiquetes que hi apareguin: per cada nom d'element HTML en suma 1, per cada classe suma 10 i per cada id 100. Per exemple, seguint l'exemple anterior, si tenim

```
.vermell {color:red}          Pes: 10
#hola-mon {color:orange}      Pes: 100
p#hola-mon {color:purple}     Pes: 1 + 100 = 101
body p {color: red}           Pes: 1 + 1 = 2
p, p[id] {color: black}       Pes: 1, 1 + 1 = 2
```

En avaluar el codi, el navegador aplicarà la tercera norma donat que és la que té més pes i mostrarà el text en lila.

També s'ha d'evitar l'ús de la directiva `!important` de CSS, la qual permet anul·lar aquest comportament quan s'afegeix a la dreta del valor d'una propietat. El seu ús complica el seguiment de la cascada per part de l'implementador i dificulta els canvis posteriors.

Anirem veient més detalls de CSS al llarg de tot aquest document.

2. Estudi de viabilitat

Abast

La renovació de l'aspecte d'Elgg es planteja, tal com hem vist, com plugin extern al nucli que sobreescriurà el codi CSS generat, tant pel propi nucli com per els altres connectors actius. A més, CSS permet recollir i tractar part de la interacció amb l'usuari (especialment els efectes de ratolí), i també incloure petites animacions a la pàgina amb CSS3. Per aquest motiu, es pot desplaçar a aquesta capa de presentació part de la experiència de l'usuari en detriment de JavaScript. Tots els comportaments que es puguin realitzar amb fulls d'estil CSS sempre seran més eficients i ràpids que les mateixes operacions programades amb JavaScript donat que la feina la porta directament el navegador i no el motor d'execució d'scripts.

Si es decideix traslladar part del comportament a CSS bé s'haurien de donar alternatives implementades amb JavaScript per els citats navegadors limitats de manera que no s'afecti la resta de navegadors, és a dir, sense penalitzar aquells usuaris que mantenen al dia el seu programari. Veurem com es pot fer més endavant. D'altra banda, i encara que un tema no seria, en principi, la part més adequada per incloure comportaments a la pàgina, pot resultar útil incorporar scripts amb comportaments genèrics fortament lligats al tema. Per posar un exemple, es podria incloure un script que transformés un menú d'enllaços en un menú desplegable només per al tema en qüestió.

Sistemes actuals

Es disposa d'un tema propi per kPAX que genera les regles d'estil generals i modifica el codi CSS d'altres plugins instal·lats en el sistema. Quan s'inclouen arxius css.php en un tema, Elgg automàticament integrarà aquest codi a la vista generada pels plugins que vulguem modificar. Elgg és completament modular. És capaç de detectar els canvis desitjats si es segueix una estructura lògica de directoris. Vegem aquesta estructura i una petita descripció del propòsit de cada arxiu o carpeta:

```
elgg/mod/NomDelTema
  + manifest.xml  Metadades del plugin o tema (dependències, llicència...)
  + start.php     Inicialitzacions per Elgg (comportaments, afegir scripts...)
  / graphics      carpeta contenidora d'arxius d'imatge.
  / languages     arxius amb les traduccions de les frases incloses al tema
  / views         carpeta per organitzar els arxius d'estil d'altres plugins
    /default            normes d'estil per medis gràfics (pantalla)
      /css/elements     regles d'estil genèriques (botons, formularis, menú,...)
      /js               codi JavaScript a incloure
      /moduleName       Una carpeta per cada mòdul que vulguem controlar amb un
                        arxiu anomenat css.php
```

El tema actual inclou codi per aplicar als connectors següents:

- custom_index: utilitat que permet mostrar als usuaris informació de la seva activitat recent i de manera personalitzada tot just connectar-s'hi. Mostra els arxius que han pujat, avisos de noves entrades o comentaris al blog, grups, favorits(entre d'altres) en una graella de contenidors div.

- file: es tracta d'un petit explorador d'arxius amb el què els usuaris poden controlar els arxius que han pujat al sistema de manera gràfica. Els mostra en format galeria d'icones, on les icones són miniatures per als arxius d'imatge.

- groups: aquest mòdul permet agrupar usuaris en conjunts i gestionar informació compartida. De cada grup existeix un perfil, unes estadístiques que es poden consultar per web i algunes utilitats.

- messages: mòdul que permet que els usuaris es puguin enviar missatges entre ells. Controla el format del missatge, la bústia, les dades de l'emissor, i també l'avís que es mostra als usuaris quan reben correu.

- profile: gestiona informació personal d'un usuari o compartida en un grup. Un perfil es veurà de manera diferent segons el mode en què s'obri: inclourà mecanismes d'edició per al propi usuari, només per la seva visualització per altres usuaris o inclús notificarà un accés indegut a usuaris no autoritzats.

- search: eina de cerca d'entitats per paraules clau (continguts, usuaris, bookmarks, etc). El mòdul afegeix un quadre de text interactiu que es ressalta quan l'element adquireix el focus i un botó d'enviament. El quadre de text inclou la paraula “Cerca” per defecte en lloc de fer servir etiquetes de formulari.

- thewire: utilitat a l'estil *microblogging* de *Twitter* que permet als usuaris afegir comentaris al flux d'activitat (the river). El plugin incorpora una àrea de text per escriure-hi i mostra el nombre de caràcters restants fins a un màxim de 140.

- Twitter: mini-aplicació que permet als usuaris incloure les darreres entrades del seu twitter personal a la seva pàgina de perfil. Els comentaris es mostren com a ítems d'una llista no ordenada amb una imatge de l'autor.

A la figura següent podem veure un exemple gràfic de com es mostra un perfil a Elgg. Podem veure també com s'integra la sortida d'altres plugins, com per exemple la cerca a la pàgina (connector “search”), el perfil (profile) i el taluer de missatges (Message Board), per ordre, de dalt a abaix:

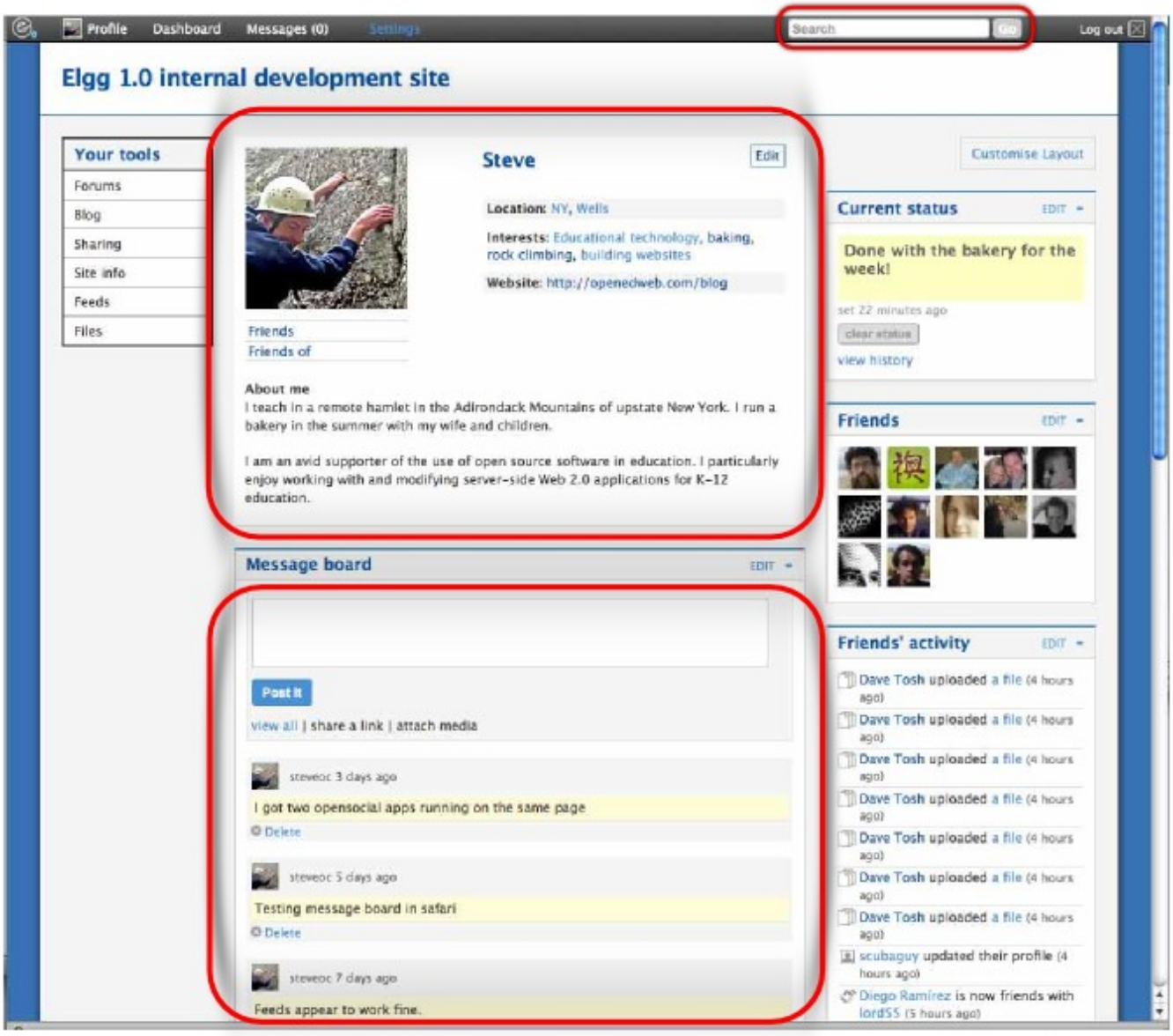

Fig. 3. Exemple de perfil a Elgg extret de http://openedweb.com/blog/2008/07/12/elgg-10-user-interface-screenshots-1/

Definició dels requisits del sistema

Requisits tècnics:

- (100) Complir amb estàndards promoguts per entitats reconegudes(W3C[1], WHATWG[2]) i àmpliament acceptats pels fabricants de navegadors.
- (100): Oferir solucions alternatives per aquells navegadors que no suportin les bondats de les darreres funcionalitats d'HTML5, CSS3 i ECMAScript5 (JavaScript en mode estricte).
- (100): Eficiència en la càrrega. Minimitzar el temps de càrrega de les pàgines en el costat del client.
- (100): Escalabilitat. Extensible a futures ampliacions.
- (80): Permetre la navegació off-line amb emmagatzematge local.
- (80): Complir amb recomanacions d'accessibilitat (WAI)
- (80): Oferir una base eficient per introduir continguts multimèdia

Requisits operatius:

- (100) Adaptat a la filosofia del portal: obert, no discriminar en marca o versió del navegador, bones pràctiques en el codi, etc. Incloure documentació/guia d'estil o bones pràctiques i compartir el codi amb finalitats educatives.
- (100) El contingut s'ha de poder visualitzar des de qualsevol navegador i dispositiu(plataforma).
- (100) L'espai web ha de transmetre un bon grau de satisfacció a l'usuari: visualment atractiu, permetre una interacció senzilla, amb una organirzació clara en subseccions i una càrrega àgil.

Legals:

- (100) Respectar la llicència base del plugin que ens servirà com a base(GPL).
- (100) Respectar drets d'autor en imatges, arxius multimèdia i codi.

Cost:

- Cost nul en llicències (fer servir productes lliures i tecnologies obertes).

1 *World Wide Web Consortium* és una entitat creada per Tim Berners Lee l'any 1994 amb l'objectiu d'estandaritzar i optimitzar els llenguatges utilitzats per l'intercanvi d'informació a la web, com l'HTML.

2 El grup "*Web Hypertext Application Technology Working Group* " va ser creat l'any 2004 per membres de la Fundació Mozilla, Apple Inc. i Opera Software davant el que consideren una " lentitud extrema" en l'evolució de les recomanacions promogudes per la W3C. Són els principals promotors de les APIs d'HTML5 i de CSS3.

Estudi i valoració de les alternatives

Existeixen diversos temes per a Elgg a la web oficial del projecte, però que no s'adapten completament als requeriments donats. A més, es prefereix partir d'un tema que ja existeix especialment per kPAX i fer-lo evolucionar.

Un dels avantatges que té el programari lliure és que no s'ha d'estar contínuament reinventant la roda i es poden aprofitar plugins existents. Existeix un plugin per Elgg creat per Evan Winslow anomenat "HTML5 for Elgg 1.8"[1] que ajuda a incloure les noves funcionalitats d'HTML a la plataforma, especialment les relacionades amb els formularis. Inclou la llibreria Modernizr.js que tal com veurem a la fase de disseny ens ajudarà, i molt, en la detecció de les capacitats del navegador de l'usuari i poder actuar en conseqüència.

Riscos

En tot projecte web existeixen riscos de funcionalitat donat la enorme varietat de dispositius (pc, portàtil, smartphone, tablet, consola de video-jocs, smartTV....), cada un amb les seves pròpies bondats i defectes. Per pal·liar aquest risc el codi haurà de ser estàndar i validat amb l'eina adequada (existeixen validadors creats per la W3C per HTML i CSS, i JSLint per JavaScript). S'hauran d'oferir solucions alternatives per navegadors més limitats.

A més, el fet de plantejar el tema com un plugin per Elgg (codi extern al nucli) ajudarà a la recuperació en cas de desastre ja que només desactivant-lo i reactivant el tema previ es recuperaria la funcionalitat habitual del portal.

El risc de rebuig es pot pal·liar oferint un disseny agradable, de fàcil navegació i àgil, que segueixi criteris de bones pràctiques en usabilitat.

1 Es pot descarregar a http://community.elgg.org/plugins/724267/1.0.1/html5-for-elgg-18

3. Anàlisi del sistema

Definició del sistema

Tal com hem vist a l'apartat anterior, la solució proposada haurà de satisfer els requisits tècnics i operatius següents:

- Complir amb estàndards web promoguts per entitats reconegudes(*W3C, WHATWG*) i àmpliament acceptats pels fabricants: HTML5, CSS3 i JavaScript en mode estricte.

- Oferir solucions alternatives per navegadors limitats. En aquesta categoria cal considerar les versions antigues de l'Internet Explorer[1] amb un percentatge actual considerable d'usuaris[2]. També cal considerar els dispositius mòbils i tauletes que encara que ofereixen un àmpli suport a les noves tecnologies web no s'ha d'oblidar que aquests aparells tenen recursos molt limitats, especialment pel que fa a capàcitat de càlcul i memòria RAM. Segons statcounter/globalStats[3], al voltant d'un 15% dels accessos a pàgines web durant l'any 2012 es van fer utilitzant algun tipus de dispositiu mòbil, i aquesta xifra no para de créixer cada dia.

- Eficiència en la càrrega: rapidesa en la descàrrega i visualització dels continguts.

- Escalabilitat. Extensible a futures ampliacions. Permetre la càrrega asíncrona d'scripts, oferir una base per les funcionalitats HTML5, facilitar la integració de nous plugins, etc.

- Permetre la navegació fora de línia: es pot indicar al navegador de l'usuari que guardi en local els arxius més comuns per evitar la descàrrega cada cop que l'usuari s'hi connecti. Ho permet l'api d'HTML5 anomenada Application cache(o AppCache), creada especialment per emmagatzemar indefinidament en local arxius estàtics.

- Oferir una base eficient per introduir continguts multimèdia: HTML5 inclou una base per fer un ús més eficient dels recursos delegant aquesta tasca a la targeta gràfica, el que alleuja la càrrega de la cpu. A més, facilita la inclusió d'aquests continguts amb els elements <video>, <audio> i <canvas>, així com la seva gestió amb JavaScript i una API nativa estàndard per cada navegador. Amb aquesta tecnologia es poden incorporar a la plataforma jocs que aprofitin les capacitats 3D que ofereix webGL[4].

A la fase de Disseny veurem més detalls i entrarem en la part tècnica relacionada amb aquests requisits.

---

1 De fet, fins la versió 9 del navegador de Microsoft no se'n pot parlar d'un suport respectable dels estàndards segons el web caniuse.com (que veurem més endavant). Ni tan sols són capaços de mostrar el text inclòs als nous elements estructurals d'HTML5 com section o article.

2 Tot just acabar 2012, un 6% dels usuaris a nivell mundial encara fan servir l'Internet Explorer 6 10 anys després del seu llançament, segons el web "The Internet Explorer 6 Countdown".
Es pot consultar a http://www.ie6countdown.com/

3 http://gs.statcounter.com/#mobile_vs_desktop-ww-monthly-201112-201212

4 Per veure un exemple, es proposa al lector visitar http://www.playmapscube.com/

Definicions de les interfícies d'usuari

Els usuaris de kPAX ja estan acostumats a la navegació en portals web. Tot i així, per raons d'usabilitat la navegació serà senzilla i tindrà els continguts ben organitzats en seccions i subseccions. Les interfícies per als usuaris registrats ja es troben definides a la documentació inicial i al plugin (veure Annex 1 amb les interfícies ja dissenyades).

Podem veure un exemple a continuació. Es tracta de la pàgina que veuran els usuaris no autenticats. Inclou un quadre de login i un enllaç al formulari de registre.

Fig 4. Pàgina inicial amb el tema previ elaborat per D. Riera

De fet, aquesta pàgina ha millorat molt en qüestió d'usabilitat si la comparem amb la pàgina que ofereix Elgg per defecte (es pot veure una captura a la pàgina següent). Aquest resulta un bon exemple de com un tema pot acabar millorant l'experiència de l'usuari, i no només pel que fa a qüestions d'aparença:

- Elimina elements innecessaris: enllaç a secció “login” dins la mateixa pàgina d'autenticació i columna buida amb molt espai mort.

- *Layout*: Descripció breu del portal a la part més destacada de la pàgina.

- Ús del *tagline: “¡Juguemos en serio!”.* És una frase molt curta que explica el propòsit del portal. Hauria de ser visible en totes les pàgines i estar més a prop del logotip o títol principal. Sempre ajuda als usuaris a recordar on són.

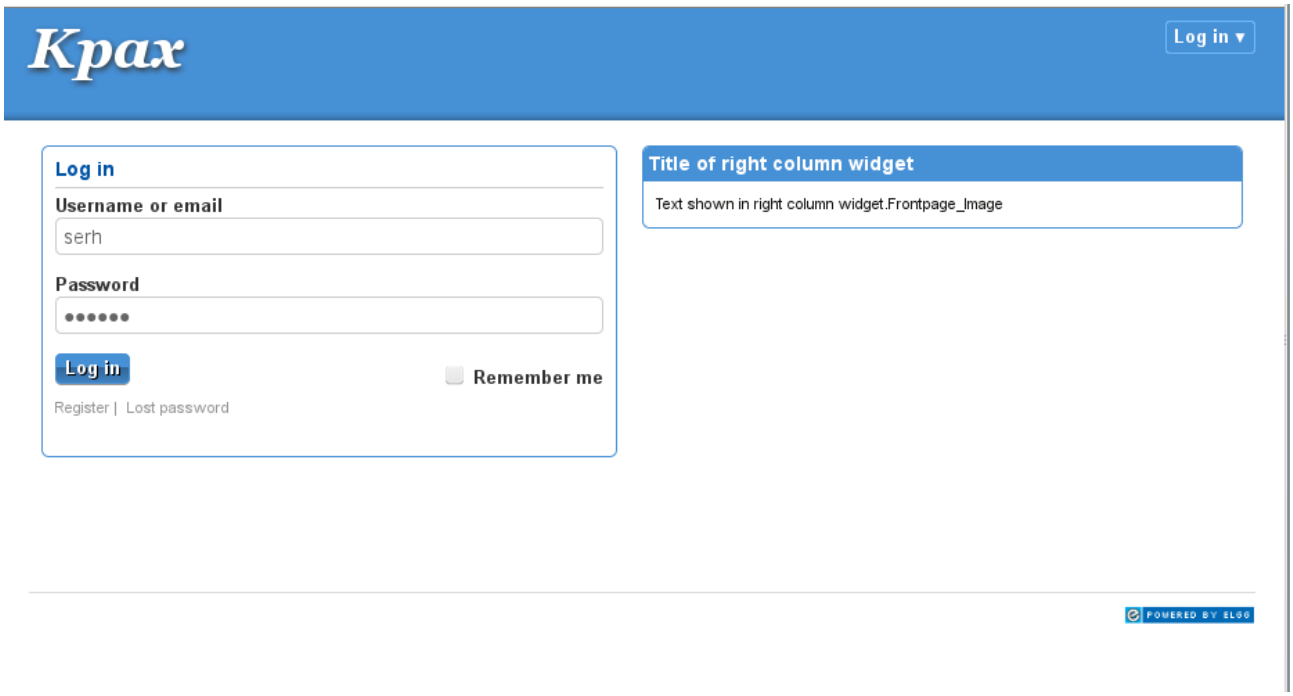

Fig 5: pàgina inicial per defecte.

4. Disseny

En aquesta fase obtindrem un plantejament tecnològic de la solució per començar el desenvolupament. Veurem les solucions tècniques que necessitarem per complir els requisits establerts a la fase d'anàlisi i introduirem el concepte de proves unitàries que es podran realitzar de manera encavalcada amb el desenvolupament.

Arquitectura

En l'estudi de viabilitat hem vist que incloure el plugin "HTML5 for Elgg 1.8" ens permetria adjuntar certes llibreries JavaScript que millorarien la disponibilitat del codi. Amb un ús adequat d'elles podem realitzar la construcció del plugin seguint una metodologia de millora progressiva (o *progressive enhancement).* Es parteix d'una base comuna i compatible amb tots els navegadors i es va fent créixer mica en mica per assegurar que no es queda cap d'ells enrere.

El primer que caldrà fer es esborrar les regles CSS per defecte del navegador, el que es coneix com "reset CSS". El que es fa es esborrar tot el format dels elements del navegador (marges, colors, tipus i tamany de lletres, etc) de manera que cap navegador hi apliqui per descuit les regles per defecte. Amb això s'aconsegueix un millor control de les normes d'estil que s'aplicaran al document i garantir que tot es veurà de la mateixa manera de manera independent al navegador. Elgg ja incorpora aquesta tècnica.

Una funcionalitat interessant que podem incloure ens la proporciona la llibreria HTML5Shiv (inclosa al seu torn a la llibreria Modernizr.js que veurem a continuació). Aquesta llibreria soluciona els problemes que tenen les versions d'Internet Explorer anteriors a la 9 en la representació dels elements estructurals d'HTML5. Com que aquests navegadors no els coneixen, simplement els ignoren. La llibreria el que fa és crear els elements mitjançant javascript i els assigna un estil per defecte [1].

Es pot incloure el seu codi dins d'un bloc condicional d'HTML (només per versions antigues de l'Internet Explorer):

```
<!--[if lt IE 9]>
<script src="dist/html5shiv.js"></script>
<![endif]-->
```

A partir d'aquest moment ja es poden utilitzar els elements estructurals d'HTML5 de la figura del marge i seguir els estàndards web sense perill.

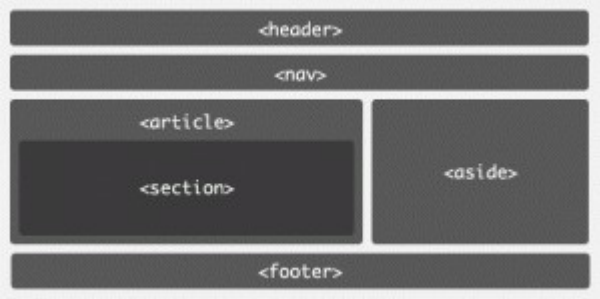

Fig. 6 Elements estructurals d'HTML5

Cal puntualitzar, però, que encara que s'afegeixin elements d'HTML5 i aquests funcionin, el plugin no canvia la definició del tipus de document, la primera línia que ha de tenir tot document

1 La solució consisteix en crear manualment cada un dels elements. A partir de la delcaració el navegador no els ignorarà: document.createElement("section");
Per a més informació consultar http://code.google.com/p/html5shiv/

HTML per indicar al navegador la versió concreta del codi utilitzat perquè el pugui interpretar correctament. En aquest cas, com que Elgg fa servir xHTML en mode estricte i l'HTML5 també deriva de l'XML (i no de l'SGML com era el cas d'HTML4) el codi seria compatible i funcionaria, però no validaria a les eines d'inspecció de la W3C. S'haurien de canviar les línies següents de l'arxiu de maquetació base /elgg/views/default/page/default.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
```

per

```
<!doctype html>
<html>
```

Una altra utilitat que proporciona el plugin "HTML5 for Elgg 1.8" és la llibreria Modernizr.js[1]. Aquesta llibreria fa un anàlisi complet de les capacitats del navegador i afegeix un marcador que consisteix en el nom d'una classe CSS afegit a l'element HTML. Fa servir *feature detection* (detecció de funcionalitats) que sempre és un mètode més fiable que utilitzar un llistat de noms de navegadors i les seves capacitats. Vegem un exemple de detecció de funcionalitats esbrinant si existeix l'objecte localStorage:

```
if (typeof localStorage.setItem === "function") {
      afegeix_classe_a_html("localstorage");
} else {
      afegeix_classe_a_html("no-localstorage");
}
```

Vegem un exemple de les funcionalitats detectades per la pàgina oficial de la llibreria amb l'ajut del Firebug de Firefox:

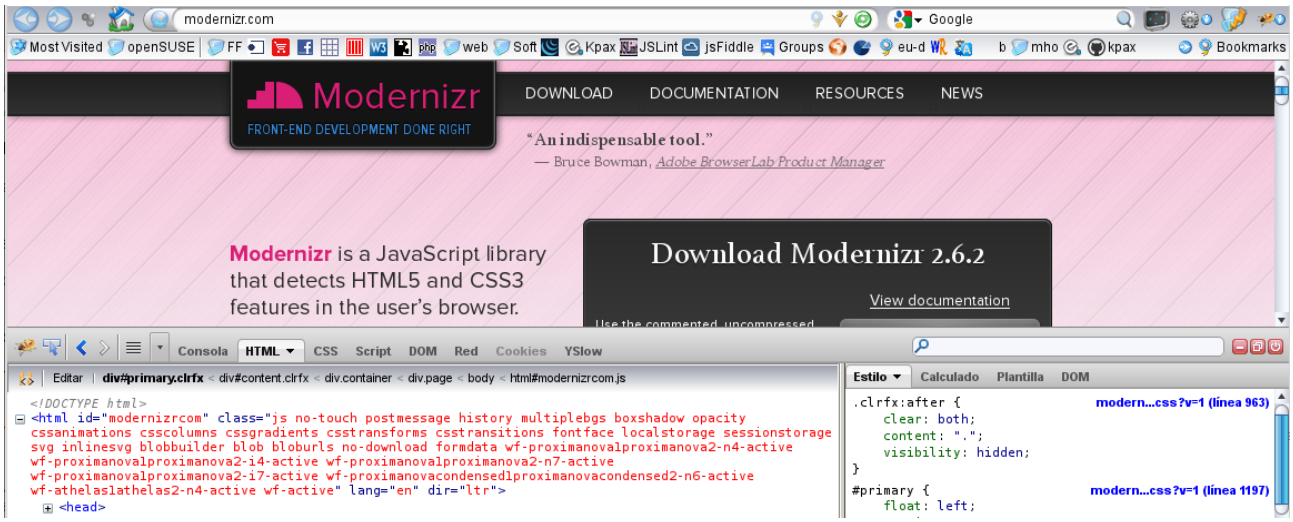

Fig 7: Exemple de funcionalitat de Modernizr.js

1 http://modernizr.com/

Amb més detall:

```
<!DOCTYPE html>
<html id="modernizrcom" class="js no-touch postmessage history multiplebgs
boxshadow opacity cssanimations csscolumns cssgradients csstransforms
csstransitions fontface localstorage sessionstorage svg inlinesvg blobbuilder blob
bloburls no-download formdata wf-proximanovalproximanova2-n4-active
wf-proximanovalproximanova2-i4-active wf-proximanovalproximanova2-n7-active
wf-proximanovalproximanova2-i7-active
wf-proximanovacondensed1proximanovacondensed2-n6-active wf-athelas1athelas2-
n4-active wf-active" lang="en" dir="ltr">
```

Donat que el marcador consisteix en el nom d'una classe CSS assignada a l'element arrel del document, ara podem aprofitar-ho per elaborar normes d'estil adaptades a les capacitats de l'usuari. Posem, per exemple, que volem que la mida de la lletra dins els botons sigui el triple del tamany normal per dispositius tàctils donat que en aquests aparells l'usuari té menys precisió. Com que tots els botons estaran per sota de l'element arrel n'hi ha prou amb afegir el nom de la classe al selector (que serà “touch” en aquest cas):

```
.touch button {
	font-size: 3em;
}
.no-touch button {
	font-size: 1em;
}
```

També es pot utilitzar per oferir solucions alternatives amb JavaScript que no afectin al rendiment dels altres navegadors. Posem per exemple el quadre de cerca utilitzat al portal. Consisteix en quadre de text que quan està buit mostra la paraula “Cerca”, i quan l'usuari hi fa click i selecciona la caixa s'esborra el text inicial (no es vol fer servir una etiqueta per al camp de text per un disseny més net). Existeix un atribut per caixes de text en HTML5 que fa aquesta tasca, l'atribut placeholder. Podrem utilitzar, llavors, l'atribut placeholder quan ho desitgem sempre si hi incloem Modernizr i una implementació d'una solució alternativa. Vegem l'exemple:

```
// En rebre l'atenció, si el seu valor és l'inicial, llavors esborrar-lo
function inputFocus() {
	if(this.value === this.placeholder) {
		this.value = “”;
	}
}

// En perdre el focus, si el seu valor és buit reestablir l'ajuda
function inputBlur() {
	if(this.value === “”) {
		this.value = this.placeholder;
	}
}
```

```
if (!Modernizr.placeholder) {
    // Aquest codi només s'executarà en navegadors antics
    var aInputs = document.getElementsByTagName(“input”), i, oInput;
    for(i = 0; oInput = aInputs[i]; i++) {
        if(oInput.placeholder) {
              oInput.onfocus = inputFocus;
              oInput.onblur = inputBlur;
        }
    }
}
```

Si la solució alternativa (o *fallback solution)* és complexa es pot desar en un arxiu a part i incloure'l amb la funció load de Modernizr. De fet, es considera una bona pràctica desar el codi alternatiu en un arxiu independent si és llarg ja que d'aquesta manera evitem que els usuaris amb navegadors actualitzats s'hagin de descarregar un codi que no necessitaran. Existeixen diverses d'aquestes solucions alternatives codificades en JavaScript que solicionen problemes comuns a la xarxa, el que es coneix com polyfill:

```
Modernizr.load({
  test: Modernizr.geolocation,     // Prova el suport a la geolocalització
  yep : 'geo.js',                  // Script amb funcions natives
  nope: 'geo-polyfill.js'          // Script alternatiu o polyfill
});
```

Al web http://caniuse.com es poden trobar molts polyfills lliures i àmpliament testejats. De fet, aquest projecte mereix una menció especial ja que ens permet saber del cert quines capacitats suporten els diferents navegadors, veure estadístiques de compatibilitat dels diferents navegadors i dispositius; i, a més, ens ofereix una solució alternativa en cas que ho necessitem. També ens pot servir per saber del cert si encara necessitem anteposar el nom del navegador (*vendor prefix*[1]*)* a determinades propietats de CSS3 o ja el podem oblidar.

1 -moz- per Mozilla Firefox; -webkit- per Google Chrome i Safari; -o- per Opera, -khtml- per Konqueror i -ms- per Windows Internet Explorer

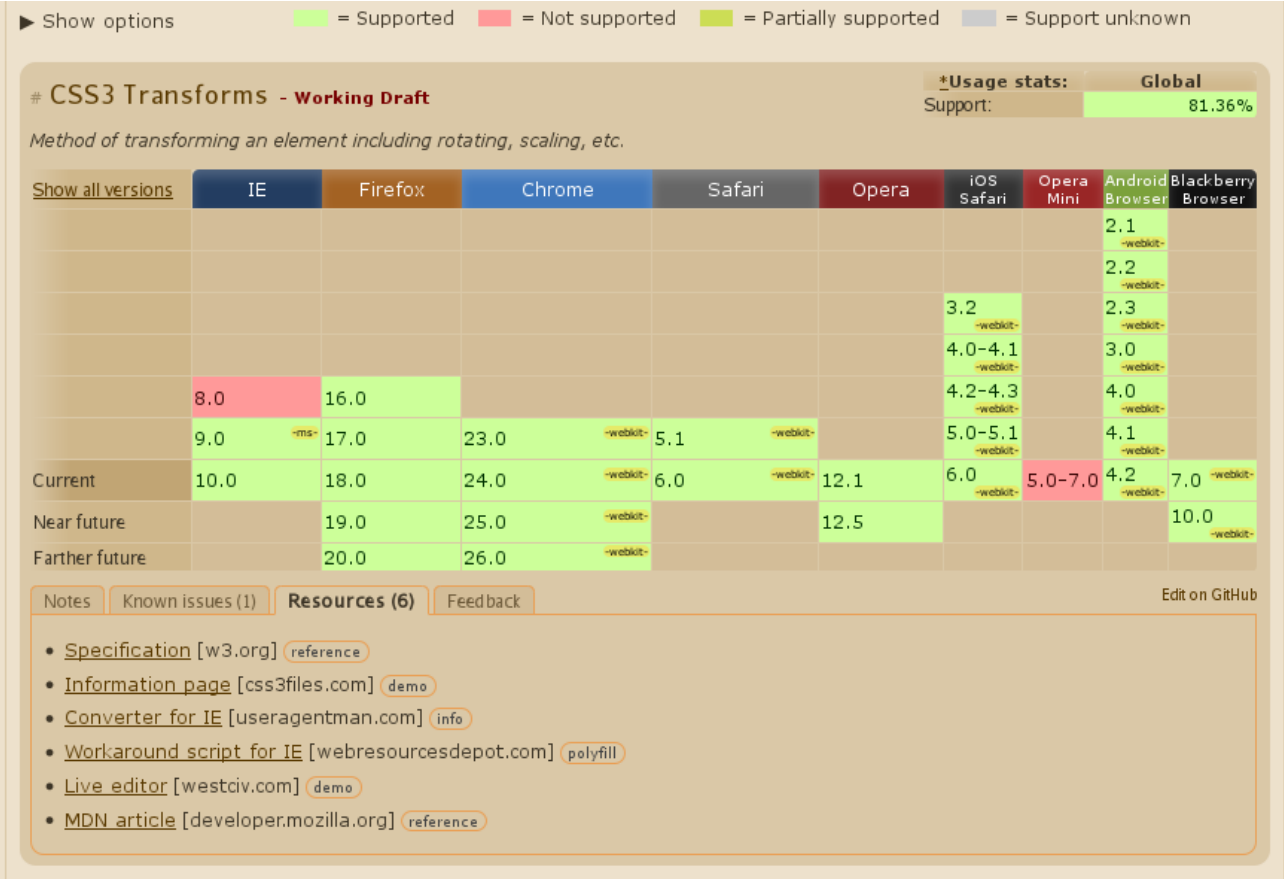

Fig 8: Taula de compatibilitat a la propietat transform de CSS3

Al gràfic anterior es pot veure una taula de suport a la propietat transform de CSS3, la qual permet traslladar, rotar, inclinar o escalar un element. A la part superior s'hi indica també que l'especificació per aquesta propietat encara es troba en fase d'esborrany *(working draft)*. Podem veure que hi ha un ampli suport (un 81% dels navegadors) i que alguns d'ells requereixen que s'anteposi el vendor-prefix al nom de la propietat (IE9, Chrome, Safari, i els navegadors de Blackberry i Android), També s'observa que no el suporta ni l'IE previ a la versió 9 ni l'Opera mini, i que per al primer existeix una solució alternativa que podrem descarregar si polsem l'enllaç “Workaround script for IE” i marcat amb l'etiqueta “polyfill”.

Un altre exemple d'aplicació CSS3:

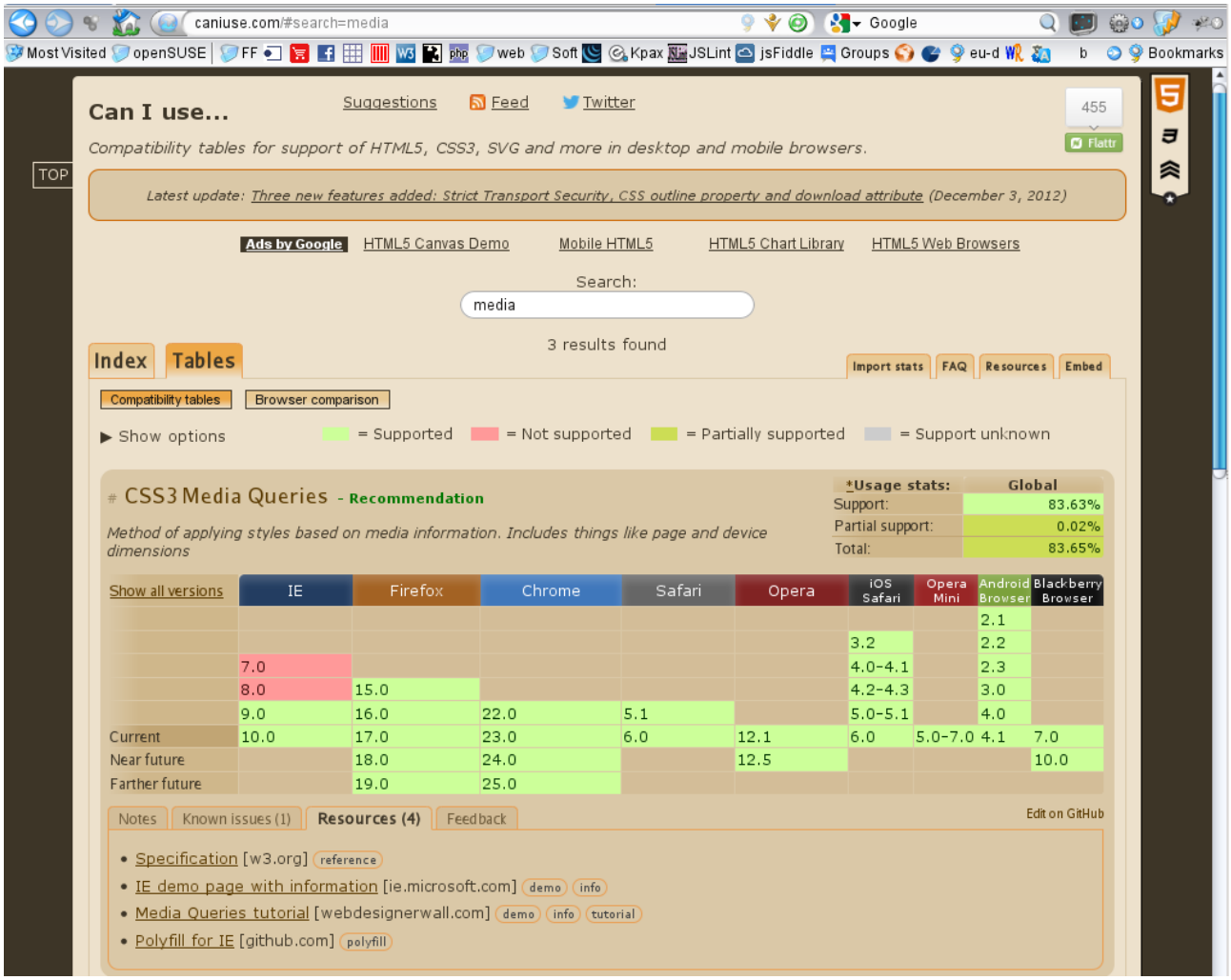

Fig. 9: Taula de compatibilitat a CSS3 Media Queries a http://caniuse.com

Al gràfic anterior es veu que ja hi ha un ampli suport a les anomenades Media Queries de CSS3, i que ja és una recomanació, que forma part oficialment de l'estàndard CSS. Les consultes de medis permeten obtenir determinades característiques del navegador i poder aplicar normes d'estil en conseqüència. D'aquestes característiques es pot destacar l'amplada de píxels de la pantalla, la orientació del dispositiu (vertical u horitzontal) i la profunditat de color. Es poden fer servir en combinació amb els selectors de dispositiu de CSS2 (pantalla, projector, tv,...) i fer el web de tipus “responsive”, és a dir, fer que el navegador respongui davant determinades condicions i adapti la maquetació per optimitzar la visualització i millorar així l'experiència d'usuari.

Podem veure un exemple d'aquest tipus d'espai a la imatge següent. S'hi pot observar com varia la disposició del menús i la mida de les imatges segons l'amplada de la finestra del navegador que mostri la pàgina:

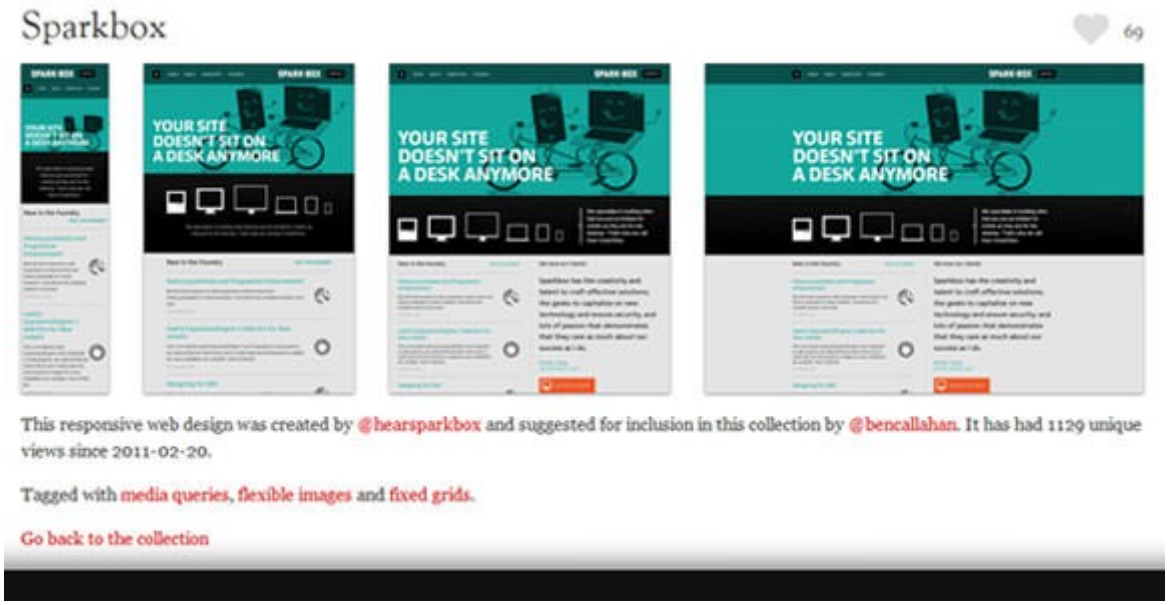

Fig 10: Disseny “responsive” de http://seesparkbox.com/

Es poden incloure diversos dissenys ràpidament en l'arxiu global de CSS, i sempre al final de l'arxiu perquè les normes d'estil no es sobreescriguin les unes a les altres. S'agrupen les normes que es volen aplicar indicant el tipus de dispositiu (la pantalla en aquest cas) i el punt de ruptura en píxels:

```
@media (screen) and (max-width:700px) {
    body {
        margin:0;
   }
   img {
        max-width:100%;
   }
....
}
@media (screen) and (max-width:480px) {
   nav a {
       font-size:0.6em;
   }
   #logo-img {
       background: url(logo-petit.png) no-repeat;
   }
}
```

També pot servir per canviar arxius d'imatge per altres més petits però amb millor definició que si s'haguessin d'escalar. A més, els arxius d'imatge inclosos dins d'una media querie no es descarreguen si no es dóna la condició que les engloba. D'aquesta manera no es penalitza l'usuari descarregant imatges que no necessitarà, a diferència d'incloure imatges a la part “global” de CSS (fora d'una secció media) donat que sempre es descarreguen encara que no es mostrin.

L'eficiència en la càrrega de continguts és molt bona. Amb l'ajut de l'extensió Firebug per Firefox podem estudiar les peticions realitzades pel navegador, el temps de descàrrega per cada una d'elles (de manera gràfica en un *timeline*) i el temps que triga a completar-se la càrrega de la pàgina i la descàrrega de tots els arxius involucrats, és a dir, quan es llança l'esdeveniment onload.

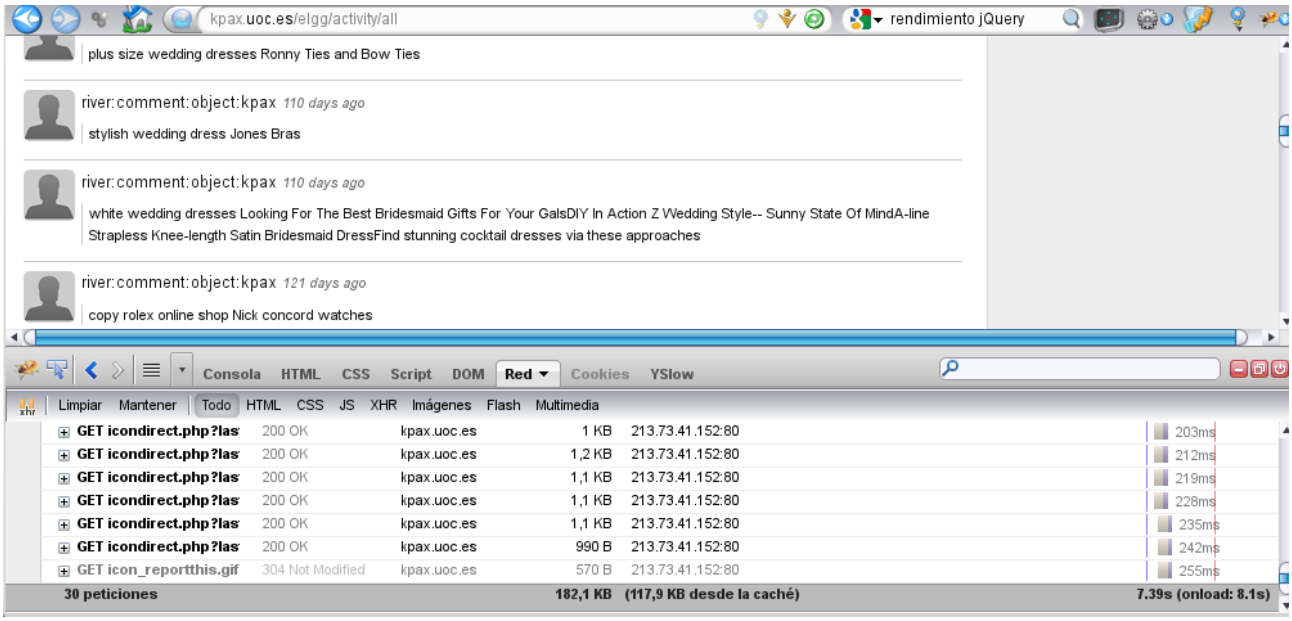

Fig 11: Anàlisi de les peticions i respostes de Firebug

A la figura anterior es pot veure que s'han realitzat 30 peticions i que la pàgina s'ha carregat completament en 8.1 segons. Per millorar el temps de descàrrega es pot provar a minimitzar el nombre de peticions. De fet, per evitar que una mateixa pàgina col·lapsi els recursos de xarxa, el nombre de peticions que un navegador pot mantenir obertes a la vegada amb un mateix servidor és molt reduït, i en la majoria de casos s'ha d'esperar a que acabi una connexió TCP per obrir-ne una de nova. Es pot observar al *time-line* de Firebug (cantonada inferior dreta del gràfic) que les connexions bloquejades o en espera es marquen en lila, mentre que les actives es mostren en color verd, i no hi sol haver més de 5 en un moment donat.

Una primera aproximació per reduir el nombre de peticions consisteix en ajuntar diverses imatges molt utilitzades en una de sola, de manera que només es faci una petició per totes elles. Aquesta unió d'imatges es coneix amb el nom d'*sprites*, i es fa servir de manera habitual per agrupar icones en un sol arxiu. Després, amb CSS podrem indicar quines coordenades de la imatge s'han de mostrar utilitzant la propietat *background-position,* suportada per tots els navegadors existents. Podem veure els *sprites* que fa servir el tema actual al marge de tota aquesta pàgina.

També es poden incrustar imatges en un full d'estil codificades en base64 i evitar una nova petició HTTP per la imatge. No es recomana aquesta tècnica per arxius grans d'imatge ja que augmentaria considerablement el tamany de l'arxiu CSS, però pot resultar adequat per inserir petites imatges lligades a l'estil de la pàgina, com els arxius de fons. Per exemple, si volguésim incloure el logo d'HTML5 podriem fer:

```
#contenidor {
      background:
url(
YOGQwMDRoTHhwWHyQhICUoKh4kHDAqJCUsJRwkJC8sJCcpLC0sJiAyNTAqNScsLC4BCQoKDQsNGQ4OGT
UkHiQ1NTU1NTU1NTQxNTU1NTU1NTU1NTU1NTU0MC81NS01LCwxLDQ0NDQyMDQsKTQ1NCw0NP/AABEIAE
AAQAMBIgACEQEDEQH/xAAbAAACAwEBAQAAAAAAAAAAAAABBwACBgUEA//EADoQAAECBAQDBQQHCQAAAA
AAAAECEQADBCEFBhIxQVFhBxMUIjKBkaHhFjNCUnHR8RVVYnKCk6Oxwf/EABoBAQACAwEAAAAAAAAAAA
AAAAAFBgECBAP/xAAhEQABBAIBBQEAAAAAAAAAAAAABAAIDERIhBAUTMUFRof/aAAwDAQACEQMRAD8Acz
aDcQW0kEXER28qhaC2mxuk8YIoEtMbgYAHkUG4iFHn/O+IYLj+ZaXCXXSSaammCdqljuNWlyAUuvUSzX
bePcupxAdos+hGZ5ngxhZxfuvBy/Tr092+9vvO/SCJnlLqQOkQXUVkWhE4Zn3GJkrA5mIThLo5neAVQl
oec6VKFtNtGlrAdXhh9mOd5ebcuU8iZXKqcVlSQue8koZRJG+kJP8ATBFsgPtK2gNqLxb1XNkwPVZIty
giNwPMHEUnEopZqpBBVoJCVc2i4DelbRZtVlaFe1oLI0UsR2hVQAvT/wBv5wR2hVRsPD8vq/nHlytRom
5n8LVSkTUMtOlYe4/SNDW1lBhuIrpajDk94kh9MkEXY8+sQzDK5mRkpWuYcaOTtthyNXoLnfTqrd3pd3
+r+cfWhzpVVGJ0lNM8OUrnJQGJ0lNZlNLL0SDUFaUMzJFx/qNM5A5o7l2
fS1bHC9jiYcaBOwmarfzlzyEAgqGzD3RbbYoTFSH3W8TiqqgL+lDxbbfQn2PFbkeYsIKf4bD7xgiVdPP
GF59mzFJWpKaqaNKEuWOobe2O/V4V4k6hmd+WJMyQPQRXPpHDxiaMP7Qpk5ZIlcgP1M4FdC
hlLmurRUSMRAn6QlkqG38pvxiJjypzW1d+CrPyMC6OSSwC0bC6mESaHG1TRR0CDoZ9Utt369IymVmbn
uUpAAQnvFAAbBiP+xqaLwuUaeoSnEBMnKAcFQUXDtZItvxjNZCRrzTNWSRppzfqSmMEXJGDWXul5RkiK
dwJxrV/qYu+wQqKktuhoKt/OGPMQCSkbuPfEuq6gz+ZRtBfVc2SOEVfWbmC+ogCwgiy2Y8kHGa+ZXyKo
SpqmHdrRawA3H4cjGVrMl1VAsnwviEg+qQrV8N/hDUCnmPwEAHyKL8RHJJw45DfgqUg6pPC0N0QPqW1H
k2rrSNUmXSJZ3mqu34Bz72jV5Zyt9H5kyomVPiJi06D5NIHHnHfKmUg9IgsooJtCLiRRGx5WnI6lPO0s
Ogfinpsbpgem6TbnEB+yraA+ktHWo5f//Z);
}
```

Per imatges de fons també es pot fer servir la nova propietat CSS gradients, que permet construir una imatge a partir de codi CSS i evitar una petició HTTP. En l'exemple següent[1] es pot veure com s'ha reduït la imatge zig-zag a 310 bytes:

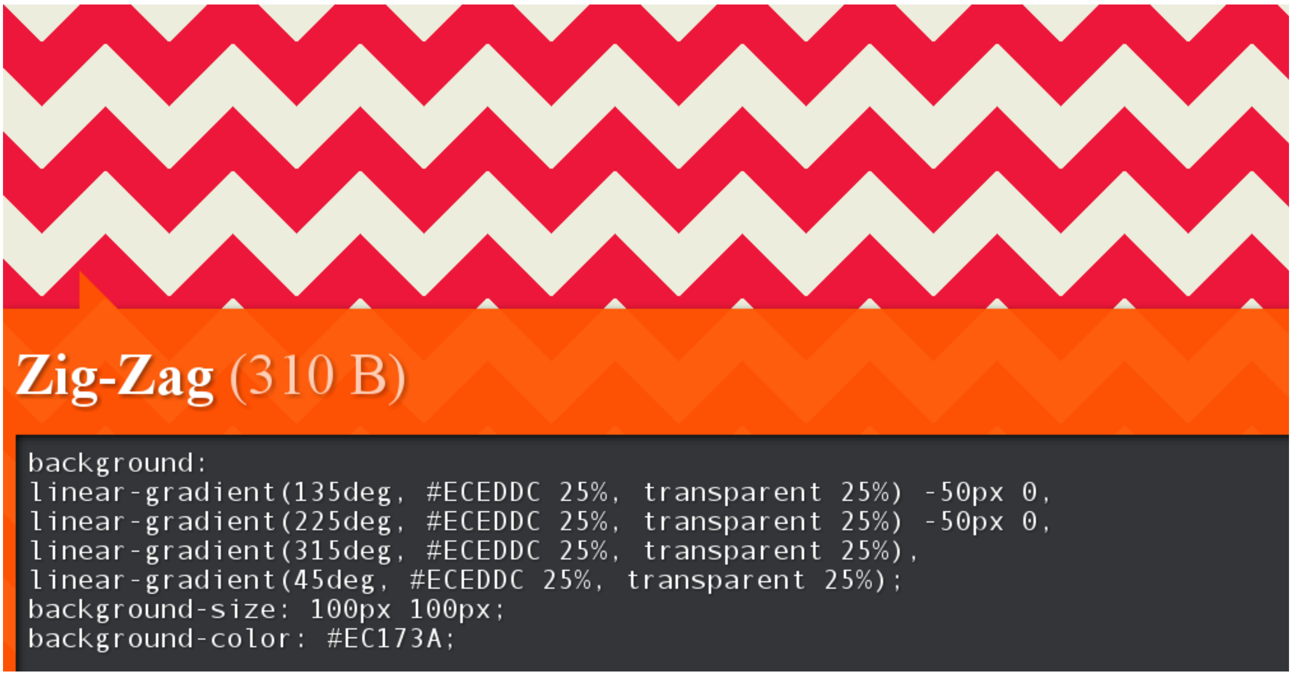

Fig12: Exemple d'imatge de fons elaborada només amb CSS3

Cal esmentar, però, que el consum de recursos de CPU pot resultar més elevat que si s'utilitzés una imatge en un arxiu donat que cada cop que el motor de renderizatge del navegador ha de fer un refresc de la pàgina haurà de fer càlculs extra.

1 Extret del web http://lea.verou.me/css3patterns/

Una altra manera de reduir el temps inicial de descàrrega és fent servir el que es coneix com a *lazy loading (*càrrega mandrosa o, de manera més concreta, càrrega sota demanda). La idea és demanar arxius estàtics només quan facin falta, en resposta a una determinada acció, i minimitzar així la quantitat de codi inicial que es descarreguen els usuaris. Per posar un exemple, es pot programar la descàrrega de les funcions *drag & drop* de la secció perfil per als casos en què l'usuari decideixi arrossegar elements, i no tenir-les constantment en memòria només per si un cas. Per demanar arxius sota demanda es pot fer servir la llibreria require.js o demanar-los a través de peticions ajax mitjançant la popular llibreria jQuery, la qual ja es troba inclosa a la plataforma Elgg.

Elgg proporciona mecanismes al nucli per incloure scripts només en determinades pàgines com a primera aproximació. En el moment en que s'inicialitza un *plugin* o, en el nostre cas, un tema, es registra l'arxiu javascript que vulguem incloure per poder referenciar-lo després només en les pàgines en què faci falta:

```
function my_plugin_init() {
        elgg_register_simplecache_view('js/my_plugin/my_javascript');
        $url = elgg_get_simplecache_url('js', 'my_plugin/my_javascript');
        elgg_register_js('my_plugin', $url);
}
```

I després:

```
elgg_load_js('my_plugin');
```

Es poden aplicar també altres bones pràctiques que ajuden a millorar el temps de càrrega de la pàgina, com per exemple, distribuir les imatges en diversos servidors per evitar el límit de descàrregues que pot obrir el navegador en paral·lel a un mateix domini. També s'ha demostrat que enllaçar el full d'estil a la secció head del document HTML i el JavaScript al final del document agilitza la representació de la pàgina donat que els navegadors quan fan la petició d'un arxiu JavaScript es solen bloquejar a l'espera de l'arxiu.

De fet, el codi JavaScript i CSS es poden minimitzar. La tècnica consisteix en substituïr el noms de variables per noms més curts, eliminar comentaris i elements redundants o innecessaris, i substiruir expressions per altres d'equivalents i més curtes. Amb aquesta tècnica, el codi de l'exemple de la pàgina 21 es reduïria a[1]:

```
function a(){this.value===this.placeholder&&(this.value="")}function b()
{""===this.value&&(this.value=this.placeholder)}if(!
Modernizr.placeholder){var
c=document.getElementsByTagName("input"),i,d;for(i=0;d=c[i];i+
+)d.placeholder&&(d.onfocus=a,d.onblur=b)};
```

En arxius grans l'estalvi és considerable: per exemple, la popular llibreria jQuery[2] 1.9 ocupa 261,1 kB en la versió completa i 90,9 kB en la versió minimitzada, menys del 35% del tamany original.

1 Minimitzat a http://closure-compiler.appspot.com/home (advanced optimization)

2 Descarregat de http://api.jquery.com

Aplicar pràctiques eficients en la codificació dels scripts també millora el temps de resposta. La pàgina jsperf permet realitzar tests de rendiment d'scripts per poder triar-ne la millor tècnica. Vegem un exemple del nombre d'operacions per segon que pot executar un navegador segons com es plantegi un bucle que faci un recorregut pels elements d'un *Array* o vector[1]:

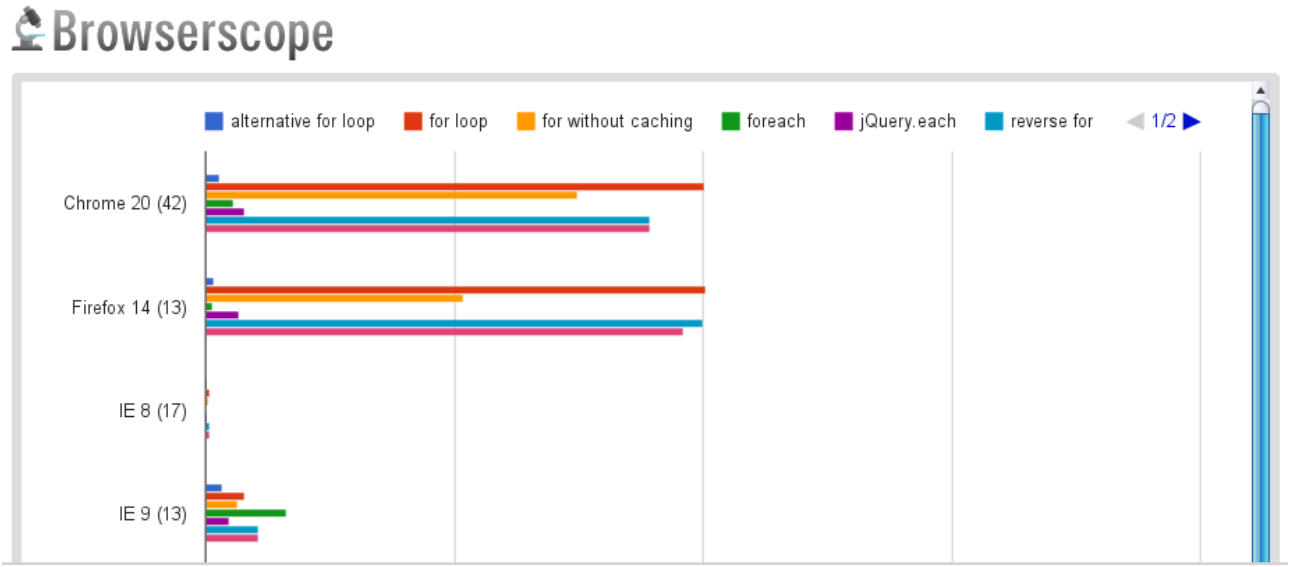

Fig13: Test de rendiment dels bucles JavaScript realitzat a jsperf.com

Del gràfic anterior es desprén que les millors pràctiques corresponen als bucles for i while tradicionals (vermell, groc, blau clar i rosa) i surten mal parats el bucle for-in (blau fosc), foreach (mètode de l'objecte natiu Array) i el mètode each de jQuery. Caldria evitar aquests últims per ser marcadament més lents. De fet, en el moment d'incloure qualsevol funcionalitat amb JavaScript és una bona pràctica plantejar-se si de veritat cal fer servir jQuery o programar directament en JavaScript natiu donat que aquesta llibreria sol penalitzar el rendiment de manera considerable.

La resta de requeriments (navegació fora de línia i la inclusió de continguts multimèdia) ja els ofereix HTML5 amb les seves API, i ja són disponibles per la majoria de navegadors de manera nativa. *Elgg*, a més, ens proporciona mecanismes per mantenir i fer evolucionar el tema (escalabilitat) ja que per ampliar el tema amb noves regles per nous plugins només cal incloure una nova carpeta i un arxiu amb el css corresponent.

1 Extret de http://jsperf.com/jquery-each-vs-for-loop/73

Especificació del pla de proves

Es farà servir una metodologia TDD (*Test Driven Development)* per les proves unitàries de les funcionalitats JavaScript: primer es desenvolupen el tests unitaris, i es va programant per complir-los. Aquest mètode agilitza el desenvolupament i evita que els errors es propaguin a fases posteriors. Elgg ja inclou JsTestDriver per realitzar aquesta tasca. També es pot fer servir per realitzar proves d'integració amb altres mòduls controlant la seva sortida.

JsTestDriver permet realitzar les proves en paral·lel en diversos navegadors, el que permet un desenvolupament més ràpid donat que no cal estar constantment guardant els arxius, obrint-los al navegador i anar polsant la tecla de refresc F5. A les figures següents podem veure un diagrama del funcionament client/servidor de JsTestDriver i com s'ha fet servir per realitzar 8 tests unitaris de cop en Mozilla Firefox, Google Chrome, Opera i Konqueror.

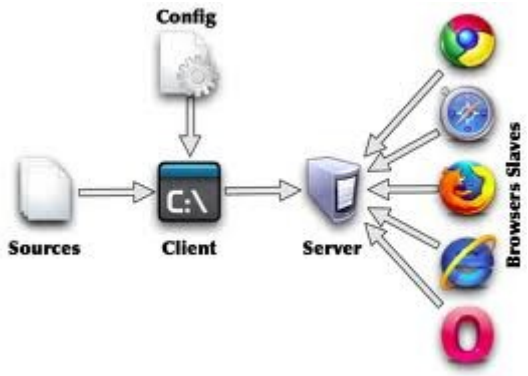

Fig 14: Diagrama de funcionament de JsTestDriver

.

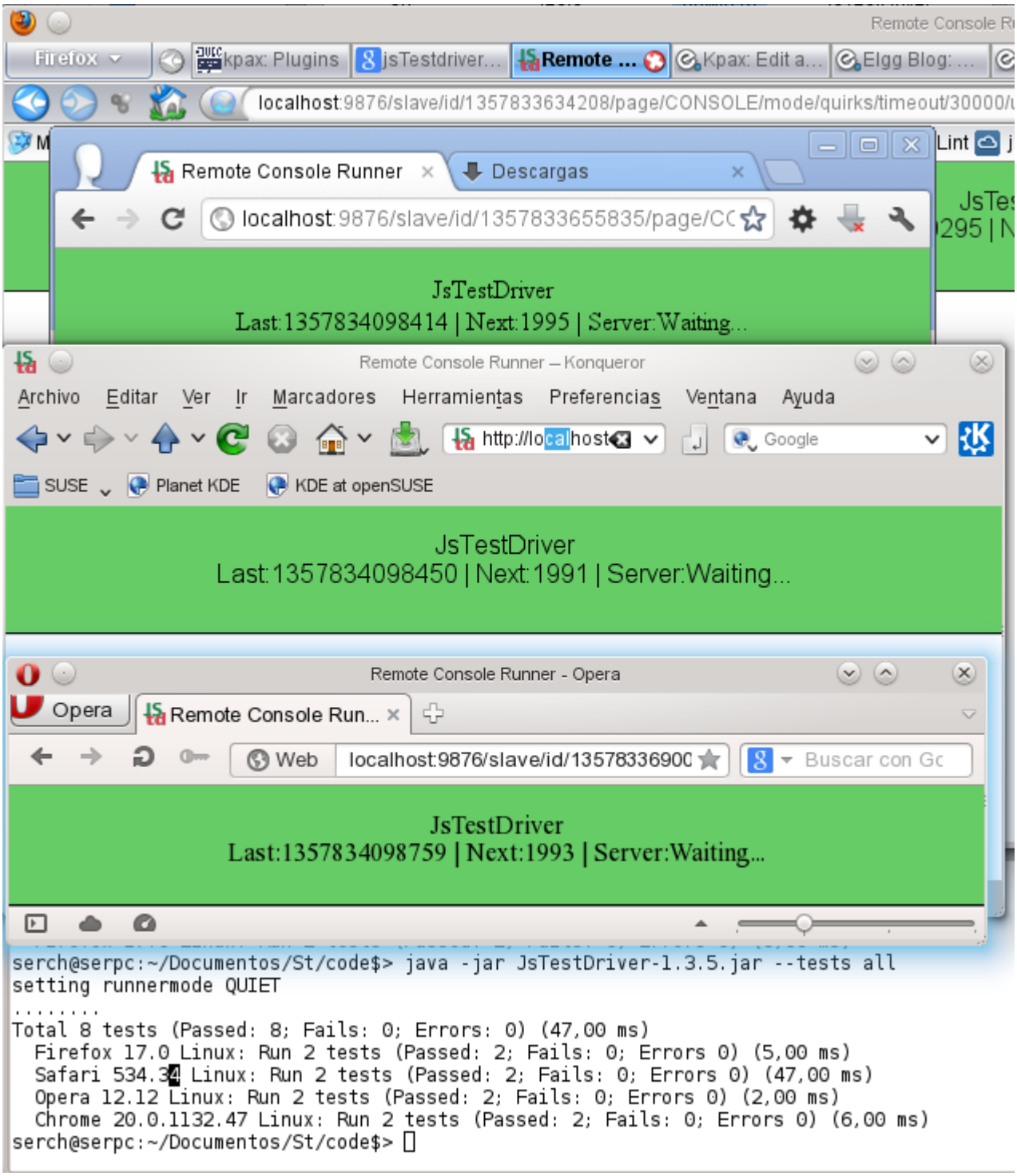

Fig 15: Navegadors capturats a JsTestDriver i execució dels tests unitaris des de la consola.

Exemple de test realitzat amb JavaScript per JsTestDriver:

```
(function(doc, ns) {

TestCase("news", {
    setUp: function() { },
    tearDown: function() { },
    "test data structures": function () {
        assertObject(oNews);
        assertArray(oNews.results);
        assertObject(ns);
        assertObject(ns.iterator);
    },
    "test inizialization": function () {
        assertFunction(ns.iterator.init);
        ns.iterator.init();
        assert(doc.getElementsByTagName("article").length === 10);
    },
    "test next functionallity": function () {
        assertFunction(ns.iterator.next);
        assertFunction(ns.iterator.hasNext);
        assert(ns.iterator.hasNext());
        for(var i = 0; i < 10; i++) {
            assertNotNull(ns.iterator.next());
        }
        assertFalse(ns.iterator.hasNext());
        assertNull(ns.iterator.next());
    },
    "test next button": function () {
        var o = document.getElementById("down"),
            spy = sinon.spy(ns.iterator, "forward");
        o.click();
        assert(o.calledOnce);
    }
});

}(document, namespace));
```

Com es pot veure, aquest test va més enllà de les simples assercions (verificar que es compleix una determinada condició, o que una variable té un valor determinat, per exemple). Permet analitzar també com i quan es criden determinades funcions (amb spyes) i simular o falsejar l'execució de funcions (amb *stubs*) o l'estat que hauria de tenir un objecte determinat (amb *mocks*). Per fer-ho, JsTestDriver incorpora la llibreria sinon.js[1], creada especialment per realitzar tests unitaris en aplicacions JavaScript.

Les proves d'integració es poden realitzar en una còpia del servidor de producció abans de la implantació per comprovar que tot funciona correctament. A més, com es veurà posteriorment en l'apartat corresponent a la implantació, es podran fer proves de rendiment més exhaustives un cop s'hagi instal·lat el connector en l'entorn real.

1 http://sinonjs.org/

5. Desenvolupament

A la fase anterior ja s'ha explicat les decisions que afectaran el desenvolupament i s'han decidit i codificat els tests unitaris necessaris per a realitzar el desenvolupament.

De programari només es necessita un editor de text pla i el JsTestDriver per anar executant els tests mentre es fan els desenvolupaments JavaScript que calguin. Per anar verificant estils o per anar provant diferents propostes resultarà més ràpid fer servir l'extensió Firebug per Firefox o Developer Tools per Google Chrome ja que permeten analitzar el flux d'herència CSS i realitzar canvis d'estil “en calent”, de manera ràpida i sense haver de modificar i guardar els arxius. A més, ofereixen eines potents per realitzar debugging, amb la inclusió de breakpoints, consultes de variables en qualsevol moment, anàlisi dels esdeveniments JavaScript llançats, i temporització de les peticions i respostes en un timeline, com ja hem vist anteriorment.

Fig 16: Chrome Developer Tools

Com a editor de text pla el més aconsellable(i obvi) és fer-ne servir un especialment adaptat a la programació, com és el cas de Sublime(veure fig. Següent). A més de resaltar les paraules clau pròpies d'HTML, CSS o JavaScript s'hi poden instal·lar extensions que ajudin a generar la documentació del codi en format jsDoc[1] a mesura que es va realitzant el desenvolupament. Si es van fent els canvis directament en el plugin instal·lat a Elgg, primer caldrà desactivar la opció "simple chaché" del panell d'administració o anirà servint la còpia que guarda a la seva memòria en lloc dels arxius modiificats.

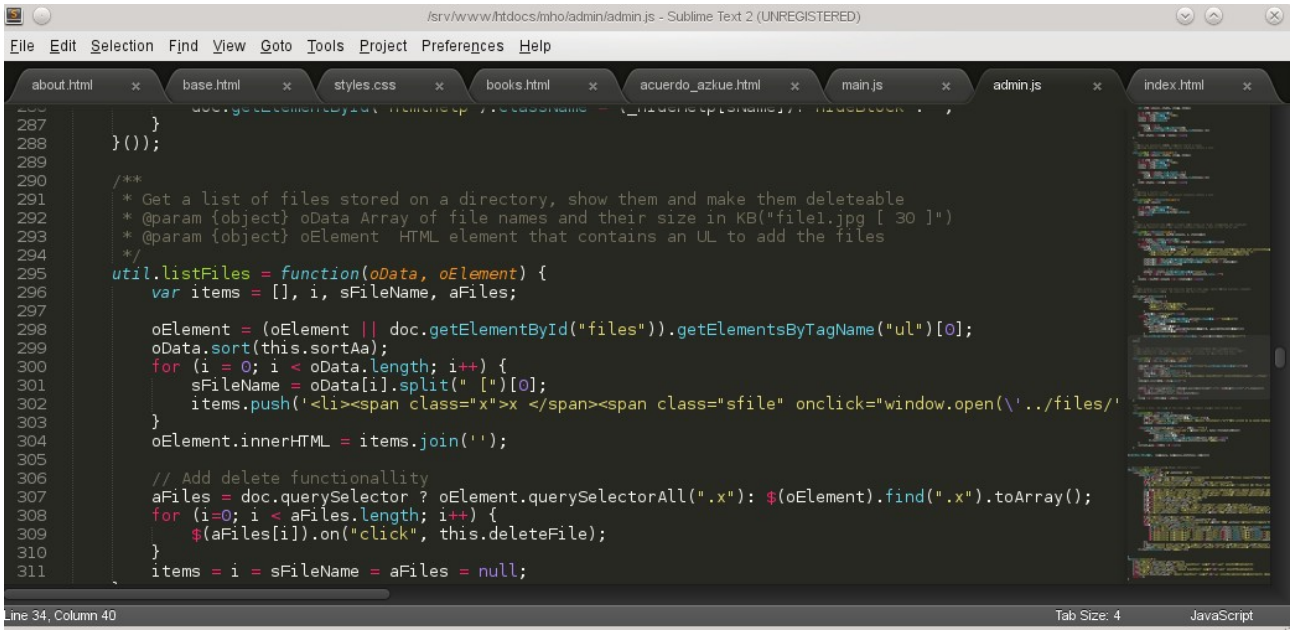

Fig 17: Sublime Text Editor

Un cop es tinguin els arxius que formaran el mòdul es pot realitzar un test d'integració en una còpia o versió reduïda d'*Elgg* en una màquina a part. Si es passen les proves d'integració i d'estrès (sobrecarregar-lo a peticions) i el plugin respon bé, llavors es podrà copiar al servidor de producció.

1 Visitar http://usejsdoc.org/ per més informació.

6. Implantació

Donat que el tema s'ha implementat de manera externa al nucli, per implantar-lo només cal copiar-lo dins el directori /mod del directori arrel de kPAX al servidor de producció i activar-lo al panell d'administració de plugins. Es recomana ubicar-lo a la part més inferior de la llista de connectors donat que Elgg els executa sempre per ordre, de dalt a abaix, i d'aquesta manera ens assegurarem que sigui el nostre tema el que sobreescrigui les normes CSS dels altres plugins i no al revés.

Un cop implantat es poden realitzar proves de rendiment en l'entorn real, a més de verificar que el contingut es mostra correctament. Les Developer Tools de Google Chrome també ofereixen eines d'auditoria de rendiment de xarxa i de codi, i ens indica com podem millorar el nostre web.

Fig 18: Auditoria de la pàgina http://localhost/elgg/activity realitzada amb Chrome Developer Tools

Una altra eina d'auditoria interessant ens la ofereix Yahoo! Inc amb YSlow. Aquesta eina es pot integrar a la coneguda extesió Firebug per Mozilla Firefox i permet realitzar proves de rendiment molt completes, a més d'oferir ajuda en línia.

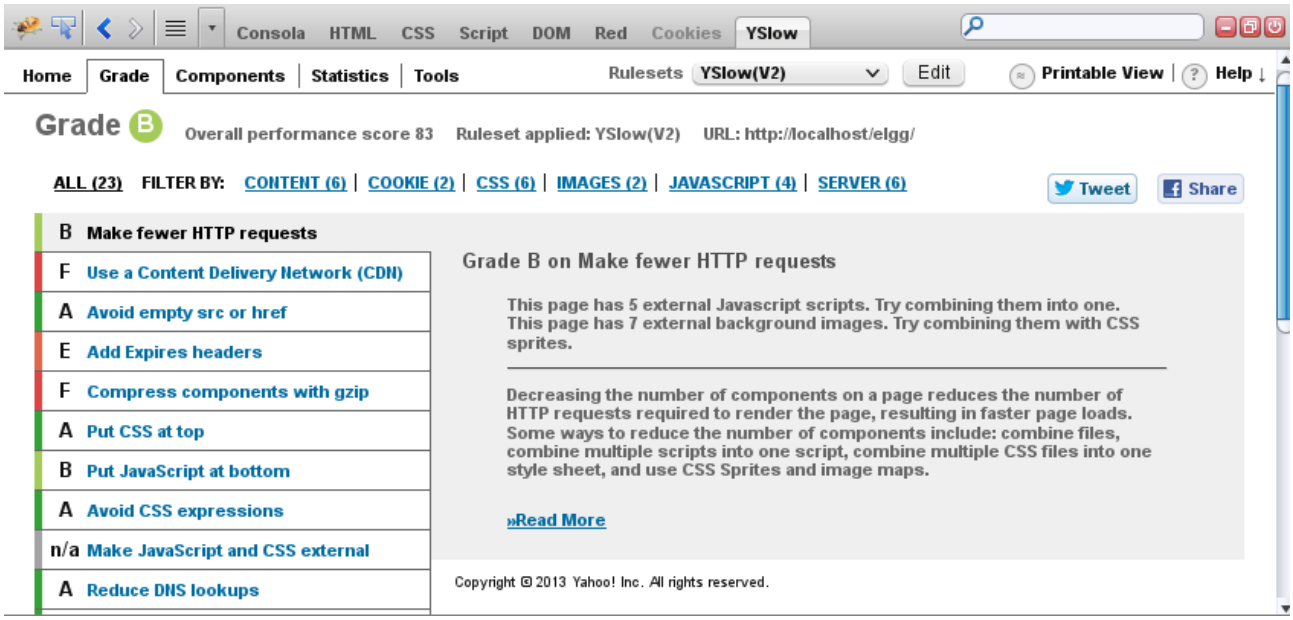

Fig 19: Informe de rendiment de la pàgina http://localhost/elgg elaborat per YSlow i Firebug

7. Manteniment

El fet que el tema s'hagi implementat com un plugin amb una estructura de directoris tan estricta facilitarà el manteniment, ja que serà més fàcil localitzar l'arxiu que conté el codi CSS que s'aplica a l'element que vulguem canviar. Si el que volem és afegir codi CSS per a un nou mòdul, caldrà incloure aquest codi en un arxiu anomenat css.php i desar-lo en una nova carpeta amb el mateix nom que el mòdul. Elgg ja l'incorporarà automàticament.

Per veure una previsualització del tema es pot fer servir el plugin develop_tools d'Elgg, al panell d'administració. Aquest connector mostra tots els elements possibles amb les normes CSS que se'ls hi aplica de manera ordenada. Amb l'ajut de l'inspeccionador d'elements de Firebug o les Developer Tools de Google Chrome es pot estudiar el codi css en detall per canviar només allò que realment faci falta canviar.

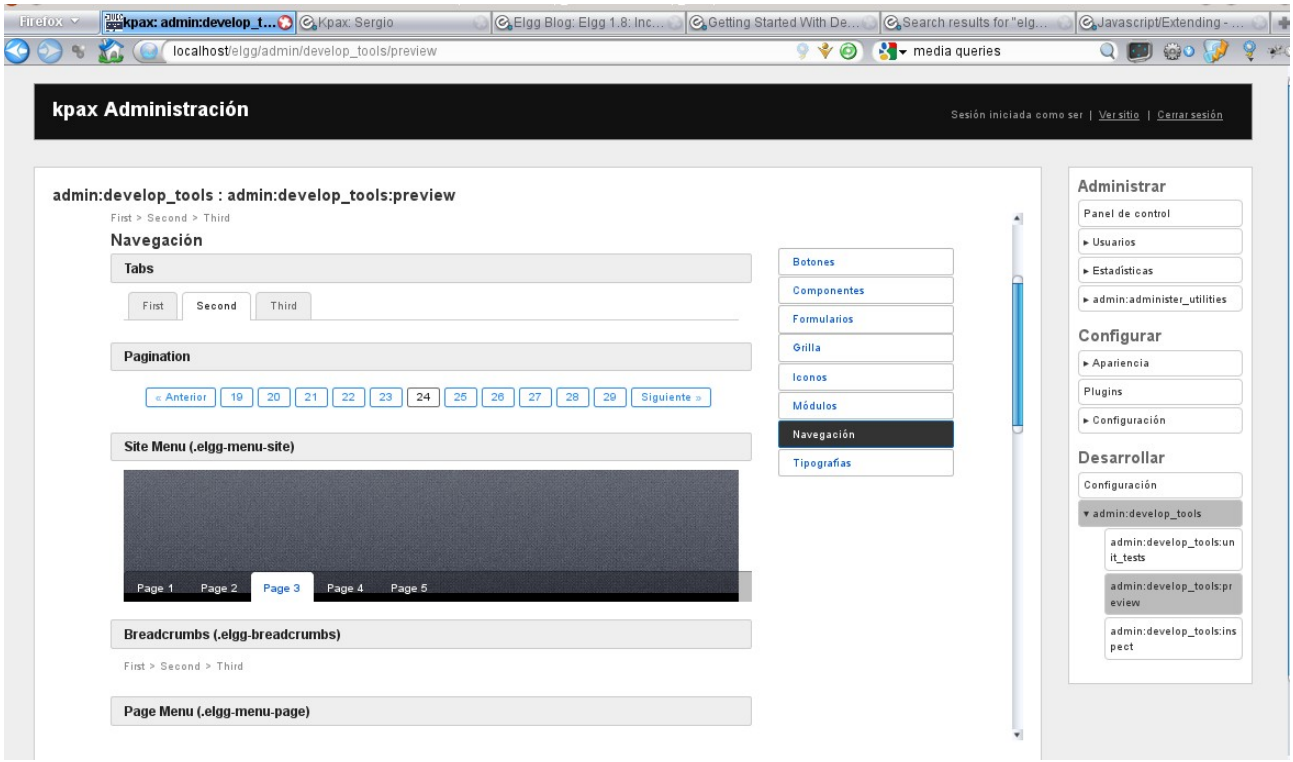

Fig. 20: connector admin develop_tools d'Elgg

8. Conclusions

En aquest treball hem estudiat com elaborar un tema propi per a un gestor de continguts que no es limiti a l'aspecte visual. Hem vist que fer un ús adequat de les funcionalitats natives de CSS i HTML que ofereixen els navegadors és sempre més eficient que realitzar la mateixa feina amb guions en JavaScript, el que comporta un estalvi significatiu en el consum de recursos i una millora en el temps de navegació. També hem vist que es pot reduir el nombre de peticions cap al servidor per agilitzar-ne la descàrrega i la visualització dels continguts al costat del client.

També hem vist que CSS aporta una capa tecnològica molt poderosa a l'hora d'adaptar l'espai web al dispositiu on es visualitza, ja sigui la pantalla d'ordinador, un projector, un dispositiu tàctil, s'imprimeixi, o fins i tot es llegeixi en dispositius braille o s'escolti mitjançant sintetitzadors de veu. A més, hem vist que amb un disseny "*responsive"* (realitzat amb *media-queries* de CSS3) es pot respondre a determinats esdeveniments provocats per l'usuari, el que permet adaptar el contingut a les noves condicions. Aquest és el cas de girar un *tablet* o canviar l'amplada de la finestra del navegador.

És important mencionar també que plantejar el tema com un connector per *Elgg* capaç de modificar la visualització del codi generat pel nucli i altres connectors en lloc de modificar directament el codi CSS d'aquests components té els seus avantatges: esdevé un únic punt de control per tot l'aspecte del lloc, simplifica la implantació i el manteniment, i també facilitarà la recuperació del sistema en cas de desastre o de rebuig del propi tema. Aquesta modularitat d'Elgg el converteix en un CMS molt potent, escalable i robust, una tria excel·lent per als propòsits de kPAX. A més de fer un ús adequat dels recursos de servidor donada una possible demanada elevada de peticions, cal que el portal faci un ús adequat dels recursos al costat del client per arribar al màxim nombre d'usuaris possibles.

Un cop instal·lat, es pot fer créixer el tema afegint un arxiu css al seu directori per cada nou connector instal·lat i Elgg l'incorporarà de manera automàtica. A més, en qualsevol moment es poden modificar els arxius d'estil del tema per realitzar canvis en l'aparença o afegir noves propietats, com per exemple animacions en CSS.

En resum, amb una navegació àgil, una funcionalitat correcta coberta amb solucions alternatives, i una maquetació que s'adapti a l'usuari (i no al revés) es pot millorar la sensació subjectiva d'experiència d'usuari. Si a més s'estructura el web i la informació que s'ofereix de manera coherent també s'optimitza el web en termes d'usabilitat.

Pel que fa al projecte que ocupa aquest document, es consideren assolits els objectius plantejats de manera teòrica per les raons tot just explicades. Ha estat una feina molt satisfactoria en posar en pràctica i millorar els coneixements en el camp del desenvolupament de *front-ends* i per poder participar en un projecte que aporta un valor a la comunitat, ja sigui de desenvolupadors o d'estudiants. KPAX es pot veure com una eina extra en l'educació de petits i grans (a tots ens agrada jugar) i especialment positiva en moments en que les retallades en educació són el pa nostre de cada dia. Esperem que només sigui el principi d'una col·laboració duradora amb aquesta xarxa oberta.

9. Recursos en línia

Dels milers d'eines d'ajuda per la construcció d'espais web existents a la xarxa es poden destacar les següents:

- http://www.w3.org Documentació i estat dels estàndards web, World Wide Web Consortium(W3C)
- http://validator.w3.org Validador de codi HTML del consorci W3C.
- http://jigsaw.w3.org/css-validator/ validador de codi CSS del consorci W3C.
- http://www.whatwg.org Web Application Technology Working Group (implementadors se navegadors). Promoció de noves funcionalitats i estàndards en tecnologies web.
- https://developer.mozilla.org/es/ Mozilla Developer Network. Portal obert a la comunitat de desenvolupadors web amb milers de documents sobre tecnologies web i demos.
- http://www.html5rocks.com/es/ Documentació i exemples d'ús d'HTML5.
- http://html5doctor.com/ Documentació, novetats i articles d'ajuda per incorporar, sobre segur, funcionalitats HTML5 en espais web.
- http://html5demos.com/ i http://html5-demos.com/. Exemples complets i funcionals d'aplicacions HTML5.
- http://caniuse.com/ Taules de compatibilitat de suport d'HTML5, CSS3, SVG i més, per navegadors en dispositius mòbils i en ordinadors. Inclou enllaços a solucions alternatives.
- http://kangax.github.com/es5-compat-table/ Taula de suport de les funcionalitats d'ECMAScript5 (JavaScript) per part dels navegadors.
- http://www.jslint.com/ Eina de control de codi JavaScript implementada per Douglas Crockford, autor de la guia *JavaScript: the Good parts.*
- http://jsperf.com/ Eina per realitzar tests de rendiment de codi JavaScript.
- http://closure-compiler.appspot.com/home Eina de minimització de codi JavaScript. Canvia els noms de les variables, elimina els comentaris i redueix les expressions redundants.

10. Bibliografia

Varis autors (2012). “Elgg's Documentation”. [en línia: http://docs.elgg.org/wiki/Main_Page]

Tommy Olsson & Paul O'Brien (2008) “The ultimate CSS Reference”. EUA:sitepoint

Dan Cederholm (2010). “CSS3 for web designers”. New York:A Book Apart. Disponible en línia.

Steve Krug (2005). “*No me hagas pensar. Aproximación a la usabilidad en la Web”*. 2ª Ed. EUA: Prentice Hall.

John Freddy Vega & Christian Van Der Henst (2011), “Guía HTML5. El presente de la web”. [e-book: http://www.etnassoft.com/biblioteca/guia-html5-el-presente-de-la-web/]

Douglas crockford (2008) “*Javascript: The Good parts*”. EUA: O'Reilly / Yahoo! Inc

Christian Johansen (2010). “*Test-Driven JavaScript Development”. EUA*: Addison-Wesley.

Annex 1: Interfícies

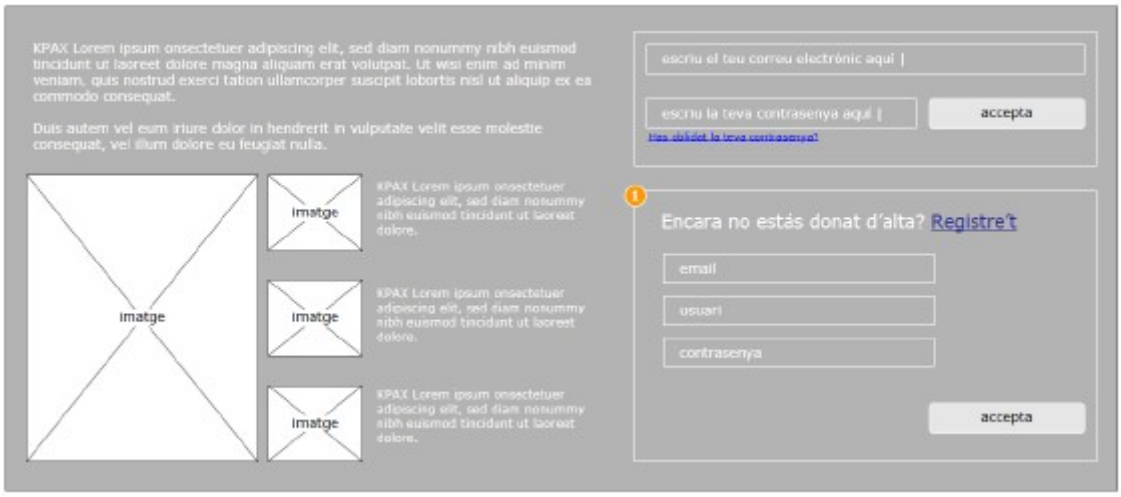

Fig.21: Pàgina inicial i de registre

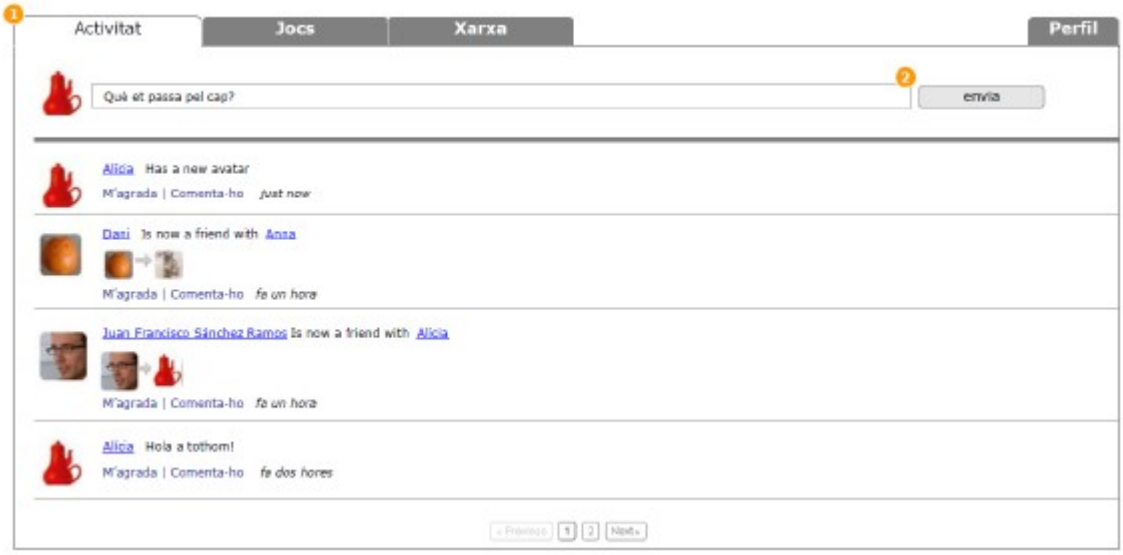

Fig.22: Activitats

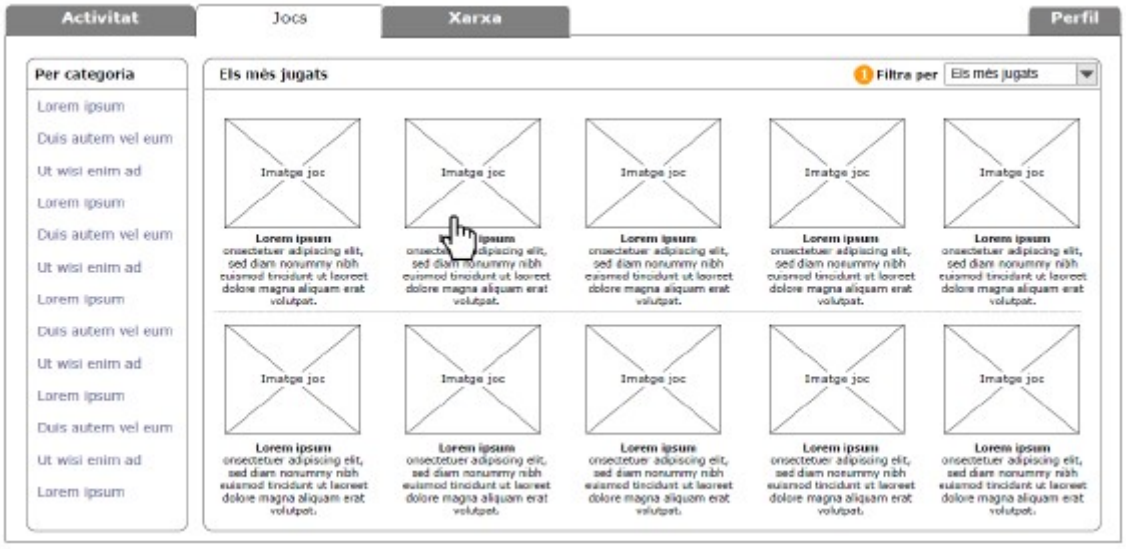

Fig. 23: Pestanya de jocs

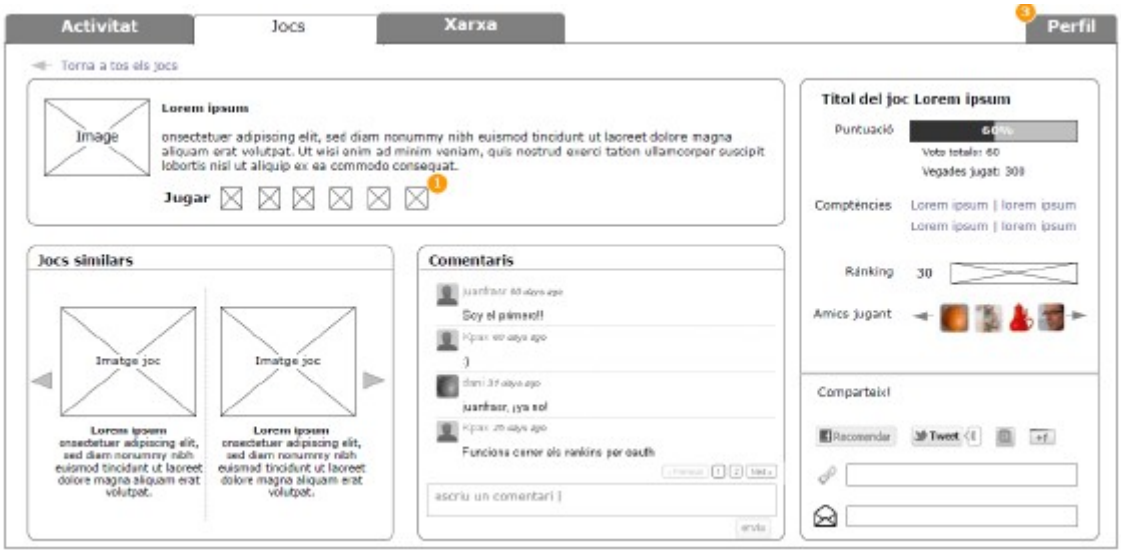

Fig.24: Detalls d'un Joc

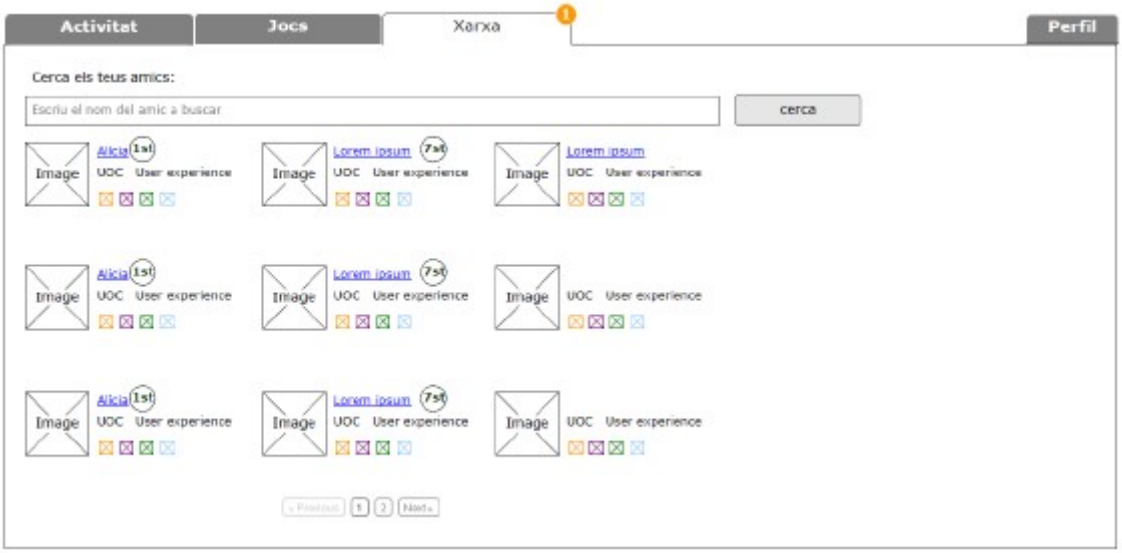

Fig.25: Xarxa d'amics

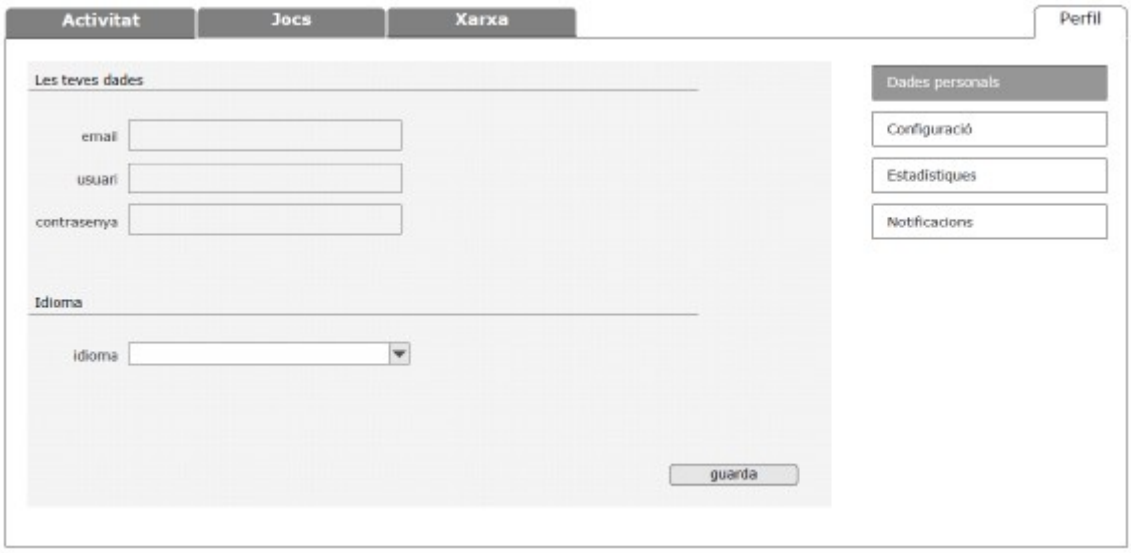

Fig 26: Perfil de l'usuari