Sourcevitality.com

Xavier Vilalta Bautista

ETIG

Miquel Àngel Senar

13 de gener de 2013

0 Resum

En aquest document s'explica el desenvolupament i desplegament d'una eina que permet obtenir informació de la vitalitat d'un projecte de programari lliure.

En realitzar projectes de qualsevol tipus, ja sigui de programació o realitzant infraestructures, moltes vegades fem servir l'ajuda de projectes de programari lliure. Un dels criteris que fem servir per a escollir si un projecte ens pot ser d'ajuda o no és l'activitat, o manca d'aquesta, del mateix. Com que aquesta informació no sempre és fàcilment accessible, aquest projecte crea una eina que permet obtenir aquesta dada de forma senzilla i còmode.

Paraules clau: vitalitat, codi font, projecte, manteniment, abandonament

Índex de continguts

Índex d'il·lustracions

Índex de taules

Índex de dibuixos

1 Introducció

En aquest capítol es fa una introducció general al projecte i la resta del document on s'entrarà més en detall en la feina realitzada.

1.1 Justificació

Avui dia, afirmar que el software lliure és un èxit és inqüestionable: des de servidor webs a sistemes operatius, passant per jocs i tota mena d'eines de desenvolupament o administració. Sigui quin sigui el projecte que emprenguem, administració, desenvolupament, etc., és molt probable que acabem trobant algun treball previ que ens sigui d'utilitat.

Aquesta gran varietat de software de vegades però, acaba sent un inconvenient, donat que és difícil escollir entre diversos projectes quin pot ser el més adient. Una de les qüestions que moltes vegades és més treballat de resoldre quant s'està fent una comparació entre diferents paquets de software lliure és fins a quin punt el projecte que ens interessa continua en desenvolupament, o si comparem més d'un projecte, quin es troba més actiu. Aquesta qüestió és rellevant en quant a la resolució d'errors (bugs), problemes de seguretat i compatibilitat amb nou software que pot fer que la nostre elecció de paquet acabi sent més i més problemàtica a mesura que avança el temps.

Aquest projecte pretén la creació d'una eina, una pàgina web, en la que sigui possible consultar, en forma gràfica, la vitalitat d'un projecte de software lliure amb la possibilitat de fer comparacions entre ells. Aquest informació s'extraurà de repositoris públics de codi com ara sourceforge.net i github.com.

1.2 Objectius

Els objectius del projecte són:

- Crear una eina àgil que permeti obtenir informació sobre la vitalitat d'un projecte
- Aconseguir que l'eina sigui tant autosuficient com sigui possible
- Utilitzar software lliure en tots els estadis del projecte (desenvolupament, manteniment, administració)

- Construir una infraestructura prou flexibles com per a que sigui senzill afegir nous orígens d 'informació

El projecte es divideix en dues parts

1. Backend

   Aquesta part s'ocuparà de l'extracció de dades i la creació de la base de dades d'informació per a utilitzar-se en el frontend. Aquest desenvolupament es farà en Python i estarà en execució en un màquina virtual dedicada.
   Ha de ser capaç de:

   1. Funcionar de manera autònoma, és a dir, amb la mínima intervenció de l'administrador possible
   2. Ser prou flexible com per acceptar ampliacions en forma d'altres pàgines de repositoris

2. Frontend

   La pàgina web pròpiament dita, en la que l'usuari podrà consultar i fer cerques i que mostrarà en forma de gràfica l'evolució d'activitat del projecte. L'objectiu és desenvolupar aquesta part fent servir Python i utilitzant una VPS dedicada.
   Els objectius són:

   1. Construir una pàgina àgil i d'aspecte modern que sigui còmode d'utilitzar
   2. Fer la pàgina tant lleugera com sigui possible per a optimitzar els recursos disponibles

1.3 Metodologia

Per determinar la metodologia a utilitzar en aquest projecte, s'han de considerar una sèrie d'elements:

1. Qui és el client

O dit d'una altra manera, qui defineix els requeriments. En aquest, en no haver-hi un client extern, la definició de requeriments es pot fer de manera àgil i és fàcil que s'adapti a futurs desenvolupaments, com tots sabem, el que moltes vegades. En realitat, si que hi ha un client extern, que és la gent que pot estar interessada en utilitzar una web d'aquest tipus, però és difícil obtenir uns requeriments precisos donat que és difícil determinar el mercat de possibles interessats i també és difícil interrogar-los sobre les seves necessitats. Això implica que el millor mètode és basar-me en la meva pròpia experiència i anar ajustant posteriorment.

2. Consideracions tecnològiques
   Aquest projecte requereix interactuar amb APIs de tercers (pàgines de repositoris). En aquestes circumstàncies, es molt possible que es trobin imprevistos tecnològics que poden obligar a tornar a definir algunes planificacions o objectius.

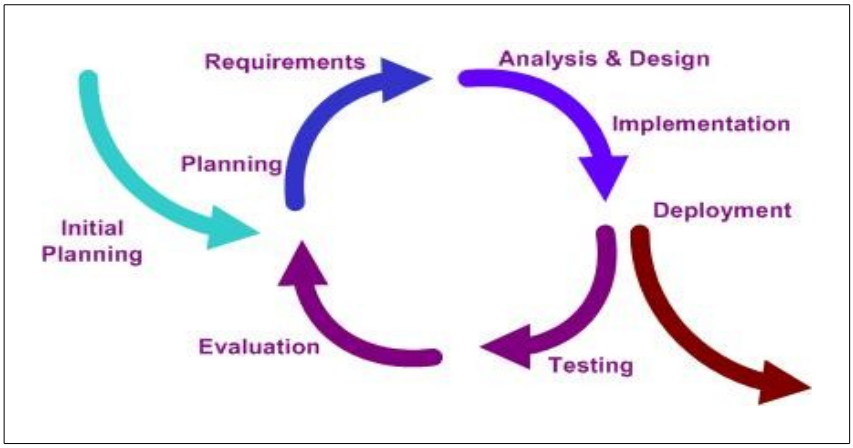
3. Recursos disponibles
   Donat que l'"equip" està format per només una persona, això ens estalvia haver d'entrar en metodologies molt feixugues pensades per incloure comunicacions àgils entre els membres que en aquest cas no són necessàries.

En aquestes condicions, una metodologia clàssica en cascada no sembla adient: es tracta d'una metodologia massa rígida que requereix un coneixement molt precís de l'evolució del projecte, per aquest motiu, és necessari algun tipus de mecanisme que permeti adaptar-se als canvis que s'aniran produint. Els mecanismes que es proposen són:

- Utilitzar un mètode iteratiu
  En aquest mètode, el model clàssic de cascada es repeteix de manera seqüencial a mesura que avança el projecte

- Ajudar-se de prototipus
  Els prototipus serviran per anar veient com evoluciona el projecte i poder fer correccions a mesura que es trobi la necessitat

El procediment es mostra en la imatge següent (de https://en.wikipedia.org/wiki/File:Iterative_development_model_V2.jpg).



Il·lustració 1: Mètode iteratiu

Per tant, primer es realitzarà una planificació inicial que correspondrà amb els objectius de la PAC2 i després s'aniran desenvolupant prototipus i avaluant com es comporten respecte als objectius finals per anar modificant el que sigui necessari.

1.4 Planificació

A continuació es detallen les tasques previstes així com la seva estimació de temps:

Àrea	Tasca	Estimació
Anàlisi i disseny	Estudi dels API de serveis de control de codi	5 jornades
	Definició d'arquitectura de software i hardware (PAC2)	5 jornades
Desenvolupament	(Backend) Creació de spider de sourceforge.net	10 jornades
	(Backend) Creació de spider de github.com	5 jornades
	(Frontend) Construcció del cercador	15 jornades
	(Frontend) Pàgina web	10 jornades
Desplegament	Preparació del sistema (VPS, altres)	5 jornades
	Domini propi	2 jornades

Taula 1: Planificació inicial

Les fites del projecte quedarien de la següent forma:

Data inicial	Data final	Fita	Contingut
01/10/2012	14/10/2012	PAC1	Pla de treball definitiu
15/10/2012	04/11/2012	PAC2	Fi de la fase de planificació inicial (arquitectura)
05/11/2012	02/12/2012	PAC3	Prototipus del Backend operatiu
03/12/2012	23/12/2012	PAC4	Prototipus del Frontend operatiu
24/12/2012	13/01/2013	Memòria final	Pàgina web completament operativa
14/01/2013	17/01/2013	Vídeo presentació	

Taula 2: Fites segons planificació inicial

1.5 Productes obtinguts

Els productes que s'obtindran com a part d'aquest projecte són:

1. Backend: El procés encarregat de recopilar les dades dels repositoris o forges junt amb el seu entorn: una màquina virtual

2. Frontend: L'aplicació encarregada de interactuar amb l'usuari i mostrar-li els resultats obtinguts pel backend

1.6 Resum de capítols

A la resta d'aquest document es detallarà com s'ha realitzat el desenvolupament i la preparació de cadascun del productes obtinguts, explicant prèviament tota l'estructura general del projecte

2 Arquitectura

En aquest capítol es detalla l'arquitectura general del projecte sense entrar en detalls de cadascun dels components que es detallaran en els capítols dedicats.

2.1 Introducció general

El sistema està dissenyat en dos components, el frontal (o frontend) i el processador (o backend)

- El processador s'encarrega de la generació i manteniment de les estadístiques necessàries sobre la vitalitat dels projectes de programari

- El frontal utilitza les dades obtingudes mitjançant el processador i les mostra en forma gràfica en funció de les cerques realitzades.

A efectes del projecte, la vitalitat o salut d'un projecte, i donat que sempre es treballa amb algun tipus de repositori de programari, es defineix com la quantitat de pujades (o commits) que es fan en un període de temps. Aquest plantejament implica que hem de ser capaços de trobar dues informacions:

1. Quins projectes existeixen

   Per aquest punt ens recolzarem en la disponibilitat de serveis d'allotjament web de programari lliure i en els seus directoris: existeixen molts serveis a la web que permeten l'allotjament de projectes de programari lliure gratuïtament o amb poc cost tals com SourceForge, GitHub, Bitbucket, Launchpad i molts d'altres. L'objectiu d'aquest projecte es centrarà en els dos primers, donat que són els més populars (dades segons Alexa) i els que compten amb més repositoris (uns 700.000 en total, segons dades de Wikipedia)
   L'eina que farem servir per obtenir les dades serà un robot o aranya web (crawler)

2. Com obtenir la quantitat de pujades per projecte per dia

   Una vegada obtingut el llistat de projectes que es treballaran, es necessita obtenir per una banda, la localització dels seus corresponents repositoris per tal de clonar-los i obtenir les dades que volem. El problema d'aquesta enfocament és que es molt difícil de portar a la pràctica

- Necessitem obtenir l'ubicació del repositori, cosa per la que no hi ha una forma genèrica possible

- Necessitem clonar una gran quantitat de dades que requereixen molta amplada de banda i molt d'espai

L'única solució factible és la de fer servir l'API que ens proporcionen tots dos serveis d'allotjament per tal d'obtenir el que necessitem que al final no és més que la quantitat de pujades per dia.

2.2 Arquitectura del processador

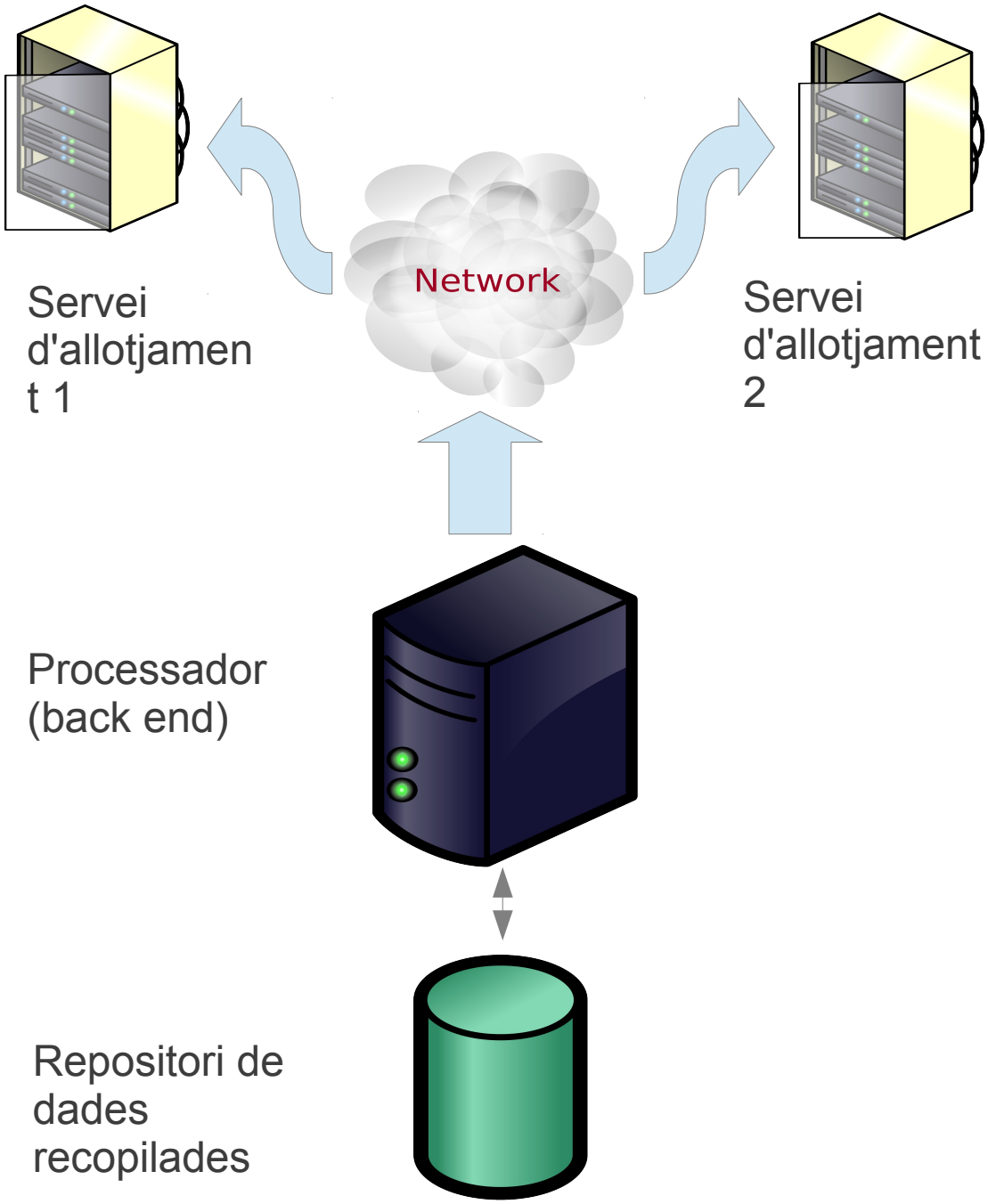
Aquest mòdul, com s'ha comentat, és el responsable de la generació de la estadístiques sobre els projectes. El funcionament per a cada servei d'allotjament escollit serà el següent

1. Mitjançant un robot web, s'obtindrà la llista de projectes allotjats

2. Fent servir el corresponent API del lloc, s'obtindrà el número de pujades per dia per projecte

Aquest procés s'anirà repetint periòdicament per anar actualitzant les dades. Es buscarà que el sistema només actualitzi les dades canviades i no totes, però això dependrà de les possibilitats que ens ofereixen els diferents serveis d'allotjament.

A continuació es mostra un diagrama de l'arquitectura del processador:



Dibuix 1: Arquitectura del processador

Per construir aquesta arquitectura es faran servir els següents components:

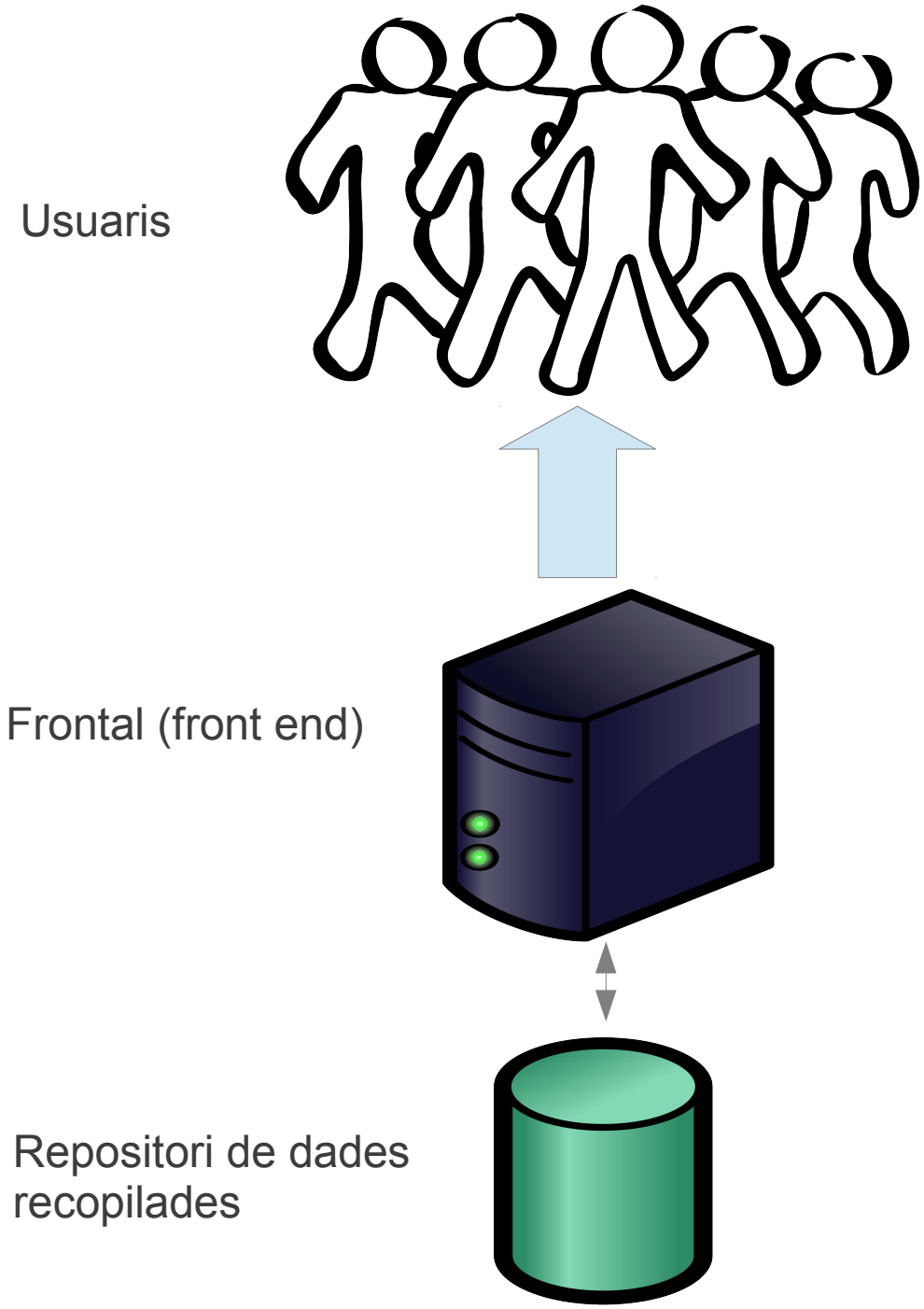
1. Python com a llenguatge principal de desenvolupament
   Python ens permet realitzar tot el desenvolupament i construcció de l'infraestructura sense haver d'anar canviant d'entorn de desenvolupament

2. Scrapy com a llibreria per realitzar el robot web
   Activa i en desenvolupament, amb bona documentació, ens ajudarà a realitzar la primera part de l'extracció de dades.

3. MySQL per a contenir les dades
   Flexible, versàtil, amb bon rendiment i ben integrat amb la resta d'eines

2.3 Arquitectura del frontal

El frontal tindrà una arquitectura similar a la reflexada a la figura següent:



Dibuix 2: Arquitectura del frontal

En aquest cas, es faran servir els següents components:

1. Flask com a microframework web en Python

Aquest entorn s'executarà al propi servidor web i serà l'encarregat d'obtenir les dades del nostre repositori i d'altres serveis i anar-los proporcionant als clients (navegadors)

2. Bootstrap com a marc per la interfície d'usuari
   Aquesta part en CSS i JavaScript s'executarà en els navegadors i serà l'encarregada de donar-nos una interfície coherent i moderna

3 Desenvolupament del processador

En aquest apartat s'explica com s'ha anat desenvolupant el processador: la seva evolució en els diferents prototipus i la justificació de les eines i recursos emprats.

3.1 Evolució

S'ha creat dos prototipus del backend, el primer fet amb SQLite i el segon amb MySQL.

En el primer prototipus es va fer servir SQLite com a motor de base de dades per motius de simplicitat. En el transcurs de les primeres proves es va determinar que potser aquest motor podria ser massa limitat, per la qual cosa es va decidir que per al segon es faria servir MySQL.

En concret, les possibles carències de la primera opció es van veure en fer les estimacions de les dades a gestionar: tot i que no es sabrà fins que no s'acabi l'execució complerta, s'estima que la base de dades només del directori de Sourceforge serien uns 5 GiB, aproximadament, si a això li afegim altres repositoris, arribem a un tamany que resultarà més fàcil de gestionar amb MySQL, tenint en compte que ens aporta:

- Més opcions d'optimització de rendiment
- Més capacitat d'escalar

En el segon prototipus ja desenvolupat en MySQL, es van trobar els següents inconvenients:

- No podia fer recollida d'actualitzacions, és a dir, cada vegada que s'executava s'ho descarregava tot, el que clarament era ineficient.
- La gran quantitat de temps necessària per executar-lo feia que fos bastant fràgil: en dues successives execucions el procés es va aturar per qüestions de hardware, que en aquest cas implica perdre (no del tot per que les dades es van bolcant a la base de dades) dies de treball.

En les següents versions, aquests inconvenients aquests inconvenients es van resoldre mitjançant un sistema que permetia continuar per l'última pàgina recopilada, de manera que el sistema es va fer més flexible a possibles errors.

El model de dades que va quedar finalment per al repositori de dades del backend és el següent:

Il·lustració 2: Model de dades del processador

3.2 Elecció de components

A continuació es detallen els components que es fan servir per al processador i els motius que justifiquen l'elecció dels mateixos. En aquest cas, l'element més important és l'eina que s'ocupa de

- Scrapy
  Com a marc d'extracció de dades web (web crawling framework), s'ha decidit fer servir Scrapy, en ser el més popular i millor suportat sobre tot si parlem d'entorn Python

- MySQL
  Com s'ha explicat abans, en primera instància es va escollir SQLite, però es va descartar.

3.3 Arquitectura física

El backend s'executa en una màquina virtual especialment preparada per a la seva execució. Executada en entorn libvirt/KVM, es tracta d'un sistema amb Ubuntu Server 12.04, al que se li han afegit un parell de components: el propi Scrapy, des del repositori oficial i una eina de monitorització (monit) que s'assegura de que el servei Scrapy no cau sense informar l'administrador.

En aquesta cas, la configuració de la màquina és bastant senzilla per què, a diferència de la que exposa el frontal, no està exposada a Internet i per tant les consideracions de seguretat són menys importants.

3.4 Resultats

Tot i que el backend està completat i ens dóna els resultats que esperàvem, en el seu estat actual, té les següents limitacions que és necessari reconèixer per poder identificar les millors formes de perfeccionar-lo:

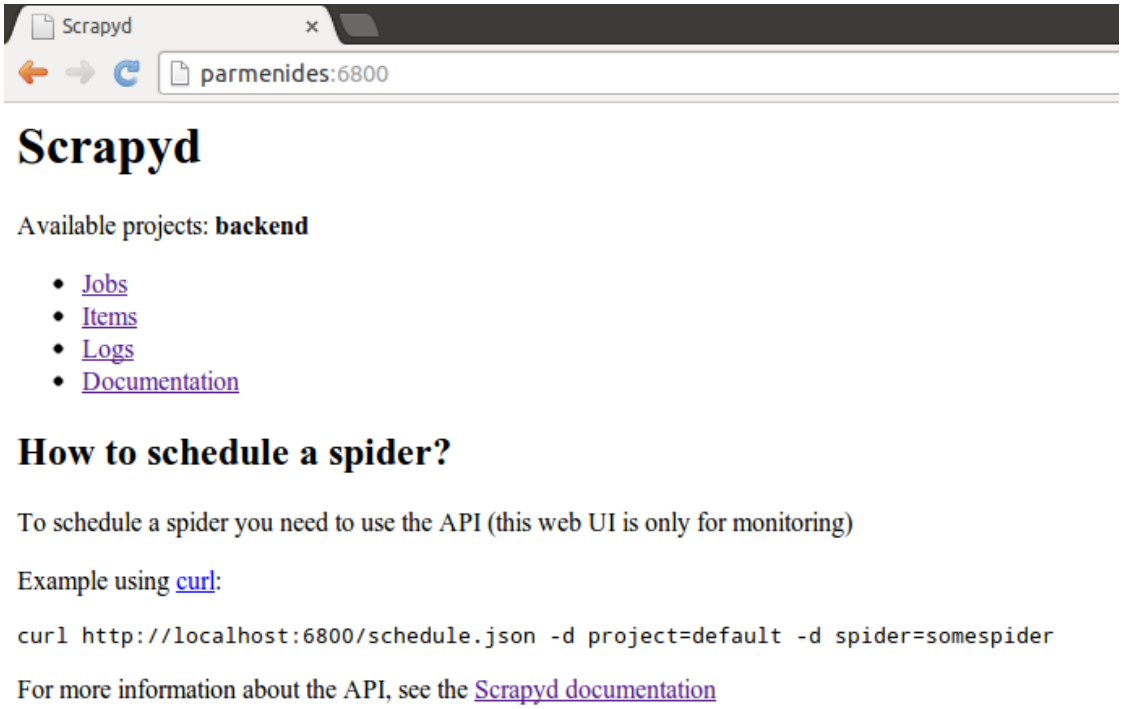
- Es bastant lent, l'estimació és que la seva execució portarà uns vint dies. Sembla una quantitat molt exagerada de temps però s'ha de tenir en compte que: l'amplada de banda de pujada no és gaire alta i la quantitat de dades a recollir

és important en tractar-se de tots els commits fets a tots els projectes d'un repositori com el de Sourceforge des d'el principi de la seva existència. Tot i així, és possible que existeixin possibilitats d'optimitzar-ho que s'hauran d'estimar més endavant.

- Està limitat a un únic repositori, Sourceforge que, tot i que segons la Viquipèdia és el que més projectes conté, no compta amb el dinamisme que pot tenir Github, per exemple.

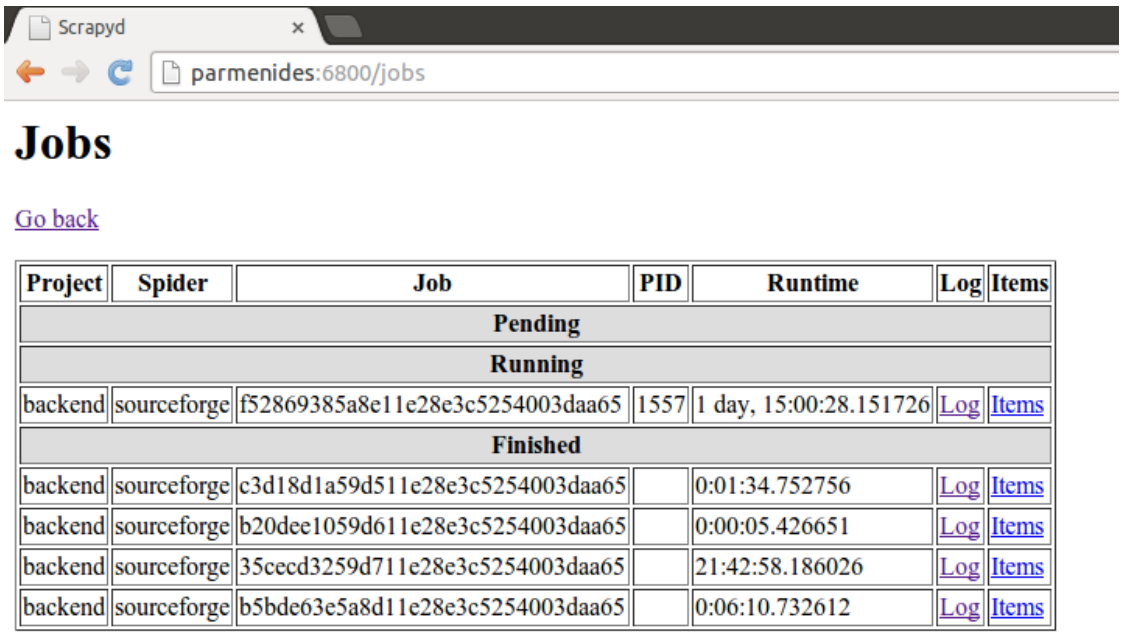
Tot i aquestes limitacions, es va decidir continuar el projecte per la part del frontend i de moment considerar aquesta versió del backend com a prou bona, deixar-la aturada i continuar el progrés en la segona àrea. Principalment per les limitacions de temps que podrien acabar per impedir tenir un projecte operatiu encara que sigui amb una funcionalitat una mica més limitada de la prevista inicialment.

A continuació es pot veure el processador en funcionament a la seva màquina virtual, integrat amb la interfície web del Scrapy:



Il·lustració 3: L'accés principal del projecte backend

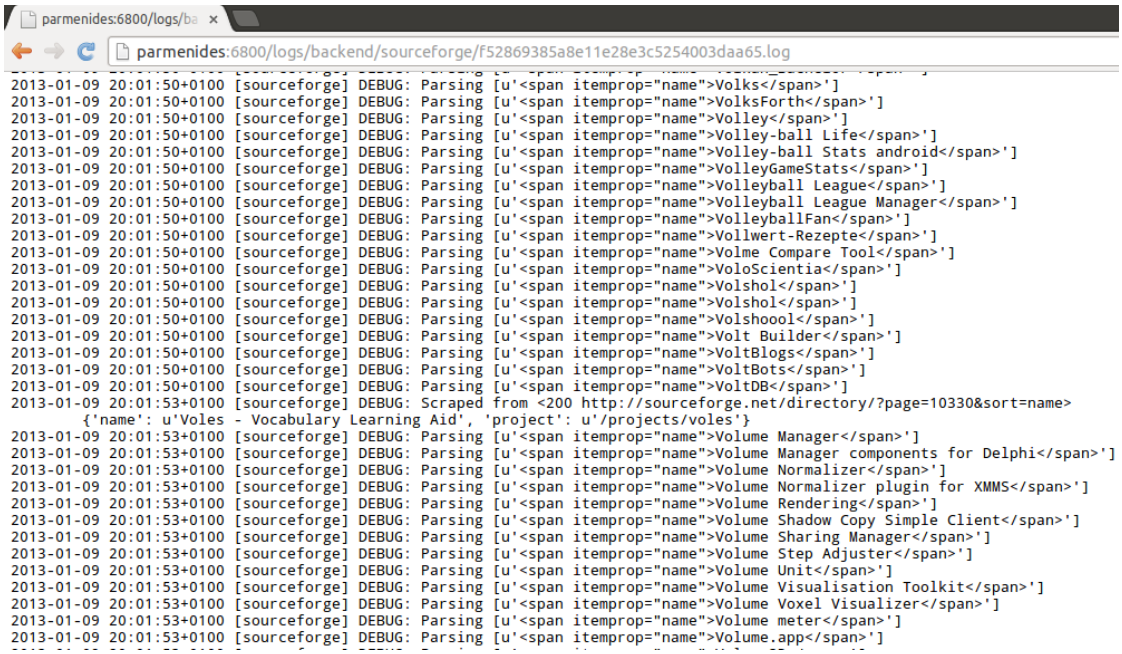
Un moment de l'execució de les tasques d'extracció de dades:



Project	Spider	Job	PID	Runtime	Log	Items
Pending						
Running						
backend	sourceforge	f52869385a8e11e28e3c5254003daa65	1557	1 day, 15:00:28.151726	Log	Items
Finished						
backend	sourceforge	c3d18d1a59d511e28e3c5254003daa65		0:01:34.752756	Log	Items
backend	sourceforge	b20dee1059d611e28e3c5254003daa65		0:00:05.426651	Log	Items
backend	sourceforge	35cecd3259d711e28e3c5254003daa65		21:42:58.186026	Log	Items
backend	sourceforge	b5bde63e5a8d11e28e3c5254003daa65		0:06:10.732612	Log	Items

Il·lustració 4: Estat d'algunes execucions

Un registre de resultats d'extracció, on es poden anar veien com s'extreuen les dades de cada projecte disponible a cada pàgina del directori:



Il·lustració 5: Registre de resultats

4 Desenvolupament del frontal

En aquest apartat s'explica com s'ha anat desenvolupant el frontal: la seva evolució en els diferents prototipus i la justificació de les eines i recursos emprats.

4.1 Evolució

El desenvolupament del frontal ha sigut la part més dificultosa pel menor coneixement inicial del desenvolupament en JavaScript.

Un dels plantejaments inicials era fer servir algun tipus de servei al núvol per a mantenir la part del frontal, però de moment es va descartar per falta de dades addicionals: s'ha de tenir en compte que el cost del manteniment al núvol depèn de conèixer molt bé quin és el tipus d'activitat que es fa, però aquestes dades no es tindran fins que el sistema estigui rodat un temps.

Al llarg del desenvolupament es va decidir que, tot i fer servir el mateix suport de dades, MySQL, era recomanable separar l'estructura de dades del frontal del processador, donat que tenen requeriments molt diferents, més endavant aquest punt es detalla una mica més. El model de dades definitiu és el següent:

Il·lustració 6: Model de dades del frontal

4.2 Elecció de components

Els elements necessaris per a la construcció del frontal han sigut molts i variats:

- La base es farà mitjançant el framework (o microframework) Flask

  Els motius pels qual s'ha escollit Flask són:

  - Simplicitat d'ús
  - Simplicitat de configuració
  - Bona documentació
  - Comunitat gran, per tant, bones possibilitats d'obtenir ajuda en forma de blocs o altres

  Altres alternatives que es van plantejar i que es van descartar són:

  - Django: Tot i que és un entorn molt potent i amb moltes possibilitats, s'ha decidit no fer-lo servir per ser massa gran per l'objectiu del projecte
  - Addicionalment, es van descartar altres alternatives no Python (com Node.js) novament per simplicitat, amb l'objectiu de construir tota la pila de l'aplicació en un únic entorn. En haver-hi un únic programador, es preferible no dispersar l'experiència

- El repositori de dades es farà amb MySQL

  Una decisió que s'ha pres durant el desenvolupament del backend ha estat separar el repositori de dades del backend del del frontend. D'aquesta manera serà més fàcil optimitzar per a les seves diferents necessitats i independitzar més el treball entre projectes. De moment, també s'ha escollit MySQL per al repositori del frontend (però amb una base de dades amb una estructura diferent), en particular per:

  - Entorn polivalent, provat i versàtil en entorns web
  - Moltes possibilitats d'optimització

  Dit d'una altra manera: MySQL, amb la informació disponible actualment és una aposta segura, encara que segurament millorable. Més endavant, amb més dades respecte al comportament, es poden provar altres opcions, per exemple, el frontend no necessita escriure al repositori, per tant, podríem treballar amb altres sistemes optimitzats per lectura sense la sobrecàrrega d'un gestor de bases de dades relacionals.

En particular, la definició de les taules fan servir una particularitat de MySQL i és que compta amb diferents motors per gestionar les dades: en aquest cas, ens en decantat per fer servir MyISAM en ser un motor que no afegeix la càrrega de la gestió ACID innecessària en aquest cas. Per aquest motiu a més, requereix menys memòria i més recursos que podrem alliberar.

- Com a servidor d'aplicacions es farà servir uWSGI

  De entre la gran quantitat de formes que hi ha de servir aplicacions web amb Python (un interessant anàlisi el trobem a http://nichol.as/benchmark-of-python-web-servers), la de uWSGI sembla de moment la més adient: Gevent es incompatible amb varies llibreries (encara que seria la millor opció en fer servir threads lleugers, més que suficients per només llegir) motiu pel qual no s'escoll
  La opció de uWSGI sembla a priori millor per la menor carga de memòria respecte a Apache.

- Servidor de contingut estàtic Nginx

  La bona integració entre Nginx i uWSGI i el seu baix nivell de recursos fa que escollim aquesta opció com a servidor web estàtic: no només s'ocuparà de servir recursos com ara estils CSS o imatges, si no que també serà el que derivarà les crides corresponents al servei uWSGI

- Initializr

  A www.initializr.com es troba una plantilla genèrica per la construcció de webs que, entre d'altres punts inclou
  - Eines de compatibilitat entre navegadors
  - Plantilles de pàgines integrades amb Bootstrap
  - Configuracions recomanades per a servidors web (Nginx, en aquest cas)

  Algunes opcions desestimades:
  - Començar de zero

    El mon del desenvolupament web és un mon molt complex i extens per fer-ho. Utilitzar plantilles preexistents permet reaprofitar un coneixement i una experiència que una persona sola no té.
  - HTML5 Boilerplate

Inclòs en Initializr però sense integració amb Bootstrap

- Twitter Bootstrap

  Aquesta llibreria de fulles d'estils permet donar un aspecte modern a la aplicació web, sense haver de ser dissenyador.

  En quant a altres alternatives, no hi ha gaires, a part de fer-ho un mateix, com es pot veure per exemple en aquesta qüestió a Stack Overflow:
  http://stackoverflow.com/questions/9212536/alternatives-to-twitter-bootstrap

- Sphinx Search

  Aquesta eina permet la cerca de text natural en les dades recopilades pel backend. Amb aquesta eina, l'usuari pot buscar per noms aproximats i de manera ràpida i eficient les informacions que l'interessen. A l'annex es veu un exemple de cerques i temps

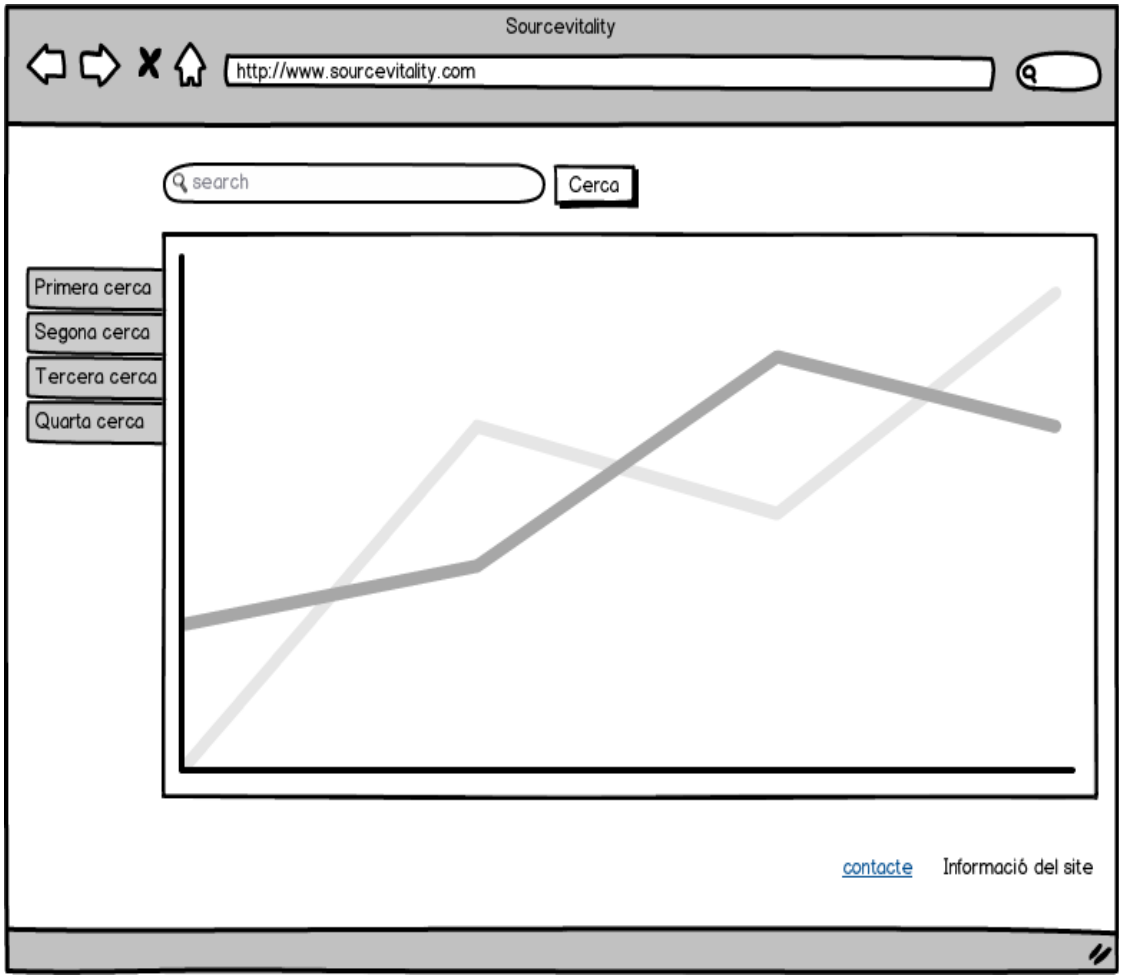
  Altres opcions, com fer servir la funcionalitat pròpia o eines com Solr s'han descartar per l'excel·lent integració de Sphinx amb MySQL que fa que la instal·lació, configuració i desenvolupament siguin molt simples, i per la qualitat de les cerques i el rendiment i recursos necessaris respecte altres opcions.

- Flot

  Aquesta llibreria en JavaScript permet la creació de gràfics de línies, permeten per tant la visualització de les gràfiques d'evolució.

  Altres opcions desestimades han sigut Highcharts, en ser de pagament, Google Chart API, descartar en necessitar connexió permanent als servidors de Google i per tant per l'excessiu consum de xarxa que a la VPS on s'executarà està limitat. També s'ha desestimat jQuery SVG en estar més menys mantingut i Raphaël en ser Flot més simple de treballar-hi i per tant, més ràpid de desenvolupar. Un bon resum de les diferents possibilitats disponibles es pot veure a http://kraskniga.blogspot.com.es/2012/06/comparison-of-javascript-data.html

A continuació es mostra un esbós de com es pretén desenvolupar l'aspecte de la pàgina web a nivell d'usuari:



Il·lustració 7: Esbós del frontal

4.3 Arquitectura física

El frontal s'executa sobre una màquina virtual allotjada a Linode, que es va escollir per les bones referències obtingudes respecte a la qualitat del servei i el suport nadiu per IPv6, Aquesta màquina virtual funciona sobre Xen i té instal·lat un Ubuntu Server 12.04 LTS. Els motius de l'elecció d'aquest entorn són familiaritat i estabilitat, en ser l'edició de llarg suport.

Els serveis que configuren l'entorn necessari són els següents, tal com s'ha comentat en els punts anteriors són:

- MySQL
- Sphinx Search
- uWSGI
- Nginx

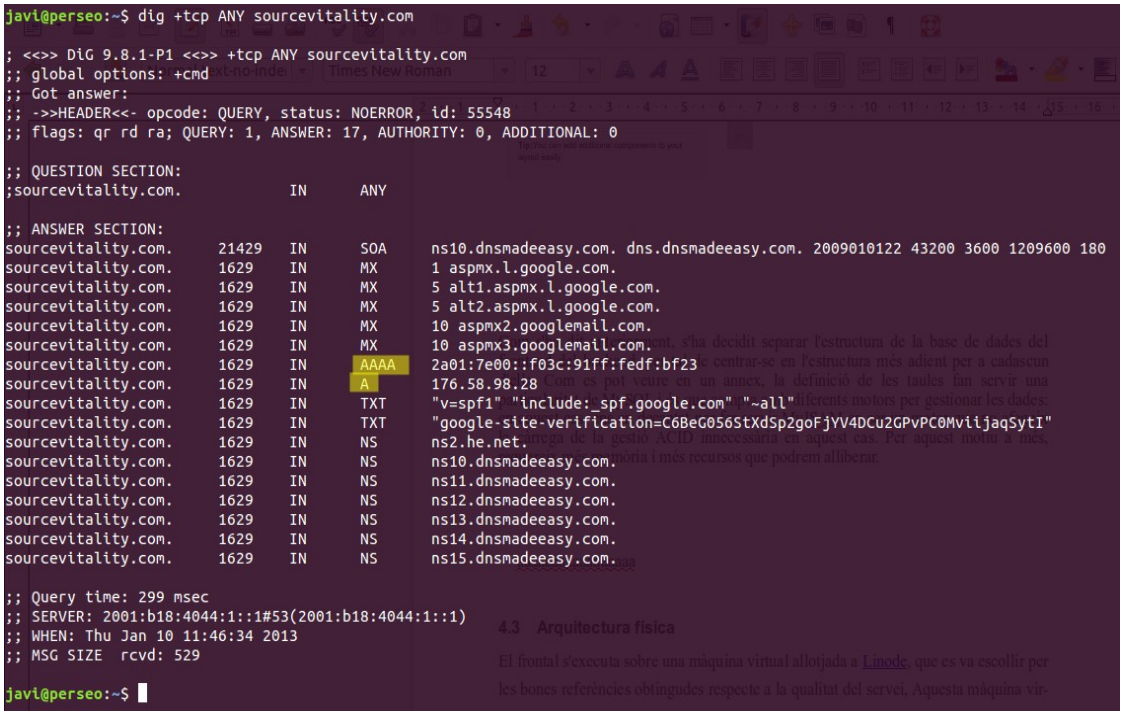
La configuració que s'utilitza està basada en l'obtinguda a Initializr, tal com s'explica anteriorment, més algunes modificacions suggerides per l'eina de seguretat w3af

A més, s'han preparat una sèrie de serveis administratius per millorar l'administració i seguretat:

- Fail2ban
  Com és habitual, tota l'administració del servidor es fa mitjançant SSH. Per millorar la seguretat es fa servir el dimoni fail2ban que bloqueja els accessos erronis al servidor.

- Monit
  Aquest servei s'ocupa de garantir que els serveis necessaris estan en marxa i no que no estan consumint massa recursos. Cas que sigui així, s'avisa mitjançant un correu electrònic o reinicia el servei corresponent. D'aquesta manera, el propi sistema és capaç de recuperar-se d'errors puntuals administratius o d'altres tipus o, com a mínim, informa l'administrador per a que prengui mesures.

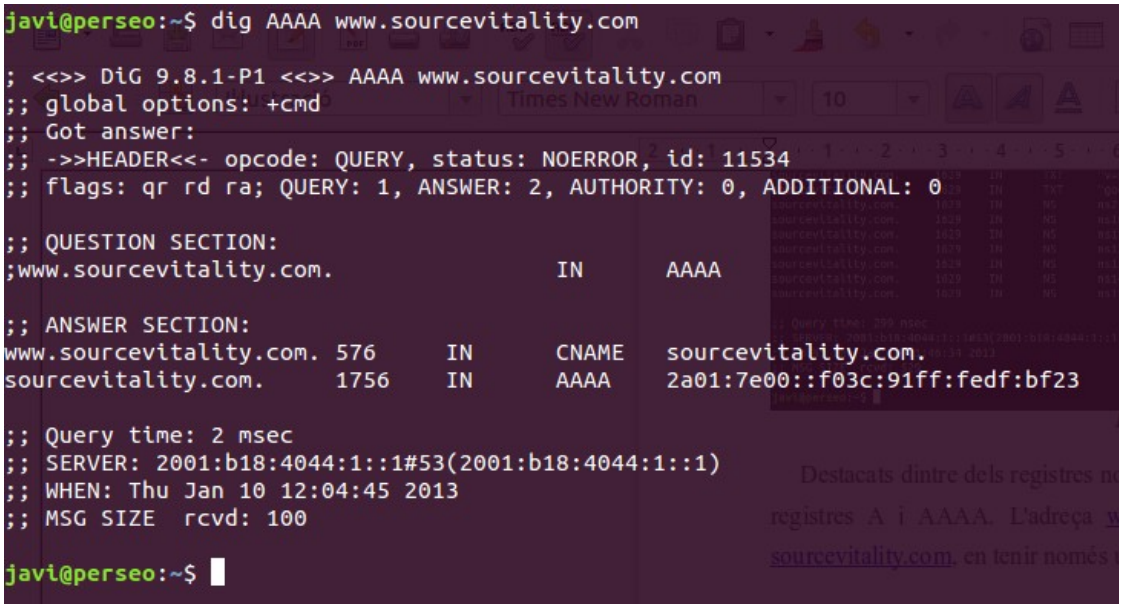
Independentment de la configuració d'aquest servidor, és necessari configurar les DNS per a que els usuaris puguin accedir. A continuació es pot veure els registres DNS per al domini:



Il·lustració 8: Registres DNS

Destacats dintre dels registres no directament implicats en l'accés a la web estan els registres A i AAAA. L'adreça www.sourcevitality.com és un CNAME del nom sourcevitality.com, en tenir només una IPv4 disponible:

Il·lustració 9: Resolució DNS IPv4 de la pàgina



Il·lustració 10: Resolució DNS IPv6 de la pàgina

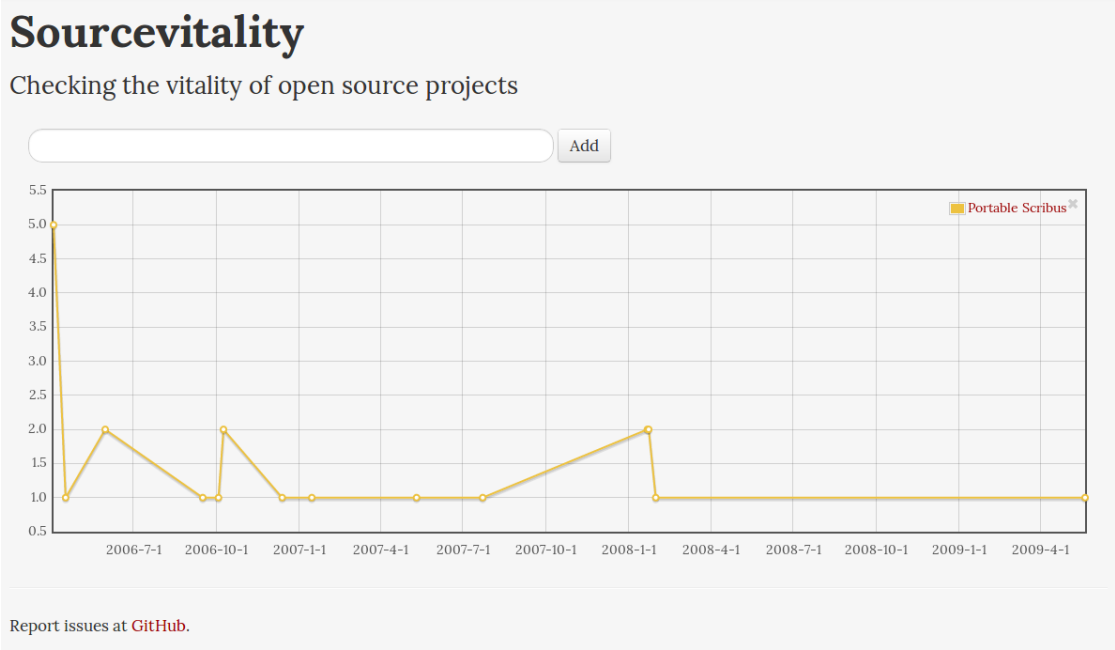
Com es pot veure, s'ha procurat que la pàgina sigui accessible tant mitjançant IPv4 com IPv6, no només en la configuració del DNS si no també en la configuració dels diferents elements per proporcionar el servei, tals com Nginx.

Tot el que impliqui publicar serveis a Internet requereix contemplar curosament el tema de la seguretat. En aquest cas s'han considerat fail2ban; l'elecció de la configuració del Nginx, avaluada, encara que sempre s'ha de prendre només com a orientació, amb eines de comprovació automàtica de vulnerabilitats; l'elecció d'una plataforma amb suport, la qual cosa implica actualitzacions de seguretats garantides; un firewall que evita accessos a ports que no siguin els del serveis publicats (80 i 22), etc.

4.4 Resultats

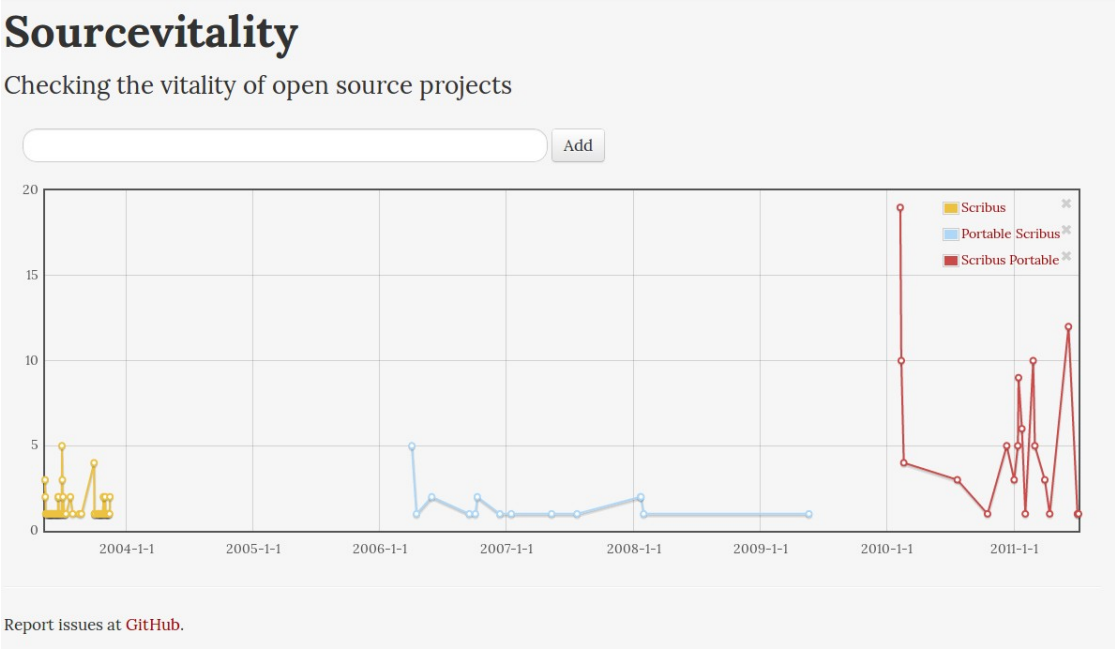
El frontal ha quedat completat amb els objectius inicials: és àgil, senzill i dóna les respostes a les preguntes que es planteges molt ràpidament.

A continuació es poden veure alguns exemples d'execució:



Il·lustració 11: Un gràfic de mostra

També és capaç de fer comparacions entre projectes.



Il·lustració 12: Una comparació

5 Valoració econòmica

Encara que sigui només a efectes informatius, és interessant fer un petit estudi econòmic del cost d'aquest projecte. Si agafem com a orientació la planificació estimada i agafem com a referència un preu per hora de 20 €, aleshores tenim que el cost, aproximat del projecte és:

Àrea	Estimació (jornades)	Cost hora	Cost total
Anàlisi i disseny	10	20,00 €	1.600,00 €
Desenvolupament	40	20,00 €	6.400,00 €
Desplegament	7	20,00 €	1.120,00 €

Taula 3: Cost del desenvolupament

Afegint-hi tots els serveis externs necessaris:

Concepte	Import
Desenvolupament	9.120,00 €
VPS (1 any)	173,07 €
Registrar (3 any, 3 dominis)	22,68 €
DNS hosting (1 any)	22,59 €
Total	*9.338,34 €*

Taula 4: Costs totals

Els valors són aproximats perquè el tipus de canvi de dòlars a euros que s'ha fet servir no ha sigut el real, però ens donen una xifra prou bona.

6 Conclusions

L'eina que ens vam proposar construir en primera instància està operativa amb els objectius que ens vam plantejar en un principi, excepte la recopilació de dades de Github que s'ha hagut de deixar al marge per requeriments de temps. La intenció és continuar treballant en aquest projecte i completar el spider de GitHub abans de fer-ho disponible a un públic més gran.

Tot i així, el gran problema són les dades recopilades de Sourceforge: la qualitat de les dades no és gaire bona principalment per dos motius:

- Una part important dels projectes que es mantenen a Sourceforge sembla que estan abandonats o han tingut molt poca activitat. Això fa que hi hagi moltes dades innecessàries.

- Per una altra banda, molts projectes populars, com ara VLC o DOSBox, per agafar alguns que es destaquen com a popular a les pàgines de Sourceforge, en realitat tenen el seu propi repositori extern, tot i que originalment si que és cert que van estar a Sourceforge. Això fa que actualment el sistema torni dades fins a una determinada data, com si el projecte s'hagués abandonat, però en realitat continua viu en un altre lloc. Serà necessari completar aquest informació per a no donar dades falses als usuaris.

Això fa que, tot i tenir tota la infraestructura muntada, encara serà necessari completar com a mínim el rastrejador de GitHub per a tenir una web més operacional.

Considerem que aquest ha estat un projecte molt complet des d'un punt de vista purament tècnic: per poder fer-se s'ha hagut de treballar una gran quantitat d'àrees, no només programació i desenvolupament en diversos llenguatges de programació (JavaScript i Python, a més de HTML i CSS), si no també aspectes d'administració de sistemes incloent-hi aspectes molt rellevants com ara seguretat.

Igualment, des d'un punt de vista més personal, treballar en aquest projecte m'ha permès treballar amb tecnologies i entorns amb els que no havia treballat gaire ni per motius professionals ni acadèmics com ara Python i JavaScript.

També és important destacar la gran quantitat de recursos i tecnologies disponibles per al professionals de la informàtica que existeixen: sense Flask, Python, Flot i MySQL, per posar uns exemples, aquest projecte hauria requerit molt més temps per completar-se. És en aquest context com una eina com la creada en aquest projecte aspira a ser un element més d'ajuda en la decisió de quines són les millors opcions per a un treball concret.

7 Glossari

Llista d'abreviatures i símbols

Terme	Definició
API	*Application Program Interface*, interfície de programa d'aplicació. Protocol de comunicació entre un programa que publica una sèrie de serveis i un programa extern
Backend	O back end, que també és correcte. Tot paquet de programari que no és visible per l'usuari final però que és necessari pel funcionament d'un sistema
crawler	Un sinònim de spider
Forja	Un lloc web que ofereix una sèrie de serveis per a la creació de projectes de programari, com ara repositoris de codi, fòrums i altres eines
Frontend	O front end, que també és correcte. Qualsevol paquet de programari que serveix com a interfície d'un sistema
GitHub	La forja de programari més popular dels últims temps
Sourceforge	Una de les forges més populars de programari lliure, que té la quantitat més gran de projectes. També és un dels més antics.
spider	Programa que fa un recorregut per un conjunt de pàgines web amb l'objectiu de recopilar dades
VPS	*Virtual Private Server*, servidor virtual ubicat a Internet que es pot utilitzar per a tot tipus de projectes sense tenir que preocupar-se del manteniment del hardware

8 Bibliografia

Aquest ha sigut més aviat un projecte de referències a la Xarxa que no pas un projecte de referències bibliogràfiques clàssiques, per la qual cosa gairebé totes les referències son en línia.

Stack Overflow, URL: http://stackoverflow.com/ [Visitat 03/01/2013].

Sphinx Search, URL: http://sphinxsearch.com/ [Visitat 03/01/2013].

Flask, URL: http://flask.pocoo.org/ [Visitat 03/01/2013].

Iterative and incremental development, URL: https://en.wikipedia.org/wiki/Iterative_and_incremental_development [Visitat 03/01/2013].

Software architecture, URL: https://en.wikipedia.org/wiki/Software_architecture [Visitat 03/01/2013].

Front end and back end, URL: https://en.wikipedia.org/wiki/Front_end_and_back_end [Visitat 03/01/2013].

9 Annexos

Annex I: Codi font

A continuació s'inclouen les fonts del processador i el frontal. La millor manera de veure aquestes fonts és a BitBucket:

Processador:

https://bitbucket.org/eudemo/sourcevitality-backend

Frontal:

https://bitbucket.org/eudemo/sourcevitality-frontend

```
"""
Crawls sourcefourge.net directory and gets items
"""

import logging
import re
from scrapy.contrib.spiders import CrawlSpider, Rule
from scrapy.contrib.linkextractors.sgml import SgmlLinkExtractor
from scrapy.selector import HtmlXPathSelector
from backend.items import SourceforgeItem
from backend.database import create_database, get_start_page,

global start_page

def process_value(value):
    m = re.search("page=(\d+)", value)
    if m:
        if m.group(1) < start_page:
            return None
    return value

class SourceforgeSpider(CrawlSpider):
    """
    Parses sourceforge.net directory

    @returns items 1
    @scrapes project name
    """

    name = 'sourceforge'
    allowed_domains = ['sourceforge.net']
    try:
        conn = create_database()
    except Exception as detail:
        logging.error('Unable to open database: ' + detail.message)
        raise
    global start_page
    start_page = get_start_page(conn, name)
    start_urls = [
        'http://sourceforge.net/directory/?sort=name&page=%s'
    ]
    rules = (
        # Get next pages skipping recursions to the first one
        Rule(SgmlLinkExtractor(allow=('sort=name', ),
                               deny=('page=%s\Z' % start_page, 'page=%s&' % start_page, ), process_value=process_value),
             callback='parse_item', follow=True, ),
        )

    def __init__(self, *a, **kw):
```

File - /home/javi/Documents/projectes/SourceVitality/src/backend/export/sourceforge.py

```
50                # TODO: Is this really necessary?
51                super(SourceforgeSpider, self).__init__(*a, **kw)
52
53            def parse_item(self, response):
54                """
55                Select item from directory based on schema
56                :rtype : SourceforgeItem[]
57                """
58                m = re.search('(?<=page=)\w+', response.request.url)
59                if m is not None:
60                    update_reference(self.conn, self.name, m.group(0
61
62                hxs = HtmlXPathSelector(response)
63                sites = hxs.select('//li[contains(@itemtype, '
64                                   '"http://schema.org/
MobileSoftwareApplication")]'
65                                   '/div/header/a[contains(@itemprop
, "url")]')
66                items = []
67                for site in sites:
68                    self.log('Parsing %s' % site.select('span').extr
act())
69                    item = SourceforgeItem()
70                    item['project'] = re.sub(r'/\?source=directory',
71                                             site.select('@href').extract()[0])
72                    item['name'] = site.select('span/text()').extrac
t()[0]
73                    items.append(item)
74
75                return items
76
```

```
#!/usr/bin/env python
#pylint: disable=E602

from fabric.api import env, put, task, settings, sudo
import importlib


def import_settings():
    """
    Import the right settings for an environment
    """

    # See http://stackoverflow.com/a/6098238/149323
    import os, sys, inspect

    cmd_folder = os.path.realpath(os.path.abspath(os.path.split(inspect.getfile(inspect.currentframe()))[0]))
    if cmd_folder not in sys.path:
        sys.path.insert(0, cmd_folder)

    return importlib.import_module('settings.%s' % env.host_string)


def sudo_user(command):
    """
    Runs a command with sudo in the environment of the destination user

    The shell command is a bit tricky because shell can be disabled for the user
    """
    s = import_settings()
    return sudo('su ' + s.SETTINGS_DESTINATION_USER + ' -l -s /bin/sh -c "' + command + '"')


@task
def copy_setup():
    """
    Copy configuration file
    """

    s = import_settings()
    with settings(warn_only=True):
        sudo_user('mkdir %s' % s.SETTINGS_DESTINATION_CONFIG_PATH)
    put('settings/' + s.SETTINGS_SOURCE_CONFIG_FILE, '/tmp/')
    sudo_user('cp -f /tmp/%s .sourcevitality/backend.cfg' % s.


@task
```

File - /home/javi/Documents/projectes/SourceVitality/src/backend/export/fabfile.py

```
47 def deploy_project(target='default'):
48     """
49     Deploys the project to a target
50     """
51
52     import scrapy.cmdline
53
54     args = [
55         'scrapy',
56         'deploy',
57         target
58     ]
59     scrapy.cmdline.execute(args)
60
61
62 @task
63 def run_spider(spider):
64     """
65     Runs a spider locally
66     """
67
68     import scrapy.cmdline
69
70     args = [
71         'scrapy',
72         'crawl',
73         '--profile=/tmp/sourceforge.profile',
74         '--pidfile=/tmp/sourceforge.pid',
75         spider
76     ]
77     scrapy.cmdline.execute(args)
78
79
80 @task
81 def schedule_spider(host, spider, project='backend'):
82     """
83     Schedules a new crawling on the specified host
84     Like curl http://localhost:6800/schedule.json -d project=
   backend -d spider=sourceforge
85     """
86
87     import urllib
88     import urllib2
89
90     req = urllib2.Request('http://%s:6800/schedule.json' % hos
   t)
91     req.add_data(urllib.urlencode({'project', project}))
92     req.add_data(urllib.urlencode({'spider', spider}))
93     urllib2.urlopen(req)
94
95 @task
96 def cancel_last_job(host, project='backend'):
```

File - /home/javi/Documents/projectes/SourceVitality/src/backend/export/fabfile.py

```
    """
    Cancel last scheduled job
    Like curl http://localhost:6800/cancel.json -d project=backend -d job=1517733c3fce11e28ba35254003daa65
    And using curl http://localhost:6800/listjobs.json?project=backend
    """

    import simplejson
    import urllib
    import urllib2

    req = urllib2.Request('http://' + host + ':6800/listjobs.json?project=' + project)
    opener = urllib2.build_opener()
    f = opener.open(req)
    j = simplejson.load(f)
    id = j['running'][-0]['id']

    req = urllib2.Request('http://%s:6800/cancel.json' % host
    req.add_data(urllib.urlencode({'project', project}))
    req.add_data(urllib.urlencode({'job', id}))
    urllib2.urlopen(req)

```

File - /home/javi/Documents/projectes/SourceVitality/src/backend/export/fabfile.py

```
# For more information about the [deploy] section see:
# http://doc.scrapy.org/topics/scrapyd.html

[settings]
default = backend.settings

[deploy]
url = http://parmenides:6800/
project = backend
version = GIT

[deploy:empedocles]
url = http://localhost:36800/
project = backend
version = GIT

```

File - /home/javi/Documents/projectes/SourceVitality/src/backend/export/scrapy.cfg

```
#!/usr/bin/env python

"""
Testing module for utils
"""

import unittest
import doctest
import backend.database


class Test(unittest.TestCase):
    """Unit tests for utils."""

    def test_doctests(self):
        # TODO: Need to review
        """Run utils doctests"""
        doctest.testmod(backend.database)

if __name__ == "__main__":
    unittest.main()

```

File - /home/javi/Documents/projectes/SourceVitality/src/backend/export/test_database.py

```python
from scrapy import signals
from scrapy.exceptions import NotConfigured
from pkg_resources import resource_filename
import sys

class RemoteDebugging(object):
    def __init__(self):
        self.server = ''
        self.port = 0
        self.pycharm_egg = ''

    @classmethod
    def from_crawler(cls, crawler):
        # first check if the extension should be enabled and raise
        # NotConfigured otherwise
        if not crawler.settings.getbool('REMOTE_DEBUGGING_PYCHARM_ENABLED'):
            raise NotConfigured
        cls.server = crawler.settings.get('REMOTE_DEBUGGING_PYCHARM_SERVER')
        if not cls.server:
            raise NotConfigured
        cls.port = crawler.settings.getint('REMOTE_DEBUGGING_PYCHARM_PORT')
        if not cls.port:
            raise NotConfigured
        cls.pycharm_egg = crawler.settings.get('REMOTE_DEBUGGING_PYCHARM_PATH')
        if not cls.pycharm_egg:
            cls.pycharm_egg = resource_filename(__name__, 'pycharm-debug.egg')

        # instantiate the extension object
        ext = cls()

        # connect the extension object to signals
        crawler.signals.connect(ext.spider_opened, signal=signals.spider_opened)

        # return the extension object
        return ext

    def spider_opened(self, spider):
        try:
            from pydev import pydevd
            # if not already registered, add to sys.path

            if self.pycharm_egg:
                if not (self.pycharm_egg in sys.path):
                    sys.path.append(self.pycharm_egg)
                pydevd.settrace(self.server, port=self.port, stdo
```

File - /home/javi/Documents/projectes/SourceVitality/src/backend/export/remote_debugging.py

```
45 utToServer=True,
46                         stderrToServer=True)
47                 spider.log("Remote debugging for Pycharm enabled
   in %s" %
48                            spider.name)
49             except ImportError:
50                 # on deploy, ignore error
51                 pass
52                 # (Only to bypass an stupid warning)
53                 from pydev import pydevd
54
```

File - /home/javi/Documents/projectes/SourceVitality/src/backend/export/remote_debugging.py

```
<!DOCTYPE html>
<!--[if lt IE 7]>      <html class="no-js lt-ie9 lt-ie8 lt-ie7"> <![endif]-->
<!--[if IE 7]>         <html class="no-js lt-ie9 lt-ie8"> <![endif]-->
<!--[if IE 8]>         <html class="no-js lt-ie9"> <![endif]-->
<!--[if gt IE 8]><!-->
<html class="no-js" xmlns="http://www.w3.org/1999/html" xmlns=
      xmlns="http://www.w3.org/1999/html"> <!--<![endif]-->
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
    <title>Sourcevitality</title>
    <meta name="description" content="Sourcevitality">
    <meta name="viewport" content="width=device-width">

    <link rel="stylesheet" href="../static/css/bootstrap.min.css">
    <style>
        body {
            padding-top: 60px;
            padding-bottom: 40px;
        }
    </style>
    <link rel="stylesheet" href="../static/css/bootstrap-responsive.min.css">
    <link rel="stylesheet" href="../static/css/main.css">

    <script src="../static/js/vendor/modernizr-2.6.2-respond-1.1.0.min.js"></script>

    <link rel='icon' href='favicon.png' type='image/png'/>
    <!-- For non-Retina iPhone, iPod Touch, and Android 2.1+ devices: -->
    <link rel="apple-touch-icon-precomposed" href="apple-touch-icon-precomposed.png">

</head>
<body>
<!--[if lt IE 7]>
            <p class="chromeframe">You are using an <strong>outdated</strong> browser. Please <a href="http://browsehappy.com/">upgrade your browser</a> or <a href="http://www.google.com/chromeframe/?redirect=true">activate Google Chrome Frame</a> to improve your experience.</p>
        <![endif]-->

<div class="navbar navbar-fixed-top">
    <div class="navbar-inner">
        <div class="container">
```

File - /home/javi/Documents/projectes/SourceVitality/src/frontend/exportToHTML/index.html

```
            <ul class="nav pull-right" id="main-menu-right">
                <li><a rel="tooltip" target="_blank" href="about.html" title="About this project and some documentation">About</a>
                </li>
            </ul>
        </div>
    </div>
</div>

<div class="container">

    <!-- Masthead
    ================================================== -->
    <header class="jumbotron subhead" id="overview">
        <div class="row">
            <div class="span12">
                <h1>Sourcevitality</h1>

                <p class="lead">Checking the vitality of open
            </div>
            {#            <div class="span6">#}
            {#                <div class="well">#}
            {#                    <script type="text/javascript"><!--#}
            {#                    google_ad_client = "ca-pub-9003595048301730";#}
            {#                    /* sourcevitality */#}
            {#                    google_ad_slot = "6560224264";#}
            {#                    google_ad_width = 728;#}
            {#                    google_ad_height = 90;#}
            {#                    //-->#}
            {#                    </script>#}
            {#                    <script type="text/javascript"#}
            {#                    src="http://pagead2.googlesyndication.com/pagead/show_ads.js">#}
            {#                    </script>                #}
            {#                </div>#}
            {#            </div>#}
        </div>
    </header>

    <div class="container-fluid">
        <div class="row-fluid">

            <div class="span12">

                <!--Body content-->
                <div class="alert alert-error fade in" id="sea
```

```
rch-error" style="display:none;">
                    <button type="button" class="close">&
times;</button>
                    Cannot find project, please try another o
ne
                </div>

                <div id="searching"></div>

                <form class="form-search">
                    <input id="inputText" type="text" class="
                           data-link="/_search_project">
                    <button id="addSeries" type="button" clas
s="btn">Add</button>
                </form>
                <div id="placeholder" style="width:100%;heigh
t:400px"></div>

            </div>
        </div>
    </div>

    <!-- Footer
     ================================================== -->
    <hr>

    <footer id="footer">
        <p>
            Report issues at <a href="https://github.com/
eudemo/sourcevitality/issues">GitHub</a>.<br>
        </p>
    </footer>

</div>

<script src="//ajax.googleapis.com/ajax/libs/jquery/1.8.3/
jquery.min.js"></script>
<script>window.jQuery || document.write('<script src="js/
vendor/jquery-1.8.3.min.js"><\/script>')</script>

<script src="../static/js/vendor/bootstrap.min.js"></script>

<script src="../static/js/plugins.js"></script>
<script src="../static/js/vendor/jquery.flot.min.js"></script
<script src="../static/js/vendor/jquery.flot.resize.min.js"><
/script>
<script src="../static/js/vendor/spin.min.js"></script>
<script src="../static/js/main.js"></script>

<script>
    var _gaq = [
        ['_setAccount', 'UA-26245188-1'],
```

```
127             ['_trackPageview']
128         ];
129         (function (d, t) {
130             var g = d.createElement(t), s = d.getElementsByTagNa
    me(t)[0];
131             g.src = ('https:' == location.protocol ? '//ssl' : '
    //www') + '.google-analytics.com/ga.js';
132             s.parentNode.insertBefore(g, s)
133         }(document, 'script'));
134 </script>
135
136 </body>
137 </html>
138
```

```
"""
Module with methods to access database
"""

import ConfigParser
import logging
import MySQLdb
import os


def init_config():
    """
    Loads default values into config
    """
    config = ConfigParser.SafeConfigParser()
    config.add_section('frontend')
    config.set('frontend', 'db', 'sourcevitality_frontend')
    config.add_section('search')
    config.set('search', 'port', '9306')
    return config

# TODO: Share between projects
def get_database_params():
    """
    Gets the database parameters for a MySQL connection from the setup or returns the default one

    A sample test:
    >>> get_database_params() != {}
    True
    """

    config = init_config()

    try:
        config.read([os.path.expanduser('~/.sourcevitality/frontend.cfg'),
                     '/etc/sourcevitality/frontend.cfg'])
    except IOError as detail:
        logging.error('Error reading config: %s', detail.strerror())
        raise

    return config

def get_database_params():
    config = _get_database_params()

    database_params = {
        'user': config.get('frontend', 'user'),
        'passwd': config.get('frontend', 'passwd'),
        'host': config.get('frontend', 'host'),
```

File - /home/javi/Documents/projectes/SourceVitality/src/frontend/exportToHTML/database.py

```
50                'db': config.get('frontend', 'db'),
51            }
52            return database_params
53
54    def get_search_params():
55        config = _get_database_params()
56
57        params = {
58                'host': config.get('search', 'host'),
59                'port': config.getint('search', 'port'),
60            }
61
62        return params
63
64
65    # TODO: Share between projects
66    def open_database(params=None):
67        """
68            Opens the database with params
69            """
70
71        if params is None:
72            params = _get_database_params()
73            conn = MySQLdb.connect(host=params['host'],
74                user=params['user'], passwd=params['passwd'],
75                db=params['db'])
76
77        return conn
78
79    def open_search(params=None):
80        """
81            Opens the search database with params
82            """
83
84        if params is None:
85            params = _get_search_params()
86            conn = MySQLdb.connect(host=params['host'],
87                port=params['port'])
88
89        return conn
90
91    def get_plotting_data(conn, projectid):
92        """
93            Gets the plotting data for the specified project
94            """
95
96        try:
97            cur = conn.cursor()
98            cur.execute("""
99                    SELECT activity_date, activities
100                   FROM activity
101                   WHERE project_rowid = %s
```

File - /home/javi/Documents/projectes/SourceVitality/src/frontend/exportToHTML/database.py

```
102             """, (projectid,))
103             raw_data = cur.fetchall()
104             plotting_data = []
105             for t in raw_data:
106                 plotting_data.append(list(t))
107             cur.close()
108             return plotting_data
109         except MySQLdb.MySQLError as detail:
110             raise detail
111 
112 
113 def get_project_info(conn, projectid):
114     """
115     Gets the plotting name for the specified project
116     """
117 
118     try:
119         cur = conn.cursor()
120         cur.execute("""
121             SELECT name, URL
122             FROM project
123             WHERE rowid = %s
124             """, (projectid,))
125         info = cur.fetchone()
126         cur.close()
127         return info
128     except MySQLdb.MySQLError as detail:
129         raise detail
130 
131 def get_project_id(conn, name):
132     """
133     Gets the project id for the specified project
134     This is used to validate what the user
135     """
136 
137     try:
138         cur = conn.cursor()
139         cur.execute("""
140             SELECT rowid
141             FROM project
142             WHERE name = %s
143             """, (name,))
144         id = cur.fetchone()
145         cur.close()
146         return id
147     except MySQLdb.MySQLError as detail:
148         raise detail
149 
150 def search_project(conn, query):
151     """
152     Search for a project. Gets a maximum of 8 results because
 is the default limit in Bootstrap typeahead
```

File - /home/javi/Documents/projectes/SourceVitality/src/frontend/exportToHTML/database.py

```
153         """
154 
155         try:
156             cur = conn.cursor()
157             cur.execute("""
158                 SELECT id, name
159                 FROM sourcevitality
160                 WHERE MATCH(%s)
161                 LIMIT 8;
162             """, (query, ))
163             raw_data = cur.fetchall()
164             search_result_id = []
165             search_result_name = []
166             for t in raw_data:
167                 search_result_id.append(t[0])
168                 try:
169                     search_result_name.append(t[1].encode('utf8'
170                 except UnicodeDecodeError:
171                     pass
172             cur.close()
173             return [search_result_id, search_result_name]
174         except MySQLdb.ProgrammingError:
175             pass
176         except MySQLdb.MySQLError as detail:
177             raise detail
178 
```

```python
from frontend.database import open_database, get_plotting_data
    get_project_info, open_search, search_project
from frontend import app
from flask import g, jsonify, render_template, request

@app.before_request
def before_request():
    g.db = [open_database(), open_search()]

@app.teardown_request
def teardown_request(exception):
    g.db[0].close()
    g.db[1].close()

@app.route('/_get_plotting_data')
def get_plotting_data():
    projectid = request.args.get('project', 0, type = int)
    project_info = get_project_info(g.db[0], projectid)
    try:
        plotting_data = {
            "projectid": projectid,
            "label": project_info[0],
            "data": get_plotting_data(g.db[0], projectid),
            "URL": project_info[1]
        }
    except TypeError:
        plotting_data = {
            "projectid": 0,
            "label": [],
            "data": [],
            "URL": []
        }
    return jsonify(plotting_data)

@app.route('/_search_project')
def search_project():
    query = request.args.get('query', 0, type = str)
    full_search = search_project(g.db[1], query)
    try:
        search_result = {
            "id": full_search[0],
            "options": full_search[1]
        }
    except TypeError:
        search_result = {
            "id": [],
            "options": []
        }
    return jsonify(search_result)

@app.route('/_get_project_id')
def get_project_id():
```

File - /home/javi/Documents/projectes/SourceVitality/src/frontend/exportToHTML/views.py

```
53      query = request.args.get('query', 0, type = str)
54      project_id = get_project_id(g.db[0], query)
55      return jsonify( { "id": project_id } )
56
57  @app.route('/')
58  def main():
59      return render_template('index.html')
60
61  @app.route('/about.html')
62  def about():
63      return render_template('about.html')
64
```

File - /home/javi/Documents/projectes/SourceVitality/src/frontend/exportToHTML/views.py

```python
from flask import Flask
from frontend.database import get_database_params, open_search

app = Flask(__name__)

# Need to be after app creation
import views

```

```
<!DOCTYPE html>
<!--[if lt IE 7]>      <html class="no-js lt-ie9 lt-ie8 lt-ie7"> <![endif]-->
<!--[if IE 7]>         <html class="no-js lt-ie9 lt-ie8"> <![endif]-->
<!--[if IE 8]>         <html class="no-js lt-ie9"> <![endif]-->
<!--[if gt IE 8]><!-->
<html class="no-js" xmlns="http://www.w3.org/1999/html" xmlns=
      xmlns="http://www.w3.org/1999/html" xmlns="http://www.w3.org/1999/html"> <!--<![endif]-->
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
    <title>Sourcevitality</title>
    <meta name="description" content="Sourcevitality">
    <meta name="viewport" content="width=device-width">

    <link rel="stylesheet" href="../static/css/bootstrap.min.css">
    <style>
        body {
            padding-top: 60px;
            padding-bottom: 40px;
        }
    </style>
    <link rel="stylesheet" href="../static/css/bootstrap-responsive.min.css">
    <link rel="stylesheet" href="../static/css/main.css">

    <script src="../static/js/vendor/modernizr-2.6.2-respond-1.1.0.min.js"></script>

</head>
<body>
<!--[if lt IE 7]>
            <p class="chromeframe">You are using an <strong>outdated</strong> browser. Please <a href="http://browsehappy.com/">upgrade your browser</a> or <a href="http://www.google.com/chromeframe/?redirect=true">activate Google Chrome Frame</a> to improve your experience.</p>
        <![endif]-->

<div class="container">

    <!-- Masthead
    ================================================== -->
    <header class="jumbotron subhead" id="overview">
        <div class="row">
            <div class="span6">
                <h1>Sourcevitality</h1>
```

File - /home/javi/Documents/projectes/SourceVitality/src/frontend/exportToHTML/about.html

```
41
42                  <p class="lead">Checking the vitality of open
43              </div>
44          </div>
45      </header>
46
47      <p>
48          How many times did you wanted to use a open source pro
    ject and you find yourself wondering if it's been
49          maintained?
50      </p>
51
52      <p>
53          Sourcevitality is a fast an easy way to answer this qu
    estion: just enter the name of the project, and you will
54          get a graphical idea of what has been the life of the
55      </p>
56
57      <p>
58          You can even compare related projects to know which on
    e is the best maintained and best suited for your job.
59      </p>
60
61      <br><br>
62
63      <p>
64          Theme from <a target="_blank" href="http://bootswatch.
    com/">Bootswatch</a>.<br>
65          Based on <a target="_blank" href="http://twitter.
    github.com/bootstrap/">Bootstrap</a>.<br>
66          Icons from <a target="_blank" href="http://glyphicons.
    com/">Glyphicons</a>.<br>
67          Web fonts from <a target="_blank" href="http://www.
    google.com/webfonts">Google</a>.
68      </p>
69
70
71      <!-- Footer
72        ================================================== -->
73      <hr>
74
75      <footer id="footer">
76          <p>
77              Report issues at <a href="https://github.com/
    eudemo/sourcevitality/issues">GitHub</a>.
78          </p>
79      </footer>
80
81  </div>
82
83  <script src="//ajax.googleapis.com/ajax/libs/jquery/1.8.3/
    jquery.min.js"></script>
```

```
<script>window.jQuery || document.write('<script src="js/vendor/jquery-1.8.3.min.js"><\/script>')</script>

<script src="../static/js/vendor/bootstrap.min.js"></script>

<script src="../static/js/plugins.js"></script>

<script>
    var _gaq = [
        ['_setAccount', 'UA-26245188-1'],
        ['_trackPageview']
    ];
    (function (d, t) {
        var g = d.createElement(t), s = d.getElementsByTagName(t)[0];
        g.src = ('https:' == location.protocol ? '//ssl' : '//www') + '.google-analytics.com/ga.js';
        s.parentNode.insertBefore(g, s)
    }(document, 'script'));
</script>

</body>
</html>

```

```
1  #!/usr/bin/env python
2
3  from fabric.api import task
4
5  @task
6  def create_database():
7      """
8      Creates the database in MySQL server
9      """
10
11
```

File - /home/javi/Documents/projectes/SourceVitality/src/frontend/exportToHTML/fabfile.py

```
1 #!/usr/bin/env python
2
3 from frontend import app
4
5 if __name__ == '__main__':
6     app.run(debug = True)
7
```

File - /home/javi/Documents/projectes/SourceVitality/src/frontend/exportToHTML/runserver.py

```
/*
 *  Main code
 */

$(function () {

    "use strict";

    var selectedItem = 0,
        select,
        data = [],
        placeholder = $("#placeholder"),
        alreadyFetched = {},
        options = {
            lines:{ show:true },
            points:{ show:true },
            legend:{
                labelFormatter:function (label, series) {
                    // series is the series object for the label
                    var URL = "";
                    if (series.URL.search("^http://") !== -1 || series.URL.search("^https://") !== -1) {
                        URL = series.URL;
                    } else {
                        URL = "http://" + series.URL;
                    }
                    return '<a target="_blank" href="' + URL +
                        '" class="close close-plot"><i class="icon-remove icon" ' +
                        'title="Remove from panel" ></i></button>';
                },
                backgroundOpacity:0
            },
            xaxis:{
                mode:"time",
                minTickSize:[1, "day"],
                timeformat:"%y-%m-%d"
            }
        };

    document.getElementById('inputText').value = "";

    // plot (empty) panel
    $.plot(placeholder, data, options);

    /*
     * Hide alert
     */
    $('.alert .close').live("click", function (e) {
        $(this).parent().hide();
```

File - /home/javi/Documents/projectes/SourceVitality/src/frontend/exportToHTML/main.js

```
49      });
50
51      /*
52       * From https://gist.github.com/1290439
53       */
54      $.fn.spin = function (opts, color) {
55          var presets = {
56              "tiny":{ lines:8, length:2, width:2, radius:3 },
57              "small":{ lines:8, length:4, width:3, radius:5 },
58              "large":{ lines:10, length:8, width:4, radius:8
59          };
60          if (Spinner) {
61              return this.each(function () {
62                  var $this = $(this),
63                      data = $this.data();
64
65                  if (data.spinner) {
66                      data.spinner.stop();
67                      delete data.spinner;
68                  }
69                  if (opts !== false) {
70                      if (typeof opts === "string") {
71                          if (opts in presets) {
72                              opts = presets[opts];
73                          } else {
74                              opts = {};
75                          }
76                          if (color) {
77                              opts.color = color;
78                          }
79                      }
80                      data.spinner = new Spinner($.extend({colo
    r:$this.css('color')}, opts)).spin(this);
81                  }
82              });
83          } else {
84              throw "Spinner class not available.";
85          }
86      };
87
88      /*
89       *  Add a project to the panel
90       */
91      // Auxiliary function to fetch the data with jQuery
92      function onDataReceived(series) {
93          // let's add it to our current data
94          if (!alreadyFetched[series.label]) {
95              alreadyFetched[series.label] = true;
96              data.push(series);
97          }
98
99          document.getElementById('inputText').value = "";
```

File - /home/javi/Documents/projectes/SourceVitality/src/frontend/exportToHTML/main.js

```
100
101         // and plot all we got
102         $.plot(placeholder, data, options);
103     }
104
105     $('#addSeries').click(function () {
106
107         // Clean eventual mistakes
108         $("#searching").spin(false);
109         $("#searching").spin();
110
111         function onProjectReceived(selectedElement) {
112
113             $("#searching").spin(false);
114
115             var error = true;
116             if (selectedElement.id !== null) {
117
118                 selectedItem = selectedElement.id[0];
119
120                 if (selectedItem !== 0) {
121
122                     // fetch one series, adding to what we
    got
123                     error = false;
124
125                     try {
126
127                         $.ajax({
128                             url:"/_get_plotting_data?project
    =" + selectedItem,
129                             method:'GET',
130                             dataType:'json',
131                             success:onDataReceived
132                         });
133
134                     } catch (e) {
135                         error = true;
136                     }
137                 }
138             }
139
140             if (error) {
141                 $("#search-error").show();
142             }
143         }
144
145         $.ajax({
146             url:'/_get_project_id?query=' + document.getEleme
    ntById('inputText').value,
147             type:"GET",
148             dataType:'json',
```

File - /home/javi/Documents/projectes/SourceVitality/src/frontend/exportToHTML/main.js

```
149             success:onProjectReceived
150         });
151
152     });
153
154     /*
155      *  Get projects with typeahead from the database
156      */
157     $('.ajax-typeahead').typeahead({
158         source:function (query, process) {
159
160             function onDataReceived(json) {
161                 $("#searching").spin(false);
162                 select = json;
163                 process(select.options);
164             }
165
166             $("#search-error").hide();
167             $("#searching").spin();
168
169             try {
170                 $.ajax({
171                     url:$(this)[0].$element[0].dataset.link,
172                     type:'GET',
173                     data:{query:query},
174                     dataType:'json',
175                     success:onDataReceived
176                 });
177             } catch (e) {
178                 $("#searching").spin();
179             }
180
181         },
182         updater:function (item) {
183             selectedItem = select.id[select.options.indexOf(i
    tem)];
184             return item;
185         }
186     });
187
188     /*
189      * Close projects
190      * Event click doesn't work because buttons are dynamic
191      */
192     $('body').on('click', '.close-plot', function () {
193         var id = $(this)[0].id - 1,
194             label = data[id].label,
195             dataAux = [];
196         if (id >= 0) {
197             for (var v in data) {
198                 if (v != id) {
199                     dataAux.push(data[v]);
```

```
200                     }
201                 }
202                 data = [];
203                 for (var v in dataAux) {
204                     data.push(dataAux[v]);
205                 }
206                 alreadyFetched[label] = false;
207                 // and plot all we got
208                 $.plot(placeholder, data, options);
209             }
210         });
211 
212         /*
213          * Select text on focus
214          */
215         $('#inputText').focus(function () {
216             $(this).select();
217         });
218         $("#inputText").mouseup(function (e) {
219             e.preventDefault();
220         });
221 
222 });
223 
```