Escola Universitària
Politècnica de Mataró

Centre adscrit a:

UNIVERSITAT POLITÈCNICA
DE CATALUNYA

GRAU EN ENGINYERIA INFORMÀTICA

YOURTALKS.COM

Servidors, API i workers

Memòria

ADRIÀ GALÍN FIGUERAS
PONENT: JOSEP ROURE ALCOBÉ

PRIMAVERA 2014

TecnoCampus
Mataró-Maresme

Dedicatòria

A la meva família.

En especial al Pare, l'àvia Maria i l'avi Quimet.

Agraïments

Donar les gràcies al meu tutor del projecte, Josep Roure Alcobé, per a la seva paciència i disponibilitat.

Donar les gràcies als meus amics i persones estimades, en especial a la Queralt, el Bere i la Judit.

Donar les gràcies més grans als meus pares per tot el que m'han donat, ja que sense ells no hagués estat capaç d'arribar fins aquí.

Finalment donar les gràcies a tothom que se les mereix.

Resum

L'objectiu del projecte YourTalks es crear una plataforma web per a la realització de conferències/classes o qualsevol altre tipus de xerrada en temps real. YourTalks vol oferir una experiència el més semblant possible a l'assistència física, a la conferència i complementar-ho amb serveis addicionals que aportin un valor afegit tant als oients com al ponent. Aquests serveis facilitaran que els locutors facin les seves explicacions més interactives i també inclouran la possibilitat d'interactuació entre tots els assistents un cop acabada la conferència.

La plataforma serà modular de manera que es pugui configurar la sala amb diferents funcionalitats i serveis segons la voluntat del ponent i dels assistents. A més ha de permetre la creació de nous serveis a mesura que vagi apareixent noves necessitats i vagi madurant la pròpia plataforma. Per a la creació técnica del projecte s'ha realitzat una extensa cerca i aprenentatge de diverses tecnologies.

Resumen

El objetivo del proyecto YourTalks es crear una plataforma web para la realización de conferencias/clases o cualquier otro tipo de charla en tiempo real. YourTalks quiere ofrecer una experiencia lo más parecida posible a la asistencia física, a la conferencia y complementarlo con servicios adicionales que aporten valor añadido tanto a los oyentes como al ponente. Estos servicios facilitarán que los locutores hagan sus explicaciones más interactivas y también incluirían la posibilidad de interactuación entre todos los asistentes una vez terminada la conferencia.

La plataforma será modular de manera que se pueda configurar la sala con diferentes funcionalidades y servicios según la voluntad del ponente y los asistentes. Además debe permitir la creación de nuevos servicios a medida que apaezcan necesidades y vaya madurando la propia plataforma. Para la realización técnica del proyecto se realiza una extensa búsqueda y aprendizaje de diversas tecnologías.

Abstract

The main objective of the project YourTalks is to create a web platform to make conferences, classes or any other type of talk in real time. YourTalks wants to offer the same experience as attending to a real conference, and complement it with extra services that add value to the attenders and the announcer. This service will help the presentations of the announcers to be more interactives, and also, after the conference, there's the possibility to interact with any of the attenders.

The platform will be modular, so the announcer can add or remove services (modules) of the conference room. Also it should allow the creation of new services for the new necessities we may found, as the platform grows up. There was an extensive research and learning of the new technologies, for the project technical development.

Índex.

Índex de figures.

Índex de taules.

Glossari de termes.

MTTR	Mean time to recover
MTTF	Mean time to failure
SLA	Acord de nivell de servei
MVP	Minimum viable product
ISP	Internet service provider
CDN	Content delivery network
DEVOPS	Desenvolupament i operacions
API	Application Programming Interface
REST	Representational State Transfer
JSON	JavasScript Object Notation
VIP	Virtual IP
CRUD	Create, Read,Update, Delete

1. Objectius.

1.1. Propòsit.

Disseny i implementació de l'estructura base de la plataforma YourTalks. YourTalks és una aplicació web que ajuda a donar conferències/classes o qualsevol altre tipus de xerrada en temps real a través d'internet. Permet l'organització de les conferències en sales amb la possibilitat de complementar-les amb diverses funcionalitats o serveis, com per exemple, un xat, videotrucades, llistat de preguntes, presentacions i enquestes, el qual permetrà una interacció entre oient-ponent i oient-oient. D'aquesta manera es vol donar un valor afegit, permetent la interactuació entre oient i ponent més enllà de la duració de la conferència i oferir una xarxa social entre ells. També es vol que sigui un punt de connexió entre tots els ponents del món.

1.2. Finalitat.

Aconseguir una plataforma web per donar conferències/classes o qualsevol altre tipus de xerrada, a través d'internet en temps real. Ajudar als locutors a realitzar les seves explicacions més interactives i amenes per al oients donant un valor afegit, permetent la interactuació entre oient i ponent més enllà de la duració de la conferència i oferir una xarxa socials entre ells.

1.3. Objecte.

Un servei web per realitzar presentacions en temps real, en model SaaS (Software as a Service) de distribució. SaaS és interessant perquè evita a l'usuari de qualsevol tipus de manteniment sobre la plataforma, tot ho gestiona el proveïdor.

1.4. Abast.

Tota la idea explicada anteriorment requereix un llarg desenvolupament, recursos i temps. S'ha de dedicar molt de temps a la investigació i estudi de noves tecnologies, perquè no s'han treballat a classe, per poder desenvolupar la plataforma. Ens centrarem en la base de l'aplicació (Core), que consisteix en crear sales, unir-te a les sales, poder xatejar amb la

gent de la sala, crear presentacions i veure presentacions en temps real. Tenir una base ben estructurada ens permetrà desenvolupar en un futur amb molta més agilitat i ràpidesa.

Pel que fa al desenvolupament s'utilitzarà Ruby[1], Ruby on Rails[2], JavaScript[3], Grunt[4], BackoneJS[5], MarionetteJS[6] i Bash[7]. Per la part de servidors s'utilitzaran eines d'automatització com ara Puppet[8], capistrano[9] i servidors web com NodeJS[10], Nginx[11], Unicorn[12].

[1] https://www.ruby-lang.org/es/
[2] http://rubyonrails.org/
[3] http://es.wikipedia.org/wiki/JavaScript
[4] http://gruntjs.com/
[5] http://backbonejs.org/
[6] http://marionettejs.com/
[7] http://www.gnu.org/software/bash/
[8] http://puppetlabs.com/puppet/what-is-puppet
[9] http://capistranorb.com/
[10] http://nodejs.org/
[11] http://nginx.com/
[12] http://unicorn.bogomips.org/

2. Metodologia de treball.

Per a la gestió del projecte es necessita un sistema d'organització enfocat a treballar en equip. Es vol utilitzar metodologies àgils, que són aquelles metodologies de desenvolupament que es basen en l'adaptabilitat de qualsevol canvi com a mitjà per augmentar les possibilitats d'èxit d'un projecte. Per a la gestió del flux de treball s'utilitzarà el mètode Kanban[13] que permet dividir el procés productiu en diferents fases perfectament limitades, un tipus de mètode per aplicar la metodologia àgil. Kanban, té dos objectius: d'una banda, aconseguir un producte de qualitat, ja que obliga cada fase del projecte a finalitzar la seva tasca correctament, i d'aquesta manera acabar amb el caos o coll d'ampolla que es pot donar en una fase del projecte en condicions normals en les quals preval la rapidesa per sobre de la qualitat del producte. Kanban està pensat per a projectes de software i per augmentar la productivitat de l'equip.

Per poder implementar aquest mètode s'ha escollit la plataforma de gestió de tasques JIRA[14] de l'empresa Atlassian[15]. JIRA és un gestor de tasques que permet als equips planificar, construir i finalitzar projectes. Pel sistema de control de versions s'ha utilitzant el producte Bitbucket[16] de l'empresa Atlassian. Permet la integració amb el sistema de control de versions GIT[17], un sistema de control de versions distribuït, dissenyat per poder suportar tant projectes petits com grans projectes amb rapidesa i eficàcia, Open Source.

[13] http://es.wikipedia.org/wiki/Kanban
[14] https://www.atlassian.com/es/software/jira
[15] https://www.atlassian.com/es/
[16] https://bitbucket.org
[17] http://git-scm.com/

3. Anàlisis de requeriments.

En aquest apartat es detalla de forma descriptiva totes les funcionalitats del projecte YourTalks. Ara només ens centrarem en desenvolupar l'estructura bàsica de l'aplicació, anomenada Core. El Core es compon de l'anàlisis de l'arquitectura dels servidors, una API REST[18] amb les funcions més bàsiques, per després poder continuar desenvolupant les altres característiques que seran necessàries per tenir el producte 'complet'. El sistema de processament de les presentacions pujades pels usuaris i el sistema de real time per a la presentació amb el módul de chat.

YourTalks és una aplicació web que ajuda a donar conferències/classes o qualsevol altre tipus de xerrada en temps real a través d'internet.

La plataforma serà modular permetent configurar la sala amb les funcios desitjades, anomenades "widgets". Els "widgets" disponibles són:

- **Videoconferència**: l'expositor fa un broadcast del seu àudio i vídeo als oients.
- **Presentacions**: poder mostrar una presentació en temps real.
- **Xat**: un xat entre tots els participants de la sala.
- **Preguntes**: un llistat de possibles preguntes a fer als participants.
- **Q&A**: preguntes que volen fer els participants al locutor.
- **Informe de comprensió**: els participants voten si entenen o no al locutor.

La plataforma es compon d'un sol tipus d'usuaris amb dos rols diferents segon l'acció a realitzar. Si l'usuari és un participant (viewer), de la sala, podrà:

- Observar en temps real la presentació de la sala.
- Visualitzar i escoltar al locutor (àudio i vídeo).
- Podrà xatejar amb els altres participants.
- Podrà formular preguntes al locutor.
- Podrà respondre les preguntes que es realitzin per part del locutor.
- Podrà donar el seu feedback sobre la presentació i el nivell de comprensió.

[18] http://en.wikipedia.org/wiki/Representational_state_transfer

Si l'usuari és un administrador (owner), el creador de la sala, podrà:

- Convidar nous participants a la sala.
- Compartir la seva càmera i àudio entre tots els participants.
- Pujar i compartir una presentació a la sala.
- Escriure en el chat.
- Formular preguntes directes als participants i veure les seves respostes.
- Visualitzar les preguntes que se l'hi formulin.
- Evaluar el nivell de comprensió dels participants.
- Controlar les presentacions des d'un altre dispositiu (per exemple el mòbil) per no haver d'estar sempre davant de l'ordinador.

Algunes de les opcions futures, després d'analitzar l'ús que fan els usuaris de la plataforma podrien ser:

- Convertir-se en una Xarxa Social de Locutors, on la gent que ha assistit a les seves presentacions podrà donar un feedback de com creu que es aquella persona com a locutora, com creu que ha sigut la presentació i valorar-la amb una puntuació.
- Creació d'una aplicació mòbil, per poder seguir les xerrades des de qualsevol lloc.
- Gestió d'events i quedades.
- Integració amb plataformes de tercers (exemple: Meetup[19]).

Es vol que la plataforma sigui un punt de connexió entre tots els conferenciants i el seu punt de referència per adquirir i mostrar nous coneixements. També es vol que la plataforma es pugui integrar amb plataformes de tercers i arribar a ser un Youtube de les presentacions online.

Per poder desenvolupar la plataforma YourTalks fa falta dividir el projecte en diferents parts pel que fa al seu desenvolupament. Per a la creció d'una aplicació escalable i modular s'ha necesitat distingir cada part com una "subaplicació".

En primer terme per treure la plataforma a producció es requereix un sistema amb alta disponibilitat, tolerant a falles, escalable de forma horitzontal i econòmic. Per poder

[19] http://www.meetup.com/

obtenir aquest tipus d'arquitectura es requereix de dos servidors físics, allotjats en datacenters separats, com a mínim per poder tenir el sistema redundat, sempre es poden afegir més host per suportar pujades de tràfics o un augment dels usuaris. Un sistema de balanceig de càrrega per distribuir les peticions en els diferents servidors. Un sistema de virtualització (per exemple, Proxmox[20]) que ens permeti muntar-lo en clúster i tenir un control de totes les màquines. Diferents servidors virtualitzats redundats per allotjar les diferents parts de la plataforma.

Per saber si els servidors estan actius ens fa falta un sistema de checking, que detecti el mal funcionament o parada d'un servidor o servei, i el mogui en el servidor de reserva. Un sistema de control o monitoreig que xequegi les constants dels servidors i actuï a partir d'unes regles definides i avisi a la persona corresponen.

Pel que fa a la gestió de les màquines es requereix d'un sistema d'automatització i orquestrat de màquines per poder realitzar la gestió dels servidors des d'un únic punt sense haver d'entrar màquina per màquina a gestionar els seus serveis.

Pel logs es requereix un sistema central on es guardin els logs de totes les màquines durant 30 dies, per a la seva consulta i anàlisi en temps real o no. D'aquesta manera ens permetrà detectar falles en les aplicacions i en el sistema.

Per la part de desenvolupament es necessita crear una aplicació escalable, que no carregui tota la lògica en un sol punt com es fa normalment, ja que requereix menys desenvolupament inicial. S'ha separat el model-vista-controlador en diferents parts, el model-controlador la API REST i la vista a l'aplicació frontend.

L'API REST, s'anomenarà Puigmal, es necessita programar amb Ruby on Rails, que ens ha de proporcionar uns 'endpoints' amb les accions disponibles. Conté tota la lògica de la base de dades. L'API correspon al MC (Model-Controlador), només interactuant aquests dos elements, ja que la lògica de la vista està delegada a una de les altres aplicacions, la de Frontend.

El Frontend, s'anomenarà Aneto, es necessita programar amb JavaScript, aquest és el consumidor de l'API REST i el que conté tota la lògica del client, la seva interfície. Es

[20] http://www.proxmox.com/es/

requereix poder muntar una aplicació de frontend amb JavaScript que sigui escalable utilitzant Backbone.Marionette. Backbone.Marionette és un Framework de JavaScript que agilitza el desenvolupament de grans aplicacions en aquest tipus de tecnologia.

L'aplicació de Real time, s'anomenarà Pedraforca, i haurà de gestionar la comunicació entre clients. Es desenvoluparà en JavaScript utilitzant NodeJS i Socket.io[21]. Aquesta comunicació serà essencial dins la plataforma, ja que donarà a l'usuari una sensació de dinamisme de la plataforma.

L'últim mòdul que forma YourTalks són els "workers", s'anomenarà Carlit, aquest mòdul té una finalitat de processament d'informació, per no provocar esperes innecessàries en els clients. Serà l'encarregat de processar les presentacions i convertir-les en imatges separades, un cop finalitzat ho comunicarà a l'aplicació de real time que al seu moment ho comunicarà al client corresponent.

Aquesta estructura ens permet, una gran modularitat per construir aplicacions entorn d'ella. Per exemple, si es vol implementar l'aplicació per a mòbil, la API ja està feta només cal que l'app mòbil consumeixi l'API REST. Si es vol donar accés a alguna de les funcions concretes de la API, el cost d'implementació serà més reduït i senzill, ja que només afect a l'API. Aquesta separació ens permetrà escalar cada servei segons la demanda.

Es vol utilitzar en la mesura que sigui possible software OpenSource per a la seva gran comunitat i reduït cost, per tant, es requereix un bona recerca de tecnologies.

[21] http://socket.io/

4. Planificació.

En aquesta fase s'estableixen els objectius i s'escullen els procediments més apropiats per a l'assoliment de cada un d'ells abans d'emprendre una acció. La planificació ens ajudarà a obtenir i repartir els recursos tan materials com humans de què es disposen per aconseguir els objectius. Així també ens ajuda a controlar l'assoliment dels objectius, fixar prioritats i tractar possibles problemes en el transcurs del projecte.

4.1. Kanban board.

La planificació d'sprints s'ha fet pensant en els objectius bàsics i l'abast que se li vol donar al software en un inici. Aquests sprints es refereixen en el que s'havia pensat que es tindria acabat per el dia de la finalització de l'sprint. Cada sprint s'ha definit en un periode màxim de 2 setmanes. La planificació del projecte es va fer entre les dates 24/03/2014 i 02/06/2014, això formen un total de 5 sprints a seguir. A cada sprint se li assignat un valor de risc (% of Error) que vol dir, la desviació possible que pot tenir l'implementació. Cada sprint englova un conjunt de tasques que han estat més detallades al JIRA, en els sprints només s'ha posat un títol i una descripció general del contingut de l'sprint. La Taula 1 mostra els sprints planificats per a la realització del projecte.

Number	Dates	Who?	Title	% of Error	Description
#1	24/03/2014 - 07/04/2014	Francesc	User Dashboard	30	Create all the interface for the User Dashboard. CRUD + Sign In
		Adrià	Puppet (Servers)	30	Setup server infrastructure with automate system and investigate about high scalability
#2	24/03/2014 - 07/04/2014	Francesc	Admin Dashboard	20	Create all the interface for the Admin Dashboard, basically statdistics (and some CRUDs)
		Adrià	API REST	20	Develop API and slideshows workers
#3	24/03/2014 - 07/04/2014	Francesc	Room (Chat)	5	Hability to create a Room, invite people and to chat with them
		Adrià	Puppet (Servers)	30	Setup server infrastructure
#4	24/03/2014 - 07/04/2014	Francesc	Room (Slideshows)	5	Hability to add a slideshow to the room and to pass the slides of it
		Adrià	API REST	10	Develop`API
#5	24/03/2014 - 07/04/2014	Francesc	Room (WebRTC)	50	Well, WebRTC
		Adrià	API REST + PUPPET	10	Fix bugs and others things
Total:				210%	

Taula 1: Planificació dels sprints. Font: el·laboració pròpia

4.2. Planificació.

En la Taula 2 es pot veure la planificació del projecte amb totes les fases.

Tasca	Subtasca	Data inici	Data fi
Avantprojecte		13/01/2014	31/01/2014
	Estudi del projecte	13/01/2014	25/01/2014
	Entrega documentació	25/01/2014	31/01/2014
Análisis i diseny		01/02/2013	
	Investigació de les tecnologies	01/02/2014	01/03/2014
	Definició de funcionalitats	01/07/2014	07/03/2014
	Disseny de prototipus	07/03/2014	
Implementació		24/03/2014	18/05/2014
	Implementació	24/03/2014	18/05/2014
	Proves	19/05/2014	06/06/2014
Memòria i presentació		05/05/2014	19/05/2014
	Memòria comuna	01/05/2014	19/05/2014
	Memòria individual	24/03/2014	18/05/2014
	Presentació	02/06/2014	13/06/2014
Entrega documentació final		02/06/2014	04/06/2014

Taula 2: Planificació amb totes les fases. Font: el·laboració pròpia

5. Estudi teòric.

En aquest apartat es fa una introducció a diferents tècniques per crear un sistema tolerant i fiable per a un producte que surt al mercat. S'enten com a sistema fiable aquell que desenvolupa la seva funció durant un període de temps determinat, quan opera en l'entorn pel qual ha estat dissenyat, creant un sistema d'alta disponibilitat i tolerant a falles, com una caiguda de tensió, un error en el programari, etc. Per exemple, si es defineix que el sistema ha d'estar disponible el 99% del dia, i es compleix, estem davan d'un sistema fiable. Normalment aquests càlculs es fan per un període més llarg (veure apartat 5.2).

Aquest és un món molt ampli i cada apartat donaria per fer un projecte final de carrera o una tesi, per això, s'ha realitzat una petita introducció en els punts més rellevants que s'han necessitat per construir la plataforma, tenint en compte el time to market, en aquest cas l'entrega del projecte final s'ha intentat tenir un minimum viable product (MVP[22]).

5.1. Administració moderna de sistemes.

"Tools can enable change in behavior and evetually change culture"

[Patrick Debois]

5.1.1. Que és DevOps?.

DevOps[23], és l'administració àgil de sistemes i també una relació més col·laborativa i productiva entre els equips de desenvolupament i els equips d'operacions. Millora la relació, augmentant-ne l'eficiència en la col·laboració i, reduint el risc de la posada a producció, associat a canvis o lliuraments freqüents. Per tant, s'ha de pensar en DevOps com la participació efectiva dels administradors de sistemes en el procés de desenvolupament d'aplicacions, utilitzant les mateixes tècniques àgils que fan servir els desenvolupadors.

L'enfocament de DevOps comparteix l'agilitat de Scrum[24] per millorar la comunicació entre client (el negoci) i el proveïdor (desenvolupadors), fomentant la confiança, compartint el pla i l'avanç en el treball, i tenint feedback constant. Igual que l'Scrum

[22] https://en.wikipedia.org/wiki/Minimum_viable_product
[23] http://es.wikipedia.org/wiki/DevOps
[24] https://www.scrum.org/

Master i el Product Owner actuen com a responsables de la coordinació entre negoci i desenvolupament.

El "DevOps engineer" s'encarrega de planificar i seguir la coordinació entre desenvolupament i operacions.

Encara que DevOps té una gran relació amb les eines que automatitzen les operacions com són Jenkins[25], Chef, Puppet, Atlassian Bamboo[26], etc., l'aspecte fonamental són els processos integrats entre desenvolupament i producció.

Podem concloure que DevOps no és un càrrec, sinó un mètode de treball (veure Figura 2) enfocat als resultats. En la Figura 1 es pot veure gràficament.

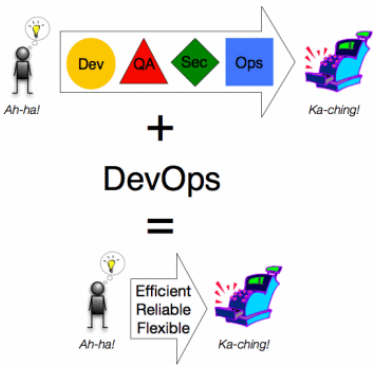


Figura 1: DevOps enfocat a resultats. Font: (Dev2Ops)

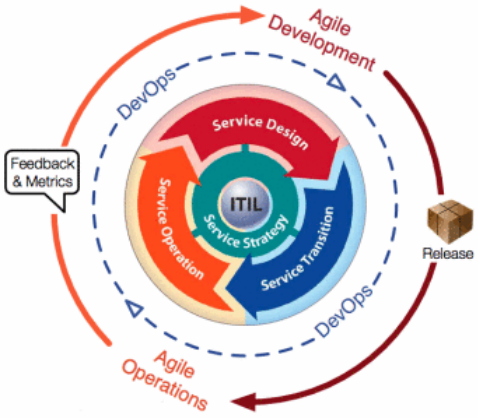


Figura 2: Procés participatiu del DevOps. Font: (Dev2Ops)

[25] http://jenkins-ci.org/
[26] https://www.atlassian.com/software/bamboo

5.1.2. Infrastructure as code.

És el concepte en que la infraestructura ha de ser tractada com a codi, és molt poderós. La configuració del servidor, els paquets instal·lats, la comunicació entre servidors. Permet gestionar la infrastructura a l'administrador de sistemes, com si fos un desenvolupador d'aplicacions. D'aquesta manera es poden realitzar tests unitaris i d'integració obtenint-ne resultats predictibles, eliminant-ne els passos manuals propensos a errors.

Aquest nou concepte, porta nous reptes. Per exemple:

- Quina versió de la infraestructura s'està utilitzant?
- Com es pot assegurar que quan un problema es soluciona, es propagarà i es rellançarà?
- Com es podrà testejar la infraestructura?

Es poden seguir les bones pràctiques de desenvolupament en la infraestructura dels nostres sistemes. Per exemple es pot:

- Etiquetar, crear branques i alliberar el codi correcte en els nostres servidors.
- Els servidors tenen un cicle de desenvolupament que cobreix les diferents etapes del codi, és a dir, desenvolupament (dev), testing (test), quality assurance (qa) i producció (prod o www).
- Provar contínuament la infraestructura quan es facin canvis.

5.2. Sistemes d'alta disponibilitat.

Actualment, les empreses depenen cada vegada més dels seus sistemes d'informació, i com és obvi es desitja que aquests siguin segurs i romanguin disponibles el major temps possible. L'anticipació és una punt clau, una augment del tràfic web pot provocar una caiguda del sistema, si no es prenen precaucions per evitar-ho, per exemple:

- La prevenció d'errors, consisteix en anticipar-se a qualsevol anomalía.
- La tolerància als errors, permet proporcionar un servei d'acord amb les especificacions, mitjançant redundància del servei, etc.
- L'eliminació d'errors, destinada a reduir-ne, per mitja d'accions correctives.

- La predicció d'errors, anticipant-nos i el seu possible impacte en el servei.
- La predicció a una alta demanda del servei.

Per a una empresa, una interrupció del sistema suposa un problema per al negoci, a causa de les conseqüències que s'en deriven. Aquests efectes poden ser:

- Costos indirectes, satisfacció dels clients, pèrdua de reputació, mala publicitat, desconfiança dels empleats, etc.
- Hores de treball addicionals per al departament de sistemes que ha de reparar l'avaria.
- Costos directes associats a la reparació del sistema d'informació (peces a reparar o substituir, ports, serveis tècnics, etc.).
- Pèrdua d'ingressos[27].

La disponibilitat es quantifica habitualment a través de l'índex de disponibilitat, que s'obté de dividir el temps durant el qual el servei està actiu pel temps total d'operació.

Les mètriques utilitzades habitualment per mesurar la disponibilitat i fiabilitat d'un sistema són el temps mitjà entre fallades o MTTF (mean time to failure) que mesura el temps mitjà transcorregut fins que un dispositiu falla, i el temps mitjà de recuperació o MTTR (mean time to recover) que mesura el temps mitjà utilitzat per tornar a la situació normal un cop s'ha produït la fallada. El temps en què un sistema està fora de servei es mesura sovint com el quocient MTTR/MTTF. Lògicament, el principal objectiu és augmentar el MTTF i reduir el MTTR de manera que es minimitzi aquest temps.

Els temps d'inactivitat d'un servei inclou tant les interrupcions programades com les que no ho són. El principal objectiu per augmentar-ne la disponibilitat d'un sistema serà minimitzar aquest temps.

L'índex de disponibilitat, es pot expressar també com un percentatge. Per exemple, si un sistema té una disponibilitat d'un 99%, al llarg d'un any es mantindrà funcionant aproximadament 361 dies, i tindrà un temps d'inactivitat de 3,6 dies. Així doncs, podríem calcular la seva disponibilitat de la següent manera:

[27] http://www.tuexperto.com/2013/08/21/caida-servidor-amazon-causa-perdidas-6-millones-dolares/

Índex disponibilitat = ((A - B)/A) x 100)

A= Hores compromeses de disponibilitat: 24 x 365 = 8760 hores/any

B= Número d'hores fora de línia per caiguda del sistema

Sent el resultat:

Índex disponibilitat = ((8760 - 24)/8760) x 100) = 99,726 %

Els fabricants o proveïdors de serveis solen utilitzar aquest percentatge en els acords de nivell de servei (SLA), per classificar el nivell de disponibilitat que s'espera d'un sistema. En la taula x podem veure el percentatge de disponibilitat relacionat amb els dies d'interrupció anual.

Percentatge de disponibilitat	Temps d'interrupció anual
98 %	7,3 dies
99 %	3,6 dies
99,9 %	8,8 hores
99,99 %	52,5 minuts
99,999 %	5,3 minuts
99,9999 %	31,5 segons

Taula 3: Relació entre el percentatge de disponibilitat i els dies d'interrupció anual. Font: (Wikipedia - Alta disponibilidad)

Aquests índex de disponibilitat (especialment aquells que passen del 99,5 % de disponibilitat) són difícils d'aconseguir, ja que és necessari poder-se recuperar de caigudes del sistema de forma transparent. La capacitat i l'interval de temps necessaris per recuperar-se davant d'aquesta eventualitat depenen directament de:

- La complexitat del sistema.
- La severitat del problema.
- La disponibilitat del personal de suport.
- La maduresa en l'administració del sistema i les seves operacions.
- Altres factors tècnics o de gestió.

Per tant es pot dir que l'alta disponibilitat és la característica que té un sistema per protegir-se o recuperar-se de interrupcions o caigudes, de forma automàtica en un curt termini de temps.

Els sistemes d'alta disponibilitat es dissenyen per eliminar o tolerar els possibles punts de fallada, per això s'empra principalment la redundància interna de components (xarxa, emmagatzemant, fonts d'alimentació, etc.), així com dels elements d'infraestructura (sistema elèctric, etc.).

Aquí es mostren varis esquemes senzills d'arquitectures d'alta disponibilitat (Figura 3 i Figura 4:

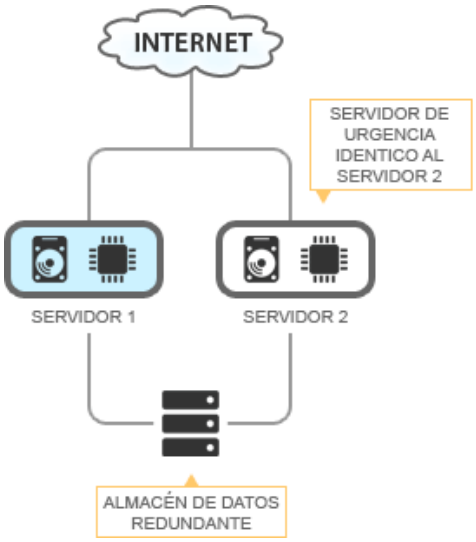


Figura 3: Esquema 1 d'arquitectures d'alta disponibilitat. Font: (Funio - Arquitectura de alta disponibilidad)

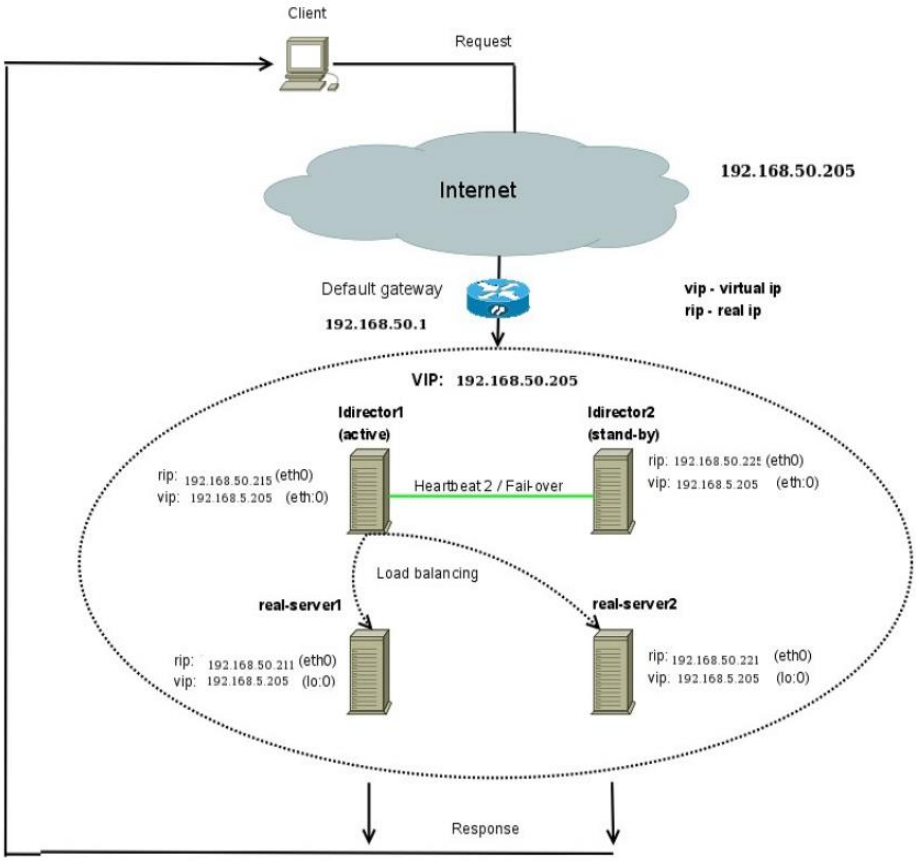


Figura 4: Esquema 2 d'arquitectura d'alta disponibilitat. Font: (Cluster de alta disponibilidad)

5.2.1. Com s'aconsegueix?.

Afegir més components a un sistema no ajuda a que aquest sigui més estable i d'alta disponibilitat. En realitat, pot provocar un efecte contrari, com més components s'afegeixin més probabilitat de fallades. Les arquitectures de sistemes moderns permeten una distribució de les càrregues de treball en diverses instàncies com una xarxa o clúster (grup), ajudant a optimitzar l'ús de recursos, millorant el funcionament, minimitzant el temps de resposta i evitant la sobrecàrrega de qualsevol sistema durant el procés de balanceig. Aquest procés també implica un canvi a un recurs en standby (adormit), un servidor per exemple, en cas que falli un servidor actiu. Els sistemes redundants que s'activen quan fallen els sistemes principals s'anomenen sistemes de failover.

Utilitzant diferents servidors d'aplicacions:

Per exemple, si hi ha un sol servidor per els nostres serveis, una web, i en un moment donat hi ha un augment sobtat en el tràfic que sobrecarrega el servidor, provocarà que s'ofereixi un servei lent, en el pitjor dels casos que el servidor pugui arribar a caure, quedant el servei interrumput.

La solució més òbvia és implementar l'aplicació en diversos servidors. Per tant, cal distribuir la càrrega entre tots ells, de manera que cap estigui sobrecarregat i el servei sigui òptim, utilitzant balancejadors de càrrega. També es pot desplegar l'aplicació en diferents servidors: un servidor pel correu, un pel contingut estàtic com a Content Delivery Network[28] (CDN).

Escalant la base de dades:

És molt important tenir la base de dades redundada o dividida (sharding) i realitzar còpies de seguretat, ja que una fallada del servidor, pot provocar una pèrdua de dades que pot ser molt costosa. La redundància és un procés que crea sistemes d'alta disponibilitat i permet tenir emmagatzemat les mateixes dades a diferents llocs, i per tan millora el rendiment i les consultes a la base de dades.

Ubicació geogràfiques diferents:

L'augment dels servidors d'aplicacions i l'escalabilitat de la base de dades és un gran pas, però si tots els servidors estan en la mateixa ubicació física i succeeix algun problema en el datacenter on es troben (talls elèctrics, algun desastre natural), el nostre servei quedarà inactiu. Per tant, és molt important tenir els servidors en diferents llocs.

5.3. Models de serveis de computació al núvol.

La computació en el núvol és un terme que resumeix una àmplia gama de serveis. Atès que el núvol és una gran col·lació de serveis, les empreses poden escollir com, quan i on desitgen utilitzar la computació en el núvol.

[28] http://es.wikipedia.org/wiki/Red_de_entrega_de_contenidos

Quan es vol desenvolupar aplicacions en el núvol s'ha de tenir en compte la forma en com es farà, ja que dins del concepte de núvol hi ha diferents formes (veure Figura 5) de fer-ho que permeten una major flexibilitat o senzillesa a l'hora de desplegar les nostres aplicacions o mantenir-les. En general, hi ha tres tipus:

Figura 5: Models de computació en el núvol. Font: (Denoe - Estructura cloud computing)

5.3.1. Sotfware as a Service (SaaS).

SaaS és un model de distribució del programari on una empresa serveix el manteniment, suport i operació que farà servir el client durant el temps que hagi contractat el servei. Proporciona als clients l'accés al mateix a través de la xarxa (generalment Internet). Les aplicacions distribuïdes en la modalitat SaaS poden arribar a qualsevol tipus d'empresa sense importar la seva mida o la seva ubicació geogràfica. Es tracta d'un model que uneix el producte (programari) al servei, per dotar a les empreses d'una solució completa que permeti optimitzar els seus costos i els seus recursos. La escalabilitat d'aquesta part de la computació en el núvol es reflecteix en la facilitat per afegir o treure usuaris que fan ús de l'aplicació. En aquest cas l'usuari no té cap control sobre la plataforma. Algunes aplicacions SaaS són: Gmail[29], Dropbox[30], etc.

[29] https://mail.google.com
[30] https://www.dropbox.com/

5.3.2. Platform as a Service (PaaS).

PaaS és un model que permet facilitar el desplegament d'aplicacions (programari) sense incórrer en el cost i en la complexitat de comprar i administrar el maquinari, el sistema operatiu i l'aprovisionament d'emmagatzematge. Una companyia per tant, pot utilitzar-ho per millorar la seva estructura de costos i tenir disponible elasticitat i escalabilitat en el servei, de manera que es pugin afegir o treure recursos en la mesura que es requereixi, minimitzant el malbaratament de recursos. En aquest model el proveïdor ofereix l'ús de la seva plataforma, mitjançant el qual l'usuari no té control sobre la plataforma ni les infraestructures però si sobre les seves aplicacions. Alguns exemples són: Google App Engine[31], Heroku[32], Microsoft Azure[33], etc.

5.3.3. Intrastructure as a Service (IaaS).

IaaS és un model que ens proporciona infraestructures tecnològiques (capacitat de processament, d'emmagatzematge, màquines, i/o comunicacions). Es pot triar quin tipus de màquines es vol utilitzar Linux o Windows, així com la capacitat de memòria o processador de cadascuna de les màquines. El maquinari ens és transparent. La principal diferència rau en encarregar-se d'escalar les aplicacions segons les necessitats, a més de preparar tot l'entorn en les màquines. Alguns exemples són: AWS[34], OVH[35], Gigas[36], etc.

5.4. Automatització de la infraestructura.

5.4.1. Introducció.

Actualment, administrar un servidor és un procés relativament senzill, es modifiquen alguns arxius de la configuració i s'executen algunes comandes. Amb dos servidors és un cas molt similar, però ara s'ha d'executar les ordres en els dos servidors. El problema comença quan en lloc d'un o dos servidors s'administren 10 o més, i si a més d'això s'afegeix virtualització en cadascun dels servidors, segurament es tornaria una tasca molt

[31] https//appengine.google.com
[32] https://www.heroku.com/
[33] http://azure.microsoft.com/es-es/
[34] http://aws.amazon.com/
[35] http://www.ovh.es/
[36] http://gigas.com/

demandant, repetitiva i avorrida mantenir actualitzades i sincronitzades les configuracions en aquests equips.

Per sort, per a la realització d'aquest tipus de tasques es disposa d'eines d'automatització i control de configuracions, que permeten mantenir centralitzada la configuració dels equips, facilitat l'administració de molts servidors, fent que sigui molt més fàcil de configurar i mantenir desenes, centenars o fins i tot milers de servidors.

Una de les fortaleses del sistema Unix[37] i derivats, és el fet que gairebé tot en el sistema es presenta a l'usuari com un arxiu. Aquesta possibilitat d'ús ha fet que moltes operacions siguin senzilles.

És molt important comprendre el procediment abans d'automatitzar. Aquesta és la primera regla de l'automatització: simplement un conjunt de passos que ja funcionen, units en una forma automatitzada.

Aquí hi ha una visió general del procés de desenvolupament de canvi automatitzat:

- Faci el canvi en un entorn de prova.
- Ajusti'l a la seva política, per exemple, faci que s'executi com un usuari no root.
- Automatitzi els passos del desplegament.
- Comprovi el desplegament en un nombre petit de hosts de prova o de desenvolupament, i confirmi que obté els efectes desitjats.
- Desplegui el canvi a tots els hosts.

Qui pot necessitar la automatització?

- Grans companyies amb molts sistemes diversos.
- Mitjanes empreses que es plantegin créixer.
- Proveïdors de serveis d'internet (ISP).
- Un centre de dades.

Quins beneficis aporta l'automatització?

- Estalvi de temps

[37] http://www.unix.org/

- Reducció d'errors.
- Documentació de les polítiques de configuració del sistema.

5.4.2. Puppet.

"Una eina open-source de següent generació per a l'automatització de servidors"

[PuppetLabs[38]]

Puppet és una eina d'administració de servidors Unix i derivats, escrita en Ruby, basada en un llenguatge declaratiu, això vol dir que no s'ha de dir a la màquina què és el que ha d'executar, sinó que se li indica "l'estat final" que es vol que tingui. Dissenyada per Luke Kenies el 200, el qual després va fundar PuppetLabs. Utilitza un esquema client-servidor per a la distribució i una llibreria per executar aquesta configuració. Puppet permet controlar l'execució de diferents serveis en una màquina, control centralitzat de fitxers de configuració, gestió i instal·lació de paquets, administració de la presència i disponibilitat de punts de muntatge, gestió d'usuaris i grups, i desplegaments de màquines. El resultat de la execució d'una comanda o script sempre és el mateix independentment de quantes vegades l'executem. Aquest principi es coneix com a idempotència: una funció obté el mateix resultat cada vegada que s'executa.

La configuració del servei s'efectua en els "manifests". El client es descarrega aquest "manifest" i aplica els que se li indiquen. Els canvis realitzats per Puppet són reportats, obtenint d'aquesta manera les confirmacions d'aquells canvis que han estat aplicats en el nostre sistema.

Si bé és cert que la programació dels "manifests" pot arribar a ser una tasca una mica tediosa, una vegada que aquests es troben programats, s'estalvia una gran quantitat de temps en la configuració i administració d'equips.

Altres eines semblans són:

Chef[39]: és una eina d'automatització d'infrastructura de sistemes, que permet desplegar aplicacions i sistemes a qualsevol ambient físic, virtual o en el núvol fàcilment. Utilitza agents per aplicar les configuracions.

[38] http://puppetlabs.com/

Ansible[40]: és una eina que permet la gestió i/o configuració de servidors i/o grups de servidors de forma remota, i utilitza una connexió ssh[41] per connectar-se als servidors.

S'ha escollit Puppet perquè és un dels primers productes que va sortir. És el més madur i el més accessible des del punt de vista d'ús, encara que es requereixen bons coneixements de Ruby. Té una gran comunitat que el segueix, i per tant hi ha molta documentació i gent que pot preguntar qualsevol dubte. És el més complet en característiques disponibles, mòduls i interfícies d'usuari.

5.5. Virtualització.

5.5.1. Introducció.

És la creació, a través de programes, d'una versió virtual d'algun recurs tecnològic, sigui un sistema operatiu, un servidor, un dispositiu d'emmagatzematge o recursos de xarxa.

Una màquina virtual: és un ordinador virtual. S'utilitza un programa per crear un ordinador lògic d'un altre de físic. Amb aquest programa es crea un o varis ordinadors lògics als quals s'assigna una part de l'ordinador físic, tenen discs durs (virtual), memòria ram (virtual), etc.

La virtualització es pot utilitzar per provar programes, diferents sistemes operatius i per donar diversos serveis, etc.

Aquest mateix plantejament d'un ordinador dins d'un altre es aplicable als servidors. Es compra una màquina potent, s'instal·la el programa que permet virtualitzar i "dins" d'aquest servidor, es creen diverses màquines virtuals: un servidor d'arxius, un servidor de correu, un servidor de comunicacions, un servidor de base de dades, i tots els serveis necessaris.

Virtualizar té molts avantatges i també inconvenients. Els avantatges són, entre d'altres:

- Un estalvi de costos perquè disminueix el nombre de servidors físics.
- Es consumeix menys electricitat.

[39] http://www.getchef.com/chef/
[40] http://www.ansible.com/home
[41] http://es.wikipedia.org/wiki/Secure_Shell

- Simplifica l'administració.
- Es redueixen les parades de servei.
- Es reparteixen els recursos amb facilitat i precisió, augmentant la eficiència en la utilització de l'espai en el centre de dades.
- Els canvis de maquinari són totalment transparents.
- Es poden realitzar proves sense risc.

Per altra banda, també té els seus desavantatges, entre d'altres:

- Requereix coneixements cada vegada més especialitzats.
- El rendiment és inferior.
- L'avaria del servidor amfitrió de virtualització afecta a totes les màquines virtuals allotjades en ell.

5.5.2. Proxmox.

Proxmox és una distribució de virtualització que ofereix la possibilitat de gestionar servidors virtuals (VPS[42]), sota el que s'anomena contenidors, amb tecnologies OpenVZ[43] i Linux KVM[44]. La gestió es realitza fàcilment amb un clic a través de la interfície web (veure Figura 6). El que fa Proxmox es crea la màquina virtual en un ambient "chroot" directament en el sistema d'arxius del servidor amfitrió, que li dóna una sèrie d'avantatges:

- Administració de recursos de forma directa. Per exemple podem incrementar la RAM, SWAP, CPU, etc. I els canvis són aplicats a l'instant en el servidor virtual.
- L'ús del mateix sistema de fitxer per a totes les màquines virtuals no sobrecarrega el sistema amfitrió.
- La creació i esborrat de màquines virtuals és gairebé instantània.
- La velocitat de les aplicacions que es tenen corrent en un servidor virtual és gairebé la mateixa, com si es tractés d'un servidor base.

[42] http://es.wikipedia.org/wiki/Servidor_virtual_privado
[43] http://openvz.org/Main_Page
[44] http://www.linux-kvm.org/page/Main_Page

Aquesta tecnologia, però, no ens permet treballar sota sistemes Windows.

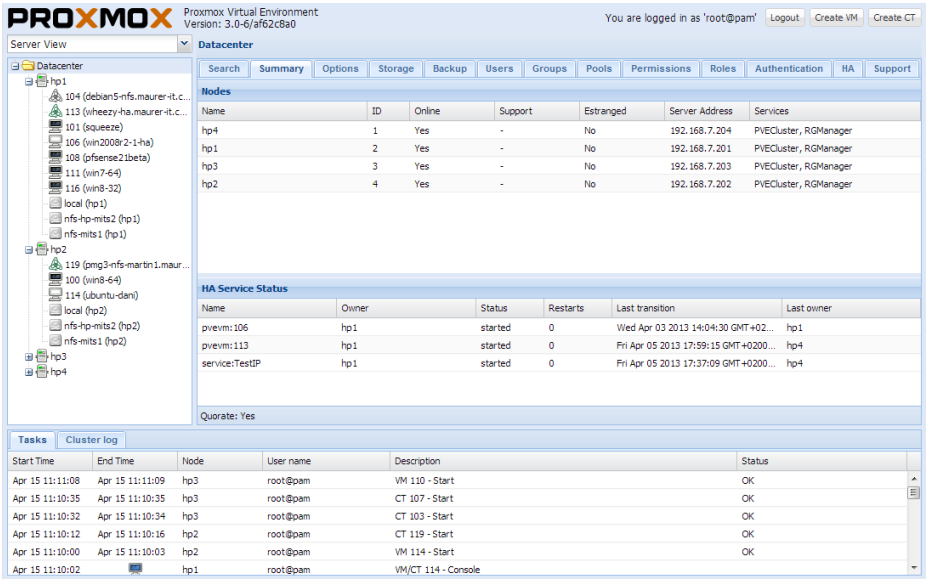


Figura 6: Interfície web Proxmox. Font: (Linuxaria)

5.5.3. Vagrant.

Vagrant és una eina escrita en Ruby i OpenSource per administrar màquines virtuals de manera simple des de la línia de comandes. Usa un entorn de virtualització basat en Virtualbox[45]. Et permet crear entorns aïllats, que poden ser reutilitzats per tot l'equip de treball en qualsevol sistema operatiu, com entorn de prova per scripts o automatització de servidors amb Puppet, Chef, etc.

Per exemple, és molt útil, quan un o diversos desenvolupadors disposen de diferents perfils i requisits en projectes. Programar en Ruby, Phyton, etc. amb diferents gestors de bases de dades, diversos sistemes operatius, diferents IDE, etc. Aquestes configuracions, a la llarga, poden provocar conflictes entre si.

Ofereix una solució basada en un aïllament per a cada perfil desitjat a través de la virtualització independent en màquines virtuals. Això no és molt diferent a altres solucions, però s'hauria de configurar el tipus de virtualització (OpenVZ, KVM, etc), instal·lar sistemes operatius, etc. Per això Vagrant actua com una eina independent de més alt nivell

[45] https://www.virtualbox.org/

on permet oblidar els detalls. D'aquesta manera, automatitza i estalvia temps a perfils tècnics diferents com poden ser desenvolupadors o equips de treball.

Hi ha molts programes de virtualització en el mercat, VMware vSphere[46], Parallels Desktop[47], XenServer[48], Windows Server 2008 R2 Hyper-V[49], etc. S'ha escollit Vagrant per la seva popularitat, comunitat i sobretot perque està escrita en Ruby. És un eina lleugera i fàcil d'instl·lar i amb comandes senzilles pots tenir un entorn replicable per a tots els desenvolupadors ràpidament.

5.6. Arquitectura del software.

5.6.1. API web.

API significa Application Programming Interface, és a dir, Interfície de Programació d'Aplicacions. Dit d'una altra manera, una API és una forma de donar un accés RESTringit a les funcions i la informació de l'aplicació sense deixar que un programador extern accedeixi directament al codi. D'aquesta forma només pot usar i executar aquelles funcions de l'aplicació que s'han decidit oferir externament.

L'API era un concepte que s'utilitzava i s'usa generalment en sistemes operatius perquè els programadors d'aplicacions poguessin cridar de forma fàcil a components ja programats del sistema operatiu. Però on ha tingut i està tenint un gran èxit actualment és en les aplicacions web.

El principal benefici d'una API es troba en l'estalvi de costos. Si s'ha instal·lat una aplicació, disposar d'una API farà que l'ús de les dades que s'estan tractant sigui relativament senzill implementar-les en una altra aplicació o servei. Per exemple, el cas d'una botiga física que utilitza un programa de facturació, que també el vol incorporar a la seva botiga online. Si es disposa d'una API, el desenvolupament d'aquesta funcionalitat serà molt més ràpida i barata.

[46] http://www.vmware.com/es/products/vsphere
[47] http://www.parallels.com/es/products/desktop/
[48] http://www.citrix.es/products/xenserver/overview.html
[49] http://es.wikipedia.org/wiki/Hyper-V

Però també es generaran noves oportunitats de negoci, ja que si es disposa d'una API se li està donant un valor afegit al producte. Els clients valoren que un programa disposi d'aquesta interfície, ja que a la llarga suposarà menys mals de cap per resoldre problemes.

Hi ha moltes empreses que ofereixen les seves API, com per exemple, Twitter[50], Facebook[51], Google[52], Slideshare[53], Amazon[54].

5.6.2. API REST.

REST o Representational State Transfer és un estil d'arquitectura o conjunt de principis, no una especificació, per dissenyar aplicacions en xarxa, basada en una arquitectura client-servidor, un protocol sense estat i comunicacions "cachesables". La idea és que, en lloc d'utilitzar els mecanismes complexos, com ara CORBA[55], RPC[56] o SOAP[57] per a la connexió entre màquines, s'usa el protocol HTTP. Per tant, s'accedeix a través d'adreces web o URL en què enviem les dades de la consulta. Com a resposta a la consulta s'obtenen dades en diferents formats, com poden ser text pla, XML, JSON, etc.

REST es va definir l'any 2000 per Roy Fielding[58], coautor principal de l'especificació HTTP.

Les aplicacions RESTful utilitzen CRUD a través de sol.licituds HTTP per enviar dades (create and/or update), llegir les dades (read) i eliminar les dades (delete).

[50] https://twitter.com/
[51] https://es-es.facebook.com/
[52] https://www.google.es/
[53] http://www.slideshare.net/
[54] http://www.amazon.es/
[55] http://es.wikipedia.org/wiki/CORBA
[56] http://es.wikipedia.org/wiki/Remote_Procedure_Call
[57] http://es.wikipedia.org/wiki/Simple_Object_Access_Protocol
[58] http://es.wikipedia.org/wiki/Roy_Fielding

En la Figura 7 es pot veure diferents peticions HTTP:

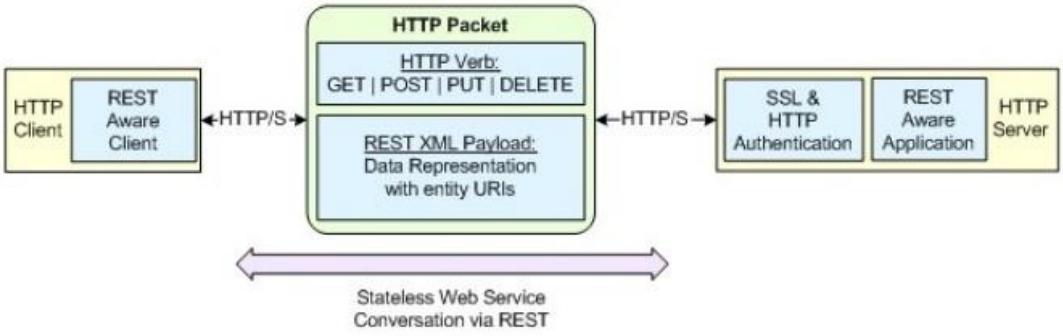


Figura 7: Peticions HTTP. Font: (PrideParrot)

Utilitzar REST simplifica l'arquitectura (millora el rendiment), les peticions es simplifiquen (augmenta la velocitat), facilitat d'escalabilitat i evolució dels seus components, accés a base de dades complexes, a peticions simples. Pel contrari, fan falta múltiples crides per aconseguir informació complexa.

Un servei RESTful es compon de:

- URI[59] del recurs.
  Per exemple: http://api.yourtalks.com/users/adriagalin/slideshows/1. Aquesta URI ens diria l'accés al recurs "Slideshow amb l'id 1 de l'usuari "adriagalin".
- El tipus de representació del recurs.
  Per exemple, podem tornar a la nostra capçalera HTTP el següent: "Content-type: application/json", de manera que el client sabrà que el contingut de la resposta és una cadena en format JSON.
- Operacions suportades.
  HTTP defineix diversos tipus d'operacions (verbs[60]), que poden set GET, PUT, POST, DELETE, PURGE, entre altres. És important saber perquè estan pensats cada verbs, de manera que siguin utilitzats correctament pels clients.
- Hipervincles.

[59] http://es.wikipedia.org/wiki/Uniform_Resource_Identifier
[60] https://es.wikipedia.org/wiki/Hypertext_Transfer_Protocol#M.C3.A9todos_de_petici.C3.B3n

La resposta pot incloure hipervincles cap a altres accions. Normalment s'inclouen el mateix contingut de la resposta, així, si per exemple, la nostra resposta és un objecte JSON, podem afegir una propietat més amb els hipervincles a les accions que admet l'objecte.

En la següent Figura 8 es mostra un exemple d'una petició GET al servei web de GitHub:

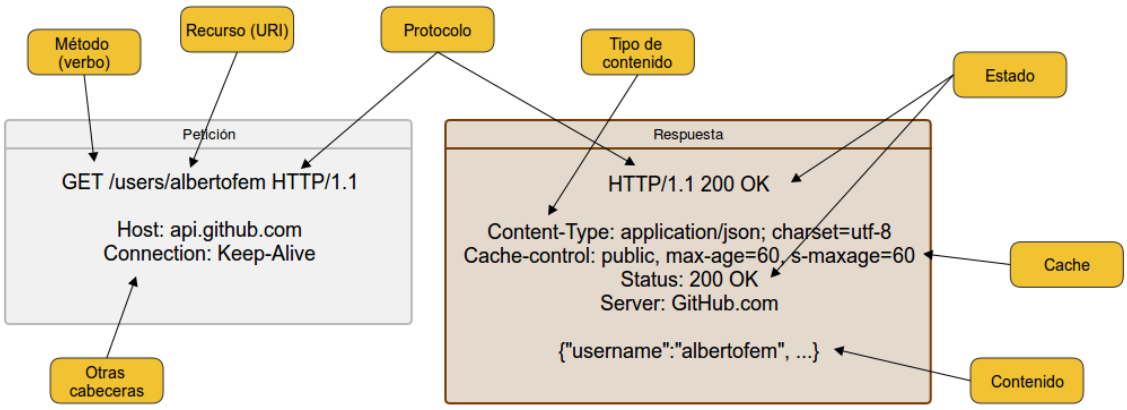


Figura 8: Exemple petició GET a Github. Font: (Github)

Hi ha varies alternatives a REST, entre d'altres:

RPC: crida a un procediment remot, és un protocol que permet a un programa executar codi resident en una altre màquina.

SOAP: defineix com dos objectes en diferents processos poden comunicar-se a través de XML.

5.7. Workers.

Un worker és un procés o tasca en segon pla, que esta escoltant a una cua de treball, per realitzar la tasca per el qual ha estat dissenyat.

Moltes vegades, es necessita fer una tasca intensiva en les aplicacions i no és lògic fer esperar el client. Alguns exemples poden ser:

- Enviament massiu de correus electrònics.
- Processament d'imatges.

- Importació i exportació de dades a la base de dades.
- Ús d'un servei extern.

Es recomana utilitzar processos en segon pla quan una petició al servidor es vagi a demorar més de 10 segons aproximadament.

6. Disseny de la plataforma.

6.1. Arquitectura de l'aplicació.

L'arquitectura de YourTalks es compon de:

Worker/s (CARLIT):

Són els encarregats de transformar les presentacions enviades per als usuaris, en imatges. La informació es guarda en el sistema de fitxers, es creen les diapositives a la base de dades MongoDB[61], i s'envia una notificació en temps real al client. També s'encarregà d'enviar un email al client, notificant l'estat de la presentació. Estan escrit en Ruby i el gestor de tasques és Sidekiq que utilitza Redis per gestionar-les.

API (PUIGMAL):

L'API s'encarrega de proveir informació a l'aplicació client, a través de peticions REST. S'ha desenvolupat amb Ruby on Rails.

API real time (PEDRAFORCA):

L'API de real time s'encarrega d'enviar notificacions en temps real, de la gestió del xat, i de la mostra de les presentacions. Esta escrita en NodeJS i Javascript. Per poder gestionar el temps real aquesta API esta subscrita a un broker Pub/Sub[62], anomenat Redis[63].

Frontend (ANETO):

És l'aplicació per la part de client, escrita totalment en Backbone.JS. Realitza peticions a la API REST per proveir-se d'informació i esta subscrita a la API de temps real.

La "llengua vehicular" de YourTalks és Ruby per la part de backend (servidor) i Javascript per la part de frontend (client).

[61] http://www.mongodb.org/
[62] http://en.wikipedia.org/wiki/Publish%E2%80%93subscribe_pattern
[63] http://redis.io/

El backend està escrit en Ruby on Rails, un framework per al desenvolupament d'aplicacions en Ruby, i Ruby per els processos en segon pla. La part en temps real en NodeJS, un entorn javascript del costat del servidor que utilitza un model asíncron i dirigit a esdeveniments, i útil en la creació de programes de xarxa altament escalables. Gran part del treball del backend és gestionar el processament de les presentacions, la peticions web que venen des del frontend i la gestió de les notificacions en temps real.

L'elecció de Ruby es deu a l'experiència prèvia amb el llenguatge. És un llenguatge molt expressiu i mantenible.

S'ha dissenyat una arquitectura en que el backend i el frontend han d'anar totalment separats. Són aplicacions independents, l'únic punt en comú és la base de dades: el backend hi escriu , el frontend en llegeix . S'utilitza una base de dades NoSQL[64], MongoDB, per la facilitat en el desenvolupament.

El backend transforma les presentacions enviades pels clients, escriu a la base de dades, proveeix d'una API per al frontend i envia les notificacions en temps real.

El frontend, anomenat Aneto, és una aplicació desenvolupada en Javascript per la part del client. El client té tota la lògica de navegació, quedant pel servidor la lògica de la funcionalitat. El client es connecta una vegada en el frontend per descarregar-se en forma d'un únic fitxer JS i un CSS miniaturitzat i comprimit aconseguint una web poc pesada. Després el client per cada petició es conectarà a la API REST per proveir-se d'informació i poder realitzar cada acció que requereixi la interacció amb l'API REST.

[64] http://es.wikipedia.org/wiki/NoSQL

En la següent figura 9 es pot veure l'estructura de l'aplicació amb el seus components interconnectats:

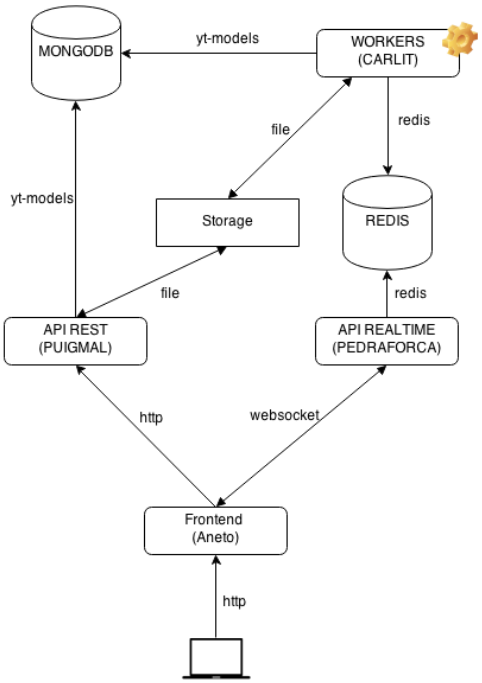


Figura 9: Estructura plataforma YourTalks. Font: el·laboració pròpia

Nota: s'ha escollit la terminologia de pic de muntanya per anomenar les diferents parts del projecte perquè els components del projecte en són aficionats.

6.1.1. Tecnologies utilitzades.

Aquí es fa una petita introducció al llenguatges de programació utilitzat per al desenvolupament de la API, els workers i l'automatització.

Ruby

És un llenguatge de programació interpretat, reflexiu i orientat a objectes, creat pel programador japonès Yukihiro "Matz" Matsumoto, qui va començar a treballar en Ruby el 1993, i el va presentar públicament el 1995. Combina una sintaxi inspirada en Python i Perl amb característiques de programació orientada a objectes similars a Smalltalk[65]. Comparteix també funcionalitat amb altres llenguatges de programació com Lisp[66], Lua[67],

[65] http://es.wikipedia.org/wiki/Smalltalk
[66] http://es.wikipedia.org/wiki/Lisp

Dylan[68] i CLU[69]. En Ruby tot és un objecte. Se li pot assignar propietats i accions a tota informació i codi. La programació orientada a objectes anomena a les propietats variables d'instància i les accions són conegudes com a mètodes. L'orientació a objectes pura de Ruby se sol demostrar amb un simple codi que aplica una acció en un nombre.

```
10.times { print "YourTalks.com – The best app" }
```

És considerat un llenguatge flexible, ja que en permet l'alteració als seus usuaris . Les parts essencials de Ruby es poden treure o redefinir-les com convingui. Intenta no restringir al desenvolupador.

Ruby es utilitzat per moltes empreses, entre elles 37Signals[70], Twitter, IBM[71], Amazon.

Ruby On Rails

És un entorn de desenvolupament web de codi obert que està optimitzat per a la satisfacció dels programadors i per a la productivitat sostenible, creat el 2003 per David Heinemeier Hansson. Permet escriure bon codi evitant repeteticions i afavorint la convenció abans que la configuració. Per tant, és un conjunt de llibreries, automatismes i convencions destinats a resoldre els problemes més comuns a l'hora de desenvolupar una aplicació web, perquè el programador pugui concentrar-se en els aspectes únics i diferencials del seu projecte en lloc dels problemes recurrents.

RoR es utilitzat per moltes empreses, entre elles Cnet[72], Twitter, Nasa[73], EA Electronic Art[74].

NodeJS

És un intèrpret Javascript, construït amb el motor de Javascript de Chrome[75], del costat del servidor. Amb l'objectiu de construir aplicacions altament escalables, i escriure un codi que manegi desenes de millers de connexions simultànies en una sola màquina física.

[67] http://www.lua.org/
[68] http://es.wikipedia.org/wiki/Dylan_(lenguaje_de_programaci%C3%B3n)
[69] http://es.wikipedia.org/wiki/CLU
[70] http://es.wikipedia.org/wiki/Smalltalk
[71] http://www.ibm.com/es/es/
[72] http://www.cnet.com/
[73] http://www.nasa.gov/
[74] http://www.ea.com/es/
[75] http://www.google.es/intl/es/chrome/browser/

NodeJS utilitza una programació orientada a esdeveniments, un model d'entrada/sortida no bloquejant que el fa lleuger i eficient, ideal per aplicacions en temps real amb molta informació que s'executen mitjançant de dispositius distribuïts. NodeJS en lloc de generar un nou fil d'execució per cada connexió, com fan altres servidors (Apache[76], Apache Tomcat), i de gastar més memòria de la màquina, cada connexió llença un esdeveniment dins del procés del servidor de Node. Va ser creat per Ryan Dahl el 2009 i la seva evolució està apadrinada per l'empresa Joyent.

NodeJS es utilitzat per moltes empreses, entre elles Yahoo[77], Linkedin[78], Google.

6.2. Arquitectura de sistemes.

En aquest apartat s'ha dissenyat un clúster actiu/passiu on funcionaran tots els serveis de la plataforma, d'aquesta manera si falla el servidor actiu o principal, el servei no s'aturarà perquè entrarà a funcionar el servidor passiu o secundari que passarà a ser l'actiu fins que es solucionin els problemes del servidor que ha fallat. En resum, la primera màquina és el node principal, i la segona és el node de suport i per tant, la feina del node de suport és fer-se càrrec dels serveis del node principal si aquest creu que la màquina no està responent.

Els usuaris accedeixen a la plataforma via http://www.YourTalks.com, i el servidor de DNS els redirigeix cap a 179.80.4.76. Es poden posar dos servidors executant la web amb diferents adreces IP, i simplement fer que el servidor de DNS redirigeixi les peticions cap al servidor de suport si el node principal queda inactiu. Aquesta opció, és una possibilitat, però si els clients tenen la IP del servidor principal en caché, continuaran entrant al sevidor caigut. Una altre possibilitat és que, si el servidor principal falla, el servidor de suport assumeixi la serva adreça IP i passi a ser ell que atengui les peticions. Aquest mètode es coneix com a IP Failover, esmentat anteriorment, i és el que s'utilitzarà. D'aquesta manera els clients continuarant accedint a http://www.YourTalks.com amb la mateixa IP, se'n transparent per al usuari.

Per tenir un sistema en alta disponible i tolerant a falles, cal tenir tots els serveis redundats. Per poder implementar l'arquitectura necessària pel projecte YourTalks es requereix de:

[76] http://httpd.apache.org/
[77] https://es.yahoo.com/
[78] https://es.linkedin.com/

- Dos servidors físics en datacenters separats.
- Un sistema de virtualització, en aquest cas Proxmox.
- Virtual IP (VIP)/IpFailover: és una adreça IP assignada a múltiples aplicacions que resideixen en un únic servidor, múltiples noms de domini, o diversos servidors, en lloc de ser assignada a un únic servidor o a la targeta de xarxa (NIC). Els paquets de dades entrants s'enviaran a l'adreça VIP que els dirigirà a les interfícies de xarxa reals. Utilitzant aquests tipus de IP, quan cau un servidor, el redireccionament cap a un altre servidor, és totalment transparent per l'usuari.
- Heartbeat[79]: amb aquest software s'estableix una comunicació entre els membres del clúster[80] (nodes) i es defineixen els serveis que es volen posar en alta disponibilitat. Un dels nodes serà el màster, que donarà el servei principal. Si cau, el/els altres nodes agafaran el relleu fins que el màster es recuperi.
- HAProxy[81]: és el servei encarregat de balancejar la càrrega. Al mateix temps, si el balencejador detecta la caiguda de uns dels servidors web, pot optar per no enviar-li més peticions. D'aquesta forma, si un dels servidors web cau, les peticions del client no es dirigiran al servidor caigut.
- VPN[82] Server: per poder-nos connectar de forma segura en els servidors i per crear una xarxa interna, per a la comunicació de les màquines.
- Tallafocs: serà l'encarregat de filtra els paquets de dades que rebrà el servidor físic.
- L'allotjament dels fitxers es realitzaria en un sistema NAS redundant amb alta disponibilitat que es buscaria en algun proveïdor. Les còpies de seguretat es realitzarien dues vegades al dia i es guardarien en una màquina externa la sistema o a un NAS.

Les petició entrant arriba a la IP virtual i és reenviada cap als servidors virtualitzats corresponent canviant la IP destí (NAT). La resposta del servidor virtualitzat torna al servidor de balanceig qui de nou canvia la IP i re-envia la resposta al servei peticionari. Com tot el tràfic passa pel balencejador, pot suposa un coll d'ampolla, per això es posa un balencejador secundari, per activar-lo en pujades del tràfic.

[79] http://www.linux-ha.org/wiki/Heartbeat
[80] http://es.wikipedia.org/wiki/Cl%C3%BAster_(inform%C3%A1tica)
[81] http://www.linux-ha.org/wiki/Heartbeat
[82] http://es.wikipedia.org/wiki/Red_privada_virtual

Perquè el node passiu pugui saber si el node actiu ha caigut, s'utilitza el programa Heartbeat. Cada node es comunica amb l'altre a través d'una connexió secundaria comprovant els seus batecs (d'aquí ve el nom de Heartbeat). El programa que realitza el canvi d'ips, és un script que parlar emb el proveïdor del hosting perque canvi la IP del servidor (veure Figura 10).

Figura 10: IP Failover OVH. Font: (OVH)

Quan el servei cau i s'ha d'utilitzar el node secundari, no es veuen les mateixes dades, per tant no tenim integritat de les dades. Hi ha moltes solucions però s'ha decantat per un NAS redundat que ofereixi algun proveïdor, d'aquesta manera ja no s'ha de tenir en compte la integritat de les dades ni de l'alta disponibilitat del sistema de fitxers. També es pot autogestionar però llavors requeria més temps i recursos, el qual no es vol malgastar si algun proveïdor ja ens ofereix aquesta opció.

Per a la base de dades s'ha optat per una replicació de la base de dades en tres servidors virtualitzats. Repartits entre les dues màquines, per tant, quan el node princpial caigui, la base de dades continuarà funcionant amb totes les seves dades, evitant la restauració d'una còpia de seguretat.

També es crearà una xarxa interna entre els dos nodes principals per poder comunicar-se entre màquines dins un entorn segur. El servidor de VPN es troba en el node principal però si el node principal cau, el node secundari agafarà el rol del servidor VPN perque no es perdi la connexió entre màquines. D'aquesta manera es manté una visió global de la configuració i estat del clúster. Ja que la comunicació entre els nodes del clúster és essencial per al funcionament d'aquest, és habitual utilitzar un canal específic, com una xarxa independent, que no es pugui veure afectada per problemes de seguretat o rendiment.

Amb aquesta arquitectura pot passar que un node deixi de funcionar però encara segueixi aixecat, accedint a certs recursos i responen a peticions. Per evitar que el node corrompi els

recursos o respongui amb peticions, s'utilitza la tècnica anomenada Fencing. La funció principal del Fencing es fer-li saber a aquest node que està funcionant incorrectament, retiri els seus recursos assignats perque els atenguin els altres nodes, i deixar-lo en un estat inactiu.

Per finalitzar es mostra una taula resum (Taula 4) dels servidors necessaris per a la plataforma i un esquema d'una arquitectura d'alta disponibilitat (Figura 11):

HOST0 (Datacenter 1)	HOST1 (Datacenter 2)
balencejador amb ReverseProxy	balencejador amb ReverseProxy
servidor d'aplicacions rails	servidor d'aplicacions rails
servidor redis	servidor redis
servidor real time	servidor real time
servidor worker	servidor worker
servidor mongo0	servidor mongo1
servidor mongo3	
NAS	NAS

Taula 4: Resum màquines necessàries per HA. Font: el·laboració pròpia

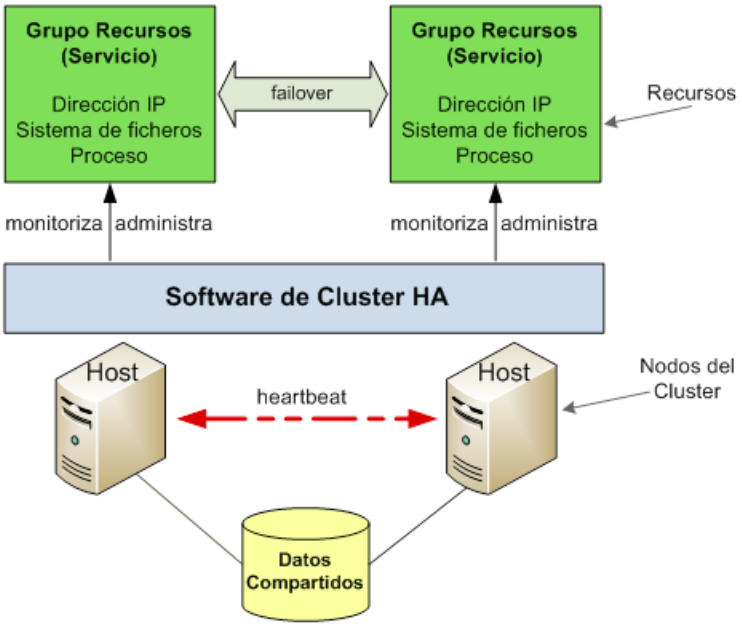


Figura 11: Arquitectura en alta disponibilitat. Font: www.google.es

6.2.1. Diagrama de l'arquitectura.

Tal com s'ha comentat anteriorment en el següent diagrama () es pot veure el sistema redundant i el servei de heartbeat pendent dels servidors.

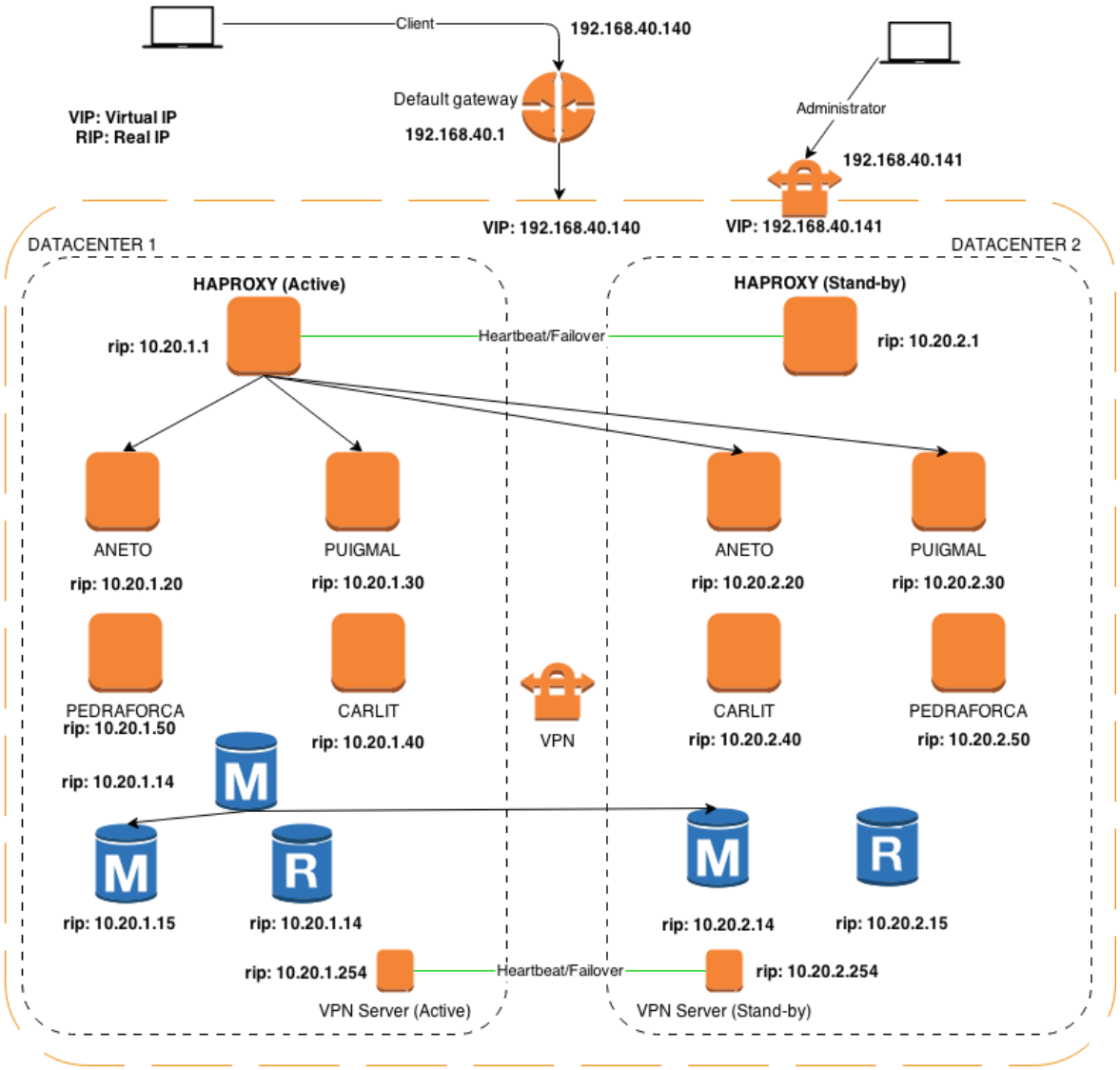


Figura 12: Esquema en HA de YourTalks. Font: el·laboració pròpia

6.2.2. Virtualització.

Per muntar tota la infraestructura planejada calen diners, però en la fase que es troba el projecte, és una despesa innecessària. Per poder realitzar les proves en un entorn de entremig entre producció i desenvolupament, que s'anomenarà preproducció o pre. S'ha

dissenyat i muntat una arquitectura de sistemes sobre una màquina física i servidors virtualizats. El servidor utilitzat és de l'empresa OVH i el sistema operatiu de virtualització és Proxmox. De moment, tota la configuració s'ha fet amb scripts, però s'ha pensat l'automatització de la creació de màquines virtuals amb el sistema Proxmox a partir de la seva API REST i la creació d'script de boostrap quan arrenqui la màquina, el qual instal·larà puppet i el repositori de puppet per a la seva execució.

Aquí es mostren les màquines creades en el sistema Proxmox per en funcionament de la plataforma en un entorn pre. Cada màquina té un hostname, el qual defineix la seva funció (veure Figura 13)

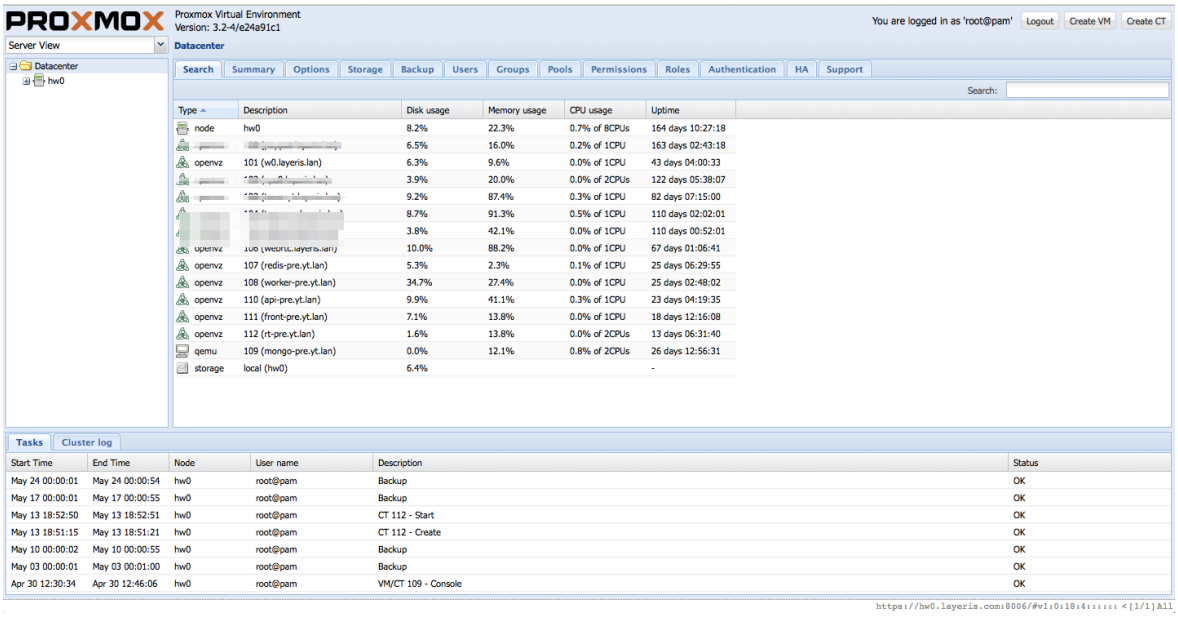


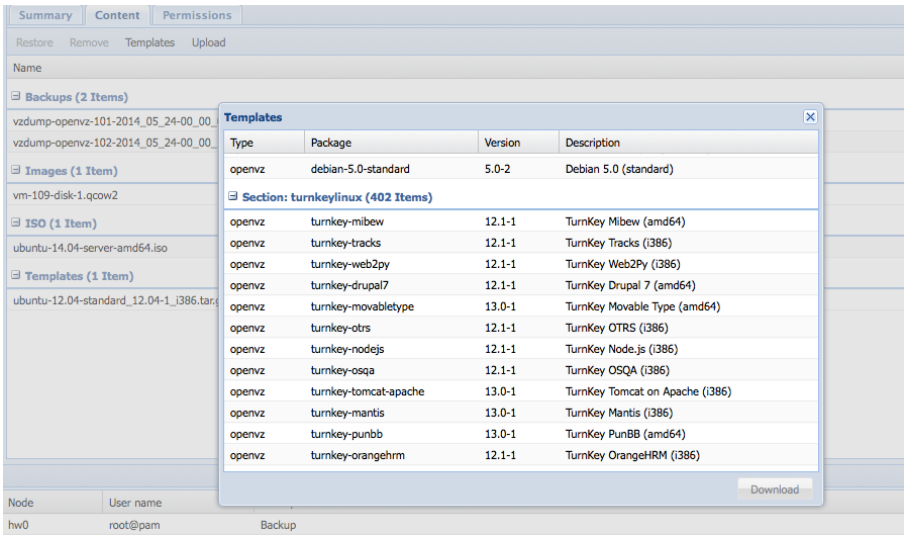
Figura 13: Hosts YourTalks. Font: Proxmox Yourtalks

CREACIÓ D'UNA MÀQUINA VIRTUAL AMB PROXMOX

Per crear màquines virtuals amb Proxmox, cal anar a la seva interfície web:

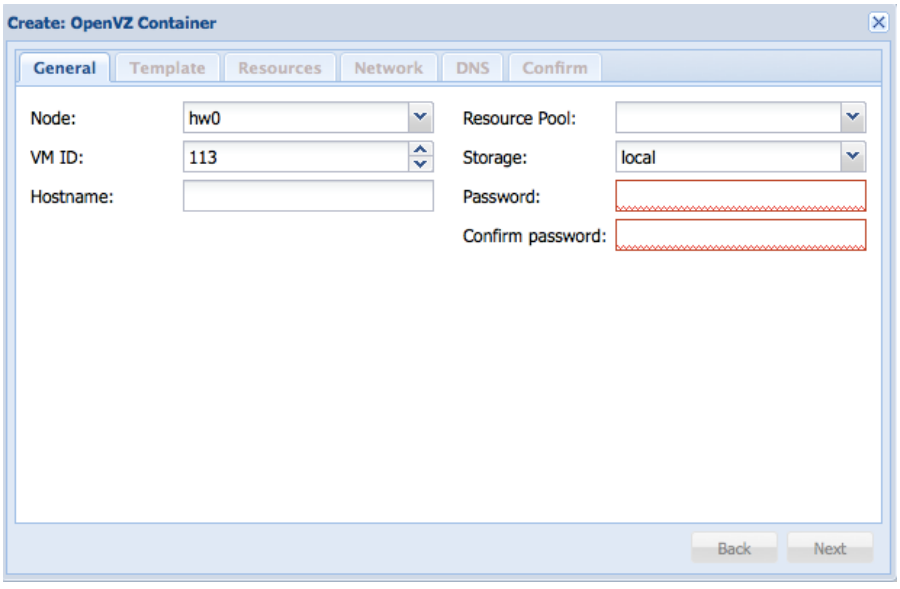
1. Es baixa el template del sistema operatiu:

   "Storage local" -> Template -> i descarreguem "Ubuntu 12.04"



2. Un cop descarregat el client, es pot crear el contenidor:

   Create CT -> Es completen les dades que tindrà el contenidor.



3. Ja es pot accedir a la màquina virtual a través de ssh.

Si ens connectem al servidor Proxmox veurem el chroot* del contenidor que s'acaba de crear.

Gràcies a la virtualització es pot administrar els recursos de la màquina virtual sense reiniciar la màquina virtual. Per exemple, s'ha creat el contenidor amb 2GB de ram, 2G de swap i 25GB de disc.

Ara es modifica els recursos a 4GB de ram, 4Gb de swap i 50Gb de disc, de la següent manera:

Seleccionar el contenidor -> Resources i s'ajusten els valors. Ara en màquina virtual es veu els canvis realitzat, de forma immediata.

```
root@hw0:~# df -h
Filesystem              Size  Used Avail Use% Mounted on
udev                     10M     0   10M   0% /dev
tmpfs                   3.2G  380K  3.2G   1% /run
/dev/md1                 20G  1.6G   17G   9% /
tmpfs                   5.0M     0  5.0M   0% /run/lock
tmpfs                   6.7G   22M  6.7G   1% /run/shm
/dev/mapper/pve-data    1.8T   25G  1.7T   2% /var/lib/vz
/var/lib/vz/private/100  10G  662M  9.4G   7% /var/lib/vz/root/100
none                    100M  996K  100M   1% /var/lib/vz/root/100/run
none                    5.0M     0  5.0M   0% /var/lib/vz/root/100/run/lock
none                    500M     0  500M   0% /var/lib/vz/root/100/run/shm
/var/lib/vz/private/102  50G  2.0G   49G   4% /var/lib/vz/root/102
none                    400M 1004K  400M   1% /var/lib/vz/root/102/run
none                    5.0M     0  5.0M   0% /var/lib/vz/root/102/run/lock
none                    2.0G     0  2.0G   0% /var/lib/vz/root/102/run/shm
```

CREACIÓ D'UNA MÀQUINA VIRTUAL AMB VAGRANT

S'ha utilitzat Vagrant perquè el DevOps pugui crear un entorn aïllat, que pugui ser reutilitzat per tot l'equip de treball en qualsevol sistema operatiu, gràcies a que Vagrant treballa amb "Vagrantfile", un arxiu on es troba la configuració de l'entorn per a una màquina virtual. Permet crear entorns de test tant per al desenvolupador com per l'administrador de sistemes, el qual testejarà scripts d'automatització realitzat en Puppet en aquest cas.

Per tenir un entorn virtualizat amb Vagrant cal instal·lar-lo i tenir instal·lat Virtualbox. (veure la documentació oficial: http://vagrantup.com).

Per crear una màquina virtual s'executa la següent comanda:

```
vagrant nit
```

Aquesta comada crea el fitxer "Vagrantfile" que és l'especificació de la màquina virtual:

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# Vagrantfile api/syntax version. Don't touch unless you know what you're doing!
VAGRANTFILE_API_VERSION = "2"

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
  # All Vagrant configuration is done here. The most common configuration
  # options are documented and commented below. For a complete reference,
  # please see the online documentation at vagrantup.com.

  # Every Vagrant virtual environment requires a box to build off of.
  config.vm.box = "base"

end
```

Com el fitxer indica, cada entorn virtual Vagrant requereix d'un box. Un box és una estrucutura utilitzada per Vagrant per a construir màquines virtuals, la majoria de boxes estan configurats per treballs en Vagrant. Aquesta boxes es poden descarregar directament de la pàgina oficial.

Per afegir un box s'han d'executar les següents comandes:

```
vagrant box add base http://files.vagrantup.com/precise64.box
```

Per inicialitzar la màquina virtual i poder entra-hi s'executa:

```
vagrant up
vagrant ssh
```

```
vagrant up
Bringing machine 'web' up with 'virtualbox' provider...
Bringing machine 'web2' up with 'virtualbox' provider...
==> web: Clearing any previously set forwarded ports...
==> web: Clearing any previously set network interfaces...
==> web: Preparing network interfaces based on configuration...
    web: Adapter 1: nat
    web: Adapter 2: hostonly
==> web: Forwarding ports...
    web: 22 => 2222 (adapter 1)
==> web: Running 'pre-boot' VM customizations...
==> web: Booting VM...
==> web: Waiting for machine to boot. This may take a few minutes...
    web: SSH address: 127.0.0.1:2222
    web: SSH username: vagrant
    web: SSH auth method: private key
    web: Warning: Connection timeout. Retrying...
==> web: Machine booted and ready!
```

Al accedir per SSH, se'ns donarà accés a un usuari anomenat "vagrant", aquest usuari té privilegis amb sudo, així que no es necessita utilitzar l'usuari root del sistema.

```
Welcome to your Vagrant-built virtual machine.
vagrant@web:~$ ls
postinstall.sh
vagrant@web:~$ ls -la
total 52
drwxr-xr-x 4 vagrant vagrant 4096 May 24 22:01 .
drwxr-xr-x 3 root    root    4096 Sep 14  2012 ..
-rw------- 1 vagrant vagrant  806 May 24 23:16 .bash_history
-rw-r--r-- 1 vagrant vagrant  220 Sep 14  2012 .bash_logout
-rw-r--r-- 1 vagrant vagrant 3486 Sep 14  2012 .bashrc
drwx------ 2 vagrant vagrant 4096 Sep 14  2012 .cache
-rwxr-xr-x 1 vagrant vagrant 6487 Sep 14  2012 postinstall.sh
-rw-r--r-- 1 vagrant vagrant  675 Sep 14  2012 .profile
drwx------ 2 vagrant vagrant 4096 Sep 14  2012 .ssh
-rw-r--r-- 1 vagrant vagrant    0 Sep 14  2012 .sudo_as_admin_successful
-rw------- 1 vagrant vagrant    6 Sep 14  2012 .vbox_version
-rw------- 1 vagrant vagrant   12 Sep 14  2012 .veewee_version
-rw------- 1 root    root     705 May 24 22:01 .viminfo
vagrant@web:~$
```

Una vegada realitzada totes les configuracions desitjades, com la configuració del Nginx, un entorn de Rails, etc. Ja es pot empaquetar la màquina virtual per a la seva distribució entre les diferents persones de l'equip de treball.

```
vagrant package
```

El que fa aquesta comanda es empaquetar l'entorn actual en un arxiu box per a la seva distribució.

6.2.3. Automatització (Puppet).

6.2.3.1. Que es vol?

Es necessita crear una infraestructura escalable i d'alta disponibilitat de forma automatitzada, i Puppet ens permet fer això. Es mostrarà la configuració en un entorn real i en un entorn de desenvolupament Vagrant a partir d'un exemple més petit, ja que la configuració real requereix més temps per poder-la desenvolupar.

6.2.3.2. Configuració en un entorn real.

Escenari 1: Puppet amb un node central, Puppetmaster.

La instal·lació del servidor central de Puppet, anomenat Puppetmaster així com la configuració d'alguns nodes que seran controlats a través d'ell. L'arquitectura de les màquines serà la següent:

- **puppet.pre.yourtalks.com (ip: 10.20.1.200**): és el servidor Puppet responsable de la configuració de les màquines.
- **web.pre.yourtalks.com (ip: 10.20.1.2):** és un client, que conte un servei web bàsic.
- **web2.pre.yourtalks.com (ip: 10.20.1.3):** és un client, que conte un servei web bàsic.

És important que el client Puppet i el servidor uppet es puguin comunicar a través del port 8140.

S'han d'instal·lar els següents paquets:

- A puppet.pre.yourtalks.com s'ha d'instal·lar:

```
apt-get install puppetmaster
```

- en els client web, web2 hem d'instal·lar:

apt-get install puppet

Aquest procediment també es podria fer executant un script a les màquines clients quan aquestes arrenquin, d'aquesta manera augmentem l'automatització, com el següent:

```
#!/usr/bin/env bash

set -e  if [ "$EUID" -ne "0" ] ; then
      echo "Script must be run as root." >&2
      exit 1
fi
if which puppet > /dev/null ; then
      echo "Puppet is already installed"
      exit 0
fi
echo "Installing Puppet repo for Ubuntu 12.04 LTS"
wget -qO /tmp/puppetlabs-release-precise.deb \
https://apt.puppetlabs.com/puppetlabs-release-precise.deb
dpkg -i /tmp/puppetlabs-release-precise.deb
rm /tmp/puppetlabs-release-precise.deb
apt-get update
echo Installing puppet
apt-get install -y puppet
echo "Puppet installed!"
```

Resolució de noms (DNS)

Un cop instal·lats els paquets, s'han de configurar les dns locals de cada client perquè resolguin el servidor de puppetmaster. S'ha d'editar el fitxer "/etc/hosts", afegint la següent línia:

```
...
...
10.20.1.200 puppet
```

Creació dels fitxers de configuració

Es crea l'estructura bàsica d'un projecte puppet, tota la configuració dels servidors es posa a git, un control de versions.

```
|— manifests
|   └— site.pp
|— modules
|   └— module
|      |— files
|      |— templates
|      └— manifests
```

El fitxer "site.pp" és el que definirà a les màquines:

Nota: hi ha moltes maneres de programar un projecte de Puppet, que inclou patrons, etc. Aquí ens hem centrat en la part més bàsica.

```
node 'puppet' {}

node   /^\w+\.pre.YourTalks\.com/
{
   class { 'apache':
      mpm_module    => 'prefork',
   }
}
```

Es defineix el node "Puppet", per si en el futur se li volem fer modificacions.

Es defineix una expressió regular, per gestionar tots els nodes iguals. /^\w+\.pre.YourTalks\.com/ instal·larà apache a tots els servidors client puppet que tingui pre.YourTalks.com en el seu nom de màquina.

Mòduls

Puppet ens permet crear mòduls, o reutilitzar mòduls (veure forge.puppetlabs.com) per crear la configuració dels diferents serveis. S'ha utilitzat el mòdul apache creat per PuppetLabs.

Es crea la carpeta "modules" a l'arrel del projecte i es descarrega el mòdul d'apache:

```
mkdir modules && cd modules
puppet module install puppetlabs-apache
```

Es puja tot en el control de versions i s'actualitza el servidor de puppetmaster amb la configuració.

Configuració del servidor Puppetmaster

Per poder servir fitxers als servidors clients Puppet, el primer que s'ha de fer és modificar el fitxer "/etc/puppet/fileserver.conf". En aquest fitxer s'especificarà a quin rang de màquines de la xarxa es vol servir els fitxers. Per exemple, si la xarxa és la 10.20.1.0 i es vol servir fitxers de Puppet a totes les màquines d'aquesta xarxa, s'afegirà la següent línia en el fitxer:

```
[files]
path /etc/puppet/files
allow 10.20.1.0/24
```

En aquest fitxer també es pot utilitzar deny per denegar l'accés a certes màquines. L'allow sempre preval sobre deny. Es especificar un nom de domini:

```
[files]
path /etc/puppet/files
allow
*.pre.YourTalks.com
```

Seguidament, s'ha de modificar el fitxer "/etc/puppet/puppet.conf". Per defecte, aquest conté la localització dels directoris dels logs, de les dades, dels certificats SSL, etc. A la configuració bàsica cal afegir-li les següents línies:

```
[main]
logdir=/var/log/puppet
vardir=/var/lib/puppet
ssldir=/var/lib/puppet/ssl
rundir=/var/run/puppet
factpath=$vardir/lib/facter
pluginsync=true
```

```
[puppetmasterd]
templatedir=/var/lib/puppet/templates
certname=puppet
autosign=false
```

A la penúltima línia (certname=puppet) s'indica que el servidor de certificació es diu Puppet, és el nom de la màquina on s'ha instal·lat Puppetmaster. En la última línia (autosign=false) s'esta indicant al servidor que no ha de signar les peticions de certificats que li arribin de forma automàtica. Al estar en una xarxa local, es posa l'autosign a false perquè els nous servidors clients Puppet puguin parlar amb el servidor Puppetmaster automàticament. De l'altre manera, posant autosign a true, s'hauria d'anar al servidor Puppetmaster i signar els certificats.

Configuració dels clients

S'ha d'editar el fitxer "/etc/puppet/puppet.conf" i afegir una línia:

```
[agent]
server=puppet
runinterval=3600
logdir=/var/log/puppet
vardir=/var/lib/puppet
ssldir=/var/lib/puppet/ssl
rundir=/var/run/puppet
factpath=$vardir/lib/facter
# pluginsync=true
```

Iniciem serveis

Perque el sistema funcioni, han d'estar iniciats els servei puppetmaster del servidor puppet i el servei Puppet en els servidors clients puppet. Si no és així, s'inicien:

```
En el servidor:
/etc/init.d/puppetmaster start

En el client:
/etc/init.d/puppet start
```

Si hi ha algun problema en el funcionament, es pot observar els fitxers de logs amb:

```
tail –f /var/log/puppet -n 100
```

El servidor client puppet contacta amb el servidor puppet cada 30 minuts per defecte.

Per provar si un canvi en la configuració dels manifests o les tasques s'executen correctament en un servidor client Puppet, parem el servei, i exectuem:

```
puppetd –t
```

Una vegada s'ha comprovat que les tasques s'executen correctament, tornem a iniciar el servei.

No cal reiniciar els serveis de Puppet i Puppetmaster després de fer canvis els fitxers de configuració o manifests. El servidor Puppetmaster els monitoriza.

Escenari 2: Masterless

S'utilizarà la mateixa configuració de puppet però cada màquina gestiona els seus Puppets. En aquesta escenari és molt important el control de versions, perquè cada màquina tingui la mateixa configuració de puppet. L'arquitectura de les màquines serà la següent:

- **web.pre.yourtalks.com (ip: 10.20.1.2**): és un client, que conte un servei web basic.
- **web2.pre.yourtalks.com (ip: 10.20.1.3):** és un client, que conté un servei web bàsic.

S'ha d'instal·lar el següent paquet:

```
apt-get install puppet git
```

Es clona la configuració de Puppet a cada client amb:

```
cd /etc/puppet/
git clone repo_url
```

Es configura cron perquè executi el Puppet cada cert temps, i consulti en el repositori si hi ha canvis. Es crea el següent script:

```
#!/bin/bash
cd /etc/puppet/
git pull origin master
puppet agent –no-daemonize --onetime
```

En el cron es posa la següent comanda perquè s'executi cada hora:

```
* 1 * * * /path/script/puppet_apply.sh
```

Nota: aquesta configuració es pot posar en un script, que s'executi quan s'inicia la màquina.

6.2.3.3. Configuració en un entorn de desenvolupament.

Per desenvolupar i testejar la configuració de Puppet s'ha utilitzat màquines virtuals amb Vagrant.

Vagrant suporta aprovisionament de servidors amb Puppet entre altres, inclou l'agent Puppet. Utilitzant vagrant tenim un flux de treball de tipus Masterless.

Es crearà un màquina vagrant senzilla, per mostrar el funcionament, on s'instal·larà un servei apache, a partir de la configuració mostrada en els punts anteriors.

Primerament es descarrega Vagrant a través de la url: https://www.vagrantup.com/downloads.html, i instal.lem el paquet.

Es crea una carpeta per al projecte i s'inicia Vagrant:

```
mdkir
YourTalks_puppet
cd YourTalks_puppet
vagrant init
```

La comanda Vagrant init ens crea el fitxer de configuració de la màquina virtual. Abans però es descarrega plantilles amb sistema operatiu, anomenades base boxes.

```
vagrant box add precise64 http://files.vagrantup.com/precise64.box
```

Ara es configura el fitxer Vagrantfile de la següent manera:

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

DOMAIN="pre.YourTalks.com"
MEMORY=512
SUBNET="10.20.1"
VAGRANTFILE_API_VERSION = "2"

puppet_nodes = [
```

```
  { :hostname => 'web', :ip => '2', :box => 'precise64' },
  { :hostname => 'web2', :ip => '3', :box => 'precise64' }
]

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
  puppet_nodes.each do |node|
    config.vm.define node[:hostname].to_sym do |node_config|
      node_config.vm.box = node[:box]
      node_config.vm.hostname = "#{node[:hostname]}.#{DOMAIN}"
      node_config.vm.network :private_network, ip: "#{SUBNET}.#{node[:ip]}"

      node_config.vm.provider :virtualbox do |vb|
        vb.customize [
          'modifyvm', :id,
          '--memory', MEMORY
        ]
      end

      node_config.vm.provision :puppet do |puppet|
        puppet.facter = {
          'fqdn' => node_config.vm.hostname,
        }

        puppet.module_path    = "puppet/modules"
        puppet.manifests_path = "puppet/manifests"
        puppet.manifest_file  = "site.pp"
        puppet.options                                    =       "--verbose       --
fileserverconfig=/vagrant/puppet/fileserver.conf"

      end
    end
  end
end
```

Es crea l'estructura mostrada anteriorment, i s'inclouen tots els fitxers.

```
mkdir –p puppet/{manifests,modules,files}
vim puppet/manifests/site.pp
```

Es crea el fitxer puppet/manifests/site.pp amb el següent contingut, ara ja no cal el node de Puppet:

```
node /^\w+\.pre.YourTalks\.com/  {
  class { 'apache':
    mpm_module => 'prefork',
```

```
  }
  apache::vhost { $::fqdn:
    port    => '80',
    docroot => '/var/www/test',
  }
  file { "/var/www/test/index.html":
    ensure => "present",
    source => [

"puppet:///files/apache_files/web/index.html"
    ],
  }
}
```

A la carpeta modules, es crea un carpeta per posar els fitxers de la web:

```
mkdir –p files/apache_files/web
```

Es crea el fitxer index.html a modules/apache_files/web:

```
<html>
  <head>
    <title>
      web test YourTalks
    </title>
  </head>
  <body>
    <h1>  Ohhh!!!  LA  LA  LA
MAGIC!</h1>
    <h2>YOURTALKS</h2>
  </body>
</html>
```

Seguidament s'instal·la el mòdul d'Apache i les seves dependències:

```
cd modules
git clone https://github.com/puppetlabs/puppetlabs-apache apache
git clone https://github.com/puppetlabs/puppetlabs-stdlib stdlib
git clone https://github.com/puppetlabs/puppetlabs-concat concat
```

Ara s'executa la següent comanda per crear la màquina virtual:

```
vagrant up
```

Si es fa un canvi en els fitxers de configuració de Puppet, i es vol aplicar els canvis, cal executar:

```
vagrant provision
```

Per veure la web, cal que s'editi el fitxer de hosts, "/etc/hosts", afegint les següents línies:

```
10.20.1.2 web.pre.YourTalks.com
10.20.1.3
web2.pre.YourTalks.com
```

Ara ja es pot anar al navegador per veure el resultat.

6.2.3.4. Arquitectura d'automatització escollida.

Per a l'arquitectura de servidors de YourTalks.com s'ha escollit l'escenari 2, masterless. Per la seva descentralització, per tant, no dependència d'un node central i simplicitat. D'aquesta forma es té un estricte control sobre com s'aplica la configuració a un servidor. Es pot tenir parel·lització, amb un servidor centralitzat, el servidor manté una única versió de la configuració. Sense el servidor central, el repositori de control de codi font manté la versió, de manera que molta gent pot treballar en paral·lel. No existeix un únic punt de fallada, mantenir un servidor central requereix temps i alta disponibilitat del servei. D'aquesta manera ja no s'ha de mantenir un altra sistema en alta disponibilitat, i es redueixen costos.

Amb el Puppet s'han realitzat scripts per veure el seu funcionament. S'ha creat un script per gestionar un wordpress i saber l'estat de la infraestructura. Tot això, en un entorn local. Es requereix de més temps per desenvolupar tota una infraestructura de màquines a través d'un codi, sobretot per realitzar una estructuració acurada i fàcil d'adaptar-se a noves necessitats.

6.3. Arquitectura de l'API.

És un API REST basada en rails, que ens permetrà crear aplicacions entorn ella, i obrir-la al públic perquè puguin crear aplicacions.

6.3.1. Diagrama de base de dades.

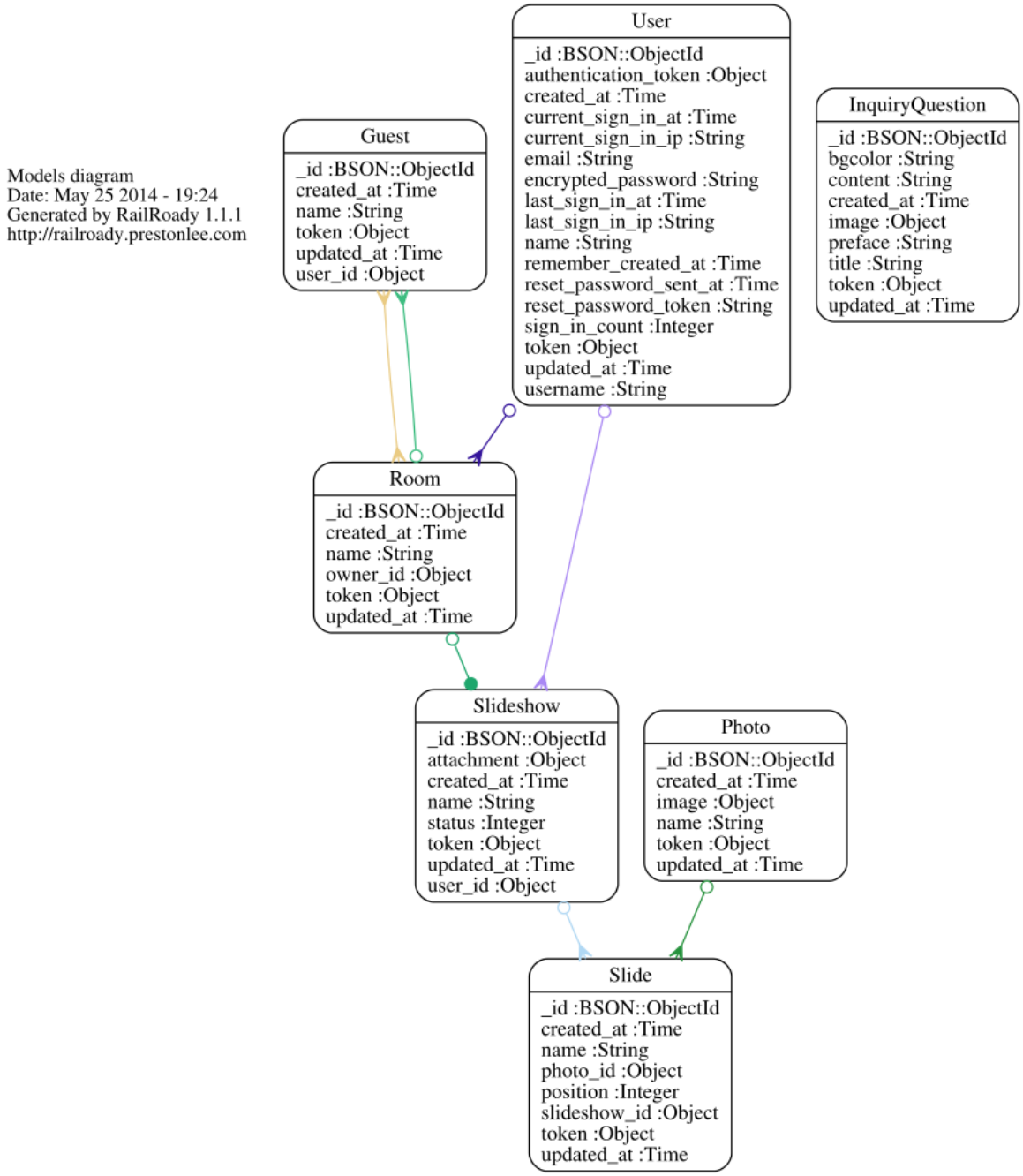


Figura 14: Diagrama de base de dades de YourTalks. Font: el·laboració pròpia

6.3.2. Seguretat.

La seguretat de l'API està implementada seguint el protocol HTTP a través de la capçalera HTTP i d'una comunicació xifrada per SSL, en un futur.

6.3.3. Proves.

Les fases de proves funcionals s'han dividit per cada controlador. A la finalització de cada controlador s'han realitzat un conjunt de proves funcionals que consisteixen en simular la interacció del frontend amb l'API detectant possibles errors abans de continuar amb els següents controladors, evitant transferir a altres parts del codi els possibles errors.

Finalment un cop completat el desenvolupament i la integració de cada part implicada, frontend, workers i l'API de temps real. S'ha realitzat una bateria de proves per comprovar el funcionament final de l'aplicació, passant per totes les funcions disponibles i amb la intenció de buscar possibles errors.

6.3.4. API Puigmal Guidelines.

Aquesta breu documentació mostra l'esquema bàsic de l'API REST de YourTalks. L'estat HTTP, els tipus de camps que s'utilitza en les respostes, el format JSON per representar l'esquema (amb Éxit o Error).

Verbs HTTP

Aquests són els diferents verbs HTTP que s'utilitzen:

Verb	Definició
GET	S'utilitza per recuperar recursos.
POST	S'utilitza per a la creació de recursos, o la realització d'accions personalitzades.
PATCH	S'utilitza per l'actualització de recursos amb les dades JSON parcials.

PUT	S'utilitza per reemplaçar els recursos o col·leccions.
DELETE	S'utilitza per eliminar els recursos

Taula 5: Verbs HTTP. Font: el•laboració pròpia

Estats HTTP

Totes les respostes de l'API utilitzen les capçaleres per identificar el que succeeix amb la resposta.

Estat	Definició
SUCCESS	
200	Quan tot va bé
201	Quan el recurs es creat correctament.
204	Sense contingut, l'acció s'ha realitzat, però la resposta no inclou una entitat.
CLIENT ERROR	
400	Sol·licitud incorrecte
401	No autoritzat, autenticació no vàlida
403	Forbidden, quan està autenticat, però no té permís per accedir al contingut.
404	No trobat, quan el recurs no es trobat.
422	Entitat no procesable, quan el recurs conté errors.
SERVER ERROR	
500	Internal Server Error, error en el servidor

Taula 6: Estats HTTP. Font: el•laboració pròpia

Tipus de camps

Aquests són els diferents tipus de camps que es poden retornar i el seu format:

- **Integers**: un enter, ex: 100, -100
- **Floats**: un decimal separat per un .(punt), ex: 1.9
- **String**: una cadena de caràcters, com per exemple: "YourTalks".
- **Array**: Una sèrie de números, cadenes hash o el tipus que es vulgui passar. Si la petició retorna una col·lecció d'objectes, per exemple: GET http://api-pre.yourtalks.com/v1/inquiry_questions, la resposta serà una matriu que conté el hash de cada inquiry question.
- **Currency**: Sempre en centaus d'euro per evitar separadors, ex: 50,60 = 5060
- **Blank**: Els camps en blanc s'inclouen com a nil.
- **Timestamp**: Totes les marques de temps es tornen en el format ISO 8601: AAAA-MM-DDTHH:MM:SSZ
- **Pictures**: Per defecte sempre es donarà la url original, per tant cal fer una substitució segons l'style que es desitgi, canviant la part original, per l'estil desitjat.

Format JSON

Aquesta és l'estructura JSON dels missatges.

- **Success**: si la sol·licitud ha tingut èxit.


```
{
   "name": "Adria",
   "email": "adria1@gmail.com",
   "username": "adria1",
   "token": "johKcydXOzBVtiMMfrTLeOdu8dY",
   "authentication_token": "jFZtKtN6_HGemayvmiaU"
}
```


- **Error**: si la sol·licitud no ha tingut èxit, es tornarà un JSON amb els errors.


```
{
   "message": "Validation Failed",
   "errors": [
      {
         "resource": "User",
         "field": "email",
```

```
            "code": 103
        }
    ]
}
```

Taula amb els diferents tipus d'error:

Codi	Nom de l'error	Descripció
100	missing	Un recurs no existeix.
101	missing field	Un camp obligatori en un recurs no s'ha establert.
102	invalid	El format d'un camp no és vàlid.
103	already exists	Un altre recurs te el mateix valor.
104	not auth	La sol·licitud necessita autenticació.
105	not found	El recurs no existeix.

Taula 7: Errors API YourTalks. Font: el•laboració pròpia

Algunes sol·licituds poden tenir errors personalitzats, s'ha de mirar la documentació de cada sol·licitud.

6.3.5. Estructura.

Users

New user

Descripció	Et permet crear un nou usuari.		
Path	{{url}}/v1/user		
Verb	POST		
Paràmetres	Paràmetre	Obligatori?	Explicació

	user_name	No	Nom de l'usuari
	user_username	Si	Nom especial per l'usuari
	user_email	Si	Email de l'usuari
	user_password	Si	Contrasenya de l'usuari
	user_password_confirmation	Si	Verificació de la contrasenya de l'usuari
Success	{ "name": "Adria", "email": "adria1@gmail.com", "username": "adria1", "token": "johKcydXOzBVtiMMfrTLeOdu8dY", "authentication_token": "jFZtKtN6_HGemayvmiaU" }		
Errors	{ "message": "Validation Failed", "errors": [{ "resource": "User", "field": "email", "code": 103 }, { "resource": "User", "field": "username", "code": 103 }] }		

Sign in

Descripció	Et permet loguejar un usuari a l'API. El resultat és el token, el token d'autenticació i l'email de l'usuari. Cal guardar aquestes dades per a l'autenticació en altres crides a la API.
Path	{{url}}/v1/user/sign_in

<table>
<tr><td>Verb</td><td>POST</td></tr>
<tr><td>Paràmetres</td><td>
<table>
<tr><th>Paràmetre</th><th>Obligatori?</th><th>Explicació</th></tr>
<tr><td>user_email</td><td>Si</td><td>Email de l’usuari</td></tr>
<tr><td>user_password</td><td></td><td>Contrasenya de l’usuari</td></tr>
</table>
</td></tr>
<tr><td>Success</td><td>{
"email": "adria@gmail.com",
"authentication_token": "FxFtW41BF5C9Bz82fdsZ",
"token": "EjfqJyATKuaaZSLNXDAbNJ3Erc0"
}</td></tr>
<tr><td>Errors</td><td>{
"message": "Invalid login attempt",
"errors": [
{
"resource": "User",
"field": null,
"code": 104
}
]
}</td></tr>
</table>

Sign out

<table>
<tr><td>Descripció</td><td>Et permet desloguejar l’usuari connectat.</td></tr>
<tr><td>Path</td><td>{{url}}/v1/user/sign_out</td></tr>
<tr><td>Verb</td><td>DELETE</td></tr>
<tr><td>Paràmetres</td><td>
<table>
<tr><th>Paràmetre</th><th>Obligatori?</th><th>Explicació</th></tr>
<tr><td>user_email</td><td>Si</td><td>Email de l’usuari</td></tr>
<tr><td>user_token</td><td>Si</td><td>És el token de l’usuari actual</td></tr>
<tr><td>user_authentication_token</td><td>Si</td><td>És el token d’autenticació de</td></tr>
</table>
</td></tr>
</table>

			l'usuari actual	

Success	Quan s'ha desloguejat correctament, es torna a l'estat 204: sense contingut del protocol HTTP.
Errors	{ "message": "You need to sign in or sign up before continuing.", "errors": [{ "resource": null, "field": null, "code": 104 }] }

Lost password

Descripció	Permet recuperar el password
Path	{{url}}/v1/user/lost_password
Verb	POST
Paràmetres	Paràmetre | Obligatori? | Explicació user_email | Si | Email de l'usuari
Success	{ "message": "You will receive an email with instructions about how to reset your password in a few minutes." }
Errors	{ "message": "Invalid reset token.", "errors": [{ "resource": "User", "field": null, "code": 105 }] }
	{ "errors": [{ "email": "Email not found"

	}] }

	{
	"errors": [{
	"message": "Missing user parameter"
	}
	]
	}

Edit password

Descripció	Permet canviar la contrasenya. Aquesta acció requereix de l'acció lost_password per funcionar.
Path	{{url}}/v1/user/update_lost_password
Verb	PATCH
Paràmetres	<table><tr><th>Paràmetre</th><th>Obligatori?</th><th>Explicació</th></tr><tr><td>user_email</td><td>Si</td><td>Email de l'usuari</td></tr><tr><td>user_password</td><td>Si</td><td>Nova contrasenya</td></tr><tr><td>user_password_confirmation</td><td>Si</td><td>Nova contranseya</td></tr><tr><td>reset_password_token</td><td>Si</td><td>Token per resetejar el password.</td></tr></table>

Success	{
	"name": "Adria",
	"email": "adriagalin@gmail.com",
	"username": "adriagalin",
	"token": "EjfqJyATKuaaZSLNXDAbNJ3Erc0"
	}

Errors	{ "message": "This email was already confirmed, please try signing in.", "errors": [{ "resource": "User", "field": null, "code": 105

	}] }

	{
	"message": "This email does not belongs to the user.",
	"errors": [
	{
	"resource": "User",
	"field": null,
	"code": 105
	}
	]
	}

Inquiry Questions

Get inquiry question

Descripció	Mostra una informació en concret de la pàgina principal.
Path	{{url}}/v1/inquiry_questions/:inquiry_question_id
Verb	GET

Success	{
	"title": "Title",
	"preface": "Preface",
	"content": "Content",
	"bgcolor": "#FF0000",
	"image": {
	"image": {
	"url": null,
	"web_big2x": {
	"url": null
	},
	"web_big": {
	"url": null
	},
	"web_small2x": {
	"url": null
	},
	"web_small": {
	"url": null
	},
	"thumb": {
	"url": null

	},
	"small_thumb": {
	"url": null
	}
	}
	},
	"token": "qgjA5T-54pkNW2bO8JWb1Wfq7MM"
	}

Errors	{
	"message": "The Inquiry question do not exists",
	"errors": [
	{
	"resource": "InquiryQuestion",
	"field": null,
	"code": 105
	}
	]
	}

List Inquiry qüestions

Descripció	Llista tota la informació de la pàgina principal
Path	{{url}}/v1/inquiry_questions
Verb	GET

Success	[
	{
	"title": "Title",
	"preface": "Preface",
	"content": "Content",
	"bgcolor": "#FF0000",
	"image": {
	"image": {
	"url": null,
	"web_big2x": {
	"url": null
	},
	"web_big": {
	"url": null
	},
	"web_small2x": {
	"url": null
	},
	"web_small": {

	<pre>"url": null
	},
	"thumb": {
	"url": null
	},
	"small_thumb": {
	"url": null
	}
	}
	},
	"token": "qgjA5T-54pkNW2bO8JWb1Wfq7MM"
	}
	]</pre>

Slideshows

Get slideshow

Descripció	Mostra una presentació en concret.
Path	{{url}}/v1/user/slideshows/:slideshow_token?email=user_email &authentication_token=auth_token
Verb	GET
Paràmetres	<table><tr><th>Paràmetre</th><th>Obligatori?</th><th>Explicació</th></tr><tr><td>slideshow_token</td><td>Si</td><td>Identificador de la presentació</td></tr><tr><td>user_email</td><td>Si</td><td>Email de l'usuari</td></tr><tr><td>authentication_token</td><td>Si</td><td>És el token d'autenticació de l'usuari actual</td></tr></table>

Success	<pre>{
	"name": "Slideshow1",
	"slideshow_status": "processed",
	"attachment": {
	"attachment": {
	"url":
	"/uploads/slideshows/slideshow/attachment/5379b8306170695556000000/bb304d231c93cfb5a47f6fc8b85b8ae7.ppt"</pre>

	<pre> }
	    },
	    "token": "bjUsvEryu2o3q88IqhQv2kC16-4"
	}</pre>

Errors	<pre>{
	    "message": "The Slideshow do not exists",
	    "errors": [
	        {
	            "resource": "Slideshow",
	            "field": null,
	            "code": 105
	        }
	    ]
	}</pre>

Get slideshows

Descripció	Llista totes les presentacions de l'usuari loguejat.
Path	{{url}}/v1/user/slideshows?email=user_email&authentication_token=auth_token
Verb	GET
Paràmetres	<table><tr><th>Paràmetre</th><th>Obligatori?</th><th>Explicació</th></tr><tr><td>user_email</td><td>Si</td><td>Email de l'usuari</td></tr><tr><td>authentication_token</td><td>Si</td><td>És el token d'autenticació de l'usuari actual</td></tr></table>

Success	<pre>{
	    "name": "Slideshow1",
	    "slideshow_status": "processed",
	    "attachment": {
	        "attachment": {
	            "url":
	"/uploads/slideshows/slideshow/attachment/5379b8306170695556000000/
	bb304d231c93cfb5a47f6fc8b85b8ae7.ppt"
	        }
	    },
	    "token": "bjUsvEryu2o3q88IqhQv2kC16-4"
	}</pre>

Create slideshow

<table>
<tr><td>Descripció</td><td>Crea una presentació.</td></tr>
<tr><td>Path</td><td>{{url}}/v1/user/slideshows/?email=user_email&authentication_token=auth_token</td></tr>
<tr><td>Verb</td><td>POST</td></tr>
<tr><td>Paràmetres</td><td>
<table>
<tr><th>Paràmetre</th><th>Obligatori?</th><th>Explicació</th></tr>
<tr><td>user_email</td><td>Si</td><td>Email de l’usuari</td></tr>
<tr><td>authentication_token</td><td>Si</td><td>És el token d’autenticació de l’usuari actual</td></tr>
<tr><td>slideshow_name</td><td>Si</td><td>Nom de la presentació</td></tr>
<tr><td>slideshow_attachment</td><td>Si</td><td>Fitxer de les presentació</td></tr>
</table>
</td></tr>
<tr><td>Success</td><td><pre>{
  "name": "Slideshow3",
  "slideshow_status": "uploaded",
  "attachment": {
    "attachment": {
      "url":
"/uploads/slideshows/slideshow/attachment/538214556170693fb8010000/01c8eccafa8165cead073ee8962113e7.ppt"
    }
  },
  "token": "En6Rqwi--tlP0Op2Epm6VP7oBdo"
}</pre></td></tr>
<tr><td>Errors</td><td><pre>{
  "message": "Validation Failed",
  "errors": [
    {
      "resource": "Slideshow",
      "field": "attachment",
      "code": 101
    }
  ]
}</pre></td></tr>
</table>

Update slideshow

<table>
<tr><td>Descripció</td><td>Crea una presentació.</td></tr>
<tr><td>Path</td><td>{{url}}/v1/user/slideshows/:slideshow_token</td></tr>
<tr><td>Verb</td><td>PATCH</td></tr>
<tr><td>Paràmetres</td><td>
<table>
<tr><th>Paràmetre</th><th>Obligatori?</th><th>Explicació</th></tr>
<tr><td>user_email</td><td>Si</td><td>Email de l’usuari</td></tr>
<tr><td>authentication_token</td><td>Si</td><td>És el token d’autenticació de l’usuari actual</td></tr>
<tr><td>slideshow_token</td><td>Si</td><td>token de la presentació</td></tr>
<tr><td>name</td><td>No</td><td>Nom de la presentació</td></tr>
</table>
</td></tr>
<tr><td>Success</td><td><pre>{
  "name": "Slideshow2-test",
  "slideshow_status": "processed",
  "attachment": {
    "attachment": {
      "url":
"/uploads/slideshows/slideshow/attachment/5379cae46170693fb3000000/78
135cbedb50a2a18f6caebe8a2a9aba.ppt"
    }
  },
  "token": "O-YMlNxSosOfGwyH5o_FSLdk47c"
}</pre></td></tr>
<tr><td>Errors</td><td><pre>{
  "message": "The Slideshow do not exists",
  "errors": [
    {
      "resource": "Slideshow",
      "field": null,
      "code": 105
    }
  ]
}</pre></td></tr>
</table>

Destroy slideshow

<table>
<tr><td>Descripció</td><td>Crea una presentació.</td></tr>
<tr><td>Path</td><td>{{url}}/v1/user/slideshows/:slideshow_token</td></tr>
<tr><td>Verb</td><td>DELETE</td></tr>
<tr><td>Paràmetres</td><td>
<table>
<tr><th>Paràmetre</th><th>Obligatori?</th><th>Explicació</th></tr>
<tr><td>slideshow_token</td><td>Si</td><td>Token de la presentació</td></tr>
<tr><td>authentication_token</td><td>Si</td><td>És el token d’autenticació de l’usuari actual</td></tr>
<tr><td>user_email</td><td>Si</td><td>Email de l’usuari</td></tr>
</table>
</td></tr>
<tr><td>Success</td><td>Quan s’ha desloguejat correctament, es torna l’estat 204: Sense contingut del protocol HTTP.</td></tr>
<tr><td>Errors</td><td>{
"message": "The Slideshow do not exists",
"errors": [
{
"resource": "Slideshow",
"field": null,
"code": 105
}
]
}</td></tr>
</table>

Slides

Show slide

<table>
<tr><td>Descripció</td><td>Mostra un diapositiva en concret</td></tr>
<tr><td>Path</td><td>{{url}}/v1/user/slideshows/:slideshow_token/slide/:slide_token?email=user_email &authentication_token=auth_token</td></tr>
<tr><td>Verb</td><td>GET</td></tr>
<tr><td>Paràmetres</td><td>
<table>
<tr><th>Paràmetre</th><th>Obligatori?</th><th>Explicació</th></tr>
<tr><td>slideshow_token</td><td>Si</td><td>Identificador de la presentació</td></tr>
<tr><td>slide_token</td><td>Si</td><td>Identificador de l’slide</td></tr>
<tr><td>user_email</td><td>Si</td><td>Email de l’usuari</td></tr>
<tr><td>authentication_token</td><td>Si</td><td>És el token d’autenticació de l’usuari actual</td></tr>
</table>
</td></tr>
<tr><td>Success</td><td><pre>{
  "name": "Slide1",
  "attachment": {
    "attachment": {
      "url":
"/uploads/slides/slide/attachment/5379b830617069555600000 0/bb304d231c93cfb5a47f6fc8b85b8ae7.png"
    }
  },
  "token": "bjUsvEryu2o3q88IqhQv2kC16-4"
}</pre></td></tr>
<tr><td>Errors</td><td><pre>{
  "message": "The Slideshow do not exists",
  "errors": [
    {
      "resource": "Slideshow",
      "field": null,
      "code": 105
    }
  ]
}</pre></td></tr>
</table>

Destroy slide

<table>
<tr><td>Descripció</td><td>Crea una presentació.</td></tr>
<tr><td>Path</td><td>{{url}}/v1/user/slideshows/:slideshow_token/slides/:slide_token</td></tr>
<tr><td>Verb</td><td>DELETE</td></tr>
<tr><td>Paràmetres</td><td>
<table>
<tr><th>Paràmetre</th><th>Obligatori?</th><th>Explicació</th></tr>
<tr><td>slideshow_token</td><td>Si</td><td>Token de la presentació</td></tr>
<tr><td>slide_token</td><td>Si</td><td>Token de l’slide</td></tr>
<tr><td>authentication_token</td><td>Si</td><td>És el token d’autenticació de l’usuari actual</td></tr>
<tr><td>user_email</td><td>Si</td><td>Email de l’usuari</td></tr>
</table>
</td></tr>
<tr><td>Success</td><td>Quan s’ha desloguejat correctament, es torna l’estat 204: Sense contingut del protocol HTTP.</td></tr>
<tr><td>Errors</td><td>{
"message": "The slide do not exists",
"errors": [
{
"resource": "Slide",
"field": null,
"code": 105
}
]
}</td></tr>
</table>

6.4. Arquitectura de TransformSlideshowWorker.

6.4.1. Transform slideshow worker code.

```
require 'redis'
require 'sidekiq'
require 'sidekiq-status'
require 'docsplit'
require 'fileutils'
require 'mongoid'
require 'carrierwave/mongoid'
require 'simple_enum/mongoid'

$:.unshift File.dirname(__FILE__)
Mongoid.load!("./config/mongoid.yml", ENV['APP_ENV'].to_s)

require './app/models/concerns/tokenable'
require './app/models/concerns/apilable'
require './app/models/concerns/authenticable'
require './app/models/slideshow'
require './app/models/slide'
require './app/models/photo'
require './app/models/guest'
require './app/models/user'
require './app/models/room'

redis_config = YAML.load_file(Dir.pwd + '/config/redis.yml')[ENV['APP_ENV']]

$redis = Redis.new(url: "redis://#{redis_config['host']}:#{redis_config['port']}/2")

Sidekiq.configure_client do |config|
  config.redis = {
    namespace: "YourTalks",
    url: "redis://#{redis_config['host']}:#{redis_config['port']}/1",
    size: 1
  }
  config.client_middleware do |chain|
    chain.add Sidekiq::Status::ClientMiddleware
  end
end

Sidekiq.configure_server do |config|
  config.redis = {
    namespace: "YourTalks",
    url: "redis://#{redis_config['host']}:#{redis_config['port']}/1",
    size: 25
  }
  config.server_middleware do |chain|
```

```
    chain.add Sidekiq::Status::ServerMiddleware, expiration: 30.minutes # default
  end
end

class TransformSlideshowWorker
  include Sidekiq::Worker
  include Sidekiq::Status::Worker
  sidekiq_options retry: 2 , backtrace: true, queue: "top_priority_slideshow_processing",
failures: true
  STEPS = 4

  def perform(slideshow_token)
    logger.info "[PROCESSING]: I'm working on -> #{slideshow_token}"
    at(1, STEPS, "Initiate process...")
    slideshow = Slideshow.find_by_token(slideshow_token)
    slideshow.update_status(Slideshow.status_processing)
    output_path = Dir.pwd +
"/uploads/tmp/slideshow_converted/#{slideshow.token}/slides/"
    at(2, STEPS, "Starting transform slideshow..")
    convert_slideshow(slideshow, output_path)
    at(3, STEPS, "Saving slides...")
    saving_slides(slideshow, output_path)
    slideshow.update_status(Slideshow.status_processed)
    slideshow.publish_slideshow_processed
    at(4, STEPS, "Finishing process...")
    # Set process
    # # puts send notification
  end

  private
  def convert_slideshow(slideshow, output_path)
    file = Docsplit.extract_images(slideshow.attachment.path, :size => '500x', :format =>
[:png], :output => output_path)
    File.delete(file[0])
  end

def saving_slides(slideshow, folder_path)
    get_png_files_paths(folder_path) {|f| slideshow.create_slide_worker(f, f)}
    #Remove tmp files
    FileUtils.rm_rf folder_path.split("/").slice!(0...-1).join("/")
  end

  def get_png_files_paths(folder_path)
    Dir.glob(folder_path + "/*.png").each do |f|
      yield f
    end
  end

end
```

6.4.2. Arquitectura de Transform slideshow worker.

Normalment els workers estan dissenyats per fer una acció en concret. El que es vol és processar una presentació powerpoint o un pdf. Si es segueix el cicle normal d'una petició i resposta del protocol HTTP ens donava un timeout. El protocol HTTP no està pensat per processos llargs. Per tant, s'ha implementat un sistema de processat en segon pla o worker.

Per realitzar aquest procés en segon pla s'ha utilitzat Sidekiq. Sidekiq és un framework pel processament en segon pla amb totes les funcions de Ruby. El seu objectiu és que sigui fàcil d'integrar amb qualsevol aplicació Rails i tenir un rendiment més eficient que altres solucions, com Resque. Sidekiq[83] utilitza Redis, com a base de dades, per gestionar les cues de treball.

En Sidekiq, pots configurar els workers per realitzar diferents tasques, i després enviar les tasques a una cua. D'aquesta manera quan un worker no està ocupat, truca a la cua per agafa una nova tasca a realitzar.

Per poder utilitzar Sidekiq, es necessita tenir instal·lada la gem sidekiq i les seves dependències (Redis, etc). A part, s'ha inclòs les dependències necessàries per poder realitzar la tasca pensada per el worker, com mongoid, per connectar-se a la base de dades, docsplit per al processament de les imatges i fitxers, entre altres.

Seguidament s'ha inclòs la creació de la connexió a la base de dades, els models i la configuració de Sidekiq necessària perquè el worker pugui fer la seva feina, incloent el mòdul de Sidekiq::Worker que hereta la funcionalitat de Sidekiq.

El mètode perfom és el punt d'entrada del worker, aquí es passa slideshow_token que es vol processar. Desrpés es busca a la base de dades, es canvia l'estat de l'slideshow a processin i es comença a convertir amb la gem docsplit, que s'encarrega de convertir la presentació en imatges. Un cop acabat el processament de la presentació, es crea una fila per slide i es relaciona amb la seva slideshow. Per finalitzar, s'actualitza l'estat del slideshow, s'envia una notificació per email, a través d'un altre worker i s'envia una notificació al servidor de temps real.

[83] http://sidekiq.org/

Per executar aquest worker desde Rails es pot fer de dues maneres. Si el worker està en la mateixa aplicació de Rails:

```
TransformSlideshowWorker.perform_async(slideshow_token)
```

Si el worker no està a la mateixa aplicació, sinó que està en servidors separats, per a un millor rendiment de l'aplicació, s'utilitza:

```
Sidekiq::Client.push( 'queue' => 'top_priority_slideshow_processing', 'class' =>
'TransformSlideshowWorker', 'args' => [self.token])
```

Aquí s'encua la tasca el sistema gestor de cues.

Quan s'executa el servidor en desenvolupament, també s'ha d'executar Redis i el procés de Sidekiq. També es pot comprovar l'activitat de Sidekiq per veure si tot està funcionant correctament. Només cal posar al fitxer routes.rb de l'aplicació el següent:

```
mount Sidekiq::Web, at:'/sidekiq'
```

On es pot veure el següent:

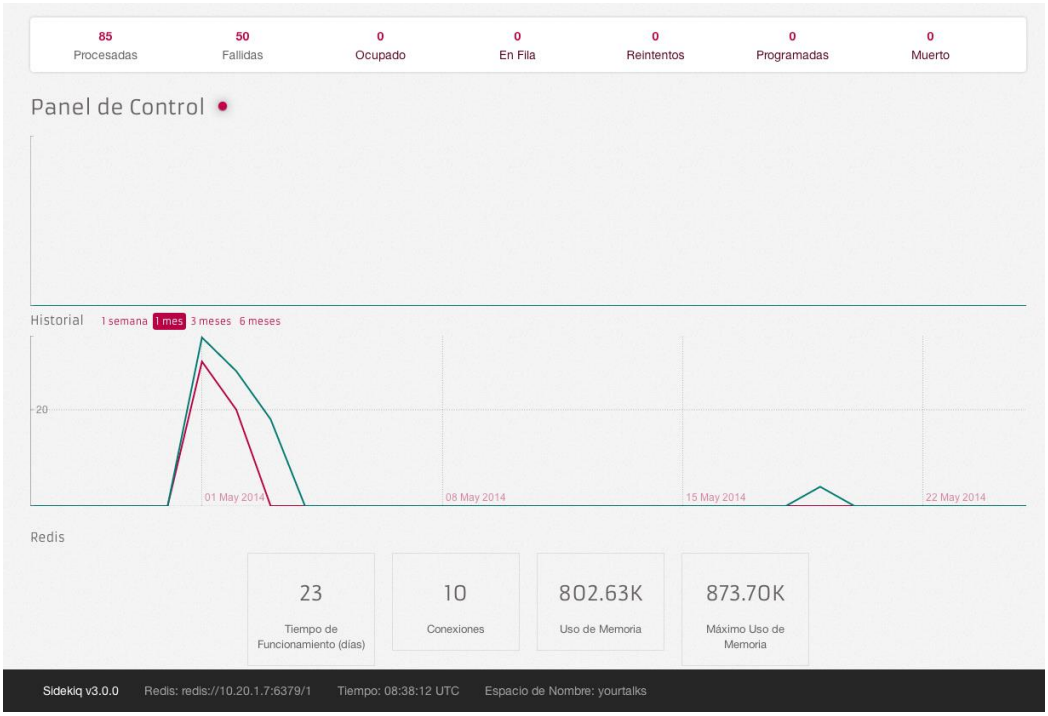


Figura 15: Sidekiq 1. Font: el·laboració pròpia

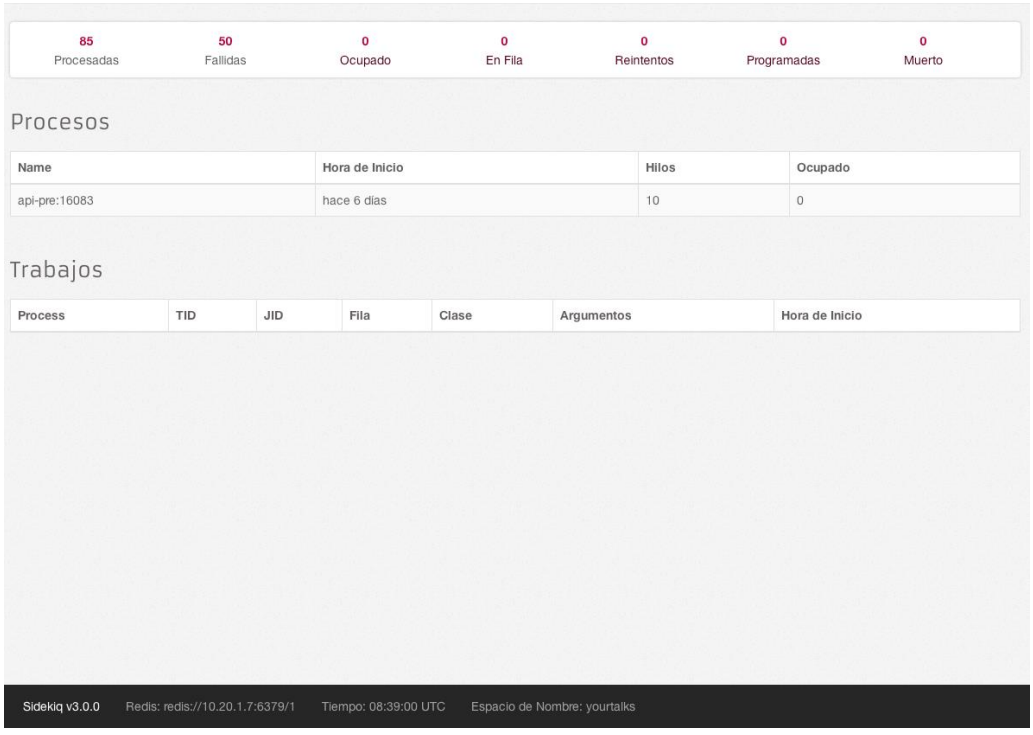


Figura 16: Sidekiq 2. Font: el•laboració pròpia

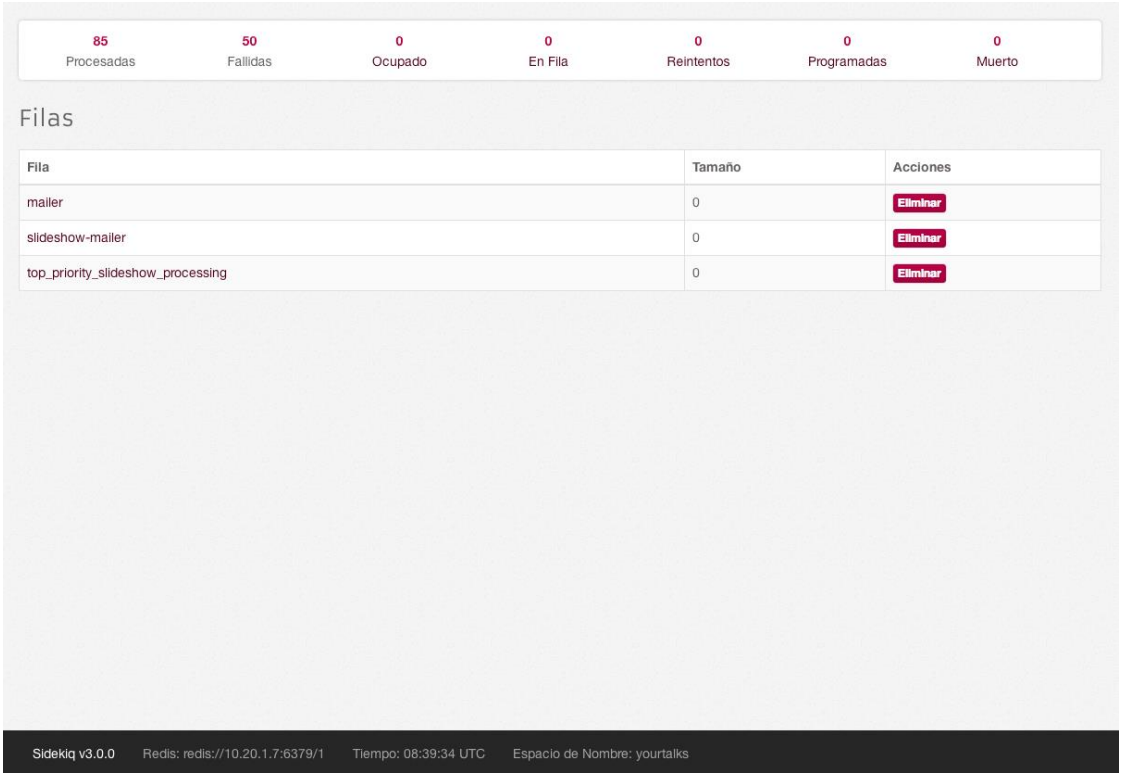


Figura 17: Sidekiq 3. Font: el•laboració pròpia

Aquí es mostra tot el procés del sistema de processament en segon plà:

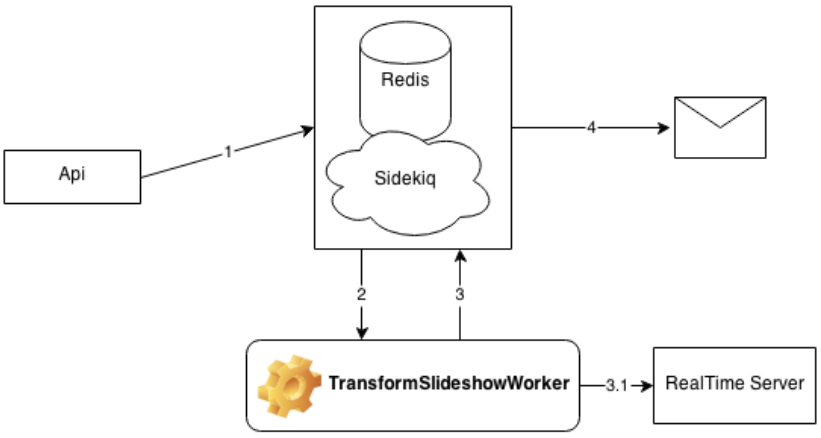


Figura 18: Worker en funcionament. Font: el·laboració pròpia

7. Punts importants sobre la implementació.

Una de les àrees claus en la producció d'aplicacions webs, és la implementació. Exactament, en la forma d'implementar. Aquesta tasca, que en principi es te poc en compte, afegeix un gran valor en el projecte ja que un procés ben elaborat d'implementació, sens dubte ajuda a reduir les despeses generals, com ara reduir el temps en que el producte es posa a producció.

Per a la implementació i desplegament de les diferents parts de l'aplicació s'ha utilitzat servidors virtualitzats en una màquina física, i Capistrano per executar múltiples scripts i automatitzar les tasques de desplegament.

S'explicarà una de les receptes per el desplegament de la infraestructura, totes mantenen la mateixa estructura. Aquí es mostra la recepta per desplegar l'API REST Puigmal. S'ha comentat cada punt, per a una millor entesa.

```
# S'importen les opcions de capistrano
require "rvm/capistrano"
# Es defineix l'entorn ruby
set :rvm_ruby_string, '2.0.0@yt-puigmal'
set :rvm_type, :user
# S'importen les tasques per gestionar les gemes
require 'bundler/capistrano'

# es defineixen els hosts per el desplegament
HOSTS_PRE="10.20.1.10"
HOSTS_WWW="10.20.1.20"

# es s'assignen les variables segons l'entorn escollit
begin
  case ENV['environment']
    when 'pre' then
      # entorn de rails
      set  :rails_env, 'pre'
      # branca de git on es troba el projecte
      set  :branch,    ENV.fetch('branch', 'pre')
      # nom de l'aplicació
      set :application, "api-pre.yourtalks.com"
      role :web,       *HOSTS_PRE
      role :app,       *HOSTS_PRE
```

```
    role :db,        *HOSTS_PRE, :primary => true

  when 'www' then
    set  :rails_env, 'production'
    set  :branch,    ENV.fetch('branch', 'master')
    set :application, "api.yourtalks.com"
    role :web,       *HOSTS_WWW
    role :app,       *HOSTS_WWW
    role :db,        *HOSTS_WWW, :primary => true

  else
    raise capistrano::Error, "Your environment is invalid, please use [pre|www]"
 end
rescue
 raise capistrano::Error, "You need to define environment variable: environment=[pre|www]\n\nFor example:\n  environment=pre cap deploy"
end

# ruta on s'allotjarà l'aplicació
set :deploy_to, "/home/web/apps/#{application}"
# usuari que executarà l'aplicació
set :user, "web"
set :runner, "web"
# repositori on es troba l'aplicació
set :repository, git@bitbucket.org:YourTalks/yt-puigmal.git
set :git_enable_submodules, true
set :deploy_via, :remote_cache
set :copy_exclude, [".git", ".gitignore"]
set :scm, :git
set :use_sudo, false
set :cache_path, -> { "#{fetch :shared_path}/system/cache" }

ssh_options[:forward_agent] = true
default_run_options[:pty] = true

# Configuració del servidor que executa rails
#set :unicorn_binary, "/usr/bin/unicorn_rails"
set :unicorn_binary, "bundle exec unicorn_rails"
set :unicorn_config, "#{current_path}/config/unicorn.rb"
set :unicorn_pid,    "#{current_path}/tmp/pids/unicorn.pid"

# Definició de tasques per al desplagament
namespace :deploy do
 task :start, :roles => :app, :except => { :no_release => true } do
   run "cd #{current_path} && #{try_sudo} #{unicorn_binary} -c #{unicorn_config} -E #{rails_env} -D"
 end
 task :stop, :roles => :app, :except => { :no_release => true } do
   run "#{try_sudo} kill `cat #{unicorn_pid}`"
```

```
  end
  task :graceful_stop, :roles => :app, :except => { :no_release => true } do
    run "#{try_sudo} kill -s QUIT `cat #{unicorn_pid}`"
  end
  task :reload, :roles => :app, :except => { :no_release => true } do
    run "#{try_sudo} kill -s USR2 `cat #{unicorn_pid}`"
  end
  task :RESTart, :roles => :app, :except => { :no_release => true } do
    stop
    start
  end
end

# Tasques per gestionar el workers
namespace :foreman do
  desc "Export the Procfile to Ubuntu's upstart scripts"
  task :export, :roles => :app do
    run "cd #{release_path} && rvmsudo bundle exec foreman export upstart /etc/init -f
./Procfile.#{rails_env} -a #{application}-workers -u #{runner} -l /var/log/#{application}-
workers"
  end

  desc "Start the application services"
  task :start, :roles => :app do
    sudo "start #{application}-workers"
  end

  desc "Stop the application services"
  task :stop, :roles => :app do
    sudo "stop #{application}-workers"
  end
  desc "RESTart the application services"
  task :RESTart, :roles => :app do
    stop
    start
  end
end

after "deploy:update", "deploy:cleanup"
after "deploy:update", "foreman:export"
after "deploy:update", "foreman:RESTart"
```

Perquè una aplicació Rails s'executi cal tenir un servidor web, en aquest cas s'ha utilitzat Unicorn. Unicorn és un servidor HTTP escrit en Ruby. Està dissenyat per servir els clients ràpidament en baixa latència, aprofitant el màxim la connexió de banda ampla i aprofitant les característiques del kernel d'Unix/Linux.

S'explicarà una de les configuracions utilitzades, totes mantenen la mateixa estructura. Aquí es mostra la recepta per desplegar l'API REST Puigmal. S'ha comentat cada punt, per a una millor entesa.

```
# segons l'entorn es defineix el socket per on emetrà unicorn
if ENV['RAILS_ENV'] and ENV['RAILS_ENV'] == 'production'
  rails_env = 'api'
  listen "/tmp/.unicorn-www.sock"
else
  rails_env = 'api-pre'
  listen "/tmp/.unicorn-pre.sock"
end
# el número de processos actius.
worker_processes 4
# directori on es troba l'aplicació
working_directory "/home/web/apps/#{rails_env}.YourTalks.com/current/"
timeout 200
# directori on es troba el procés de unicorn
pid "/home/web/apps/#{rails_env}.YourTalks.com/current/tmp/pids/unicorn.pid"
# ruta on es troba el log d'errors de unicorn
stderr_path "/home/web/apps/#{rails_env}.YourTalks.com/current/log/api-pre.stderr.log"
# directori on es troba el log de unicorn
stdout_path "/home/web/apps/#{rails_env}.YourTalks.com/current/log/api-pre.stdout.log"

# hack, per gestionar les connexions cap a la base de dades, quan el procés de unicorn es reinicia.
preload_app true
GC.respond_to?(:copy_on_write_friendly=) and
  GC.copy_on_write_friendly = true

before_fork do |server, worker|
  defined?(ActiveRecord::Base) and
    ActiveRecord::Base.connection.disconnect!
end

after_fork do |server, worker|
  defined?(ActiveRecord::Base) and
    ActiveRecord::Base.establish_connection
end
```

8. Objectius aconseguits.

En línies generals s'han complert els objectius principals establerts inicialment, que era tenir la base robusta de l'aplicació per poder desenvolupar totes les millores i versions futures amb agilitat i rapidesa.

Els objectius tant a nivell funcional com els de l'ús de la tecnologies s'han dut a terme satisfactòriament resolent els diferents problemes trobats durant la implementació del projecte. A mesura que s'ha anat desenvolupant el projecte s'ha anat anotant noves característiques i millores, per un projecte que està en constant desenvolupament, veure versions futures.

S'ha implementat la base de la plataforma YourTalks, amb consistència, millorant el nostre coneixement dels llenguatges de programació, permetent-nos crear una plataforma més robusta i fiable.

Un altre objectiu a destacar i una gran dificultat ha estat integrar totes les parts del projecte perquè funcionessin correctament, ja que cada part de l'aplicació requeria configuració i coneixements diferents.

S'ha pogut crear tot el core de l'API REST per a una primera versió en un entorn aïllat.

El sistema de worker s'ha completat al 100% permetent explotar tota la funcionalitat i escalabilitat.

Codi font organitzat i documentat per permetre un fàcil manteniment i ampliació de funcionalitats.

S'ha respectat la metodologia àgil, permetent-nos tenir flexibilitat, i temps per el desenvolupament dels objectius plantejats.

9. Estudi econòmic.

Per a realitzar una bona estimació dels costos, s'ha dividit en dos grups, d'una banda els costos generats pels recursos humans i d'altra banda els costos que provenen dels recursos materials. A tots i cadascun dels costos inclosos en aquest apartat se'ls ha afegit l'impost sobre el valor afegit (IVA).

En primer lloc ens centrarem en els costos procedents dels recursos humans i per això farem ús de JIRA. S'ha realitzat a partir de les tasques introduïdes a JIRA i una aproximació de les hores necessàries per desenvolupar el projecte fins a la seva entrega inicial. Aquí es descarta els costos futurs, en properes versions.

Personal	Hores de treball (aprox)	Preu/hora	Total	Desviació hores aprox	Preu total
Anàlisis de requeriments					
Gestor del projecte	20 h	35 €/h	700 €	20 %	840 €
Desenvolupador 1	450 h	30 €/h	13500 €	20 %	16200 €
Desenvolupador 2	350 h	30 €/h	10500 €	20 %	12600 €
Administrador de sistemes	100 h	35 €/h	3500 €	20 %	4200 €
Total del projecte	920 h		28200 €	20 %	33840 €

Taula 8: Relació costs econòmics. Font: el·labo

Com es pot observar a la taula anterior, el cost total procedent dels recursos humans és de 33840 €. També es pot veure el temps dedicat al projecte de cadascuna de les persones, el salari que cobra per hora i el salari total per a la seva participació en el projecte.

Per estimar els costos procedents dels recursos materials tindrem en compte tant la despesa produïda pel programari com pel maquinari.

Material	Cost
Ordinador portàtil (Macbook Pro)	180 €
Ordinador portàtil (Macbook Air)	180 €
Material fungible	50 €
Impressora	100 €
Total	510 €

Taula 9: Costs material. Font: el•laboració pròpia

Com es pot veure en la taula anterior, els costos generats pels recursos materials ascendeixen a 510 €. Per tant, tenint en compte la despesa generada tan pels recursos materials com pels humans, el cost total estimat del projecte és de 34350 €.

9.1. Moneteització

S'ha fet un petit brainstorming de les possibles sortides de negoci de l'aplicació. Algunes de les quals són:

- Mostrar anuncis via Adwords o plataformes d'afiliats.
- Freemium:
    - Oferir un servei gratuït fins a un total de X sales al mes.
    - Oferir només alguns mòduls gratuïts i altres de pagament.
    - Sales amb límit de gent.
- Premium:
    - Accés a totes les característiques de la plataforma.

10. Conclusions.

L'objectiu principal d'aquest projecte ha estat la realització de l'estructura base i l'investigació de les tecnologies per desenvolupar YourTalks que consisteix en crear una plataforma web per a la realització de conferències/classes o qualsevol altre tipus de xerrada en temps real.

La principal conclusió en el desenvolupament del treball ha estat la presa de decisions per escollir les eines a utilitza ja que existeixen moltes tecnologies per realitzar les mateixes accions, però cada una amb les seves particularitats.

La dificultat més gran que ens hem trobat ha estat l'aprenentantage de les noves tecnologies utilitzades, com per exemple MariontteJS o Puppet, amb el qual no havíem treballat mai. Aquest fet va provocar que les estimacions inicials no s'aproximesin a la realitat del desenvolupament.

Des de l'inici s'ha portat un control de l'estat del projecte a través del software JIRA i utilitzant les metodologies àgils, en concret, Kanban com una de les seves aplicacions. S'ha intentat seguir la planificació dels sprints, i segons l'evolució de les tasques s'han anat redefinint el següents sprints degut a les dificultats trobades. Aquest ha estat el punt clau que ens ha permès dur a terme amb èxit el projecte i complir amb els objectius establerts en l'apartat Abast. En l'anàlisi de requeriments s'ha intentat descriure amb el màxim de detall tots els aspectes que havia de cobrir l'aplicació per evitar que durant la implementació apareguessin masses canvis, tot i que, com és habitual en Kanban, els requisits van canviant segons les necessitas de cada moment.

Un dels grans reptes del projecte ha estat les tasques d'integració dels diferents components utilitzats en el projecte d'una forma escalada i modular. D'una banda els servidors, el frontend, l'API, el real time, els workers. Un dels principals objectius de la integració ha estat dissenyar l'arquitectura modular de la plataforma per a realitzar futures ampliacions.

Durant la realització del projecte, s'ha m'han plantejat diferents reptes, havia de dissenyar una arquitectura de sistemes que fos escalable i tolerant a falles, per suportar totes les necessitats de la plataforma. Una API REST ràpida i fàcil d'integrar amb els altres

components de la plataforma i un sistema de "workers" que s'encarreguessin de processar les presentacions en segon pla.

Amb les tecnologies que he investigat, m'he trobat amb el problema de quin llenguatge escollir i quina era la millor metodologia de treball personal, havia d'escollir unes tecnologies que estesin ben documentades i utilitzades per altre gent. Necessitava utilitzar tecnologies que em permetessin desenvolupar àgilment, amb rapidesa i amb molta documentació per solucionar qualsevol problema de la forma més ràpida possible i he complert l'objectiu marcat amb escreix.

Pel que fa Puppet he descobert que és una eina molt potent que permet diferents configuracions i dissenys d'automatització adaptables a qualsevol arquitectura de sistemes. He vist que m'ajudarà a gestionar i agilitzar la infrastructura de la plataforma, quan aquesta creixi. L'altre punt fort és que utilitzant Vagrant podré crear entorn de test de forma ràpida, simple i fàcil. Si en un futur entren més persones a treballar en el projecte, els podré posar un entorn de test en 10 minuts. També he vist que introduir Vagrant en el flux de treball, pot agitlitzar el desenvolupament de YourTalks.

En aquest projecte s'ha plantejat la base tecnològica, la metodologia de treball, i la base de la plataforma del que haurà de ser YourTalks en un futur.

Un dels punts que he trobat a faltar, per falta de temps, ha estat la recerca d'informació de l'escalat automàtic de les màquines segons unes mètriques predefinides.

En general, s'han complert els objectius marcats a l'inici del mateix. També destacar, com a aspecte molt important, l'aprenantatge de Puppet, Ruby on Rails, i el dissney d'arquitectura de sistemes. M'ha aportat nous coneixements que em seran molt vàlids en el món laboral i en el futur de YourTalks, la millor plataforma de conferències del món, en un futur.

Com ha conclusió final podem dir que s'ha implementat l'estructura base per crear una plataforma que permeti oferir una experiència el més semblant possible a l'assistència física a una conferència.

Per això, podem concloure que s'han assolit amb èxit tots i cadascun dels objectius establerts.

11. Referències.

Allamaraju, S. (2010). *RESTful Web Services Cookbook.* Sebastopol: O'Really.

Ángel Ramos, A., García-Morán, J. P., Picouto, F., Grijalba, J., Mayan, M., García, Á., et al. (2008). *Instala, administra, securiza y virtualiza entornos linux.* Madria: Ra-Ma.

Arundel, J. (2013). *Puppet 3 Cookbook.* Birmingham: Pack Publishing.

Backbone.JS. (sense data). Consultat el 2014, a http://backbonejs.org/

Bash. (sense data). Consultat el 2014, a http://www.gnu.org/software/bash/

Burns, A., & Copeland, T. (2012). *Deploying Rails. Autmoate, Deploy, Scale, Maintain, and Sleep at Night.* Dallas: The Pragmatic Bookshelf.

Campi, N., & Bauer, K. (2009). *Administración de sistemas LINUX/UNIX. Automatización de tareas y procesos.* Madrid: Anaya Multimedia.

Capistrano. (sense data). Consultat el 2014, a http://capistranorb.com/

Cluster de alta disponibilidad. (sense data). Consultat el 2014, a http://lobobinario.blogspot.com.es/2011/09/cluster-de-alta-disponibilidad-balanceo.html

Denoe - Estructura cloud computing. (sense data). Consultat el 2014, a http://www.denoe.es/test/wp-content/uploads/estructura-cloud-computing.png

Dev2Ops. (sense data). Consultat el 2014, a http://dev2ops.org/2010/11/devops-is-not-a-technology-problem-devops-is-a-business-problem/

Funio - Arquitectura de alta disponibilidad. (sense data). Consultat el 2014, a https://es.funio.com/arquitectura-alta-disponibilidad

Github. (sense data). Consultat el 2014, a https://github.com/linkedin/REST.li/wiki/REST.li-User-Guide

Hashimoto, M. (2013). *Vagrant. Up and Running.* Sebastopol: O'Really.

Linuxaria. (sense data). Consultat el 2014, a http://cdn.linuxaria.com/wp-content/uploads/2013/04/Screen-PVE-Datacenter-Summary.png

Loope, J. (2011). *Managing Infrastructure with Puppet.* Sebastopol: O'Really.

Marionette. (sense data). Consultat el 2014, a http://marionettejs.com/

Nginx. (sense data). Consultat el 2014, a http://nginx.com/

Node.JS. (sense data). Consultat el 2014, a http://nodejs.org/

OVH. (sense data). Consultat el 2014, a https://www.ovh.es/servidores_dedicados/ip_failover.xml

PrideParrot. (sense data). Consultat el 2014, a http://www.prideparrot.com/blog/archive/2012/3/creating_a_REST_service_using_asp_net_web_API

Ruby. (sense data). Consultat el 2014, a https://www.ruby-lang.org/es

Ruby On Rails Website. (sense data). Consultat el 2014, a http:/www.rubyonrails.org

The Javascript Task Runner. (sense data). Consultat el 2014, a http://gruntjs.com/

Turnbull, J. (2007). *Pulling String with Puppet.* New York: Apress.

Turnbull, J., & McCune, J. (2013). *Pro Puppet.* Apress.

Turnbull, J., Lieverdink, P., & Matotek, D. (2009). *Pro Linux System Administration.* New York: Apress.

Unicorn. (sense data). Consultat el 2014, a http://unicorn.bogomips.org/

What is Puppet? (sense data). Consultat el 2014, a http://puppetlabs.com/puppet/what-is-puppet

Wikipedia - Alta disponibilidad. (sense data). Consultat el 2014, a https://es.wikipedia.org/wiki/Alta_disponibilidad

Wikipedia. (sense data). *Javascript.* Consultat el 2014, a http://es.wikipedia.org/wiki/JavaScript