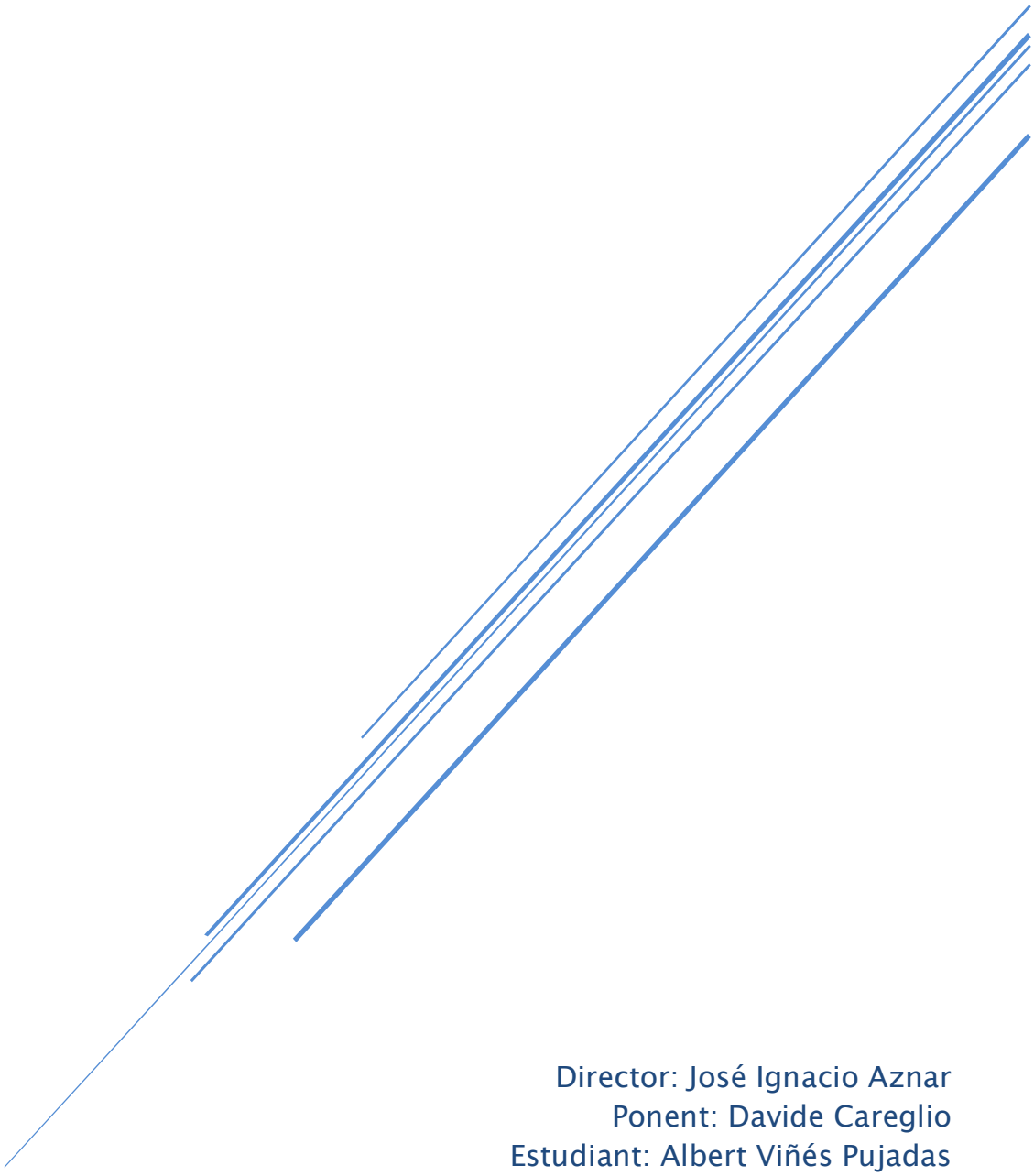
Network service orchestration for SDN based Data Centers: an OpenStack approach



Director: José Ignacio Aznar
Ponent: Davide Careglio
Estudiant: Albert Viñés Pujadas

Resum

L'entorn dels centres de dades és sota un dels majors canvis tecnològics dels últims anys degut al Cloud Computing. Fins fa poc, la infraestructura del centre de dades no oferia cap altra solució a l'increment de la demanda dels usuaris que l'adquisició de més recursos i per tant l'increment dels costos. No obstant, les plataformes actuals per gestionar els centres de dades o *clouds* manquen d'usabilitat i llibertat de gestió dels recursos. En aquest TFG es mostra com és possible crear una interfície gràfica de gestió de *clouds* enfocada als seus usuaris i adaptada als futurs centres de dades en l'àmbit del projecte europeu FP7 COSIGN.

Resumen

El entorno de los *Data Center* se encuentra bajo uno de los mayores cambios tecnológicos de los últimos años debido al *Cloud Computing*. Hasta hace poco, la infraestructura de los *Data Center* no ofrecía otra solución a el incremento de demanda de los usuarios que la adquisición de más recursos y por lo tanto el aumento de los costes. No obstante, las plataformas de gestión *Cloud* actuales carecen de usabilidad y libertad de gestión de los recursos. En este TFG se muestra como es posible crear una interfaz gráfica de gestión de *clouds* enfocada a sus usuarios y adaptada a los futuros *Data Centers* en el ámbito del proyecto europeo FP7 COSIGN.

Abstract

The Data Center environment is under one the major technological changes of the last years due to the appearance of Cloud Computing. Until very recently, the infrastructure of Data Centers could just satisfy the increase of user's data growth by the addition of more IT resources and therefore the rise of the CAPEX costs. However, the current Cloud Management platforms lack of usability and freedom for managing the resources. This thesis shows how it's possible to create a user friendly Cloud Management user interface ready for the future Data Center in the European FP7 COSIGN project.

1. Introducció i motivació

El TFG s'ha desenvolupat dins el projecte europeu FP7 COSIGN, que té com a objectiu dissenyar i implementar una nova arquitectura per els futurs centres de dades mitjançant orquestració dels recursos per software i commutadors òptics d'alta capacitat. Aquesta nova arquitectura permetrà nous casos d'ús i solucionar les mancances i limitacions de l'arquitectura actual.

Actualment, les tecnologies i infraestructures utilitzades als centres de dades es veuen limitades per l'increment d'ús d'Internet per part del públic global. Tal increment només pot ser solucionat mitjançant l'adquisició de més recursos físics, fet que comporta un augment de la dificultat de gestió del centre de dades. Davant d'aquest repte, SDN (de l'anglès Software Defined Network) ha crescut com a solució per l'orquestració i gestió de la xarxa. Tanmateix, no existeix cap solució comercial usable y complerta encara preparada per aquest increment de demanada i les eines comercials de gestió Cloud existents són poc usables o bé estan limitades en quan funcionalitats de gestió dels recursos.

L'objectiu d'aquest TFG és desenvolupar una interfície gràfica per a la gestió de centres de dades d'acord amb el cas d'ús Virtual Data Center del projecte europeu FP7 COSIGN. A més, la interfície ha de ser segura, fiable, robusta i enfocada a l'usuari. El cas d'ús Virtual Data Center consisteix en arribar a virtualitzar tot el centre de dades, no només com fins ara que s'ofereix virtualització de servidors, sinó també de la xarxa en funció de les necessitats de l'usuari.

Per tal d'assolir els objectius s'ha utilitzat OpenStack, eina de gestió de Clouds *open-source,* ampliant la interfície web que ofereix el seu mòdul Horizon de gestió del centre de dades i així complir els requeriments del cas d'ús. La programació de la interfície s'ha realitzat amb el estàndard HTML5, CSS3, Javascript i jQuery mentre que la programació del servidor s'ha dut a terme amb Django, *framework de Python*. El resultat final ha sigut validat satisfactòriament conforme els requeriments mitjançant diferents experiments de funcionalitat i seguretat.

En aquesta memòria hi podrem trobar l'escenari i context en el que s'ha desenvolupat el TFG, l'anàlisi de l'estat de l'art, els objectius, la planificació, el pressupost, la sostenibilitat, el disseny, la implementació, la validació posterior i les regulacions aplicables. Per últim, es conclou amb les conclusions i el futur treball.

2. Escenari i context

L'escenari i context d'aquest Treball Final de Grau està conformat per les xarxes òptiques dins de centres de dades on el control es realitza en base a tecnologies Software Defined Network (SDN) i l'orquestració de recursos mitjançant OpenStack. Hi ha quatre elements fonamentals en aquest escenari i context:

- Cloud Computing.
- Software Defined Network.
- Orquestració mitjançant OpenStack.
- Xarxes òptiques de centres de dades amb commutadors de nova generació.

2.1 Cloud Computing

El terme *Cloud Computing* és bastant ampli i algunes vegades pot causar confusió degut a l'increment d'aquest tipus de tecnologia. És per això que abans de descriure el context i escenari veurem les característiques principals de Cloud Computing segons la National Institute of Standards and Technology [1]:

- Servei disponible de forma automàtica i sota demanda: Un usuari pot començar a utilitzar un recurs *cloud* (emmagatzematge, màquina virtual, etc...) sense ninguna intervenció per part de l'operador de l'empresa que proporciona el servei.

- Virtualització dels recursos: la virtualització és un element fonamental del *Cloud* Computing ja que garanteix l'entrega de recursos informàtics compartits al mateix servidor, reduint així els costos de la infraestructura i augmentant la flexibilitat i fiabilitat del hardware existent.

- Accés a través de la xarxa: Els recursos estan disponibles a través de la xarxa (Internet o algun altre tipus de xarxa pública o privada), mitjançant mecanismes estandarditzats que permetin l'ús de diversos clients, des d'ordinadors a telèfons mòbils.

- Els recursos s'agrupen en pools en un model de tinència múltiple (*multi-tenancy*): els recursos son compartits en general per múltiples clients, que poden disposar d'ells sota demanda i als que se'ls hi ha de garantir aïllament i seguretat, malgrat que comparteixin els recursos.

- Elasticitat: Es un nou concepte relacionat amb el *cloud* i que porta al extrem el concepte d'escalabilitat, ja que els recursos que pot necessitar l'usuari del *cloud* poden créixer i decréixer de forma ràpida en funció de les seves necessitats i demanda dels usuaris finals.

- Pagament per ús: la utilització real que fa els recursos del usuari del *cloud* i no una tarifa per trams es el que defineix el pagament dels serveis de *cloud*. En funció del pagament, s'obté una disponibilitat dels recursos i característiques del servei diferents.

Els serveis que s'ofereixin als usuaris i compleixin aquests requisits es poden considerar pròpiament *Cloud Computing* i donat que son requisits amplis, normalment es classifiquen en funció del tipus de servei que ofereixin en tres capes: IaaS, SaaS i PaaS.

- *Infraestructure as a Service* (IaaS): S'ofereix principalment emmagatzematge i capacitats de computació (màquines virtuals) com a servei al núvol. Els administradors de sistemes de tot el món es poden plantejar muntar un servei a una màquina física, una màquina virtual o al cloud, oferint aquesta última opcions molt interessants per desplegaments de demanda variable. Els serveis de cloud IaaS més coneguts son els de Amazon Web Services, RackSpace Cloud o Joyent.

- *Platform as a Service*: Aplicació completa per al desenvolupament i desplegament de software. Els desenvolupadors de software veuen facilitades les tasques de proves i desplegament si opten per una plataforma de desenvolupament al núvol. Algunes de les opcions més conegudes de PaaS són Google App Engine, Windows Azure, Red Hat OpenShift o Heroku.

- *Software as a Service*: Aplicació completa que s'ofereix com a servei al núvol. Es l'exemple més conegut i utilitzat de cloud computing en el que qualsevol usuari utilitza el software de determinada empresa, com per exemple els serveis de Google, Microsoft Office 365, Dropbox i un llarg etcètera.

La capa més completa i complexa de totes es la de IaaS i serà en la que ens centrarem en aquest TFG. Un aspecte implícit des del principi en las característiques de *Cloud* Computing és que és un servei ofereix a tercers, però les interessants tecnologies varen causar que des de fa alguns anys diferents organismes i empreses comencessin a treballar de la mateixa manera però amb els seus recursos. Per tant, també es poden classificar els modes de cloud en funció del seu àmbit d'ús en:

- Privat: proporciona els mateixos avantatges que un *cloud* públic, incloent elasticitat i fiabilitat però a través d'una infraestructura privada. Al contrari que el model públic, que proporciona serveis a vàries organitzacions, un *cloud* privat està dedicat exclusivament a una sola organització.

- Públic: el proveïdor de servei ofereix recursos, com aplicacions i emmagatzematge, disponibles al públic general a través d'Internet. Els serveis de *cloud* públic poden ser gratuïts o bé oferts en un model de pagament per ús que evita despeses innecessàries als usuaris.

- Híbrid: Alguns serveis es gestionen al *cloud* privat i altres es transfereixen a un públic, o fins i tot fent ús de l'elasticitat del *cloud* es passen recursos del *cloud* privat al públic en funció de les necessitats. Aquests *clouds* normalment utilitzen una API comú que permet una bona integració entre ells i permeten una major flexibilitat i més opcions per a l'emmagatzematge de dades.

La figura que es mostra a continuació (veure Figura 1) resumeix els fonaments i bases sobre els que s'assenta el Cloud Computing.

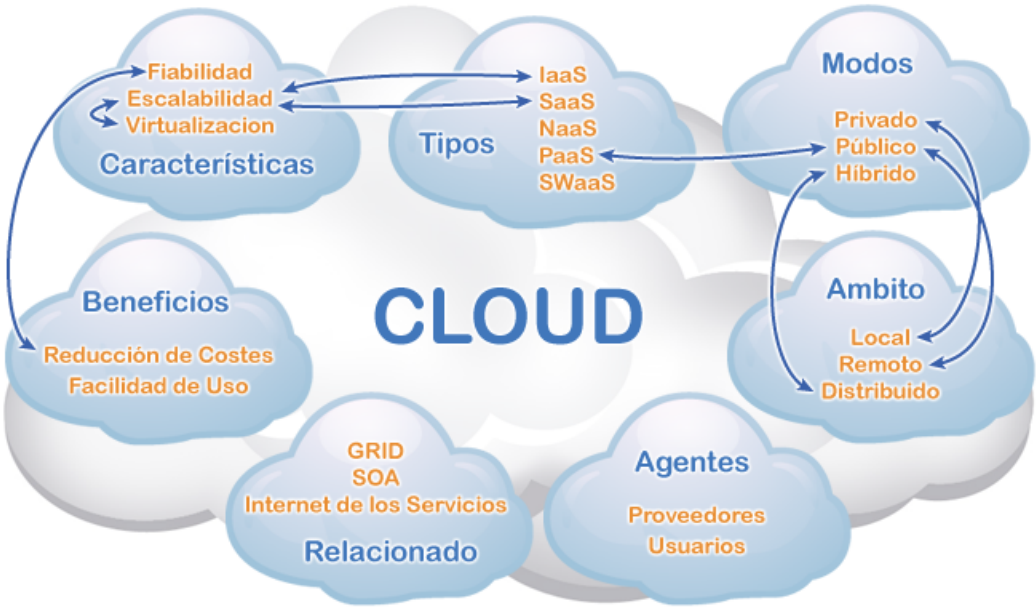


Figura 1: fonaments i bases del Cloud Computing Referència: cortesia de i2CAT

2.2 Projecte COSIGN

El projecte COSIGN (Combining Optics and SDN In next Generation data centre Networks) [2] és un projecte europeu FP7, és a dir de recerca, que es va iniciar el 2014 amb una duració de 36 mesos. El projecte aprofitarà la integració adequada de hardware òptic avançat amb les últimes tecnologies de software per millorar el rendiment, l'escalabilitat i la gestió de la xarxa i recursos IT. El TFG forma part d'aquest projecte europeu.

2.2.1 Consorci del projecte

El consorci COSIGN reuneix una combinació única de d'empreses privades, universitats, centres de recerca i proveïdors de servei amb l'experiència i els recursos necessaris per oferir noves solucions de xarxa escalables i amb garanties. Aquestes són les entitats que conformen el consorci:

- Technical University of Denmark (DTU)
- Nextworks (NXW)
- Fundació i2CAT
- POLATIS LTD
- University of Bristol (UNIVBRIS)
- University of Southampton (UNISOUTH)
- Technische Universiteit Eindhoven TU/e (TUE)
- PhotonX Networks
- IBM Israel – Science and Technology LTD
- OFS Denmark
- Interoute S.P.A. (IRT)
- Venture Photonics Ltd.

2.2.2 Objectius de COSIGN

L'objectiu de COSIGN és definir i implementar una arquitectura de centres de dades horitzontal i escalable mitjançant tecnologies òptiques, control de xarxa mitjançant SDN i orquestració amb OpenStack. Per tal de definir i implementar una arquitectura de centre de dades eficient i escalable, COSIGN desenvoluparà commutadors de transmissió òptica i tecnologia d'orquestració de xarxes basada en Software Defined Network (veure Software Defined Network), concretament amb OpenDaylight i OpenStack (veure OpenStack i OpenDaylight). Els pilars fonamentals del projecte són commutadors òptics nous, orquestració software i els nous casos d'ús facilitats per aquestes tecnologies.

La transformació del centre de dades que preveu el consorci és de canvis majors en la infraestructura del *Data Plane* d'avenços significatius en el *Control Plane* i d'innovació en la gestió i automatització de xarxes. Els objectius tècnics concrets són els següents:

1. A la xarxa del centre de dades, específicament al Data Plane, el projecte té com a fita assolir els següents punts:

    - Commutadors òptics *top-of-the-rack* d'alta densitat de ports, d'alta capacitat de transmissió i escalables.

    - Interconnexions del centre de dades purament òptiques de baixa latència.

    - Acotació espacial d'alta densitat per donar suport a la densitat de ports, així com l'escalabilitat de l'arquitectura utilitzant noves tecnologies òptiques com *multicore fibres*.

2. Construir a partir del paradigma *Software Defined Network* les capacitats necessàries per cobrir els requeriments de les xarxes òptiques.

3. Implementar el concepte "*Data Center Infrastructure as a Service*" desenvolupant els mecanismes necessaris per la composició i gestió de múltiples centres de dades virtuals aïllats i concurrents que comparteixen la mateixa infraestructura física.

2.2.3 Impacte esperat

Amb els diferents objectius exposats a l'apartat anterior, s'espera que l'impacte del projecte sigui notable ja que permetrà a nous models de negoci per proveïdors de servei, a operadors de centres de dades i a proveïdors de contingut proporcionar serveis *Cloud* sostenibles i innovadors a tota la ciutadania sigui a casa o arreu. A continuació s'exposen els motius:

- COSIGN enfortirà la influència de la industria europea a la definició de la nova generació de centres de dades.

- COSIGN permet innovar a la industria. Com a tal, busca dissenyar i desenvolupar xarxes *intra-Data Center* eficients, escalables, programables i econòmiques.

- COSIGN desbloquejarà la naturalesa rígida i tancada de les xarxes òptiques actuals gràcies a l'extensió del paradigma Software Defined Network i la implementació de nous commutadors òptics *top-of-the-rack*.

- COSIGN disposa d'un consorci que s'ha construït per incloure socis tant com amb impacte al mercat com amb estratègies sòlides en projectes d'estandardització.

2.2.4 Commutadors òptics

Encara que el TFG desenvolupa part de la capa software és interessant veure les característiques dels commutadors per saber fins a quin punt la solució global és innovadora.

Els nous commutadors desenvolupats dins el projecte COSIGN es caracteritzen per l'elevada capacitat transmissió de fins als 720Gbps, interconnexions de baixa latència, l'alta densitat de 128 ports i l'escalabilitat que ens ofereix el seu tamany i tecnologia. A la figura 2 podem veure l'aspecte del commutador desenvolupat per el *partner* TUE i a la figura 3 podem comprovar la millora de tamany respecte els *racks* dels centres de dades actuals.

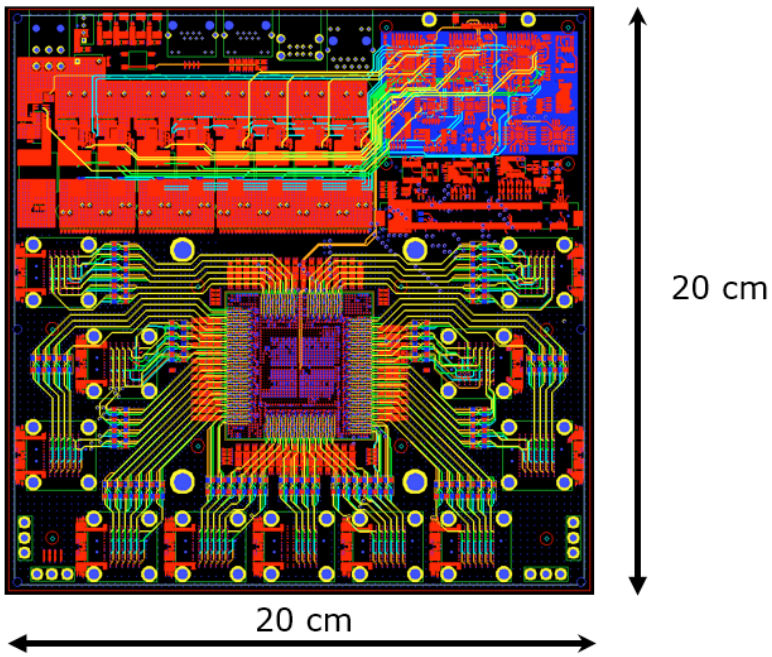


Figura 2: circuitería del commutador desenvolupat per el partner TUe. Referència: cortesia de TUE.

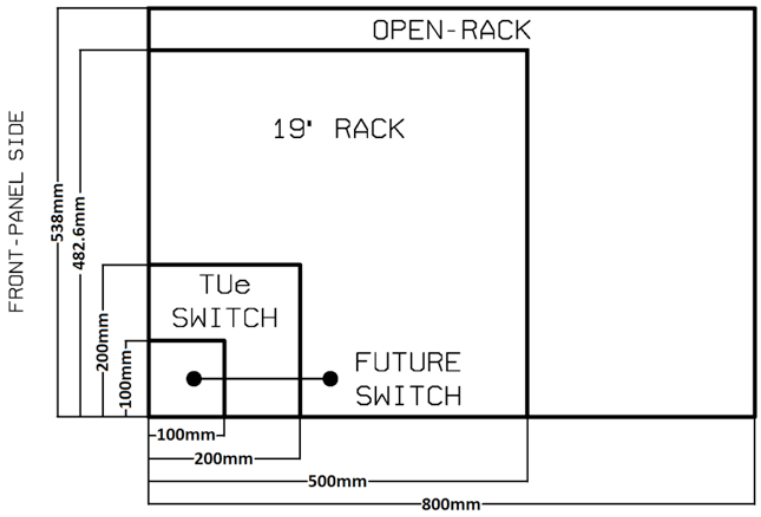


Figura 3: mides del commutador actual i del que s'està desenvolupant comparat amb l'espai del rack estàndard. Referència: cortesia de TUe.

2.2.5 Cas d'ús Virtual Data Center

Aquest cas d'ús del projecte COSIGN consisteix en virtualitzar els recursos de xarxa i computacionals del centre de dades oferint a l'usuari sol·licitar i gestionar una petició de recursos, a partir d'ara anomenada petició VDC (Virtual Data Center), que s'adapti a les seves necessitats.

Es desplegaran varies instàncies de VDC (resultat de processar les peticions VDC) aïllades entres elles a la mateixa infraestructura física. Cada instància de VDC demana una fracció de xarxa i computació del centre de dades per a gestionar aplicacions que s'executin mitjançant aquests recursos. Els nous recursos per possibilitar aquest cas d'ús els anomenarem node virtual i enllaç virtual (veure Figura 4).

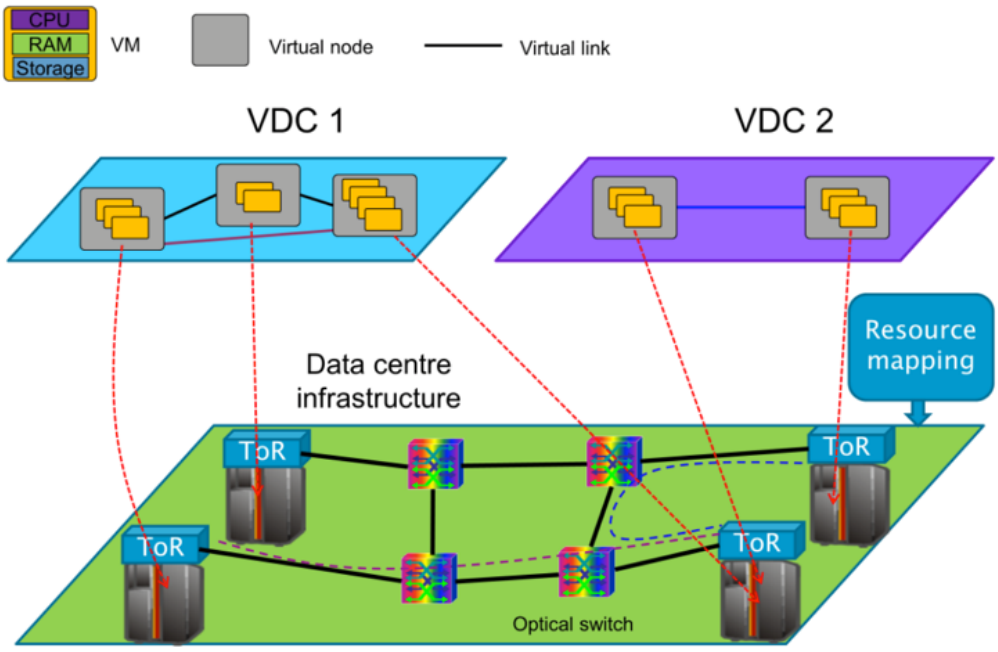


Figura 4: traducció dels recursos virtuals als físics. [3]

Els nodes virtuals estan composats per màquines virtuals repartides pel centre de dades però aquesta col·locació és transparent a l'usuari, és a dir, desconeix a quin servidor s'allotgen les màquines virtuals.

Les màquines virtuals requereixen recursos (RAM, CPU, Disc...) i una imatge de sistema operatiu per tal de ser inicialitzades. La imatge del sistema operatiu ha de poder ser personalitzada per l'usuari, que ha de poder escollir totes aquestes característiques en el moment d'inicialitzar les màquines virtuals.

Aquests nodes virtuals poden tenir connectivitat entre ells amb un ample de banda especificat per l'usuari. El recurs que possibilita aquesta connectivitat és l'enllaç virtual i també es transparent a l'usuari.

El responsable de processar les peticions és el Mòdul d'Algoritmes, desenvolupat per el departament de Teoria i Senyal i Comunicacions de la UPC dins del projecte COSIGN, que ofereix una REST API on poder consultar la instancia de VDC de l'usuari, enviar una petició nova i esborrar els recursos de la instancia executant-se en aquell moment. El Mòdul d'Algoritmes és conscient dels recursos disponibles al centre de dades i en rebre una petició VDC escollirà la col·locació més apropiada dels recursos.

En resum, els nodes i enllaços virtuals es tradueixen en vàries màquines virtuals distribuïdes en diferents servidors i diferents enllaços físics que fan que la connexió sigui possible amb l'ample de banda especificat per l'usuari i el Mòdul d'Algoritmes processa les peticions.

2.3 OpenStack

El projecte OpenStack [4] és una plataforma de *Cloud Computing* de programari lliure que amb tan sols cinc anys de desenvolupament s'ha convertit en una de les principals opcions per implementar un *cloud* públic o privat. OpenStack proporciona una solució de IaaS fàcil d'implementar a través d'un conjunt de serveis interrelacionats.

2.3.1 Principis fundacionals

OpenStack no depèn d'una sola empresa, sinó que s'ha constituït una fundació, la OpenStack Foundation, dels quals els membres són elegits entre tots els participants i els seus principis es resumeixen en el següents punts:

- Llicència Apache 2.0.
- Procés de disseny obert.
- Repositoris de codi font oberts.
- Tots els procediments de desenvolupament han de ser documentats.
- Disseny modular flexible gràcies a l'ús d'APIs.
- Orientat a l'adopció estàndards oberts.
- Comunitat oberta.

2.3.2 Components

Els serveis interrelacionats que hem mencionat al principi són mòduls amb una funció concreta. OpenStack És conegut per la seva extensibilitat gràcies a les seves APIS fàcils d'utilitzar i per la seva modularitat. Cadascun d'aquests mòduls està implementat amb Python, llenguatge també de codi obert, i s'encarrega d'una funció específica com per exemple de la computació, xarxa, gestió de les imatges, monitorització o orquestració entre d'altres. A continuació es detallen els mòduls més utilitzats d'OpenStack.

Mòdul	**Descripció**
Horizon	Interfície web de gestió
Nova	Gestió d'instàncies (màquines virtuals)
Cinder	Emmagatzematge de volums (block storage)
Keystone	Autentificació i autorització
Swift	Emmagatzematge d'objectes (object storage)
Glance	Gestió d'imatges per les instàncies
Neutron	Controlador de xarxes definides per software (SDN)
HEAT	Motor d'orquestració per desplegar aplicacions complexes descrites en arxius de text segons l'especificació HOT (Heat Orchestration Template)

Taula 1: components d'OpenStack, també coneguts com mòduls

El mòdul que pren importància al TFG aquí explicat és Horizon, implementat amb Django [5], que desplega un servidor web i ofereix una interfície gràfica amigable a l'usuari per gestionar i monitoritzar tota la infraestructura del centre de dades.

2.3.3 Exemple típic de funcionament

En funció dels components instal·lats i de seva distribució, el procés d'iniciar una instància serà més o menys complex, però el podem resumir en els següents passos:

1. L'usuari s'autentifica a Keystone, bé directament o a través de la interfície web Horizon, obté un token de sessió que el permetrà realitzar accions amb la resta de components d'OpenStack sense necessitat de tornar a autentificar-se. Aquestes accions estaran limitades pels permisos del rol que tingui l'usuari a Keystone.

2. L'usuari sol·licita a Glance la llista d'imatges disponibles. Aquestes imatges poden ser allotjades pel mòdul Swift o bé directament a l'equip de l'usuari.

3. Després de seleccionar una imatge i especificar les característiques de la instancia (flavors), l'usuari sol·licita a Nova que la instanciï. Nova automàticament escull el node del cloud (servidor) més adequat per executar la instancia si l'usuari no l' especifica.

4. Per últim, Neutron s'encarrega de la configuració de la xarxa virtual per la instància.

2.4 Software Defined Network

Les xarxes definides per software, en anglès *Software Defined Network*, és una arquitectura de xarxes computacionals amb l'objectiu de facilitar la implementació i implantació dels serveis de xarxa. Es distingeix per la seva escalabilitat, dinamisme i per l'increment d'usabilitat per als administradors de xarxa. Tot això és possible mitjançant la separació del *Data Plane* (Software) del *Control Plane* (Hardware), de manera que la gestió de xarxes ja no depèn d'una solució Hardware, sinó d'un controlador software més intel·ligent i preparat per assolir el màxim rendiment.

Actualment les tecnologies actuals de cap manera estan pensades per a conèixer els requeriments d'usuaris, proveïdors i empreses. A més a més, fer front a l'increment als requeriments del públic mundial és impossible amb les xarxes tradicionals [6].

2.4.1 Limitacions actuals

A continuació es detallen les limitacions de les xarxes, infraestructura i software actuals [7]:

- **Complexitat**: Generalment, per acomodar les xarxes a les necessitats dels usuaris, la industria ha millorat la seguretat i l'eficiència dels protocols de xarxa. Tanmateix, aquests protocols es defineixin d'una manera aïllada, resolent un problema aïllat enlloc de buscar una acció conjunta.

- **Polítiques incoherents**: Per implementar una política que doni abast a la xarxa completament, els administradors de xarxa han de configurar milers de mecanismes aparatosos. Per exemple, una tasca trivial com afegir una màquina virtual a la xarxa pot portar hores fins a que l'administrador encarregat reconfigura les llites d'accés (ACL) a tota la xarxa.

- **Falta d'escalabilitat**: a la vegada que les demandes dels Centres de Dades augmenten ràpidament la xarxa ha de créixer al mateix temps i de la mateixa manera. De tota manera, la xarxa es torna més complexa amb la suma dels aparells nous que a més a més requereixen ser configurats i gestionats.

- **Dependència del venedor**: les noves capacitats i serveis perseguits per proveïdors i empreses es veuen frenats pels cicles de producció dels fabricants i venedors, que poden tardar fins a més de cinc anys.

- **Augment dels costos:** al mateix temps que la xarxa i els recursos d'un centre de dades s'incrementen, també augmenta el CAPEX (Capital expenditure) degut a la necessitat de comprar més Hardware per donar solució a la demanda dels usuaris finals. A la figura 5 podem observar els resultats d'una enquesta feta a les companyies operadores.

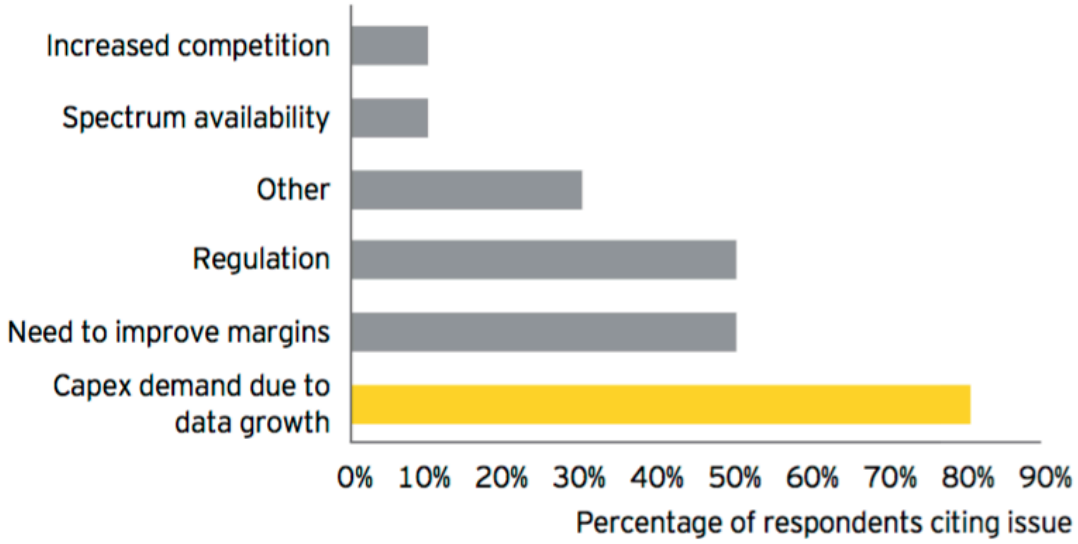


Figura 5: problemes majors al que s'enfronten les companyies operadores [8].

2.4.2 Necessitat d'una nova arquitectura

Els serveis de xarxa sorgits en els últims temps estan portant a les xarxes actuals al seu límit. A continuació s'exposen alguns dels motius que ens porten a la necessitat d'una nova arquitectura de xarxa:

- **Augment dels serveis *cloud* i *Big Data***: degut a la gran acollida per part del públic i empreses, l'agilitat d'operació, la infraestructura i l'escalabilitat d'aquests serveis requereixen escalabilitat dinàmica de computació, d'emmagatzematge i de recursos de xarxa.

- **Elevada carga de treball:** els administradors de xarxa es veuen sota la pressió de mantenir la seguretat i fiabilitat a les xarxes mentre que cada dia nous dispositius mòbils tenen accés a Internet.

- **Heterogeneïtat de protocols:** al contrari de les aplicacions client-servidor, les aplicacions modernes creen tràfic màquina a màquina mitjançant servidors i bases de dades abans de que l'usuari obtingui la resposta.

2.4.3 Arquitectura SDN

Encara que la terminologia està en constant modificació, els termes més comuns per els components de l'arquitectura són els següents:

- **Capa d'aplicacions:** Les aplicacions SDN son programes dels usuaris finals que usen els serveis de SDN a través de les APIs superiors (*Northbound*) de la capa de control. Permeten simplificar i automatitzar tasques de provisionament, configuració i gestionar nous serveis a la xarxa.

- **Capa de control:** és un controlador centralitzat que permet als desenvolupadors abstraure's de la topologia de xarxa i usar-ne les seves funcionalitats. El controlador SDN elimina la intel·ligència de commutació i encaminament de les dades dels nodes que realitzen tal funció, passant-ho al controlador SDN, que pren les decisions i selecciona el millor camí per el tràfic. Això permet que l'arquitectura sigui agnòstica als protocols usats entre les interfícies i ofereix el potencial d'unificar i simplificar la seva configuració.

- **Capa d'infraestructura:** està constituïda per els nodes de xarxa que realitzen la commutació i encaminament de paquets. Proporciona un accés obert programable a través d'una API (*Southbound*).

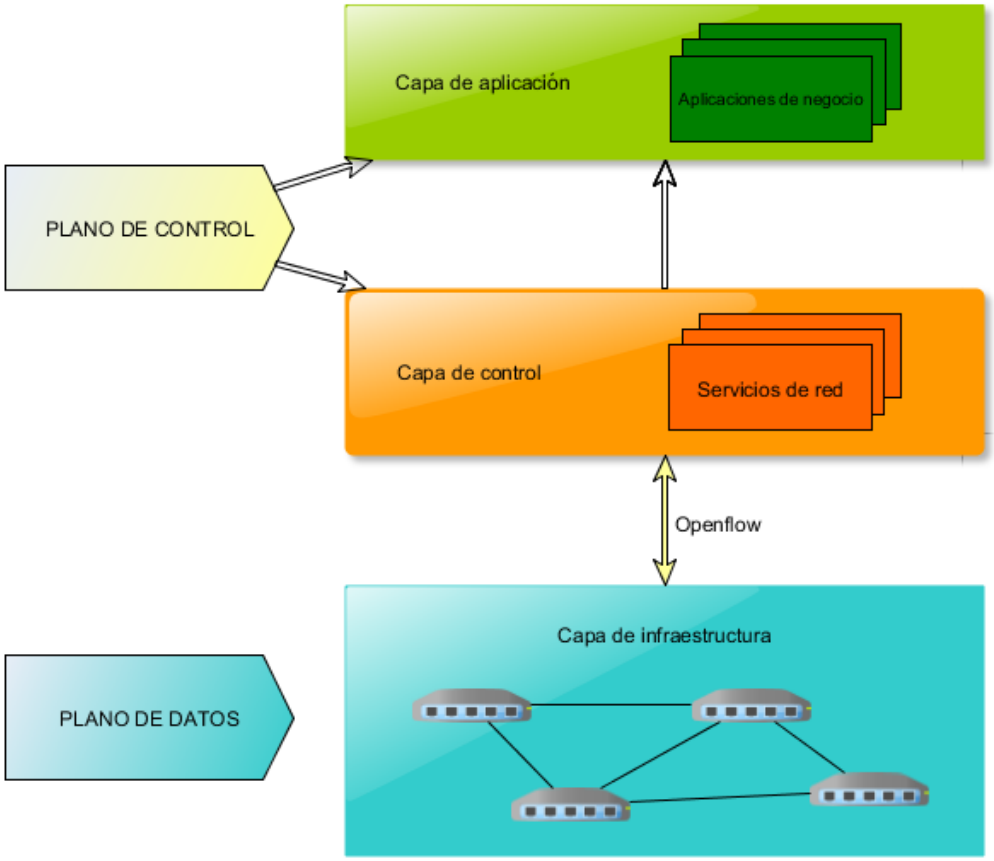


Figura 6: Arquitectura de SDN [9].

2.4.4 Estandardització

La ONF (Open Network Foundation) es un consorci industrial sense ànim de que està liderant l'evolució de SDN i estandarditzant els elements crítics d'aquesta nova arquitectura, tal com el protocol OpenFlow. Aquest consorci va ser fundat l'any 2011 per Deutsche Telekom, Facebook, Google, Microsoft, Verizon i Yahoo! contant actualment amb més de 140 companyies membre.

2.4.5 Automatització

En gran mesura, l'automatització amb SDN resideix a les APIs obertes *Northbound* i *Southbound*. La promesa de SDN de crear una infraestructura molt més àgil i flexible és desenvolupada principalment mitjançant la transformació de xarxa a una xarxa més programable. A continuació es descriuen tres exemples d'ús principals:

- **Ajustar els fluxos**: Aquest cas es centra en els protocols, com Openflow, que permeten al controlador SDN interaccionar amb els nodes de xarxa al pla de dades per ajustat com ha de fluir el tràfic a la xarxa, ajudant a resoldre la demanda d'aquesta.

- **Suport d'aplicacions**: Aquest cas es preocupa per la coordinació, automatització i la gestió d'excepcions d'una xarxa per adaptar millor les necessitats de cadascuna de les aplicacions que fan ús de la xarxa.

- **Automatització de xarxes**: Per últim, aquest cas es centra en la no interferència del administrador de xarxa en les accions que realitza la xarxa de forma automàtica quan hi ha algun problema i si aquest succeeix la intel·ligència de la xarxa s'encarrega de solucionar-ho.

2.4.6 Beneficis de SDN

Per obtenir el màxim rendiment, ús i simplicitat, les operadores de telecomunicacions, proveïdors de serveis al cloud i les empreses poden utilitzar la tecnologia SDN a tota la xarxa, des del centre de dades al escriptori, independitzant el hardware del pla de control de les aplicacions i automatitzar la xarxa.

A més a més, de les xarxes definides per software o SDN és la reducció dels costos en equipament i operació de xarxa, que com hem vist anteriorment és la major preocupació de les companyies de telecomunicacions.

En conclusió, és degut a la reducció dels costos, l'automatització, la programabilitat i el suport d'aplicacions que les tecnologies SDN ofereixen una solució a les complexitats i limitacions existents de les xarxes de centre de dades d'avui en dia i agilitzar el negoci entorn al model IaaS.

2.5 OpenDaylight

OpenDaylight [10] és un projecte de codi obert escrit amb Java organitzat per The Linux Foundation l'any 2013. El seu objectiu és accelerar l'adopció de xarxes definides per software (SDN) i crear una base sòlida per a les funcions de xarxa virtuals (Network Function Virtualization). Tot i que no té impacte en el desenvolupament d'aquest TFG, OpenDaylight és part de l'escenari global on s'executarà la solució i lidera la transformació de les tecnologies SDN.

2.5.1 Principis fundacional

Inspirats amb l'exemple d' OpenStack, els membres fundacionals del projecte tenen com objectiu construir plataforma oberta que utilitzin totes les empreses, evitant que els sistemes propietaris limitin el creixement del mercat i al mateix temps reduir els costos de desenvolupament. Aquesta plataforma SDN oberta i comú inclou àrees com el controlador SDN, els protocols de les aplicacions de xarxa, les interfícies d'usuari, els commutadors virtuals i per últim les interfícies físiques dels dispositius.

Tot i els seus membres fundacionals, OpenDaylight no té una sola companyia o conjunt de companyies que lideren el projecte, sinó que opera a través d'una comunitat oberta i activa a la qual qualsevol s'hi pot unir. El requisit per formar-ne part és contribuir a l'obertura de la comunitat, sigui oferint ajuda al desenvolupament, al màrqueting o a la direcció del projecte. No hi ha cap requisit monetari per formar part del projecte.

En concret, Big Switch Networkds, Brocade, Cisco, Citrix, Ericsson, IBM, Juniper Networks, Microsoft, NEC, RedHat i VMware son els membres fundacionals del projecte, els quals donen software i recursos d'enginyeria per ajudar aquest projecte de codi obert a definir la plataforma SDN oberta de referència dels propers anys.

2.5.2 Aspectes tècnics

Mitjançant el suport als estàndards oberts com l'estàndard de xarxa OpenFlow, OpenDaylight proporciona un marc comú de codi obert. La primera versió del projecte, anomenat Hydrogen, va ser llançat al febrer de 2014 i consistia en un controlador SDN de funcionalitat simple, xarxes virtuals d'overlay, complements (plug-ins) de protocols i millores dels dispositius de commutació. Després de l'alliberament de Hydrogen, els desenvolupadors preparaven la propera versió: Helium, que va ser la que va fer popular OpenDaylight entre desenvolupadors i operadores.

La principal avantatge d'aquesta proposta és que elimina les barreres d'adopció, ja que algunes organitzacions no es volen comprometre amb un venedor o fabricant en particular que pot acabar bloquejat al futur. Al ser una solució SDN, com hem vist en apartats anteriors, les empreses que l'utilitzin podran optar per tecnologies de diferents venedors de manera que seran Inter operables.

3. Objectius i estat de l'art

3.1 Objectius del TFG

3.1.1 Objectius principals

- **OP1 adquirir coneixements profunds sobre *Data Center networks*** en entorns cloud, així com les tecnologies i soluciones més actuals. Comprendre la seva estructura, funcionament i utilització per ser capaç d'implementar-ne extensions.

- **OP2 dissenyar la interfície web per el cas d'ús VDC**: Aquesta interfície ha d'oferir a l'usuari poder configurar i operar el centre de dades virtual sempre que tingui connexió a Internet. Els requeriments d'aquesta interfície seran proposats per l'alumne i validats pel consorci de COSIGN.

- **OP3 implementar una solució operativa**: La solució ha de ser vàlida i operativa per el cas d'ús Virtual Data Center (veure 2.2.5 Cas d'ús Virtual Data Center) del projecte COSIGN i amb impacte dins el context global d'un projecte tant competitiu com és COSIGN.

- **OP4 integrar amb el Mòdul d'Algoritmes**: Per tal de poder operar el centre de dades és necessària implementar una comunicació fiable amb el Mòdul d'Algoritmes desenvolupat per la UPC. Aquesta comunicació ha de ser possible mitjançant REST APIs.

3.1.2 Objectius secundaris

- **OS1 millorar el rendiment**: El mòdul d'OpenStack Horizon incorpora moltes funcionalitats pròpies que ja estan programades. Aquest objectiu implica no utilitzar funcionalitats pròpies del mòdul, sinó programar-ho des de zero i comprovar que el temps necessari és inferior. També s'analitzarà el codi d'Horizon per veure si es poden optimitzar accions típiques de funcionament.

- **OS2 protegir la interfície d'atacs**: Actualment qualsevol lloc web a Internet és susceptible de ser atacat. Degut a aquest fet i a la importància que té una interfície per gestionar un centre de dades, és important protegir-la per garantir-ne la integritat. Els punts febles més comuns a un lloc web són els formularis susceptibles a execució de codi Javascript, injecció SQL i la falsificació de peticions (CSRF).

- **OS3 usabilitat i experiència**: La usabilitat és la mesura de la qualitat de l'experiència que té un usuari quan interactua amb un producte o sistema. Degut a que es tracta d'usabilitat web s'haurà de dissenyar el lloc web per a que els usuaris puguin interactuar amb els elements de la forma més fàcil, còmoda i intuïtiva possible. Amb l'objectiu de garantir la millor experiència es facilitaran notificacions de possibles errors o respostes del sistema i es reduirà el número de clics necessaris per dur a terme les operacions.

- **OS5 garantir la compatibilitat de la solució**: Tractant-se d'una interfície web, s'ha de garantir el nombre màxim de navegadors i dispositius compatibles. Això no només implica que la programació sigui compatible entre navegadors, sinó que el disseny CSS sigui responsiu, és a dir, que s'adapti al dispositiu.

3.2 Cloud Service Management

La prestació dinàmica de serveis, plataformes o aplicacions cloud (IaaS, PaaS o SaaS) no es produeix perquè sí, sinó que és degut a la gestió eficaç que ofereixen les solucions de Cloud Service Management. A més a més de les millors pràctiques per assolir una administració efectiva de tots els elements associats amb el servei cloud, les plataformes de Cloud Service Management permeten als proveïdors estar capacitats per les contínues variacions de demanda en un entorn altament elàstic gràcies a l'aprovisionament automàtic de recursos [11].

Mitjançant la gestió i monitorització exitosa de serveis cloud, els proveïdors poden utilitzar la qualitat que proporciona el seu servei per diferenciar-se de la competència dins del que continua sent un mercat amb molt públic potencial i d'alta repercussió global. A més a més, una gestió eficaç dels serveis cloud també ajuda a reduir els costos econòmics i per conseqüència ampliar el marge de benefici. Tan mateix, assolir aquests objectius pot ser difícil en un entorn virtual en el que la visibilitat i el control del que succeeix en temps real són limitats.

Així doncs, és important conèixer els processos clau en la gestió dels serveis cloud ja que, com en tot negoci, l'objectiu és garantir la satisfacció del client. Essencialment, la gestió dels serveis cloud adapta els recursos de la infraestructura al acord de nivell de servei (SLA) del client. Això implica ajustar les operacions i les polítiques de tots els recursos implicats en l'entrega del servei amb l'objectiu d'orquestrar els recursos per garantir l'aprovisionament necessari, adaptable i efectiu.

Tot i que els venedors han desenvolupat varies plataformes privades de Cloud Service Management i de monitorització, amb infraestructura inclosa, per a que les empreses construeixin i gestionin els seus propis clouds privats, existeixen solucions gratuïtes com OpenStack (veure OpenStack) que permeten construir una plataforma de Cloud Service Management a partir d'una infraestructura pròpia.

Si hem entès el cas d'us Virtual Data Center (veure 2.2.5 Cas d'ús Virtual Data Center) i els objectius principals del TFG, sabem que la fita és dissenyar i implementar una interfície per a gestionar un centre de dades amb dos nous recursos (node virtual i enllaç virtual). Amb altres paraules, dissenyar i implementar una interfície de Cloud Service Management amb noves característiques. Per tant, abans de dissenyar i implementar la solució, estudiarem les solucions més esteses al mercat com Amazon Elastic Compute Cloud (Amazon EC2) i Rackspace Cloud, les analitzarem i compararem per conèixer si compleixen els requeriments d'aquest TFG, els seus punts febles i les seves fortaleses per així poder dissenyar i poder implementar una solució millor.

3.3.1 Amazon Elastic Compute Cloud

El servei Amazon Elastic Compute Cloud [12](Amazon EC2, nascut el 2006) tot i ser una de les solucions més populars de Cloud Management Service, falla en varis aspectes si ho comparem amb els requeriments d'aquest TFG. A continuació s'exposen les raons:

- No facilita emmagatzematge, sinó que cal contractar un servei a part. Aquest servei es coneix com Amazon Simple Storage Service (Amazon S3) i es troba en un apartat diferent de la interfície a la d'Amazon Web Services (veure Figura 7).

- No proporciona connectivitat amb un ample de banda garantit entre màquines virtuals. En cas que l'escenari requereixi ample de banda garantit entre els nodes de computació, el servei no és capaç de complir amb el requisit.

- L'usuari no té la possibilitat d'usar imatges .iso pròpies. Només pot escollir entre una llista de sistemes operatius. Aquesta és una limitació greu per usuaris que requereixin un ús avançat com operar amb imatges .iso pròpies. Per tant, l'usuari està obligat a construir l'entorn de cada màquina virtual.

- Els recursos assignats a les màquines virtuals estan definits per defecte. L'usuari només pot escollir entre una llista de recursos (RAM, CPU, HDD...) predefinits.

- No està pensat per usuaris que no siguin experts i s'inclouen molts productes innecessaris a la mateixa interfície (veure Figura 7). Requereix llegir molta documentació, a vegades per a fer operacions trivials.

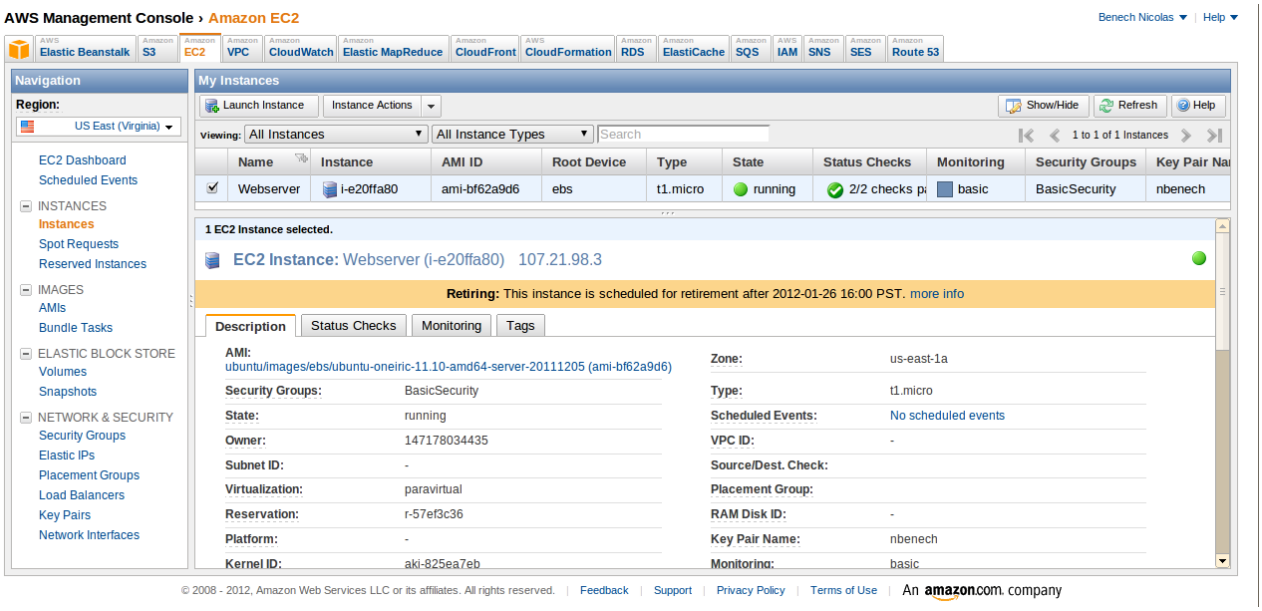


Figura 7: Panell de control d'Amazon EC2.

3.3.2 Rackspace Cloud

Rackspace Cloud és una altra plataforma de Cloud Service Management semblant a Amazon EC2 i que utilitza OpenStack. Tot i que és difícil de conèixer els detalls de les organitzacions per raons estratègiques, el seu tamany com organitzacions poden ser comparats amb els seus beneficis: 27 milions de dòlars de beneficis per Rackspace contra 415 milions de dòlars de beneficis per Amazon durant el primer quadrimestre del 2013 [13]. La plataforma Rackspace va néixer el 2006 i ofereix els següents serveis que ens resulten familiars:

- Cloud Servers: Comparable amb EC2. Es el nom de la marca per a un servei combinat de computació.

- Cloud Tools: És el nom per aplicacions composades que s'executen al servei Cloud Servers. Per exemple, en cas de necessitar un entorn PHP tenen una plantilla anomenada Zend, per analitzar el rendiment tenen Cloudkick, per monitoritzar aplicacions tenen CopperEgg, per a una base de dades MySQL tenen Xeround... etcètera.

- Cloud Sites: és una solució de PaaS (veure apartat 2.1) que dona accés a una configuració d'infraestructura predefinida. Per exemple, en cas de necessitar allotjar un lloc web en un servidor web, més serveis de bases de dades i correu electrònic es paga un preu mensual per tot això com a unitat.

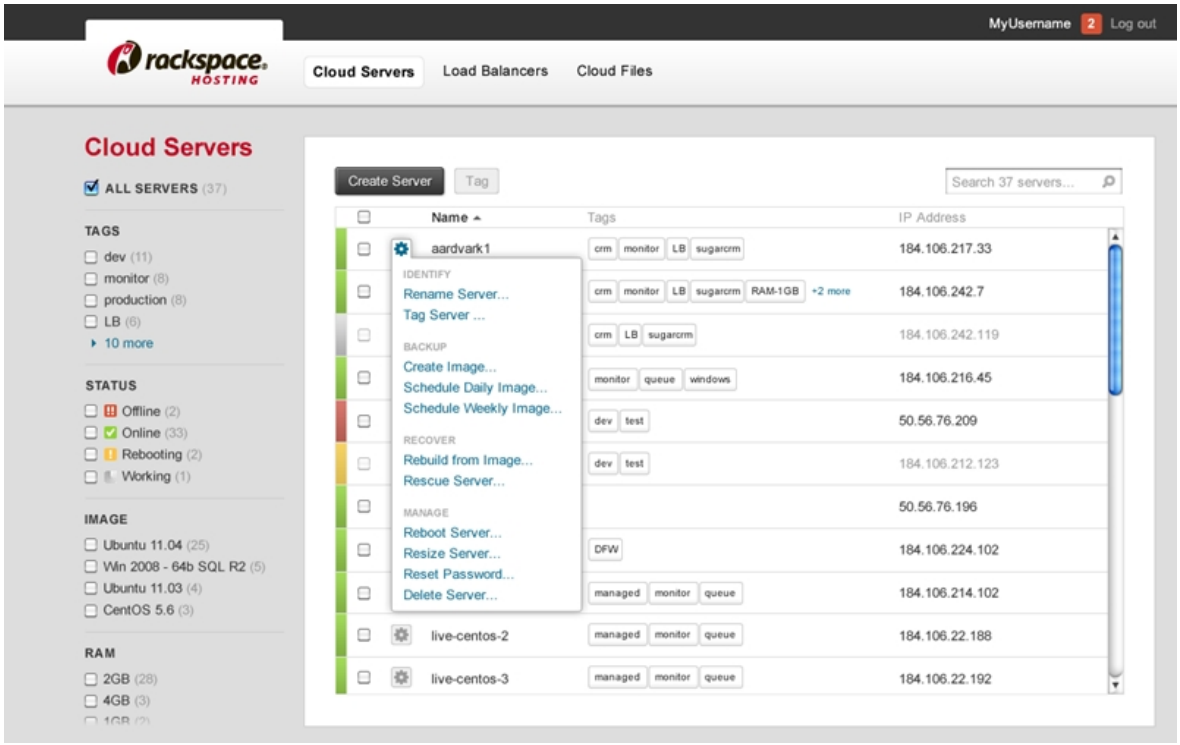


Figura 8: Panell de control de Rackspace Cloud

3.3.2 Comparació d'Amazon EC2 i Rackspace

A continuació es detalla una taula comparativa de pros i contres entre Rackspace i Amazon Web Services (AWS, que inclou Amazon EC2 i Amazon S3).

	Pros	**Contres**
Rackspace Cloud	Interfície molt més usable i clara que la de AWS. Permet utilitzar imatges de màquines virtuals pròpies. Adequat per a organitzacions de tamany mitjà. Més estable que AWS; menys talls en el servei. Millor servei d'atenció al client.	Generalment més car que AWS. Més petit que AWS, disposa de menys centres de dades distribuïts al món. Molts menys desenvolupadors que hi treballen. No garanteix un ample de banda desitjat entre els nodes computacionals. No admet flavors personalitzats.
Amazon EC2	Amb diferencia, el major proveïdor cloud. Preus molt competitius. Elevat número de desenvolupadors. Múltiples centres de dades arreu del planeta.	Menys rendiment que Rackspace [14]. Es cobra l'emmagatzematge no utilitzat i les operacions de lectura i escriptura a disc. Tampoc garanteix un ample de banda desitjat entre els nodes computacionals. No permet usar imatges pròpies per les màquines virtuals. Tampoc admet flavors personalitzats.

Taula 2: comparació entre Rackspace i AWS.

Actualment, cap de les solucions al mercat permet garantir un ample de banda entre nodes que agrupin màquines virtuals i per tant, cap d'elles pot ser aprofitable. Per tant s'utilitzarà el mòdul Horizon d'OpenStack ja que no només permet llibertat d'ús d'imatges pròpies i recursos personalitzats per a les màquines virtual, sinó que en tractar-se d'un projecte de programari lliure podem accedir al codi font per modificar-lo i satisfer les condicions del cas d'ús Virtual Data Center.

4. Tasques i planificació

El treball realitzat es descompon en un total de 18 tasques. Es va iniciar el 5 d'octubre del 2015 com i es va acabar el 18 de Juny de 2016.

4.1 Descripció de les tasques

Estudi del projecte COSIGN

Aquesta primera tasca va consistir en llegir molta documentació del projecte europeu COSIGN per tal de conèixer quines tasques hauria de desenvolupar. Aquesta documentació és interna del projecte i es en forma de deliverables, organitzats en diferents "work package", on es descriu la feina preliminarment.

Instal·lació i estudi d'OpenStack

Un cop assabentat de l'àmbit del projecte, el següent pas lògic era instal·lar Openstack i estudiar-ne l'arquitectura, escalabitat, modularitat i com es podia modificar.

Instal·lació i estudi d'OpenDaylight

Després d'aconseguir instal·lar Openstack mitjançant Devstack, tocava instal·lar Opendaylight en la mateixa màquina ja que és el controlador usat dins COSIGN.

Integració d'Openstack i Opendaylight

Amb ambdós ja instal·lats, es va procedir a integrar Opendaylight amb OpenStack per a així poder fer ús de totes les funcionalitats d'Opendaylight des d'OpenStack.

Formació amb Python

Degut a que els mòduls d'OpenStack estan escrits amb Python 2.7 i sobretot Horizon, el mòdul que aixeca un servidor i ofereix a l'usuari una interfície gràfica per falta de coneixements previs va ser imprescindible dedicar un mes d'aprenentatge a Python.

Formació amb Django

Un cop adquirits coneixements sòlids amb Python, llenguatge amb el que corre Django, era necessari formar-se amb aquest *framework* web ja que el mòdul d'OpenStack Horizon està programat amb Django.

Definició dels requeriments del TFG

Aquesta tasca es correspon a entendre, definir i acordar amb la resta de socis involucrats dins el projecte el disseny de la interfície, el seu contingut i quines funcionalitats ha de tenir. Es va debatre amb trucades de tipus setmanals on es presentava la feina feta i les propostes pertinents.

Instal·lació del servidor Horizon

Per a poder ampliar Horizon és necessari instal·lar-lo com a mòdul de desenvolupament separat de la instal·lació original d'OpenStack.

Preparació del servidor

Un cop el servidor està instal·lat és necessari configurar-lo i generar els nous apartats, és a dir tots els directoris i fitxers per a les pàgines web, que seran afegits al mòdul original.

Topologia del VDC

Aquesta tasca es correspon a programar amb jQuery, CSS3 i la llibreria Vis.js una topologia de recursos físics i de xarxa per el cas d'ús de Virtual Data Center. jQuery ens permet gestionar les dades dels objectes JSON més eficientment que amb Javascript, CSS3 s'utilitza per a donar els estils a la pàgina i Vis.js ens permet crear una representació topològica amb drag & drop, zoom i físiques de moviment dels nodes del graf.

Representació detallada del VDC

Després d'acabar la representació en forma de topologia, cal mostrar també la informació detallada de cada node computacional així com dels seus enllaços d'una manera clara i fàcil d'entendre.

Gestió de la petició del VDC

L'usuari que fa ús de la interfície genera una petició de recursos. La tasca implica la programació i gestió d'un objecte JSON que inclou tota la informació de la petició

Integració amb l'orquestrador

Arribat aquest punt la interfície gràfica ja ha sigut finalitzada i cal integrar-la amb un mòdul nou d'algoritmes extern al projecte, que processa la petició de recursos VDC. Això implica instal·lar aquest nou mòdul i habilitar un canal de comunicació bidireccional amb Horizon i l'orquestrador mitjançant REST APIs.

Fase de prova i correcció

Com a tot projecte, és la fase responsable de provar totes les funcionalitats, corregir errors i resoldre vulnerabilitats.

Recuperació de peticions

Un cop la integració sigui completada es desenvoluparà un mecanisme per a que l'usuari pugui guardar les seves peticions de VDC i recuperar-les gràcies a l'orquestrador.

Redacció de la memòria

El temps previst per a la redacció i supervisió de la memòria.

Preparació de la presentació

El temps previst per a preparar la presentació final del TFG.

4.2 Alternatives i desviacions

Degut al anticipat inici del projecte s'ha disposat de suficient temps per a adquirir els coneixements necessaris i desenvolupar tot tal i com estava previst. Tanmateix, existeix la possibilitat de que si els socis del projecte no arriben a temps o bé retarden els inputs necessaris, com per exemple l'orquestrador, no es pugui afegir funcionalitats com la recuperació de peticions.

No hi ha hagut modificacions temporals respecte la feina realitzada i la planificació original però sí de contingut. La tasca Recuperació de peticions havia de servir per a guardar un històric de les diferents topologies del centre de dades virtual però des del consorci de COSIGN s'ha decidit que aquesta funcionalitat és adient realitzar-la al Mòdul d'Algoritmes (veure apartat 7) ja que usa una Bases de Dades amb les propietats de la instància i per tant s'evitaria feina redundant. Tanmateix, s'ha aprofitat el temps per afegir dues funcionalitats: copiar el text JSON al porta retalls amb un sol clic i pujar una instància especificada dins un arxiu de text per posteriorment modificar-la, eliminar-la i desplegar-la.

4.3 Llistat de les tasques Gantt

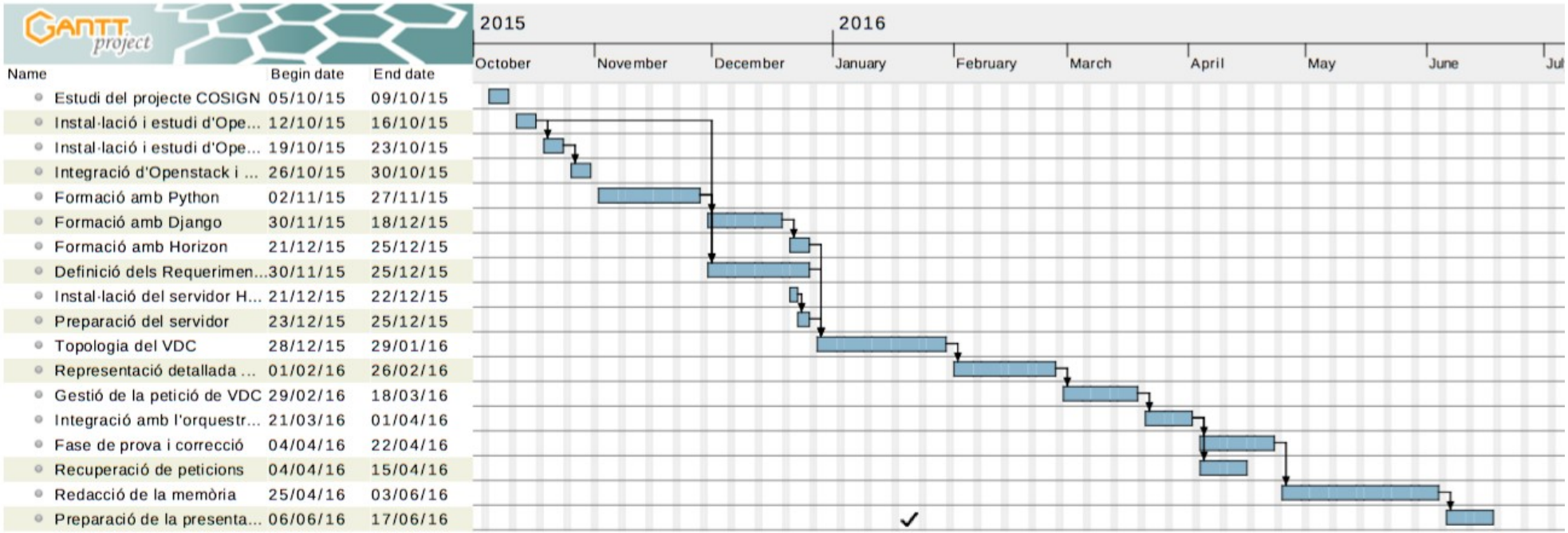
A sota podem veure la llista de les tasques del diagrama de Gantt, així com la seva duració. Al següent apartat també podem trobar el diagrama on podem veure-hi la distribució temporal de les tasques i la seves dependències.

Tasks

Name	Begin date	End date
Estudi del projecte COSIGN	05/10/15	09/10/15
Instal·lació i estudi d'Openstack	12/10/15	16/10/15
Instal·lació i estudi d'Opendaylight	19/10/15	23/10/15
Integració d'Openstack i Opendaylight	26/10/15	30/10/15
Formació amb Python	02/11/15	27/11/15
Formació amb Django	30/11/15	18/12/15
Formació amb Horizon	21/12/15	25/12/15
Definició dels Requeriments del TFG	30/11/15	25/12/15
Instal·lació del servidor Horizon	21/12/15	22/12/15
Preparació del servidor	23/12/15	25/12/15
Topologia del VDC	28/12/15	29/01/16
Representació detallada del VDC	01/02/16	26/02/16
Gestió de la petició de VDC	29/02/16	18/03/16
Integració amb l'orquestrador	21/03/16	01/04/16
Fase de prova i correcció	04/04/16	22/04/16
Recuperació de peticions	04/04/16	15/04/16
Redacció de la memòria	25/04/16	03/06/16
Preparació de la presentació	06/06/16	17/06/16

4.4 Diagrama de Gantt

Gantt Chart



4.5 Metodologia

Durant l'especificació dels requeriments i el desenvolupament s'han fet trucades setmanals amb els *partners* (socis del consorci COSIGN) per validar i posar en comú la feina realitzada. També dins l'àmbit del projecte COSIGN s'ha documentat les tasques realitzades per afegir-ho als deliverables propis del projecte. Aquests deliverables els validen revisors de la Comissió Europea conforme el projecte avança segons el que s'espera. Per tant, existeix un major compromís que en altres possibles circumstancies respecte l'assoliment de les fites planejades.

A més de les trucades setmanals, també s'ha disposat de l'ajuda i supervisió d'un *project manager* també en forma de reunions setmanals. Tal *project manager* també és membre del projecte COSIGN, fet que en facilita la gestió i possibilita aportar l'ajuda i perspectiva adequades.

Per al desenvolupament s'ha utilitzat el software de control de versions Git. Aquest software ens permet mantenir un control en les diferents versions del codi i ens assegura que podrà ser revertit a versions anteriors per contenció d'errors. La versió final del codi també serà analitzada per revisors de la Comissió Europea.

5. Pressupost

5.1 Identificació de recursos

En aquest apartat es recopilaran els recursos hardware i software necessaris així com la seva relació amb l'organització de les tasques del projecte.

5.1.1 Recursos hardware i software

A. HP ProBook 63060b
B. Servidor Linux Remot
C. Ubuntu 14.04
D. OpenStack
E. OpenDaylight
F. Sublime Text
G. Editor de text Vim
H. Skype
I. Microsoft Office 2010
J. Windows 7 Professional
K. Bitbucket

5.1.2 Assignació de recursos per activitat

Activitat	Recursos
Planificació del projecte i contextualització	
Estudi del projecte COSIGN	A, C
Instal·lació i estudi d'Openstack	A, B, C, D
Instal·lació i estudi d'Opendaylight	A, B, C, E
Integració d'Openstack i Opendaylight	A, B, C, D, E
Formació amb Python	A, C, F
Formació amb Django	A, C, F
Formació amb Horizon	A, B, C, F
Anàlisi i entorn	
Definició dels requeriments del TFG	A, H, I, J
Instal·lació del servidor Horizon	A, B, C, D
Preparació del servidor	A, B, C , D, G
Desenvolupament	
Topologia del VDC	A, B, C , D, G, K
Representació detallada del VDC	A, B, C , D, G, K
Gestió de la petició de VDC	A, B, C , D, G, H, K
Integració amb l'orquestrador	A, B, C , D, E, G, H, K
Fase de prova i correcció	A, B, C , D, G, H, K
Recuperació de peticions	A, B, C , D, G, H, K
Etapa final	
Redacció de la memòria	A, I, J
Preparació de la presentació	A, I, J

Taula 3: ús de recursos per activitat del diagrama de Gantt.

5.2 Amortització i estimació

A continuació es detallen els costos associats als recursos hardware i software, així com la seva amortització prevista. Aquesta amortització es calcula tenint en compte que el coeficient màxim permès pel Govern d'Espanya és del 25%. Per tant l'amortització s'expressa amb la següent formula:

$$\frac{(preu*0{,}25*hores_{projecte})}{(anys\ útils*365)} = \frac{(preu*150)}{4*365}$$

5.2.1 Amortització del Hardware

Recurs	Unitats	Preu	Vida útil	Amortització
HP ProBook 63060b	2	800€/Unitat	4 anys	164,4€
Servidor Linux Remot	1	15€/Mes	4 anys	9,25€
Total	--	1690€	--	173,65€

Taula 4: cost i amortització dels recursos Hardware.

5.2.2 Amortització del Software

Recurs	Preu	Vida útil	Amortització
Ubuntu 14.04	0€	--	--
OpenStack	0€	--	--
OpenDaylight	0€	--	--
Sublime Text	55€	--	5,65€
Editor de text Vim	0€	--	--
Microsoft Office 2010	79€ *	4 anys	8,12€
Skype	--	--	--
Windows 7 Professional	0€*	4 anys	--
Bitbucket	0	--	--
Total	134€	--	13,77€

Taula 5: cost i amortització dels recursos Software.

El Sistema Operatiu *Windows 7* és gratuït per estudiants de la Facultat d'Informàtica de Barcelona i els estudiants universitaris poden obtenir el paquet d'ofimàtica *Microsoft Office* amb un preu reduït.

5.2.3 Recursos Humans

A la següent taula s'indica la dedicació en hores i el seu cost en funció dels diferents rols que participen al projecte.

Rol	Hores estimades	Preu per hora [15]	Cost estimat
Project Manager	100	35€	3.100€
Desenvolupador	500	22,5€	7.500€
Total	600	--	10.600€

Taula 6: cost dels recursos humans.

5.2.4 Estimació total dels costos

A continuació es mostra el cost total previst del projecte. El desglossament es pot trobar a l'apartat anterior a les taules 2, 3 i 4. Els imprevistos, expressats després d'aquest apartat, no es comptabilitzen ja que tots els serveis afectats són externs i per tant les possibles despeses no van a càrrec del projecte. En el pitjor dels casos poden retardar l'execució de les tasques per hores.

Concepte	Cost
Hardware	173,65€
Software	13,77€
Recursos Humans	10.600€
Total	10.787,42€

Taula 7: cost total del projecte.

5.3 Imprevistos i contingències

Els imprevistos més probables en un projecte d'aquestes característiques són els següents:

- Pèrdua d'avenços: durant el desenvolupament no es perdrà informació ja que s'usarà un repositori a Bitbucket per a control de versions.

- Caiguda del servidor: aquesta incidència no es pot solucionar directament ja que cal esperar que el servei d'atenció contesti i el servei de manteniment el repari. Això ha passat i amb menys de trenta minuts ha estat solucionat.

- Fallada de la xarxa: en cas que la sortida a Internet falli, caldrà avisar al manteniment tècnic i esperar que solucionin el problema. També s'ha donat el cas i pot implicar no accedir al servidor en remot i per tant no poder treballar durant algunes hores o bé experimentar lentitud alhora de navegar a Internet.

Es realitza una reunió de seguiment i supervisió setmanalment amb el cap del projecte. En aquestes reunions s'estudia l'estat del projecte per anticipar les necessitats que poden sorgir, preveure possibles errors o fer adaptacions a la solució d'aleshores.

6. Sostenibilitat

En aquest apartat es comentarà i es justificarà la sostenibilitat del projecte en els camps econòmic, social i ambiental. A continuació es mostra l'avaluació que s'ha realitzat en els diferents camps:

Dimensió	Puntuació sobre 10
Econòmica	9
Social	7
Ambiental	8

Taula 8: valoració de les dimensions del projecte.

6.1 Dimensió econòmica

S'ha escollit els productes més útils, lògics i raonables en funció de les necessitats de cada tasca per a executar de manera satisfactòria cadascuna de les fases. El temps previst a cada tasca és proporcional a la seva importància. A més a més, es farà ús de tecnologies existents com jQuery o Bootstrap per agilitzar la feina i reduir la càrrega de treball. Observant el resultat expressat a la taula 7 (veure Taula 7: cost total del projecte.) podem veure que el projecte és viable si tenim en compte que el projecte COSIGN disposa de 10.000.000€ de pressupost i que la feina presentada al projecte està justificada degut al valor que aporta.

El resultat final serà competitiu perquè oferirà un nou servei d'alta capacitat òptica i escalabilitat al mercat de IaaS [16]. Aquest resultat permetrà als proveïdors d'aplicacions operar amb serveis innovadors i competitius que aporten valor a la tecnologia actual.

6.2 Dimensió social

El projecte està pensat per ser usat a tot els països del primer món, ja que es on els Centres de Dades operen majoritàriament. Els qui usin el servei seran proveïdors d'aplicacions i serveis corrent a sobre una infraestructura i que per tant, necessiten gestionar-ne la xarxa i recursos de manera remota. Hi haurà tants usuaris finals com usuaris facin ús d'un servei que corre sobre la solució que proposa i implementa el projecte.

A més a més, farà possible la creació de nous models de negoci per proveïdors de serveis, d'aplicacions, propietaris de Centres de Dades per proporcionar serveis *Cloud* a ciutadans a casa o a qualsevol lloc.

6.3 Dimensió ambiental

La taula 3 (veure Taula 3: ús de recursos per activitat del diagrama de Gantt.) especifica els recursos que s'utilitzen per a cada etapa del projecte. El seu consum és purament elèctric i en tenir-ne alguns serveis externs com ara el servidor remot, que està virtualitzat, es redueix l'impacte i ajuda a un estalvi energètic major. A més a més, tot el material físic es aprofitable per a altres projectes.

Finalment, no es controla l'energia que pugui a consumir tota la infraestructura final però, degut als nous commutadors òptics de baix consum elaborats per COSIGN podem afirmar contribuirà globalment al Centre de Dades ja que els commutadors òptics consumeixen la meitat [17] que els commutadors de tecnologia Ethernet.

7. Disseny i implementació

7.1 Requeriments

En aquesta secció s'especifiquen els requeriments inicials de la solució implementada. Cal recordar que aquest requeriments han sigut definits per l'alumne, proposats al consorci COSIGN i validats per el mateix.

7.1.1 Requeriments funcionals

- Els usuaris han de poder fer *log-in*.
- Els usuaris han de poder fer *log-out*.
- Els usuaris han de poder veure la instancia VDC en el seu estat actual.
- Els usuaris han de gestionar la petició VDC. Això implica les següents operacions:
    - Afegir node virtual amb un nom especificat per entrada d'usuari.
    - Afegir enllaç virtual amb un ample de banda especificat per entrada d'usuari.
    - Afegir màquines virtuals als nodes virtuals. L'usuari ha de poder especificar les característiques de les màquines virtuals.
    - Editar els noms dels nodes virtuals.
    - Esborrar màquines virtuals dels nodes virtuals.
    - Esborrar nodes virtual.
    - Esborrar enllaç virtual.
- Els usuaris han de poder desplegar els recursos especificats a la petició VDC.
- Els usuaris han de poder esborrar la instancia VDC.
- Els usuaris han de poder veure els nodes virtuals i la seva informació.
- Els usuaris han de poder veure els enllaços virtuals i la seva informació.
- Els usuaris han de poder navegar per la interfície web.

7.1.2 Requeriments no funcionals

1. Un projecte (grup d'usuaris) només pot gestionar una instancia VDC.

2. Les instancies VDC han d'estar aïllades entre elles (*tenant isolation*).

3. Cada projecte, anomenat tenant d'OpenStack, pot gestionar una sola instancia VDC.

4. Compatibilitat: la solució ha de ser compatible amb els navegadors Google Chrome i Mozilla Firefox.

5. Integritat: el sistema ha de ser robust i mantenir la seva integritat, assegurant que no és subjecte de possibles errors interns. A més, l'usuari no ha de poder introduir dades errònies.

6. Extensibilitat: les funcionalitats del sistema han de poder ser incrementades o modificades sense afectar la resta de funcionalitats.

7. Escalabilitat: la solució ha de poder escalar i mantenir un funcionament correcte si l'usuari fa ús de molts recursos VDC.

8. Consistència: els resultats de les mateixes operacions han de ser iguals.

9. Seguretat: la solució ha d'estar protegida de possibles usos fraudulents per part dels usuaris.

10. La solució ha de ser integrada amb el Mòdul d'Algoritmes (veure apartat 2.2.5 Cas d'ús Virtual Data Center).

11. La solució ha de poder ser instal·lada a diferents equips o màquines virtuals amb el sistema operatiu Ubuntu 14.04 i amb la versió Liberty d'OpenStack instal·lada prèviament.

7.2 Disseny

7.2.1 Disseny preliminar

Degut a la claredat dels requeriments i l'habilitat prèvia en programació web no ha sigut necessari realitzar dissenys preliminars a la implementació. Per tal de mantenir la coherència al mòdul Horizon, que serà ampliat per afegir-hi la interfície gràfica del cas d'ús Virtual Data Center, es mantindran els estils i modes de funcionament.

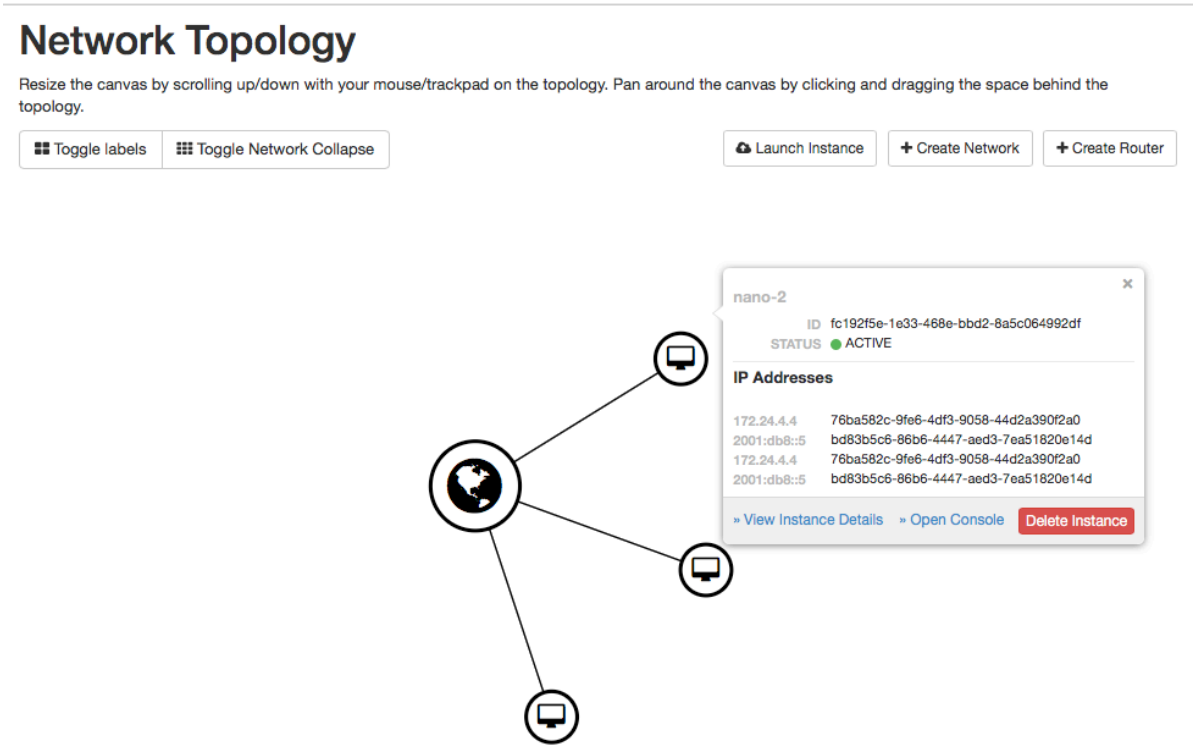


Figura 9: apartat Network Topology d'Horizon.

A la figura anterior podem veure una representació dels recursos en forma de graf que permet fer *zoom* i *drag & drop* als elements i veure els detalls dels nodes en fer-hi clic. És una manera visual de veure els recursos actuals i és per aquest motiu que es farà una representació semblant, però millorada, de la topologia de la instancia VDC.

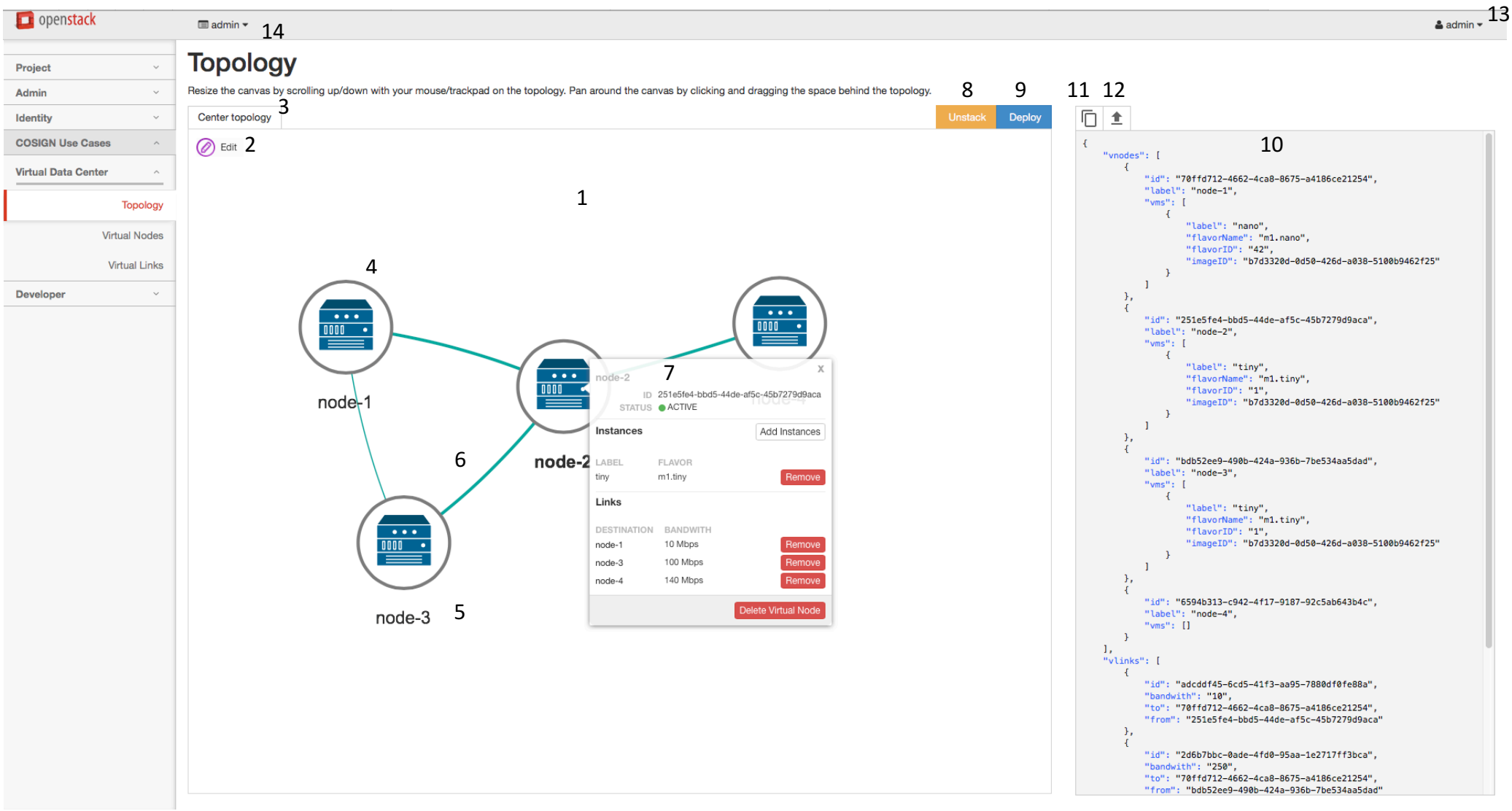
S'ha decidit dividir la interfície en tres pàgines diferents: una primera amb una topologia en forma de graf amb *drag & drop* i zoom habilitats i una descripció JSON de la petició VDC, una segona pàgina on es mostri la informació detallada dels nodes virtuals i una tercera per la informació dels enllaços virtuals. Les funcionalitats es centralitzaran a la primera pàgina per tal d'evitar redundància i reduir la possibilitat d'errors de consistència. Els tres primers requeriments no funcionals es compleixen si es manté el funcionament actual d'Horizon.

Per tal de fer possible la comunicació entre l'ordinador de l'usuari i el Mòdul d'Algoritmes es desenvoluparà una REST API al servidor Horizon que aïlli la informació que pot ser objecte d'ús fraudulent i només s'enviarà la informació necessària a l'usuari.

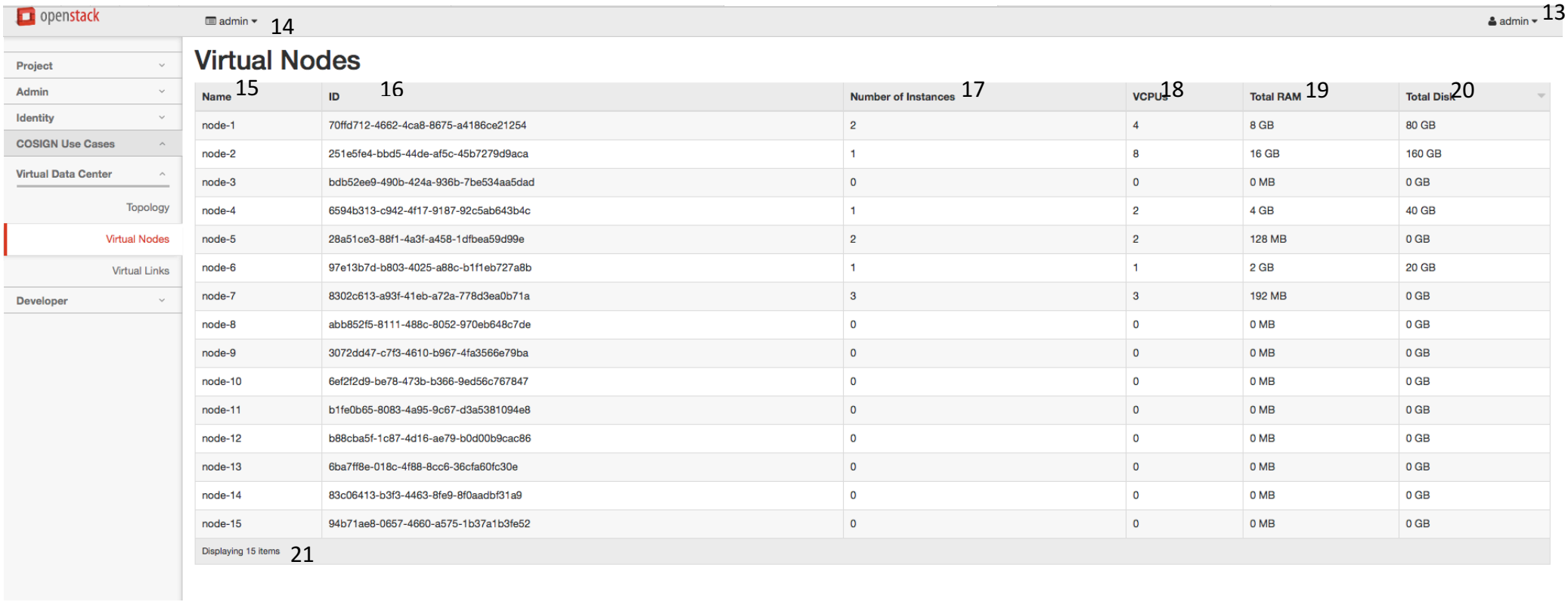
7.2.2 Disseny final

En aquest apartat es mostra el disseny final de la solució i una numeració per els elements rellevants explicats a l'apartat 7.2.3 Descripció dels elements.

7.2.2.1 Secció Topology

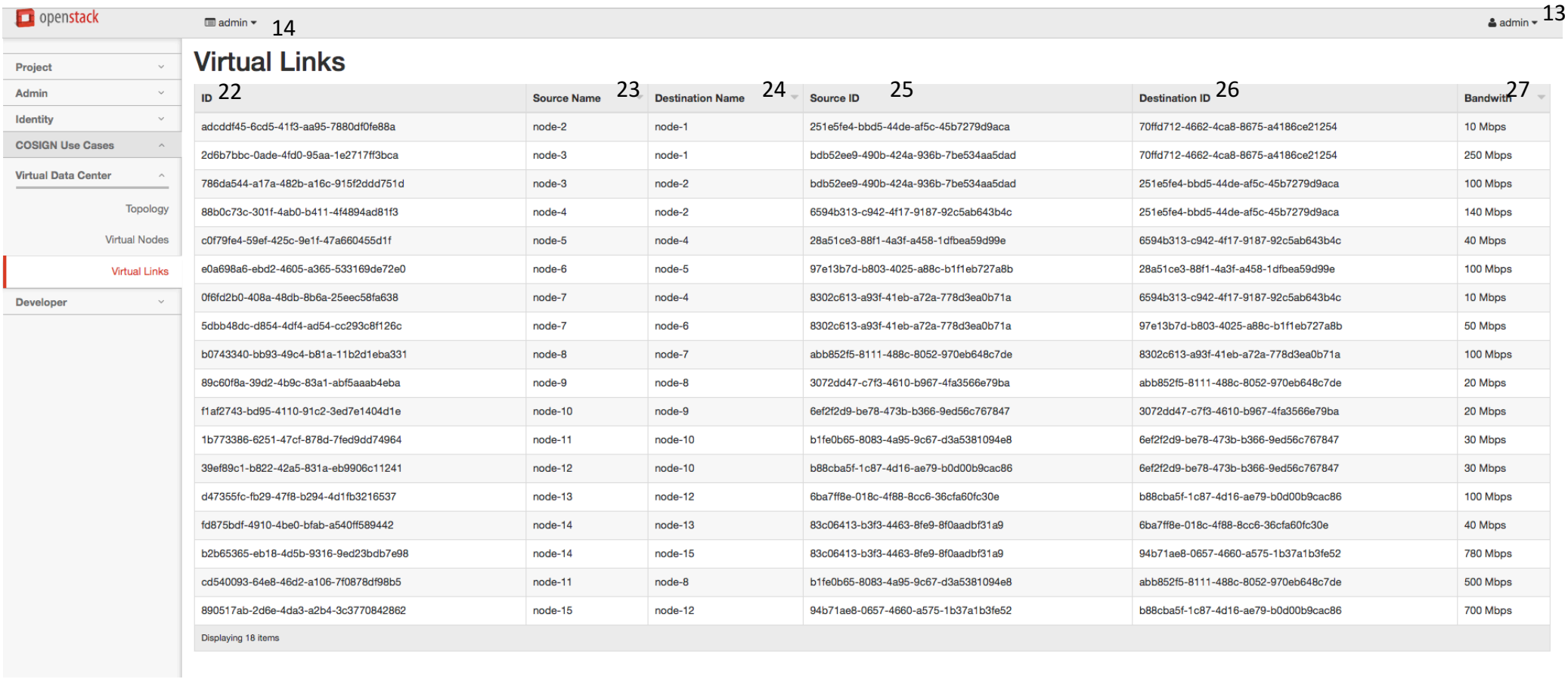


7.2.2.2 Secció Virtual Nodes

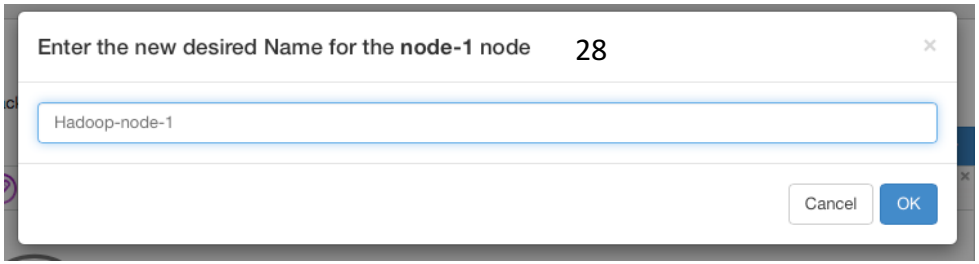


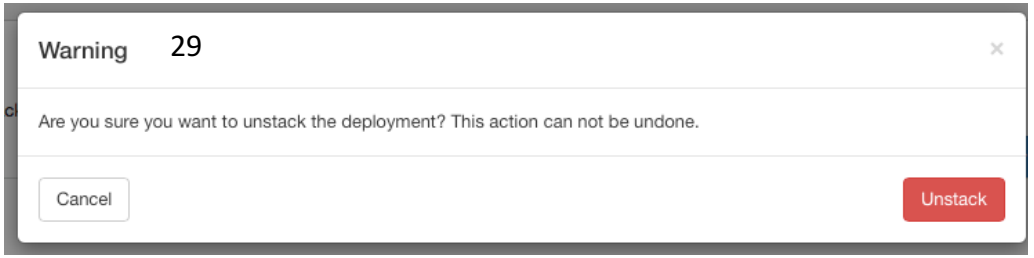
Name	ID	Number of Instances	VCPUs	Total RAM	Total Disk
node-1	70ffd712-4662-4ca8-8675-a4186ce21254	2	4	8 GB	80 GB
node-2	251e5fe4-bbd5-44de-af5c-45b7279d9aca	1	8	16 GB	160 GB
node-3	bdb52ee9-490b-424a-936b-7be534aa5dad	0	0	0 MB	0 GB
node-4	6594b313-c942-4f17-9187-92c5ab643b4c	1	2	4 GB	40 GB
node-5	28a51ce3-88f1-4a3f-a458-1dfbea59d99e	2	2	128 MB	0 GB
node-6	97e13b7d-b803-4025-a88c-b1f1eb727a8b	1	1	2 GB	20 GB
node-7	8302c613-a93f-41eb-a72a-778d3ea0b71a	3	3	192 MB	0 GB
node-8	abb852f5-8111-488c-8052-970eb648c7de	0	0	0 MB	0 GB
node-9	3072dd47-c7f3-4610-b967-4fa3566e79ba	0	0	0 MB	0 GB
node-10	6ef2f2d9-be78-473b-b366-9ed56c767847	0	0	0 MB	0 GB
node-11	b1fe0b65-8083-4a95-9c67-d3a5381094e8	0	0	0 MB	0 GB
node-12	b88cba5f-1c87-4d16-ae79-b0d00b9cac86	0	0	0 MB	0 GB
node-13	6ba7ff8e-018c-4f88-8cc6-36cfa60fc30e	0	0	0 MB	0 GB
node-14	83c06413-b3f3-4463-8fe9-8f0aadbf31a9	0	0	0 MB	0 GB
node-15	94b71ae8-0657-4660-a575-1b37a1b3fe52	0	0	0 MB	0 GB

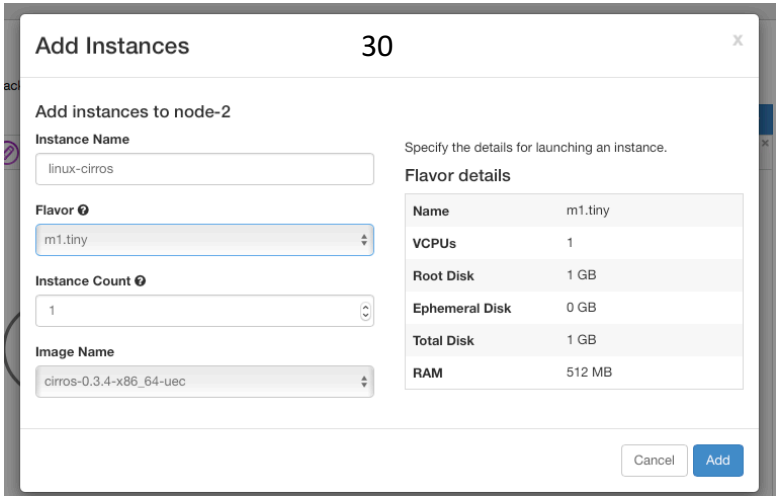
7.2.2.3 Secció Virtual Links

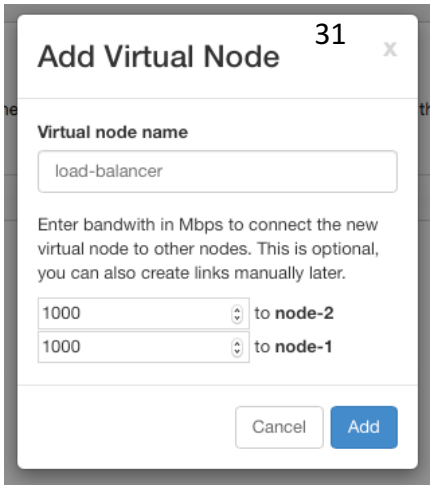


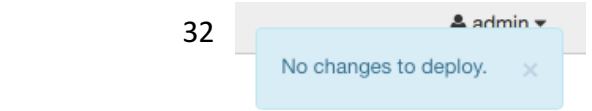
7.2.2.4 Diàlegs, formularis i notificacions

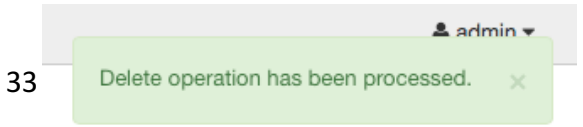


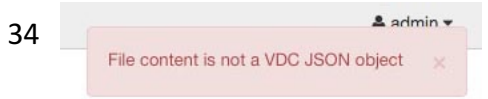


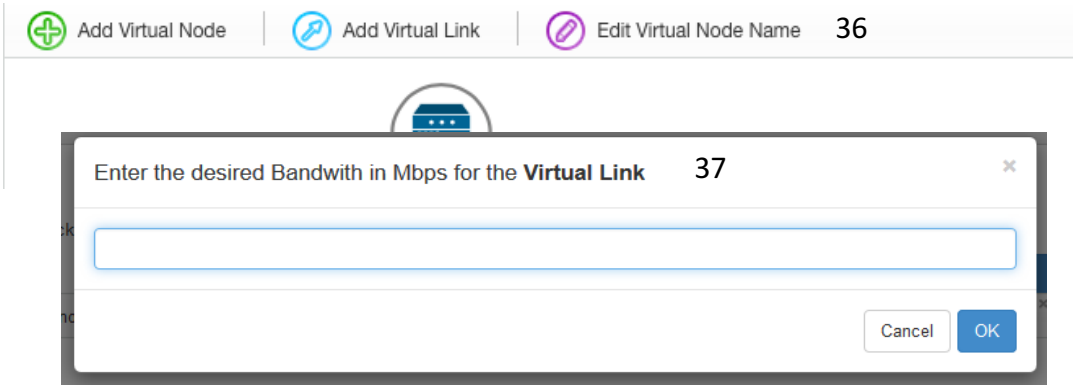












7.2.3 Descripció dels elements

En aquest apartat es descriuen els elements enumerats a l'apartat anterior.

1. **Representació topològica**: En aquest requadre es pot visualitzar dinàmicament la topologia de la instància VDC d'acord amb els canvis s'apliquin a la petició. S'hi pot fer zoom, moure el conjunt i moure nodes individualment. La topologia inclou físiques de moviment.

2. **Botó d'edició**: Fent clic en aquest botó apareix una barra d'eines (element 36) que ens permet afegir nodes virtuals, enllaços virtuals i canviar el nom dels nodes virtuals.

3. **Botó de centrar i estabilitzar**: Serveix per a centrar la topologia en el requadre i estabilitzar-la, és a dir, aturar les físiques de moviment en aquell moment.

4. **Representació de node virtual**: Aquest dibuix representa un node virtual. Quan està desplegat es mostra amb aquest color i si no s'ha desplegat es mostra de color gris.

5. **Nom del node virtual**: Per defecte és el nom que s'ha introduït en el moment d'afegir el nou node virtual però es pot editar mitjançant el botó d'edició.

6. **Representació d'enllaç virtual**: Aquest cable representa un enllaç virtual i el seu estat: si ha sigut desplegat es mostra d'aquest color i en cas contrari es mostra de color gris.

7. **Informació del node Virtual**: En fer clic a un node virtual apareix aquest camp d'informació. Indica l'estat del node virtual, el seu identificador, les seves màquines virtuals i els enllaços virtuals connectats. També hi podem veure botons d'acció com esborrar-ne una màquina virtual, esborrar un enllaç virtual, esborrar el node sencer amb els seus enllaços i afegir-hi màquines virtuals.

8. **Botó Unstack**: Esborra la topologia, la petició i allibera els recursos de la instancia VDC.

9. **Botó Deploy**: Desplega de nou la instancia VDC d'acord amb la petició VDC.

10. **Descripció JSON**: Descriu amb un objecte JSON les propietats de la petició VDC.

11. **Botó Copy**: Copia al porta-retalls de l'usuari la descripció JSON. Un cop l'usuari hi ha fet clic, pot copiar el objecte JSON amb les últimes modificacions. Aquesta funcionalitat no és disponible amb Safari per motius de seguretat.

12. **Botó Upload**: En fer-hi clic l'usuari pot seleccionar un arxiu de text on hi hagi un objecte JSON que descrigui una instància VDC. Un cop seleccionat l'arxiu, es validen els camps i es substitueix la representació topològica i la descripció JSON.

13. **Selecció d'usuari**: Permet sortir de la sessió, anar als ajustaments i veure l'ajuda.

14. **Selecció de projecte**: En fer-hi clic es despleguen els projectes als que pertany l'usuari. Els recursos com màquines o xarxes virtuals d'un projecte estan aïllats des de la resta de projectes i no en coneixen l'existència. Això es coneix com a *tenant isolation*.

15. **Columna Name**: Mostra el nom del node virtual. Totes les columnes són ordenables asdecenment o descendentment fent clic al títol de la columna.

16. **Columna ID**: Mostra l'identificador del node virtual.

17. **Columna Number of Instances**: Número de màquines virtuals del node virtual.

18. **Columna VCPUs**: Mostra el total de CPUs virtuals que requereix el node virtual.

19. **Columna RAM**: Mostra el total de la memòria RAM que necessita el node virtual.

20. **Columna Total Disk**: Mostra el total d'emmagatzematge que es reserva per al node.

21. **Peu de taula**: Indica el número d'elements que es mostren.

22. **Columna ID**: Mostra el nom de l'enllaç virtual.

23. **Columna Source Name**: Mostra el nom del node virtual d'origen de l'enllaç virtual.

24. **Columna Destination Name**: Mostra el nom del node virtual de destinació de l'enllaç.

25. **Columna Source ID**: Mostra l'identificador del node virtual d'origen de l'enllaç.

26. **Columna Destination ID**: Mostra l'identificador del node virtual de destinació de l'enllaç.

27. **Columna Bandwith**: Mostra l'ample de banda en Mbps reservat per l'enllaç virtual.

28. **Formulari d'edició de nom**: Es mostra quan l'usuari ha fet clic a la barra d'eines. Permet a l'usuari canviar el nom d'un node virtual.

29. **Confirmació d'operació Unstack**: Es mostra quan l'usuari clica al botó Unstack per confirmar que l'usuari vol dur a terme l'operació. En cas que no hi hagi canvis no es permet realitzar l'operació, no es demanarà confirmació a l'usuari i es mostra una notificació d'error informant del problema.

30. **Formulari d'afegir màquines virtuals**: Es mostra si l'usuari ha fet clic al Botó "Add Instances" que podem veure a l'element 7 (Informació del node virtual). Aquest diàleg ens permet especificar les característiques de les màquines virtuals (instàncies) que volem agrupar al node virtual tals com els recursos, la imatge, un nom i el número d'instàncies que volem afegir amb les mateixes característiques.

31. **Formulari d'afegir node virtual**: Es mostra quan l'usuari fa clic a "Add Virtual Node" a l'element 36 (Barra d'eines). Aquí podem especificar el nom del node virtual i també connectar-hi enllaços virtuals directament només especificant l'ample de banda desitjat cap a altres nodes virtuals ja afegits.

32. **Notificació informativa**: Exemple de notificació informativa. Aquest tipus de notificació s'utilitza per guiar i informar a l'usuari, com per exemple quan no es pot desplegar la instancia VDC perquè ja ha sigut desplegada i no hi ha hagut canvis.

33. **Notificació d'èxit**: Exemple de notificació d'èxit. Es mostra quan una operació s'ha realitzat correctament, com per exemple quan la instancia VDC es desplega correctament.

34. **Notificació d'error**: Exemple de notificació d'error. Es mostra quan hi ha algun error i quan no es poden realitzar operacions de "Deploy" o "Unstack" perquè no hi ha hagut canvis a la petició o bé perquè la instància VDC està buida.

35. **Confirmació d'operació Deploy**: Es mostra quan l'usuari clica al botó "Deploy" (element 9) per confirmar que realment vol desplegar la topologia. En cas que no hi hagi canvis a la instancia VDC no es permet realitzar l'operació, no es demana confirmació a l'usuari i es mostra una notificació d'error informant del problema.

36. **Barra d'eines**: Es fa visible quan l'usuari fa clic al botó d'edició (element 2) i es pot amagar si l'usuari la tanca clicant a la creu, fet que tornaria a mostrar el botó d'edició. Hi podem trobar les opcions d'afegir un node virtual, editar el nom d'un node virtual i afegir un enllaç virtual. La primera opció mostra el diàleg d'afegir node virtual (element 31), la segona el diàleg d'edició de nom (element 28) i la tercera el diàleg de crear enllaç virtual (element 37).

37. **Formulari de crear enllaç virtual**: Es mostra quan es fa clic al botó "Add Virtual Node" que podem veure a la barra d'eines (element 36) i mitjançant una operació de Drag & Drop l'usuari indica els nodes a connectar. Un cop ha especificat els nodes, el diàleg espera que l'usuari entri l'ample de banda desitjat que forçosament serà convertit a un número.

7.2.4 Mètodes de la interfície

Els mètodes de la interfície es classifiquen en mètodes de configuració i mètodes d'operació. Ens referim a mètodes de configuració a aquells mètodes que tenen com a finalitat configurar la petició VDC mentre que els mètodes de d'operació tenen efectes reals sobre els recursos físics del centre de dades.

7.2.4.1 Mètodes de configuració

- **Afegir node virtual**: Afegeix un node virtual (element 4) a la petició VDC amb el nom introduït per l'usuari (element 31). S'executa si l'usuari clica "Add Virtual Node" a la barra d'eines (element 36) i completa el procediment. L'usuari pot afegir-hi enllaços virtuals si hi ha altres nodes existents especificant l'ample de banda desitjat entre el node existent i el que s'està afegint.

- **Afegir enllaç virtual**: Afegeix un enllaç virtual entre dos nodes virtuals amb un ample de banda especificat per l'usuari. S'executa si l'usuari clica "Add Virtual Node" a la barra d'eines (element 36), selecciona un node d'origen, arrossega l'enllaç cap al node de destinació i introdueix l'ample de banda desitjat (element 37).

- **Afegir màquines virtuals**: Afegeix màquines virtuals a un node virtual. S'executa si l'usuari clica a "Add Instances" a l'element 7 i completa el formulari d'afegir màquines virtuals (element 30). En aquest formulari l'usuari ha d'escollir els recursos entre una llista de "flavors" predefinits o personalitzats, la imatge del sistema operatiu i especificar un nom de la màquina o màquines a afegir.

- **Editar el nom del node virtual**: Edita el nom del node virtual seleccionat. S'executa si l'usuari clica a "Edit Virtual Node Name" de la barra d'eines i completa el formulari d'edició de nom (element 28).

- **Esborrar node virtual**: Esborra el node virtual seleccionat. S'executa si l'usuari clica a "Delete Virtual Node" a l'element 7. Aquesta operació també esborra les màquines virtuals agrupades dins el node virtual i tots els seus enllaços virtuals connectats.

- **Esborrar enllaç virtual**: Esborra l'enllaç virtual del node virtual seleccionat. S'executa si l'usuari clica a "Remove" a una fila d'un element d'enllaç virtual a l'element 7.

- **Esborrar màquina virtual**: Esborra la màquina virtual del node virtual seleccionat S'executa si l'usuari clica a "Remove" a una fila d'un element de màquina virtual a l'element 7.

- **Llegir petició VDC**: Substitueix la petició VDC (element 10) que es mostra en aquell moment per una petició descrita en un arxiu de text. Es fa una comprovació senzilla de que la petició es correcta i si no ho és no es substitueix la petició o instancia actual. En cas que la comprovació inicial acceptés la petició, el Mòdul d'Algoritmes s'assegura que no hi hagi errors abans de processar la petició. S'executa si l'usuari clica el botó Copy (element 11).

7.2.4.2 Mètodes d'operació

- **Desplegar VDC**: Envia la petició VDC actual al servidor mitjançant una petició HTTP Post per a que el servidor i generi una nova petició amb un identificador privat d'OpenStack anomenat "tenant ID" al cos de la petició cap al Mòdul d'Algoritmes, que utilitza l'identificador per gestionar, processar la petició i desplegar els recursos a la topologia física. Per últim, el Mòdul d'Algoritmes respon a la petició HTTP, el servidor envia aquesta resposta al navegador client i es mostra el resultat a l'usuari en forma de notificació. S'executa si l'usuari clica el botó Deploy (element 9).

- **Eliminar VDC**: Envia una petició HTTP Post al servidor per esborrar la instancia de VDC. El servidor crea una nova petició HTTP Delete amb el "tenant ID" a la URL com a paràmetre cap al Mòdul d'Algoritmes, que esborra tota la informació i elimina els recursos de la instancia VDC executant-se en aquell moment. Per últim, el servidor envia la resposta de la petició HTTP Delete que ha enviat al Mòdul d'Algoritmes al client. S'executa si l'usuari clica el botó Unstack (element 8).

7.2.4.3 Altres mètodes

- **Copiar petició VDC al porta-retalls**: Copia la descripció JSON de la petició VDC (element 10) al porta-retalls de l'usuari. Posteriorment l'usuari pot copiar la descripció a qualsevol camp de text, fet que li permet guardar-ho en arxius de text per després poder llegir la petició VDC (element 11) d'un arxiu per facilitar el procediment de generació de la topologia. S'executa si l'usuari clica al Botó Copy (element 11).

- **Canviar de projecte**: L'usuari pot navegar entre els diferents projectes als que pertanyi. Cada projecte té un límit de recursos disponibles, assignat per l'administrador de la infraestructura, i tal com s'ha explicat abans, els projectes estan aïllats entre ells i no poden visualitzar ni modificar els altres projectes (multi-tenancy). Per a canviar de projecte l'usuari ha de clicar l'element 14.

- **Iniciar sessió**: Per tal d'entrar a la interfície gràfica l'usuari s'ha d'autentificar mitjançant les seves credencials a un panell d'entrada inicial per iniciar la sessió. Les credencials poden ser gestionades per l'usuari ha iniciat sessió.

- **Sortir de la sessió**: L'usuari pot acabar la seva sessió actual mitjançant l'element 13.

Els mètodes de configuració tenen com a objectiu configurar la petició de VDC. A més de la representació topològica, la petició es mostra amb un objecte JSON a l'element 10 de la secció Topology. Un cop s'ha desplegat la petició mitjançant el mètode d'operació Deploy, aquesta passa a ser la instancia VDC en execució a la infraestructura.


```
{
    "vnodes": [
        {
            "vms": [
                {
                    "label": "nano-1",
                    "flavorName": "m1.nano",
                    "flavorID": "42",
                    "imageID": "b7d3320d-0d50-426d-a038-5100b9462f25"
                }
            ],
            "id": "a48810b8-5511-4b53-91bb-bec592f24dfc",
            "label": "node-1"
        },
        {
            "vms": [
                {
                    "label": "tiny-1",
                    "flavorName": "m1.tiny",
                    "flavorID": "1",
                    "imageID": "b7d3320d-0d50-426d-a038-5100b9462f25"
                },
                {
                    "label": "tiny-2",
                    "flavorName": "m1.tiny",
                    "flavorID": "1",
                    "imageID": "b7d3320d-0d50-426d-a038-5100b9462f25"
                }
            ],
            "id": "d43f79c7-e256-411f-954a-7031733abae7",
            "label": "node-2"
        },
        {
            "id": "9517554e-ba53-4bbc-a58b-a026b69f1788",
            "label": "node-3",
            "vms": []
        }
    ],
    "vlinks": [
        {
            "to": "a48810b8-5511-4b53-91bb-bec592f24dfc",
            "from": "d43f79c7-e256-411f-954a-7031733abae7",
            "id": "ec5deacd-bd0f-4be7-8427-0e40b459b2e8",
            "bandwith": "1024"
        },
        {
            "id": "9e492f56-2fbf-4840-ac46-ce8fb0baf4b5",
            "bandwith": "530",
            "to": "a48810b8-5511-4b53-91bb-bec592f24dfc",
            "from": "9517554e-ba53-4bbc-a58b-a026b69f1788"
        }
    ]
}
```


Figura 10: Representació JSON de la petició VDC .

7.3 Implementació

7.3.1 Característiques del servidor i entorn

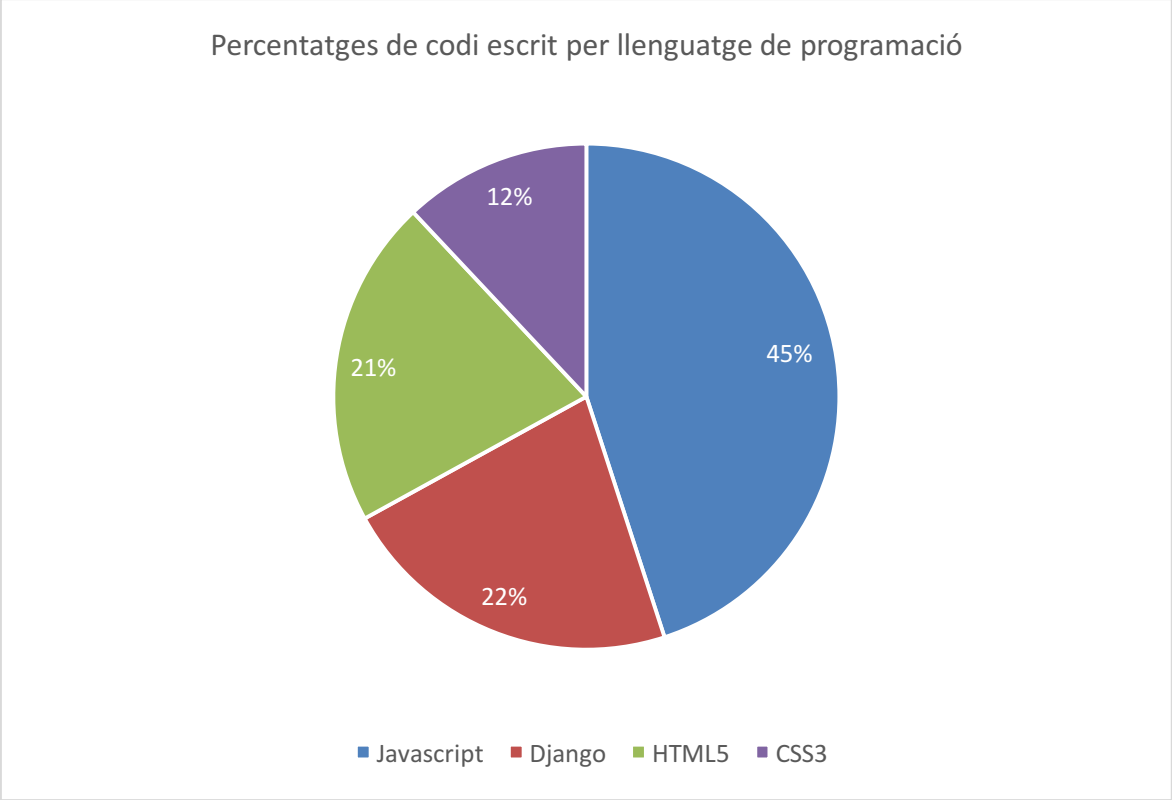
El servidor que allotja la interfície i processa totes les peticions és el mateix on s'ha dut a terme el desenvolupament. Es tracta d'un servidor Linux remot amb la versió Liberty d'OpenStack instal·lada per a poder estendre, és a dir ampliar, el mòdul Horizon i així complir objectius i requeriments.

OS: Ubuntu Server 14.04 LTS
CPU: Intel Xeon E5620 2.40Ghz - 8 nuclis
RAM: 10GB
Disc: 40GB

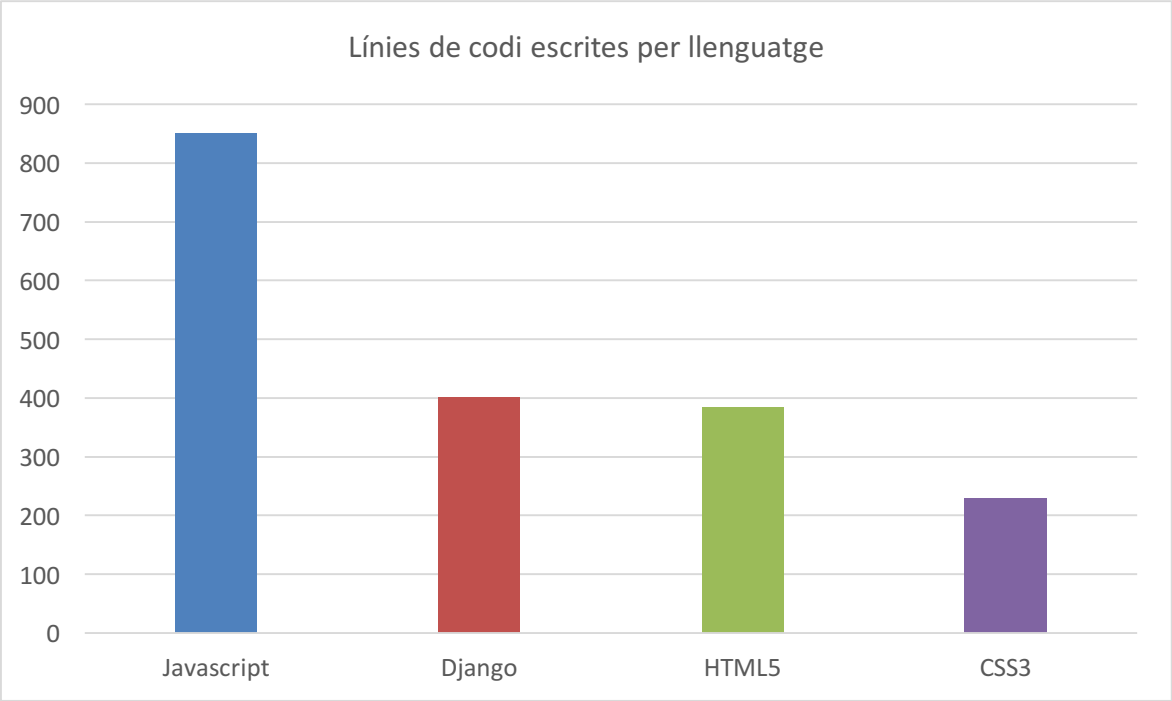
7.3.2 Tecnologies utilitzades

Degut a que la solució proposada és una interfície web i que l'entorn de desenvolupament es troba en un servidor remot Linux, el protocol usat per a la comunicació és HTTP i SSH és el protocol per accedir al servidor remot.

Per dur a terme la implementació s'han utilitzat en gran mesura llenguatges de programació web: Javascript i jQuery, HTML, CSS3 i Django. Recordem que per desenvolupar el servidor s'ha partit del mòdul Horizon d'OpenStack, que està escrit amb Django 1.8 (*framework* web de Python 2.7).



Gràfic 1: percentatges d'ús dels llenguatges de programació utilitzats.

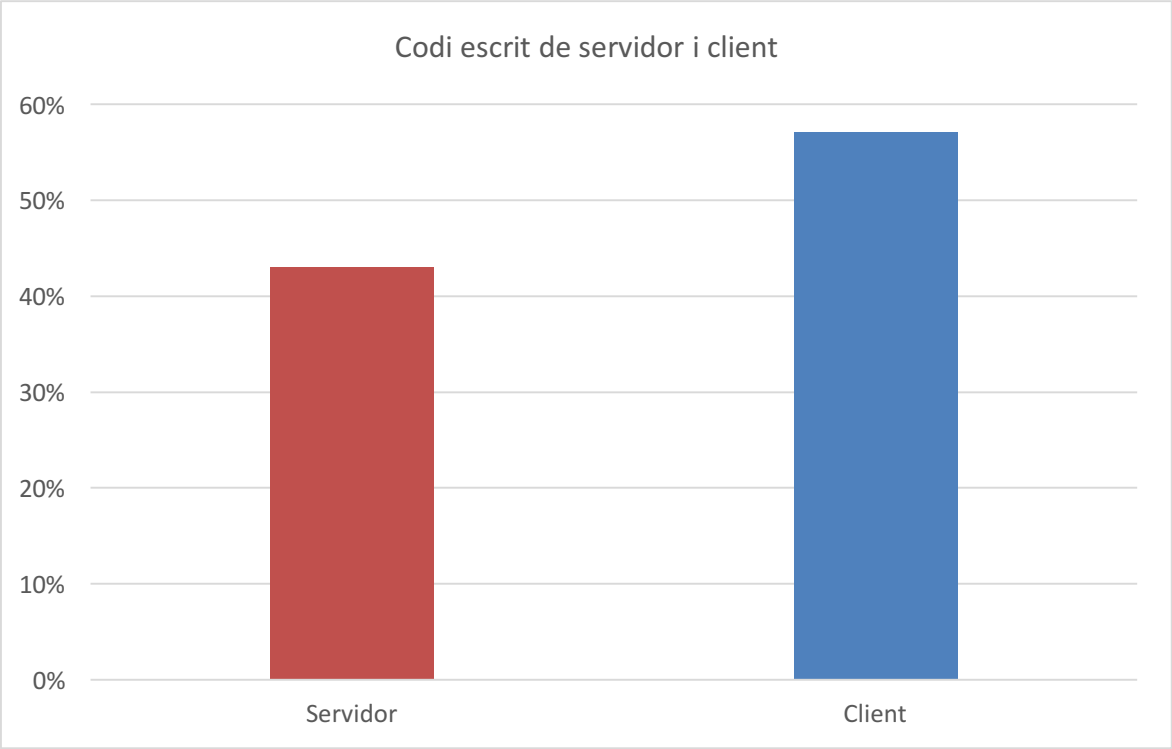


Gràfic 2: línies de codi escrites per llenguatge.

El client, popularment anomenat *front-end*, s'ha implementat amb Javascript i fa ús de llibreries de les següents llibreries:

Llibreria	**Descripció**	**Llicència**
Bootstrap	Més que una llibreria és un *framework* de CSS3 que Horizon ja inclou per defecte i ha permès crear fàcilment un disseny responsiu.	Apache 2.0
jQuery	*Framework* de Javascript que n'amplia les funcionalitats i permet modificar el HTML i CSS.	MIT
Vis.js	És una llibreria de visualització que permet crear representacions amb forma de graf i gestionar-ne les dades. S'ha utilitzat per crear la representació topològica (element 1).	Doble llicencia Apache 2.0 i MIT
Bootstrap-growl	Llibreria que utilitza jQuery i Bootstrap usada per afegir notificacions dinàmicament a la interfície (elements 32, 33 i 34).	MIT
Tablesorter	Llibreria que utilitza jQuery usada per organitzar les files de les taules dels apartat Virtual Nodes i Virtual Links (veure apartats 7.2.2 i 7.2.3) ascendentment o descendentment.	MIT

Taula 9: Llibreries utilitzades i descripció.



Gràfic 3: codi total escrit de back-end (servidor) i front-end (client).

Al gràfic 3 podem veure el total de codi que s'ha creat i a on s'executa. El servidor Horizon a més de tot el codi Python ha de processar els arxius HTML, ja que s'utilitzen els *templates* de Django per a generació de codi HTML. La resta de codi es correspon als arxius de Javascript i CSS que interpreta el navegador.

7.3.3 Arquitectura de la comunicació

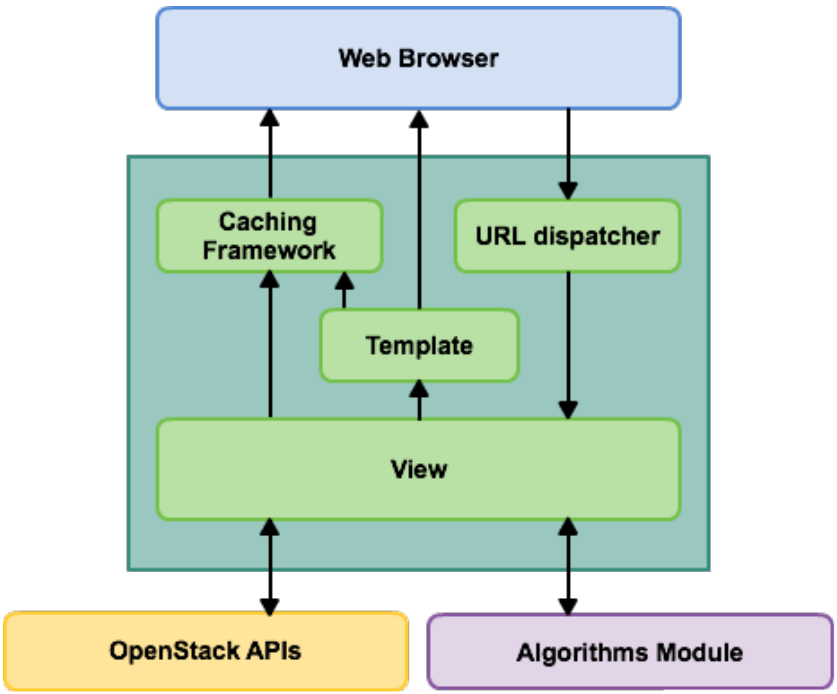


Figura 11: Visió global de la comunicació entre client i el servidor Horizon modificat. Referència: elaboració pròpia.

A continuació s'expliquen els passos d'un exemple típic del funcionament de la comunicació entre client i el servidor en base a la figura anterior:

1. Quan un navegador web realitza una petició al servidor, el URL dispatcher assigna la petició de pàgina a la funció de vista (View) especificada a urls.py i la crida. Si la memòria cau (cache) està activada, aquesta funció comprova si hi existeix una versió de la pàgina i pot estalviar-se passos posteriors retornant la pàgina en memòria cau.

2. La funció de la vista, normalment a views.py, executa l'acció demanada, que normalment implica usar les APIs d'OpenStack o del Mòdul d'Algoritmes per llegir informació i mostrar-la. També pot realitzar altres tasques.

3. Les crides a les APIs retornen les dades amb objectes Python o JSON a la funció de la vista. Tot i que en aquesta comunicació no s'usen bases de dades, en un altre escenari es podria donar el cas en que s'utilitzessin bases de dades relacionals que Django admet per defecte.

4. Després d'executar les tasques requerides, la vista retorna una resposta HTTP (després de passar les dades a través d'un *template*) cap el navegador web. Opcionalment, la vista pot guardar una versió de la resposta HTTP a la memòria cau per una duració de temps especificada.

Els *templates* de Django ofereixen una sintaxis fàcil d'usar per tal de construir la pàgina que l'usuari acaba veient. Per exemple, si l'usuari sol·licita veure la pàgina Virtual Nodes (veure apartat 7.2.2.2 Secció Virtual Nodes), el *template* inclou un bucle que afegeix una fila a la taula per cada node virtual.

7.3.4 Estructura i funció dels arxius

El servidor Horizon està conformat d'una composició complexa d'arxius de configuració, scripts i codi. Tanmateix, veurem la composició d'arxius i directoris de l'apartat Topology (veure apartat 7.2.2.1 Secció Topology) per a comprendre'n millor el funcionament:

```
.
├── __init__.py
├── panel.py
├── templates
│   └── topology
│       ├── add_instances.html
│       ├── add_node.html
│       ├── balloon.html
│       ├── cards.html
│       ├── dependencies.html
│       └── index.html
├── tests.py
├── urls.py
└── views.py
```

Figura 12: Stdout d'executar la comanda `$tree .` *al directori de l'apartat Topolgy.*

A la Figura 12 podem veure arxius dels que parlàvem a l'apartat anterior. L'arxiu urls.py (veure Figura 13) indica al URL dispatcher l'assignació de direccions URL a pàgines o accions definides a l'arxiu views.py (veure Figura 14). Tal assignació ens permet crear REST APIs d'una manera molt senzilla. Les funcions de la Figura 14 són cridades en executar els mètodes d'operació (veure apartat *7.2.4.2 Mètodes d'operació*) excepte la funció index, que renderitza la pàgina per defecte quan el navegador entra al lloc web.

Al directori templates/topology s'hi guarden tots els arxius HTML que el servidor processa abans d'enviar la resposta a l'usuari. L'arxiu panel.py és únicament de configuració del servidor on, entre d'altres coses, s'especifica el nom visible del l'apartat. Finalment, l'arxiu __init__.py no té impacte ja que no inclou cap instrucció.

```
urlpatterns = patterns(
    '',
    url(r'^$', views.IndexView.as_view(), name='index'),
    url(r'^get_vdc/$', views.get_vdc, name='get_vdc'),
    url(r'^delete_vdc/$', views.delete_vdc, name='delete_vdc'),
    url(r'^submit_vdc/$', views.submit_vdc, name='submit_vdc')
)
```

Figura 13: Fragment de codi de urls.py.

```
def submit_vdc(request):
    vdc = json.loads(request.POST['json'])
    vdc['tenantID' ] = tenant_id
    vdc = json.dumps(vdc)
    headers = {'content-type': 'application/json'}
    r = requests.post("http://127.0.0.1:12119/orchestrator/algorithms/vdc/",
            vdc, auth=HTTPBasicAuth('admin', 'password'), headers=headers)
    try:
        vdc = json.loads(r.text)
        vdc.pop("tenantID")
    except:
        vdc = r.text
    return HttpResponse(json.dumps(vdc))

def delete_vdc(request):
    r = requests.delete("http://127.0.0.1:12119/orchestrator/algorithms/vdc/?tenantID="+tenant_id,
            auth=HTTPBasicAuth('admin', 'password'))
    return HttpResponse(json.dumps(r.text))

def get_vdc(request):
    r = requests.get("http://127.0.0.1:12119/orchestrator/algorithms/vdc/?tenantID="+tenant_id,
            auth=HTTPBasicAuth('admin', 'password'))
    return HttpResponse(json.dumps(r.text))

def index(request):
    index = "index"
    return render(request, 'cosign/topology/index.html',context)
```

Figura 14: Fragment de codi de views.py.

El servidor també emmagatzema els arxius de codi Javascript, arxius de codi CSS i imatges, que poden ser utilitzats per la restes d'apartats (veure Figura 15). Els arxius són enviats al client en cas de que s'incloguin al template HTML.

La gran majoria dels arxius Javascript són per l'apartat Topology excepte els arxius jquery.tablesorter.pager.js i jquery.metadata.js. que són utilitzats a l'apartat Virtual Nodes i Virtual Links (veure Figura 15).

```
.
├── css
│   ├── buttons.css
│   ├── topology.css
│   └── vis.min.css
├── img
│   ├── acceptDeleteIcon.png
│   ├── addNodeIcon.png
│   ├── stack-gray.svg
│   ├── stack-green.svg
│   ├── stack-red.svg
│   └── zoomExtends.png
├── js
│   ├── add_instances.js
│   ├── balloon.js
│   ├── bootbox.min.js
│   ├── buttons.js
│   ├── jquery.bootstrap-growl.min.js
│   ├── jquery.metadata.js
│   ├── jquery.tablesorter.pager.js
│   ├── network.js
│   ├── others.js
│   ├── secure_input.js
│   ├── show_request.js
│   ├── upload_json.js
│   └── vis.min.js
└── scss
    └── cosign.scss
```

Figura 15: Arxius estàtics de CSS, Javascript i imatges.

7.3.5 Graf d'estats

El servidor requereix un *log-in* inicial d'usuari. Un cop s'han validat les seves credencials l'usuari pot navegar entre els diferents apartats. Per editar la instancia de VDC és necessari que l'usuari vagi a l'apartat Topology, del qual en pot marxar sortint de la sessió d'usuari o navegant a un altre apartat.

La instancia VDC en tot moment pot ser editada encara que sigui desplegada (deployed) o s'esborrin els recursos (cleared). Per confirmar els canvis és necessari desplegar la petició (deploy). No requereix una operació de creació, sinó que la instancia VDC estarà buida inicialment esperant a rebre una petició de recursos per a ser instanciada i en cas de rebre una nova petició, s'esborrarà i s'instanciarà de nou. A la següent figura s'indiquen els possibles estats:

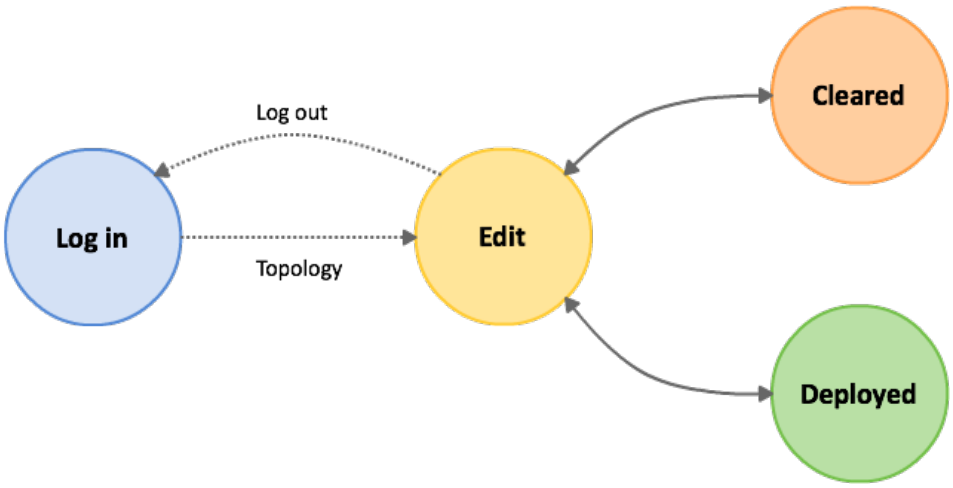


Figura 16: graf d'estats de la solució.

8. Validació

8.1 Funcionament

Amb l'objectiu de validar el bon funcionament de la solució es durà a terme l'execució de diferents experiments i en diferents navegadors per demostrar els següents criteris:

1. **Fiabilitat**: el funcionament és correcte en un període de temps arbitrari.

2. **Repetibilitat**: repetir els mateixos experiments donen sempre els mateixos resultats.

3. **Reproductibilitat**: repetir els mateixos experiments en diferents plataformes donen el mateix resultat.

Es provaran els mètodes d'operació mitjançant la REST del servidor des de la interfície un cop hem fet *log-in*. S'obviaran els mètodes de configuració manuals ja que no es poden demostrar si no és amb un vídeo. Per tant, s'utilitzaran arxius de text que descriguin una petició VDC mitjançant el mètode de lectura de petició i posteriorment se'n sol·licitarà el desplegament. A partir d'ara anomenarem experiments a aquests arxius de text.

Els mètodes de la REST API del servidor utilitzen autentificació HTTPBasicAuth per comunicar-se amb el Mòdul d'Algoritmes i recordem que hi afegeixen l'identificador tenant ID. Aquest mètodes són els següents:

POST /cosign/submit_vdc/	
Cos de la petició	Objecte JSON que descriu una petició VDC correctament.
Descripció: envia la petició VDC al Mòdul d'Algoritmes amb una petició HTTP POST.	
Resposta d'èxit: vdc registered(201)	
Respostes d'error: tenantid required(400), all vlink fields required(400), all vm fields required(400), all vnode fields required(400), vnode parameters of vlink is not well defined(400), error(500)	

Taula 10: descripció del mètode submit_vdc de la REST API del servidor.

POST /cosign/delete_vdc/	
Cos de la petició	-
Descripció: envia una petició HTTP DELETE al Mòdul d'Algoritmes per esborrar els recursos de la instancia VDC.	
Resposta d'èxit: deleted vdc(200)	
Respostes d'error: tenantid not found(404), tenantid required(400), error(500)	

Taula 11: descripció del mètode delete_vdc de la REST API del servidor.

Encara que no es s'utilitzarà exhaustivament durant la validació, hi ha un tercer mètode de la REST API que s'executa automàticament:

GET /cosign/get_vdc/	
Cos de la resposta	Objecte JSON que descriu la instancia VDC.
Descripció: envia una petició HTTP GET al Mòdul d'Algoritmes per obtenir la instancia	
Resposta d'èxit: 200 OK	
Respostes d'error: tenantid not found(404), tenantid required(400), error(500)	

Taula 12: descripció del mètode get_vdc de la REST API del servidor.

A continuació es descriuen els experiments:

Nom de l'experiment	**Descripció**
Cas base	Petició VDC de dos nodes virtuals, amb un enllaç virtual i una màquina virtual a cada node.
Cas mig	Petició VDC de cinc nodes virtuals amb dos màquines virtuals en cada node i quatre enllaços virtuals.
Cas extrem	Petició VDC de vint nodes virtuals amb cinc màquines virtuals en cada node i vint-i-cinc enllaços virtuals.

Taula 13: descripció dels experiments.

Per cada execució d'un experiment es mostrarà la petició i la resposta HTTP a les consoles de desenvolupadors de Google Chrome i de Mozilla Firefox, si els resultants són els mateixos podem assumir que la reproductibilitat és correcte.

Amb l'objectiu de comprovar la fiabilitat del sistema, es deixarà el servidor executant-se des del dilluns 20 de juny i s'executaran els experiments el dimecres 22 de juny en horari laboral.

A la primera execució del primer experiment també es mostraran elements de la interfície. Per últim, s'executarà dos cops cada experiment en cada navegador per assegurar la repetibilitat i reproductibilitat del sistema d'acord amb els requeriments.

8.1.1 Cas base

En començar l'experiment es té la petició de recursos buida. S'usarà un arxiu que descriu aquest experiment amb un objecte JSON per ser llegit per el client Javascript i per últim es desplegarà.

Execució 1 amb Firefox

Després de fer log-in el servidor processa la petició GET dels arxius correctament, que són enviats al client i aquest els interpreta i mostra a l'usuari. En aquest moment el sistema està preparat per dur a terme la primera execució.

Es procedeix a llegir l'arxiu del cas base i la interfície ens respon amb una notificació indicant-nos que la petició està preparada per ser desplegada.

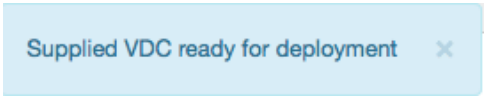


Figura 17: notificació informativa de que la petició està llesta per ser desplegada.

En aquest moment la representació de la topologia té aquest aspecte:

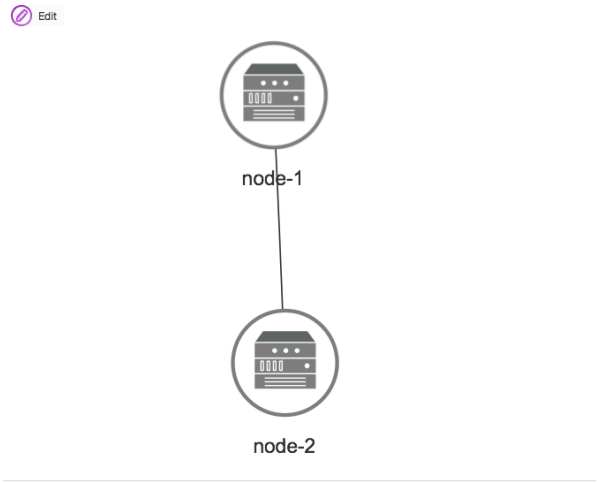


Figura 18: representació topològica del cas base abans de ser desplegat.

Es procedeix a desplegar la petició VDC mitjançant el botó Deploy. Visualment l'usuari observa una notificació d'èxit, un efecte de *highlight* a sobre la representació topològica i un canvi de colors de la mateixa. Tanmateix, els aspectes visuals no són rellevants per aquesta validació i per tant només els veurem en aquesta primera execució.

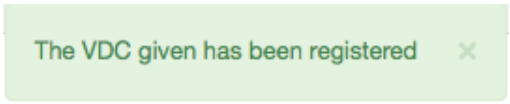


Figura 19: notificació d'èxit de que s'ha desplegat el VDC correctament.

Aquest és l'aspecte de la representació de la topologia en aquest moment:

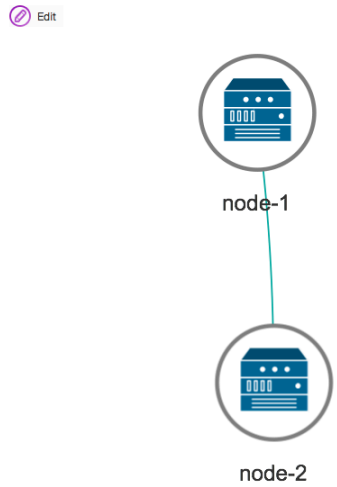


Figura 20: representació topològica del cas base un cop ha sigut desplegada.

Amb la consola de desenvolupadors de Firefox podem veure en els detalls de la petició HTTP que la resposta és 200 OK (veure figura 21).

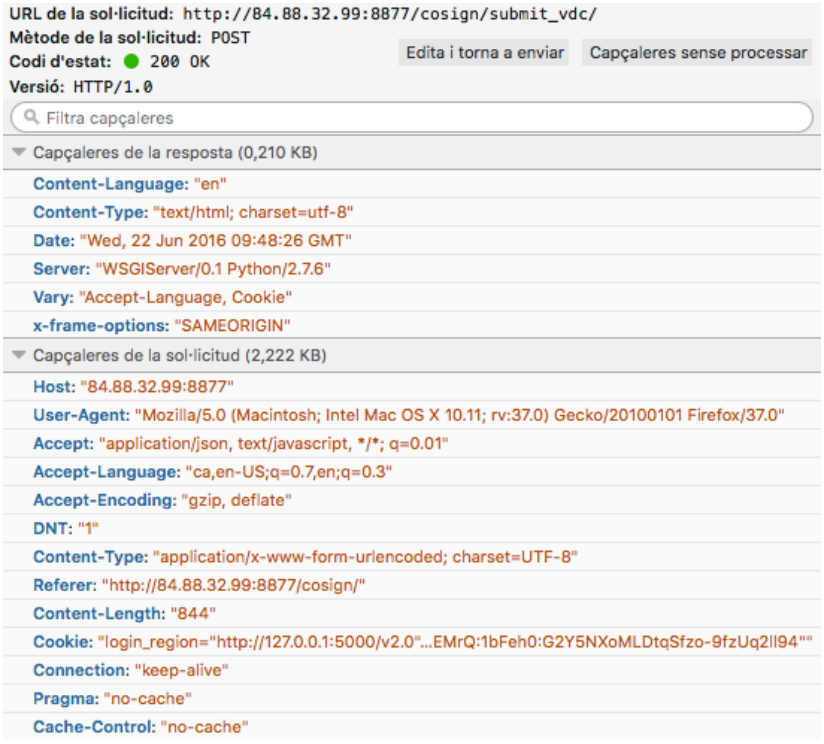


Figura 21: petició HTTP de l'operació Deploy de la primera execució del cas base amb Firefox.

Per finalitzar, abans de procedir amb la següent execució fem clic al botó Unstack per esborrar la instancia VDC. Degut a que la resposta és 200 OK (veure figura 22 a continuació) es mostra una confirmació (veure element 33 de 7.2.2.4 Diàlegs, formularis i notificacions).

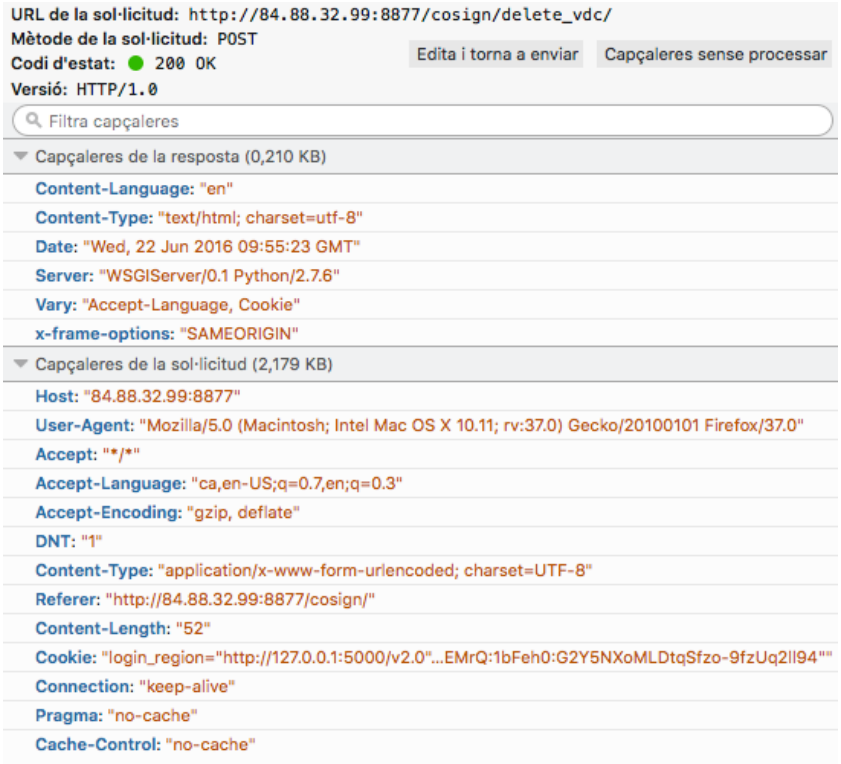


Figura 22: petició HTTP de l'operació Unstack amb Firefox.

Execució 2 amb Firefox

De manera anàloga a la primera execució, es repetiran tots els passos però d'ara en endavant s'obviaran els elements gràfics de la interfície. Degut a que anteriorment hem executat l'operació Unstack, podem procedir amb la segona execució llegint de nou la petició VDC del cas base.

La interfície ens notifica que la petició està preparada per ser desplegada i procedim a desplegar-la. A continuació podem veure la petició HTTP pertinent i que la resposta era l'esperada (veure figura 23). El resultat és el mateix.

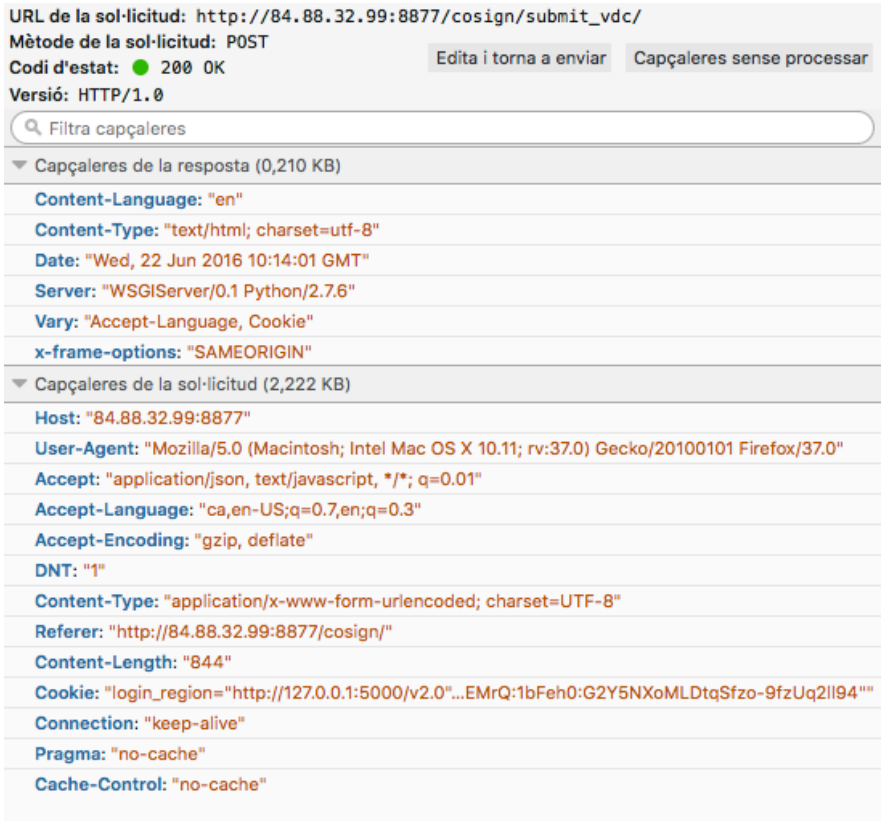


Figura 23: petició HTTP de l'operació Deploy de la segona execució del cas base amb Firefox.

Execució 1 amb Chrome

Es repetirà el mateix procediment de les execucions anteriors obviant els elements gràfics i centrant-nos amb la resposta de la petició HTTP. Abans de començar recordem que és necessari fer *log-in* ja que estem amb un altre navegador.

Un cop hem entrat i hem esborrat la instancia VDC mitjançant la operació Unstack, procedim a realitzar el mateix procediment de l'apartat anterior.

Un cop s'ha llegit la petició VDC del arxiu del cas base, s'executa la operació Deploy i s'analitza la petició HTTP (veure figura 24).

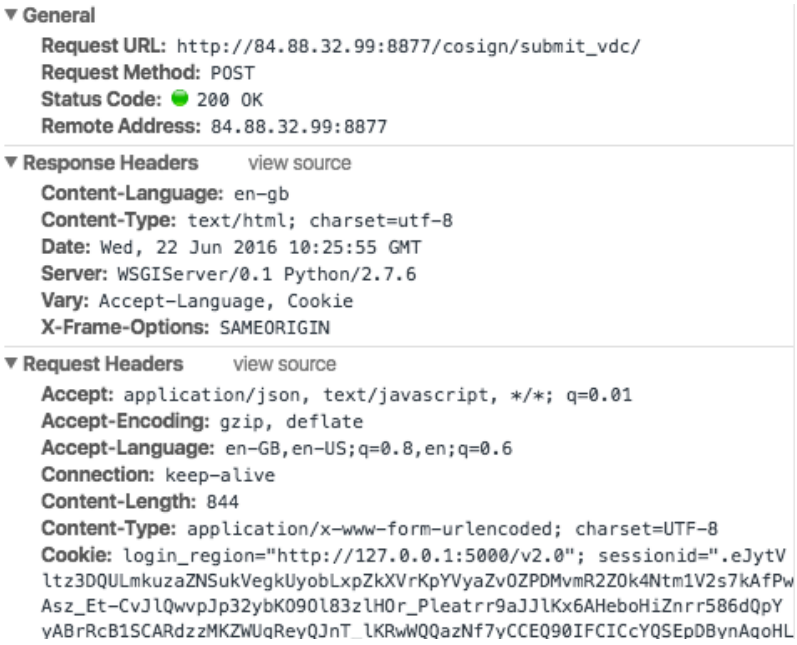


Figura 24: petició HTTP de l'operació Deploy de la primera execució del cas base amb Chrome.

Execució 2 amb Chrome

De manera anàloga a l'execució anterior, s'executen tots els mateixos passos excepte el log-in inicial i abans de començar s'executa la operació Unstack per executar l'experiment de nou. Observem que el resultat és l'esperat a la figura 25.

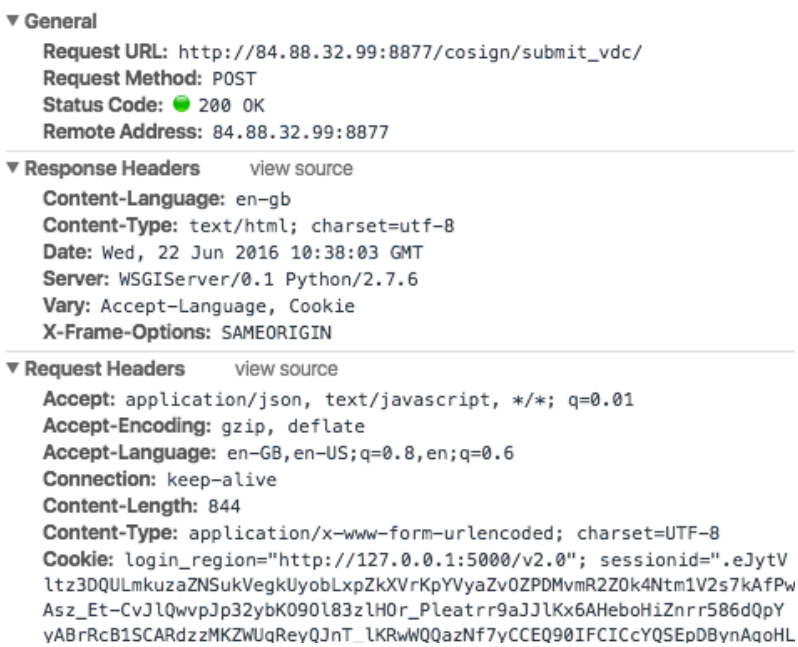


Figura 25:petició HTTP de l'operació Deploy de la segona execució del cas base amb Chrome.

Després d'executar el primer experiment dos cops en cada navegador i analitzar les peticions HTTP podem assegurar que el funcionament és l'esperat. A l'apartat següent incrementarem la dificultat amb un experiment de més complexitat.

8.1.2 Cas mig

Abans de començar s'esborrarà la instancia VDC. S'usarà un arxiu que descriu aquest experiment amb un objecte JSON per ser llegit per el client Javascript i per últim es desplegarà.

La representació topològica d'aquest cas és la següent:

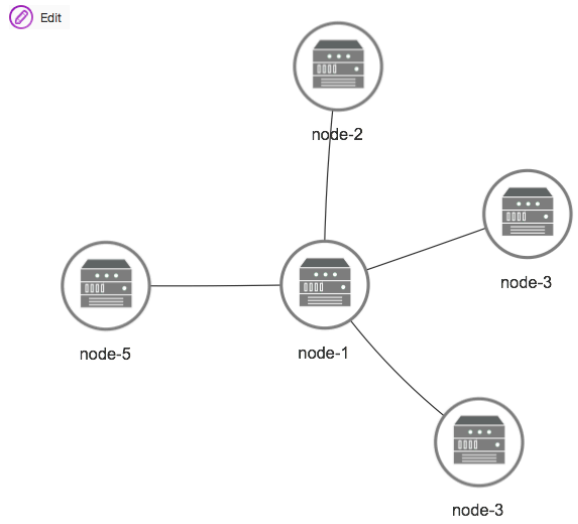


Figura 26: Representació topològica del cas mig abans de ser desplegat.

Execució 1 amb Firefox

Executem l'operació Deploy i comprovem que el resultat és correcte a la petició HTTP (veure Figura 27).

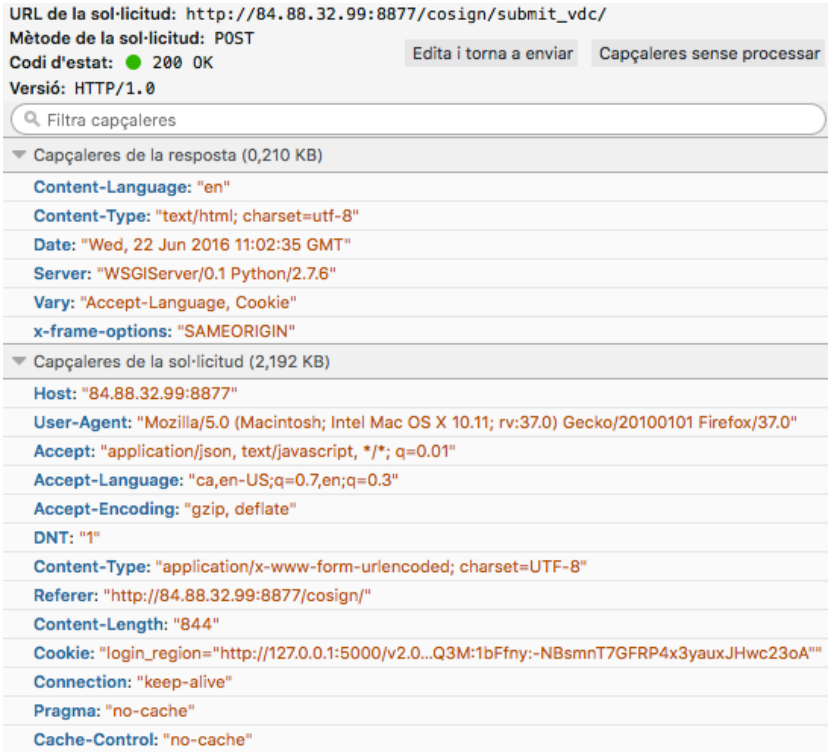


Figura 27: petició HTTP de l'operació Deploy de la primera execució del cas mig amb Firefox.

Execució 2 amb Firefox

Esborrem la instancia VDC i executem el mateix procediment. Analitzem la petició HTTP i veiem que el resultat és correcte (veure figura 28).

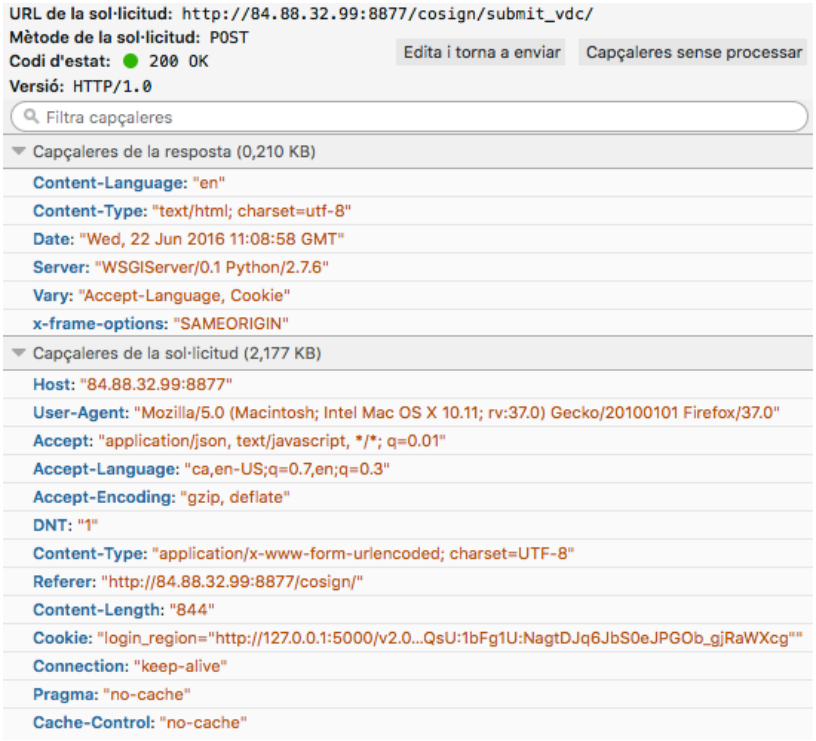


Figura 28: petició HTTP de l'operació Deploy de la segona execució del cas mig amb Firefox.

Execució 1 amb Chrome

Canviem al navegador Chrome, esborrem la instancia VDC i executem el mateix procediment. Finalment, analitzem la petició HTTP i veiem que el resultat és correcte (veure figura 29).

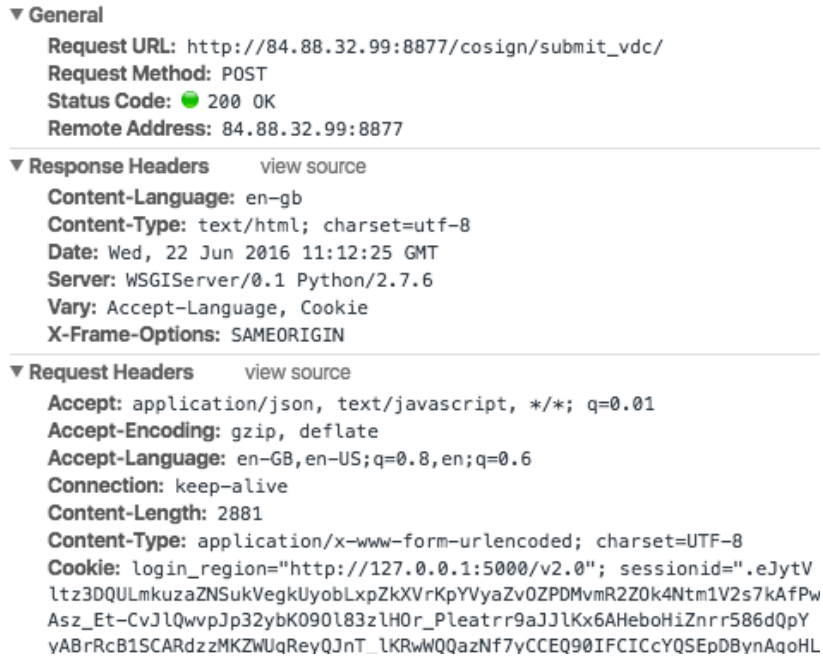


Figura 29: petició HTTP de l'operació Deploy de la primera execució del cas mig amb Chrome.

Execució 2 amb Chrome

Esborrem la instancia VDC, executem el mateix procediment i veiem que el resultat és l'esperat (veure figura 30).

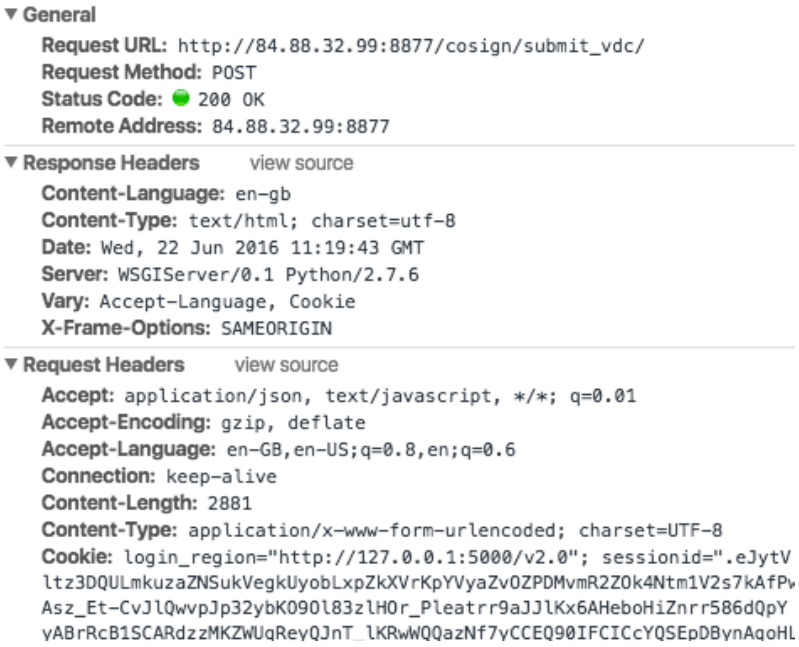


Figura 30: petició HTTP de l'operació Deploy de la segona execució del cas mig amb Chrome.

Havent executat el segon experiment satisfactòriament podem passar al següent i últim experiment, on es posarà a prova la interfície i el servidor per processar una petició JSON de 870 línies.

8.1.3 Cas extrem

S'executarà un procediment de manera anàloga als altres experiments i s'analitzaran els resultats. La representació topològica d'aquest cas és la següent:

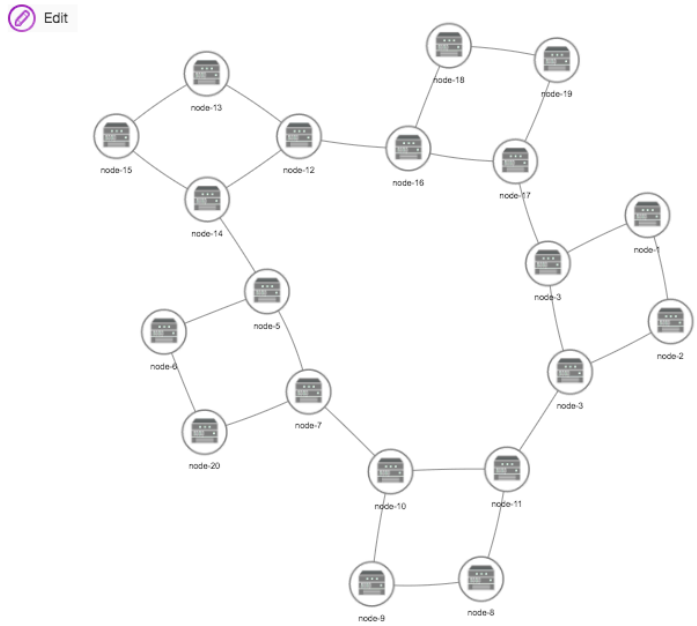


Figura 31: Representació topològica del cas extrem.

Execució 1 amb Firefox

Executem el mateix procediment i observem que el resultat és l'esperat (veure Figura 32).

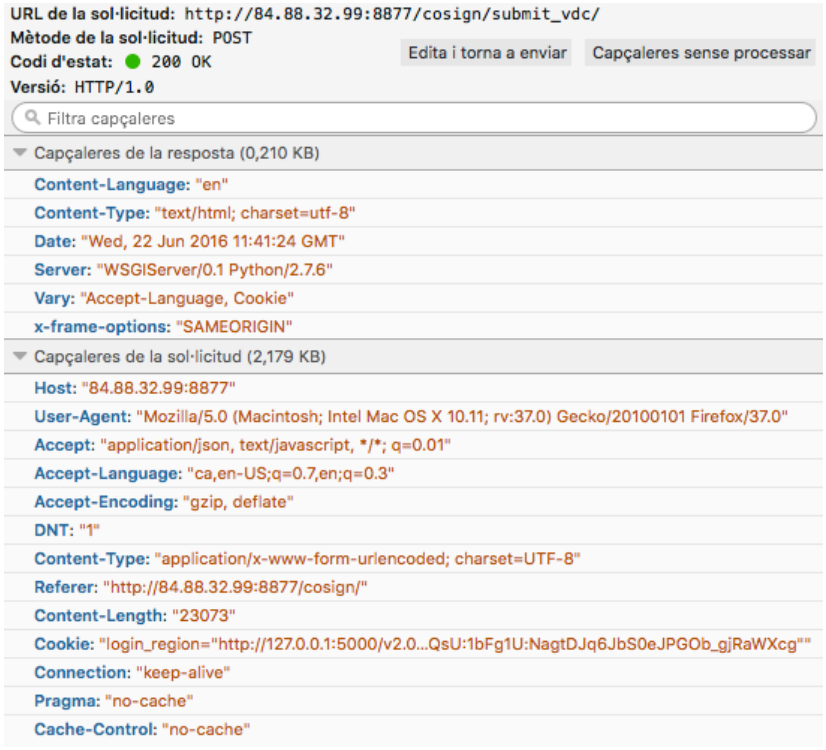


Figura 32: petició HTTP de l'operació Deploy de la primera execució del cas extrem amb Firefox.

Execució 2 amb Firefox

El resultat de la segona execució amb Firefox també és l'esperat (veure Figura 33).

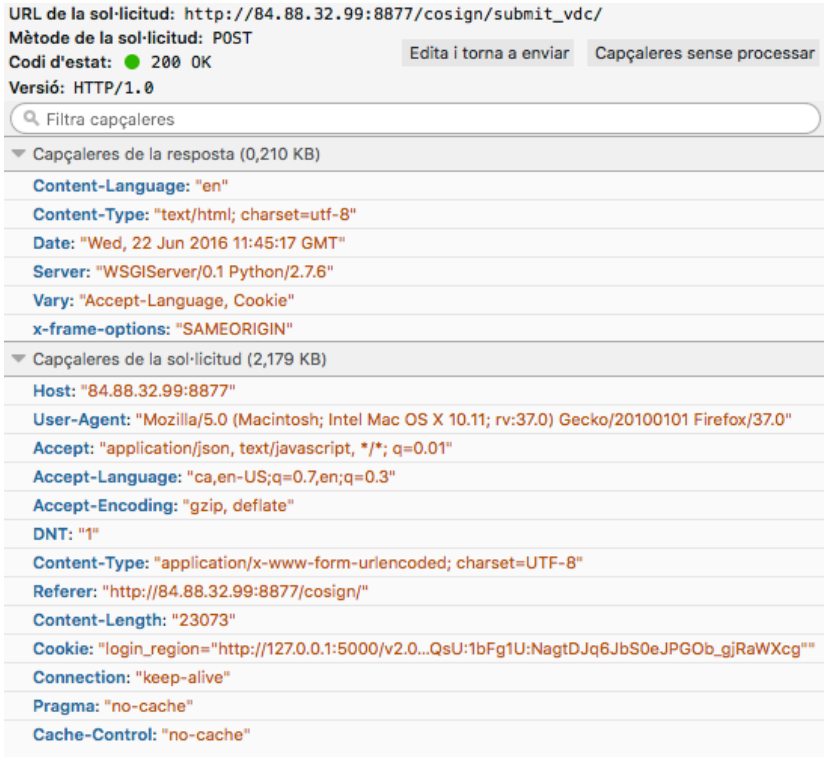


Figura 33: petició HTTP de l'operació Deploy de la segona execució del cas extrem amb Firefox.

Execució 1 amb Chrome

Anàlogament als altres casos, canviem al navegador Chrome, esborrem els recursos VDC i executem el mateix procediment. Finalment, analitzem la petició HTTP i veiem que el resultat és correcte (veure figura 24).

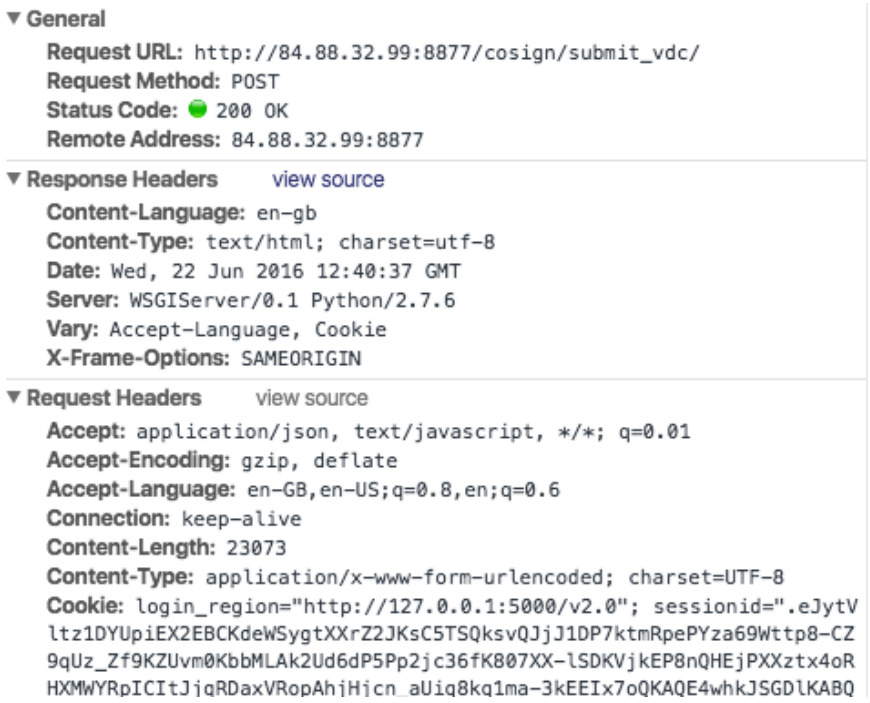


Figura 34: petició HTTP de l'operació Deploy de la primera execució del cas extrem amb Chrome.

Execució 2 amb Chrome

Després d'esborrar els recursos VDC i executar l'operació Deploy inspeccionem la petició HTTP i veiem que el resultat torna a ser correcte.

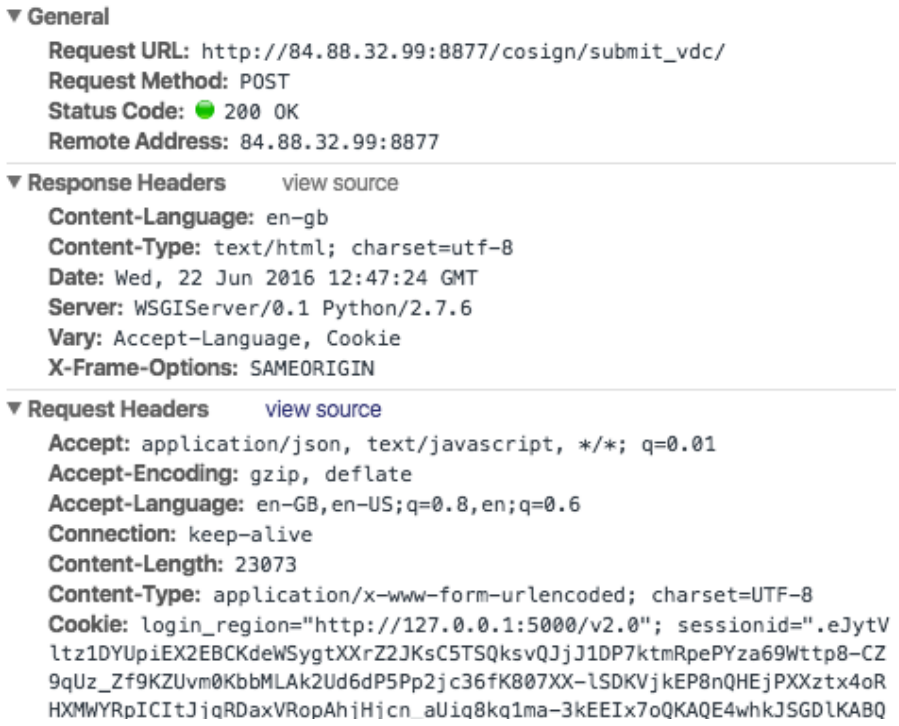


Figura 35: petició HTTP de l'operació Deploy de la segona execució del cas extrem amb Chrome

Hem conclòs les execucions dels tres experiments a Firefox i Chrome en un període de temps des de les 10:14 fins a les 12:47 satisfactòriament i amb el mateix resultat. Per tant, podem confirmar el bon funcionament de la interfície amb els criteris definits al principi d'aquest apartat: fiabilitat, repetibilitat i reproductibilitat.

8.2 Seguretat

Els experiments anteriors assumeixen un ús correcte i no fraudulent de la interfície. Per tant, no s'ha validat que sigui suficientment segura. Amb l'objectiu de comprovar la seguretat de la interfície es buscaran vulnerabilitats dins dels atacs informàtics més comuns.

8.2.1 XSS

XSS (Cross-site Scripting) [18] és una de les vulnerabilitats més comunes dels llocs web a Internet. Consisteix en introduir codi Javascript a través de formularis per modificar el comportament del lloc web.

S'han detectat els formularis o accions que poden ser víctimes d'atacs XSS. A continuació s'analitzarà el seu comportament quan s'hi introdueix el següent codi Javascript:

```
<script> alert("XSS Attack!!") </script>
```

- Formulari d'afegir node virtual: si s'introdueix el codi maliciós com a nom del node, s'afegeix el node virtual amb normalitat però sense nom. Els camps per connectar-lo a amb altres enllaços són per força números.

- Formulari d'afegir enllaç virtual: si s'introdueix el codi maliciós com a l'ample de banda del node l'operació és automàticament cancel·lada i es mostra una notificació dient que l'entrada ha de ser un número.

- Formulari d'afegir màquines virtuals: només es pot introduir el codi maliciós al nom de la màquina virtual. Si es confirma el formulari amb codi maliciós les màquines virtuals s'afegeixen a la petició però sense nom.

- Formulari d'editar nom de node virtual: si s'introdueix el codi maliciós com a nom nou del node s'accepta l'operació però el nom quedarà buit.

- Lectura de petició VDC mitjançant un fitxer: en cas que l'objecte JSON escrit a l'arxiu contingui el codi maliciós, no s'accepta l'operació i s'avisa a l'usuari mitjançant una notificació que el contingut del fitxer no és vàlid.

Encara que hi ha camps dels formularis on només hi podem escriure números, aquesta limitació es pot eliminar mitjançant la consola de desenvolupadors. Per aquest motiu, també es realitza una comprovació de seguretat eliminant qualsevol codi trobat a tots els camps dels formularis.

8.2.2 CSRF

CSRF (Cross-site Request Forgery) [19] és un tipus d'*exploit* maliciós d'un lloc web en el que un accions no autoritzades són enviades per un usuari o servidor externs en el que el servidor confia. En el cas d'aquesta solució podríem suposar que algun codi maliciós extern intenta realitzar accions d'operació de la instancia VDC. Tanmateix, això no és possible ja que Django proporciona protecció envers aquesta classe d'atacs mitjançant un *token* anomenat "csrf_token" [20] que és necessari per realitzar peticions al servidor i només es pot obtenir un cop s'han validat les credencials d'usuari.

8.2.3 Injecció SQL

La injecció SQL [21] és un atac en el que s'insereix codi SQL maliciós als formularis per a que després el servidor SQL l'executi i retorni informació. Afortunadament, no s'han utilitzat bases de dades més que les existents al mòdul Keystone d'OpenStack que ja estan protegides. Per tant, el servidor no és vulnerable a aquest atac.

8.2 Usabilitat

Donat que no resultava factible realitzar un informe d'usabilitat que requeriria un estudi amb diferents usuaris, el desenvolupament s'ha realitzat sota una sèrie de criteris per procurar garantir la usabilitat i bona experiència de l'usuari. Els criteris elegits són els que s'exposen a continuació:

- S'ha evitat l'ús de *banners* dinàmics al web
- S'ha evitat sobrecarregar visualment les pàgines.
- S'ha assolit un temps de càrrega del lloc web ràpid.
- La navegació és consistent i simple.
- Es demana confirmació d'operacions no reversibles.
- Es notifica a l'usuari del resultat de les operacions o de possibles errors.
- El disseny és responsiu, preparat per ordinadors d'escriptori i tauletes mòbils.
- El disseny és estètic i minimalista.
- El tamany dels botons és adequat.
- Els colors utilitzats canvien en funció de la classe d'error, notificació o acció.
- S'evita que l'usuari cometi errors com proporcionar una petició VDC no vàlida mitjançant un arxiu de text.

9. Regulacions aplicables

No es guarden dades personals d'usuari i per tant, no és necessari realitzar cap acció que requereixi el compliment legal del tractament de les dades. Tanmateix, sí que s'utilitzen llibreries amb diferents llicències.

Les llibreries emprades estan llicenciades amb llicències de caràcter obert com MIT o Apache 2.0. La llicencia MIT permet usar, copia, modificar, distribuir i/o vendre de copies del software sempre que es mantingui la capçalera original als arxius de codi. En canvi, la llicencia Apache 2.0 també dona plena llibertat d'ús però requereix un arxiu LICENSE amb la còpia de la llicencia i un arxiu NOTICE que indica si hi ha avisos obligatoris del software present, al directori principal de projecte. S'han respectat ambdues condicions, tant de les llicencies MIT com les llicencies Apache 2.0.

10. Conclusions i futur treball

Després d'haver acabat el projecte, podem arribar a les següents conclusions:

- El futur dels centres de dades passa necessàriament per utilitzar el nou paradigma de gestió de xarxes SDN i per crear nous serveis IaaS.

- OpenStack és la plataforma *open-source* referent en la creació de serveis IaaS.

- Les eines comercials actuals no solucionen el problema que planteja el cas d'ús Virtual Data Center. Només permeten virtualitzar recursos del servidor i per tant no poden garantir una connectivitat amb l'ample de banda entre els diferents nodes. A més, manquen de possibilitats de gestió i personalització dels recursos.

- S'ha desenvolupat una interfície gràfica web per el cas d'ús Virtual Data Center satisfactòriament. La solució implementada permet agrupar les màquines virtuals en nodes virtuals i dotar els nodes de connectivitat entre ells amb un ample de banda especificat per l'usuari.

- Al començament del projecte es tenia la intenció d'implementar un historial de les peticions VDC. No obstant, es va haver de descartar. Aleshores, enlloc d'implementar l'historial es va decidir afegir dues funcionalitats: lectura de peticions VDC en arxius i copiar la petició VDC al porta retalls de l'usuari.

- S'ha validat que la interfície implementada és fiable i segura mitjançant diferents experiments aplicats en diferents entorns. En concret, s'ha validat amb els navegadors Mozilla Firefox i Google Chrome i s'han solucionat els possibles errors de seguretat XSS, SQL Injection i CSRF.

- Haver separat la comunicació del client Javascript del Mòdul d'Algoritmes amb el servidor Horizon com intermediari ha fet possible aïllar les dades sensibles i eliminar la possibilitat d'ús fraudulent de les seves operacions.

- Durant el desenvolupament s'ha garantit la usabilitat de la interfície i l'experiència d'usuari. Al no disposar de recursos per fer un estudi d'usabilitat, s'han seguit els criteris d'usabilitat i experiència més coneguts.

- S'ha evitat utilitzar funcionalitats ja programades realitzant una programació pròpia dels esdeveniments i peticions. Tal decisió ha implicat observar un millor rendiment en els apartats de la solució que ens els existents d'Horizon.

- Finalment, s'ha aconseguit arribar als objectius inicials amb la creació d'una interfície que compleix amb tots i cadascun els requeriments.

Encara que la solució ha sigut desenvolupada correctament, creiem que hi ha varies possibilitats de millora degut a la magnitud del projecte. A continuació s'exposen possibles millores:

- Si el Mòdul d'Algoritmes ho admet al futur, permetre a l'usuari especificar més detalls de les màquines virtuals com aplicar-hi *Security Groups* (regles Firewall) o permetre que s'iniciïn a partir d'un *snapshot*.

- Si el Mòdul d'Algoritmes ho admet al futur, limitar l'ús de recursos de xarxa disponibles (els recursos de computació ja poden ser limitats) i mostrar a l'usuari l'ús actual dels recursos disponibles.

- En fer clic a un node virtual a la topologia, oferir a l'usuari veure els detalls del node virtual a una pàgina sencera dedicada a aquell node.

- Traduir la interfície a més idiomes i mostrar-la amb l'idioma corresponent del navegador de l'usuari.

- Afegir una acció per esborrar totes les màquines virtuals dins un node virtual de cop.

- Afegir una acció per esborrar tots els enllaços virtuals connectats a un node virtual de cop.

11. Referències

[1] The NIST Definition of Cloud Computing
http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

[2] Cosign | Advanced Optical Hardware and SDN for Next-Generation Data Centres
http://www.fp7-cosign.eu/

[3] Deliverable D4.2 COSIGN Orchestrator low level architecture and prototype design
http://www.fp7-cosign.eu/wp-content/uploads/2015/08/D4-2-v12-final_submitted.pdf

[4] https://www.openstack.org/

[5] https://www.djangoproject.com/

[6] http://www.gartner.com/newsroom/id/3136417

[7] https://en.wikipedia.org/wiki/Software-defined_networking

[8]http://www.ey.com/Publication/vwLUAssets/Future_network_operations/$FILE/Future_network_operations.pdf

[9] https://es.wikipedia.org/wiki/Redes_definidas_por_software

[10] https://www.opendaylight.org/

[11] http://searchcloudprovider.techtarget.com/feature/Cloud-service-management-and-cloud-monitoring-for-providers-A-primer

[12] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html

[13] https://www.upguard.com/articles/aws-vs-rackspace-let-the-cloudwars-begin

[14] http://www.yutaaoki.com/blog/rackspace-vs-amazon-aws-ec2-performance-comparison

[15] http://www.tusalario.es/main/salario/comparatusalario

[16] http://www.interoute.es/what-iaas

[17] http://www.gizmag.com/fujitsu-low-power-optical-switch/16901/

[18] https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

[19] https://www.owasp.org/index.php/CSRF

[20] https://docs.djangoproject.com/en/1.9/ref/csrf/

[21] https://www.owasp.org/index.php/SQL_Injection