TREBALL DE FI DE CARRERA

Títol del TFC: Abiquo Provisioning Portal

Titulació: Enginyeria superior en Informàtica (pla 2003)

Autor: David López Padilla

Director: Xavier Fernández

Ponent: Jordi Nin Guerrero

Data: 21 de Gener de 2013

Resum

Abiquo Provisioning Portal neix amb la idea de **millorar la gestió dels entorns clouds privats** a través d'un servei cloud que treballarà sobre una infraestructura com a servei (IaaS) i gestionarà centres de dades de forma dinàmica i escalable, permetent que els client puguin auto-gestionar les seves plataformes **de forma senzilla**, avançant un pas més **per oferir serveis cloud a usuaris amb necessitats concretes i coneixements limitats**.

Gràcies a aquest servei es facilitarà **l'aprovisionament de infraestructures**, configurades i llestes per treballar en producció, **i del software personalitzat** que es requereixi. Els propis clients seran qui gestionaran els seus entorns, i podran modificar els recursos de les màquines desplegades sota demanda, segons necessitats puntuals. D'aquesta manera es podran reduir costos en manteniment i posta en marxa, ja què en cas de necessitar més recursos l'ampliació es realitzarà a través del portal, i gràcies a què la infraestructura s'administrarà de forma dinàmica es podrà optar per pagar només pels recursos que s'estiguin utilitzant, l'anomenat ***pay as you go***, deixant de banda els antics models de negoci que cobren per mensualitats, sense tenir en compte si els recursos s'estan aprofitant o no.

Per tant, **el benefici serà per les dues parts implicades**: **els consumidors** poden contractar serveis per un període de temps curt i pagaran només pel temps que tinguin les màquines virtualitzades**, i els centres de dades** podran aprofitar els recursos de forma més eficient.

A la Maria Josep, per estar sempre al meu costat.

Als meus pares, Imma i Joan, pel seu suport i comprensió.

A la Janina, per fer-me riure sempre que ho necessito.

A en Xavi, per integrar-me en la seva empresa.

A en Jordi, per la seva ajuda i orientació.

Gràcies a tots.

ÍNDEX

INTRODUCCIÓ

Abiquo és una empresa especialitzada en software destinat a implantar Infraestructures com a servei en centres de dades. Actualment el producte es troba en producció en grans empreses arreu del món les quals poden oferir entorns virtualitzats sota demanda altament escalables, com les que descriurem a continuació:

Claranet

Empresa internacional que es troba oferint serveis cloud a través de la plataforma d'***Abiquo*** utilitzant desplegaments multi-regió en 6 països diferents.

Interoute

Empresa internacional que es troba oferint serveis cloud a través de la plataforma d'***Abiquo*** utilitzant desplegaments multi-regió també en 6 països diferents.

Bluefire

Empresa internacional que gestiona 8 centres de dades per oferir serveis cloud a través d'***Abiquo***.

Abiquo Provisioning Portal neix amb la idea de millorar la gestió dels entorns clouds privats a través d'un servei cloud que treballarà sobre una infraestructura com a servei (IaaS) i gestionarà centres de dades de forma dinàmica i escalable, permetent que els client puguin auto-gestionar les seves plataformes de forma senzilla, avançant un pas més per oferir serveis cloud a usuaris amb necessitats concretes i coneixements limitats.

Gràcies a aquest servei es facilitarà l'aprovisionament de infraestructures, configurades i llestes per treballar en producció, i del software personalitzat que es requereixi. Els propis clients seran qui gestionaran els seus entorns, i podran modificar els recursos de les màquines desplegades sota demanda, segons necessitats puntuals. D'aquesta manera es podran reduir costos en manteniment i posta en marxa, ja què en cas de necessitar més recursos l'ampliació es realitzarà a través del portal, i gràcies a què la infraestructura s'administrarà de forma dinàmica es podrà optar per pagar només pels recursos que s'estiguin utilitzant, l'anomenat ***pay as you go***, deixant de banda

els antics models de negoci que cobren per mensualitats, sense tenir en compte si els recursos s'estan aprofitant o no.

Per tant, el benefici serà per les dues parts implicades: els consumidors poden contractar serveis per un període de temps curt i pagaran només pel temps que tinguin les màquines virtualitzades, i els centres de dades podran aprofitar els recursos de forma més eficient.

CAPÍTOL 1. OBJECTIUS DEL PROJECTE

L'objectiu del projecte és la creació d'un portal cloud que permeti gestionar els serveis de IaaS a través de la plataforma creada per Abiquo. Amb aquest servei s'espera que es puguin cobrir les noves necessitats detectades pels clients de la plataforma, sobretot els proveïdors de serveis, els quals busquen noves línies de negoci que podrien ser explotades mitjançant el portal i facilitaria el procés d'auto-aprovisionament de infraestructures configurades i llestes per treballar en producció, i del software personalitzat que es requereixi per part de clients.
A més, es vol simplificar l'accés als recursos físics mitjançant connexions VNC i es vol fer accessible a través de qualsevol dispositiu mitjançant la potència del HTML5 combinada amb websockets. Per fer-ho, es desenvoluparan dues aplicacions SaaS, una que permeti accedir des de qualsevol navegador però optimitzada per pantalles de sobretaula o portàtil i l'altra també accessible des de navegador però optimitzada per dispositius mòbils.
En definitiva, aconseguir que un usuari amb necessitats concretes i coneixements limitats sigui capaç d'accedir a un catàleg d'ofertes pre-configurades i, en pocs clics, pugui desplegar infraestructures privades, auto-aprovisionar-se del programari que necessiti i utilitzar-lo, accedir a les màquines virtuals personalitzades o bé eliminar-les i deixar de pagar pel servei utilitzat, seguint el concepte del *pay as you go.*

CAPÍTOL 2. ESTAT DE L'ART

2.1. Estat de l'art en *cloud computing*

En aquesta secció descriurem els diferents estats de l'art que es realitzaran, on s'estudiarà les diverses solucions adoptades a productes semblants i les tecnologies que s'han emprat.

Actualment moltes empreses inclouen la paraula 'cloud' en la seva definició. Però que representa realment? En aquest apartat explicarem què significa, quines són les diferències entre serveis de cloud i les empreses més importants què l'utilitzen.

2.1.1. Introducció

El *cloud computing* es refereix a l'ús de recursos hardware i software per oferir-los com un servei a través de la xarxa (típicament Internet). En seu nom prové del fet d'utilitzar capes com a abstracció de la complexa infraestructura que conté.

Actualment es troben definits diversos tipus de cloud computing:

- *Infraestructure as a service (IaaS)*
- *Platform as a service (PaaS)*
- *Software as a servce (SaaS)*
- *Storage as a service (STaaS)*
- *Security as a service (SECaaS)*
- *Data as a service (DaaS)*
- *Test environment as a service (TEaaS)*
- *Desktop as a service (Daas)*
- *API as a service (APIaaS)*

El model de negoci que més s'està desenvolupant actualment és el que uneix SaaS o PaaS + IaaS, és a dir, aquell en el que els proveïdors de servei administren els recursos de forma dinàmica mitjançant un "Intrastructure as a service" i els serveixen a clients finals a través de "Platforms as a service" o "Software as a service". (o en cursiva les expressions en angles)

Això permet que els usuaris finals puguin accedir a aplicacions cloud a través d'un navegador web, d'un escriptori lleuger o aplicacions mòbils, mentre que la capa de negoci i les dades d'usuari es gestionen de forma remota en un servidor, on les empreses puguin servir les seves aplicacions de forma més ràpida i eficient, millorant l'administració i el manteniment, permetent també que

la infraestructura ajusti els recursos més ràpidament per adaptar-se a les necessitats de negoci.

2.1.2. Els orígens

L'origen del terme cloud computing no està clar, però es creu que ha derivat de la utilització dels núvols per referir-se a xarxes en esquemes de sistemes informàtics i de comunicació. Aquest terme s'utilitza com una metàfora d'Internet, basat en l'ús normalitzat d'una forma semblant a un núvol per denotar una xarxa d'esquemes de telefonia i més tard per representar Internet en els diagrames de xarxes informàtiques com una abstracció de la infraestructura que representa.
Durant la dècada de 1990, les empreses de telecomunicacions, que anteriorment oferien serveis dedicats punt a punt, van començar a oferir xarxes privades virtuals (VPN) amb una qualitat de servei comparable al que ja oferien però a un cost molt menor. Gràcies a que van propiciar la utilització sota demanda, van ser capaços d'utilitzar l'ample de banda total de la xarxa de forma més eficient.

En aquest context, el símbol de núvol s'utilitza per separar la responsabilitat del proveïdor i el que era la responsabilitat dels usuaris.

El primer concepte respecte a computació en el núvol es remunta a la dècada de 1950, quan es comencen a fer servir els *mainframe* en el món acadèmic i a les empreses. Degut a què el cost de comprar un servidor central era molt gran s'havia de trobar models de negoci que facilitessin el retorn de la inversió. Els mainframe permeten que diversos usuaris comparteixin tant l'accés físic a l'ordinador des de múltiples terminals, així com el temps de CPU i de recursos.

Mentre els ordinadors es van tornar més freqüents, es van explorar noves formes de millorar els temps de càlcul a gran escala, per garantir l'ús òptim de la infraestructura, plataforma i aplicacions amb accés prioritari a la CPU i l'eficiència per els usuaris finals.

Amazon va tenir un paper clau en el desenvolupament de la computació en el núvol mitjançant la modernització dels seus centres de dades, els quals, igual que la majoria de les xarxes informàtiques, s'utilitzaven només al 10% de la seva capacitat en pics de demanda. Després d'estudiar els avantatges que suposava l'arquitectura cloud van veure que podrien servir recursos més fàcilment, i amb aquest objectiu en ment va néixer Amazon Web Service (AWS) al 2006.
Després de veure el potencial que se'n podia extreure, molts van dedicar grans esforços a construir API's compatibles amb el nou servei de Amazon i, al 2008, eucaliptus es va convertir en la primera API REST *opensource* en utilitzar el servei de AWS. A partir d'aquest moment van començar a sortir moltes revisions utilitzant la base d'eucaliptus, i duran el mateix any, *OpenNebula* va alliberar el primer SDK per implementar cloud públics i privats.

L'1 de març de 2011, IBM va crear la fundació Smarter Computing per donar suport al programa Smarter Planet, on el cloud computing és una peça

fonamental. Actualment moltes empreses estan veient les oportunitats de negoci que ofereix el cloud computing, i estan creant nous serveis que aprofitin aquest tipus d'arquitectura.

2.1.3. Sistemes similars i conceptes clau

Aquesta secció descriu sistemes que comparteixen característiques cloud.

- ***Autonomic Computing***

Són sistemes informàtics capaços d'auto-gestionar-se sense interacció humana.

- ***Model client-servidor***

El terme client-servidor es refereix a qualsevol aplicació distribuïda que diferencïi entre els proveïdors de serveis (servidors) i consumidors de serveis (clients)

- ***Grid Computing***

Forma de computació distribuïda i en paral·lel que es basa en la utilització de diversos ordenadors que treballen de forma orquestrada per simular un súper computador que pugui realitzar tasques computacionals costoses.

- ***Mainframe***

Són ordinadors potents utilitzats principalment per les grans organitzacions per a aplicacions crítiques, com ara les relacionades amb estadístiques del cens, la policia i els serveis secrets d'intel·ligència, la planificació de recursos empresarials i el processament de transaccions financeres.

- ***Peer-to-peer***

Arquitectura distribuïda sense la necessitat d'una coordinació central, on s'utilitzen als participants tant de proveïdors com de consumidors de recursos (en contrast amb el tradicional model client-servidor).

- ***Jocs Cloud***

El joc és renderitzat i executat en el servidor, i posteriorment les dades s'envien als client per la xarxa, de forma que els dispositius que s'utilitzin per jugar ja tenen la imatge i so processat. Com a avantatges cal destacar que el joc serà independent del client que s'utilitzi per jugar-lo, i que els requeriments del client son ínfims en comparació al servidor.

2.1.4. Característiques

La computació en núvol presenta varies característiques distintives. Agilitza l' aprovisionament de recursos per la infraestructura als clients, mols cops a través de API's REST que permeten la interacció entre servei i clients de forma transparent als usuaris.

El model de negoci que promou la infraestructura cloud es el de lloguer per consum, cosa que ajuda a les empreses que no disposen de infraestructura pròpia i volen tenir aquest servei externalitzat quan el necessitin. A més no caldrà que el client afronti els grans costos de comprar la infraestructura, ja què seran servits per un proveïdor segons l'ús que se'n faci. Aquest punt és clau per les *s'arrup* que no poden invertir en infraestructura, però al mateix temps, no poden renunciar a utilitzar-la pel tipus de serveis que ofereixen; així que per a moltes els serveis cloud són un bé indispensable.

Un altre punt a destacar si parlem del Ses és el fet que els usuaris poden accedir al software cloud sense requerir més que un navegador web, cosa que simplifica la distribució i millora l'experiència dels usuaris finals perquè poden accedir al servei des de qualsevol computadora que disposi de navegador web.
La tecnologia de virtualització permet als servidors i dispositius d'emmagatzematge la compartició de recursos i l'alta escalabilitat en cas que sigui necessària. Això millora l'eficiència d'utilització i ocupació de sistemes que normalment s'utilitzen només al 10-20%.

Pel que fa a la seguretat és sovint tan bona o millor que altres sistemes tradicionals, en part perquè els proveïdors són capaços de dedicar recursos a la solució dels problemes de seguretat que molts clients no poden permetre. No obstant això, la complexitat de la seguretat és molt més gran quan les dades es distribueix sobre una àrea més àmplia o major nombre de dispositius, sobretot en instàncies *multi-tenant*, on el sistema es comparteix per usuaris no relacionats. A més, l'accés d'usuari als registres d'auditoria de seguretat pot ser difícil o impossible. Instal·lacions de núvols privades estan en part motivats pel desig dels usuaris a mantenir el control sobre la infraestructura i evitar la pèrdua de control de la seguretat de la informació.

El manteniment de les aplicacions de computació en núvol és més fàcil, ja que no requereix ser instal·lat en l'ordinador de cada usuari i es pot accedir des de diferents llocs.

Finalment, un punt distintiu es l'auto-aprovisionament dels recursos. L'autoservei sota demanda permet als usuaris obtenir, configurar i desplegar serveis en el núvol ells mateixos utilitzant catàlegs de serveis cloud, sense requerir l'assistència d'IT.
El requisit d'auto-servei de cloud computing és disposar d'una empresa que aprovisioni el catàleg de plantilles que seran desplegades en serveis cloud. Les plantilles contenen configuracions pre-definides per establir serveis en el núvol, i inclouen detalls específics de configuració per a diferents infraestructures de núvol, amb informació sobre els servidors per a tasques específiques, com ara allotjament d'aplicacions, bases de dades, llocs web, etc. Moltes també

inclouen servei web predefinit, el sistema operatiu, base de dades, configuracions de seguretat i balanceig de càrrega. Un cop es disposa d'aquesta plantilla s'haurà de desplegar en un entorn cloud (com ara Amazon EC2), normalment a través d'un portal d'autoservei.

2.1.5. Models de servei

Els proveïdors de cloud computing ofereixen els seus serveis d'acord a tres models diferents:
Infraestructura com a servei (IaaS), plataforma com a servei (PAAS) i software com a servei (SaaS).

- ***Infraestructura com a servei (IaaS)***

La infraestructura com a servei es un model de cloud que es basa en el aprovisionament de recursos de forma dinàmica i altament escalable, ja què serveix els recursos a través de màquines virtuals, les quals es gestionen mitjançant hipervisors. Aquests disposen d'una base de dades d'imatges en cru de varis tipus de serveis, com ara servidors web, distribucions Linux o d'altres sistemes operatius, tallafocs, enrutadors, etc. Un cop el client demanda un servei en concret es genera una nova instància en el hipervisor, reservant els recursos que s'hagin sol·licitat i utilitzant com a sistema operatiu la imatge seleccionada.

El fet que sigui un hipervisor i que no s'interactuï directament amb la màquina física es precisament per aprofitar l'escalat dinàmic de recursos si ens convé, ja què podríem sol·licitar un nombre més gran de CPU's, RAM, disc, etc... de forma transparent. Altres recursos als núvols IaaS inclouen imatges d'una biblioteca de màquines virtuals, emmagatzematge d'imatges en brut i basats en arxius, tallafocs, balanceig de càrrega, adreces IP, xarxes virtuals d'àrea local (VLAN) i paquets de software.

En aquest model, és l'usuari qui es fa responsable del manteniment del sistemes i del programari. Així doncs, el cost reflectirà en la quantitat de recursos que s'han assignat i consumit.
Exemples de IaaS inclouen: Amazon CloudFormation (i els serveis bàsics, com ara Amazon EC2), Cloud Rackspace, Terremark, Windows Azure Màquines Virtuals, Google Ets Engine. i Joyent.

- ***Plataforma com a servei (PAAS)***

En el model PAAS, els proveïdors de cloud ofereixen una plataforma de computació típicament incloent el sistema operatiu, l'entorn de programació de llenguatge d'execució, base de dades i servidor web. Els desenvolupadors d'aplicacions poden desenvolupar i executar les seves solucions de programari en una plataforma en el núvol, evitant el cost de la infraestructura i de la gestió d'aquest, a més d'evitar-se problemes amb la instal·lació i configuració de programari. Amb algunes ofertes PAAS, els servidors permeten l'escalat dinàmic per adaptar-se a la demanda de les aplicacions, de forma que no cal assignar recursos manualment.

Alguns exemples són: Amazon Elastic Beanstalk, Foundry Cloud, Heroku, Force.com, EngineYard, Mendix, Google App Engine, Windows Azure Ets i OrangeScape.

- ***Software com a servei (SaaS)***

En aquest model, els proveïdors de cloud contenen el programari en un servidor i l'ofereixen a través de la xarxa, sense que el client necessiti cap requeriment especial per executar-lo, només una connexió a la xarxa i un navegador. Els usuaris d'aquest cloud no gestionen ni la plataforma, ni la infraestructura on s'executa el programa. Així s'estalvien els costos de manteniment i posada a punt. Però el que diferencia a una aplicació cloud de la resta d'aplicacions és l'alta escalabilitat, de la mateixa forma com s'ha comentat en els models anteriors, ja què es pot aconseguir en temps d'execució la clonació de tasques en diverses màquines virtuals per satisfer la demanda, fent que alguns nodes s'encarreguin de la orquestració i vagin distribuint la carga de treball entre diverses màquines. Aquest procés és transparent per a l'usuari, ja que només veu un punt d'accés únic.

A més a més, per aprofitar els recursos al màxim aquestes aplicacions poden ser multiusuari, és a dir, qualsevol màquina serveix a més d'una organització d'usuaris cloud.
El model de negoci per a les aplicacions SaaS és típicament una tarifa plana mensual o anual per usuari, així que el preu és escalable i ajustable segons el nombre d'usuaris.
Alguns exemples de SaaS són: Google Apps, innkeypos, Online Quickbooks, bizx SuccessFactors, Vídeo Platform Limelight, Salesforce.com, Microsoft Office 365 i OnLive.
Clients cloud
En aquesta secció es descriuen els tipus de clients que utilitzen els models descrits anteriorment. Els accessos es fan a través de qualsevols dispositiu que tingui una connexió a la xarxa, com ara ordinadors de sobretaula, portàtils, *tablets* i/o *smartphones*. Alguns d'ells gestionen íntegrament totes les dades mitjançant algun servei cloud, mentre que altres només l'utilitzen com a servei d'emmagatzematge elàstic (*blob storage*).
Alguns exemples són el Chromebook, un notebook basat en un navegador que aprofita la potència de l'Ajax i el HTML5 per no requerir cap software addicional al navegador que porta preinstal·lat de fàbrica. D'altres, com ara el servei onlive, permeten renderitzar els jocs en servidors i mostrar-los als clients mitjançant una connexió semblant al que coneixem com vnc.

2.1.6. Tipus de Cloud Computing

- ***Cloud Públic***

En aquest tipus de cloud l'emmagatzematge i altres recursos es posen a disposició del públic en general a través un proveïdor de serveis. Aquests serveis són gratuïts o oferts en un model de pagament per ús. En general, els proveïdors públics de serveis en el núvol com Amazon AWS, Microsoft i Google, són propietaris i operadors de la infraestructura d'accés, i s'ofereixen només a través d'Internet (no tenen opció de connectivitat directa).

- ***Cloud Community***

Aquest tipus de cloud es basa en la organització d'una comunitat amb interessos comuns (seguretat, compliment, jurisdicció, etc), els quals administren i allotgen internament o per tercers. Les avantatges són que es troba més controlat el tipus de clients que en formen part, però no és tan car com tenir un cloud privat.

- ***Cloud Hibrid***

Consta de la composició de dos o més clouds de diversos tipus (privat, comunitari o públic).
Gràcies a aquesta unió, l'arquitectura pot assumir una altra tolerància a fallades, gràcies al re-desplegament en noves màquines al cloud públic, en combinació amb facilitat d'ús a nivell local sense dependre de la connexió a Internet, garantint també la seguretat en les dades més sensibles. També es sol utilitzar per fer proves puntuals en entorns més grans que els que es tenen en producció durant períodes de temps limitats, evitant així la gran inversió que suposaria comprar la infraestructura per fer els tests.

- ***Cloud Privat***

Finalment, el cloud privat és aquell en el que coneixem en quin centre de dades es desplegaran les màquines o serveis que necessitem; en tot moment podríem saber la situació de les dades i tenir la certesa que aquestes es troben segures en un centre privat. Aquest tipus de cloud també permet a les empreses a que s'auto gestionin els recursos de forma interna, evitant així la carrega de treball en els departament de IT per aprovisionar noves màquines o recursos. Si es disposa de un cloud privat gestionat com a IaaS, els propis departaments de desenvolupament o testeig poden auto aprovisionar-se les màquines que necessitin de forma gairebé instantània, agilitzant el procés de producció.

2.1.7. Privadesa i seguretat

El model cloud ha estat àmpliament criticat a nivell de seguretat, ja que el fet de emmagatzemar o utilitzar recursos de forma remota implica que es faci en màquines que no són propietat dels clients (cloud públic), i es rumoreja que les empreses de serveis que ofereixen espai cloud controlen el contingut que s'utilitza en els seus servidors per evitar traspassar la línia legal en alguns casos i, en d'altres, estudiar l'ús que en fan els clients dels seus recursos.
Un exemple molt criticat va ser el programa que la NSA va fer en secret, en col·laboració amb AT&T i Verizon, operadores mòbils, en el que es van registrar 10 milions de trucades a Nord-Amèrica per estudiar el comportament dels usuaris. Per tant, el proveïdor de serveis en núvol (CSP) es qui pot comprometre la privacitat en les dades segons on es despleguin les màquines virtuals. Aquest fet provoca que no es tingui control sobre la situació geogràfica on es guardarà la informació, provocant problemes legals sobre jurisdicció.
Encara que hi ha empreses com US-EU Safe Harbor que han intentat canviar la forma legal d'aquests models de negoci, els proveïdors com Amazon encara atenen als mercats principals (típicament els Estats Units i la Unió Europea) mitjançant el desplegament de la infraestructura local i permet als clients seleccionar "zones de disponibilitat", cosa que permet controlar la regió geogràfica on s'està emmagatzemant el contingut.

2.1.8. El codi obert

Gràcies a les grans comunitats de codi obert que existeixen han proporcionat eines per la creació d'implementacions de cloud computing. Alguns exemples destacats són Hadoop i Cloud de VMware Foundry. Al novembre de 2007, la Fundació per al software lliure va crear la llicència Affero General Public, una versió de la GPLv3 amb la intenció de solucionar els problemes legals associat amb el programari lliure dissenyat per executar-se en xarxa.

2.1.9. Els estàndards actuals

La majoria dels proveïdors actuals exposen les seves API's juntament amb una extensa documentació per a la seva explotació (sovint sota una llicència Creative Commons), però també és una implementació pròpia i limita la interoperabilitat entre altres APIs de tercers. Actualment s'estan adoptant estàndards, com ara el REST, per garantir la futura connectivitat, portabilitat i transparència entre serveis.

2.1.10. Sostenibilitat

Tot i que el cloud computing es una forma de "green computing" , no s'ha realitzat encara cap estudi oficial que ho demostri, tot i que el tipus de gestió dinàmica de recursos on només es gasta el que es necessita sembla que ho corrobori. Els grans centres de dades se solen trobar en climes freds, on s'afavoreix de la refrigeració natural. Països com ara Finlàndia, Suècia i Suïssa, estan utilitzant aquest criteri per atreure inversions que creïn nous centres de dades per explotar-se mitjançant arquitectures cloud.

2.1.11. Ús fraudulent dels serveis

De la mateixa forma que es poden adquirir recursos de forma privada per realitzar activitats il·legals també es poden utilitzar recursos cloud pels mateixos fins. Amazon ha patit moltes denúncies degut a atacs originats des dels seus serveis.

2.2. Estat de l'art de portals cloud privats

2.2.1. Definició de portal cloud

Com s'ha vist en els apartats anteriors el cloud computing és una gran forma d'aprofitament dels recursos, que amplia l'escalabilitat en les aplicacions que l'utilitzen, afavoreix la descentralització de serveis, redueix costos energètics i amplia els marges de benefici dels proveïdors de serveis. Per tant, ens trobem amb un model de negoci molt rentable. Amb la idea d'aprofitar aquest sistema en la massa critica d'usuaris que utilitzen pocs recursos en comparació amb les grans empreses molts proveïdors de serveis estan creant portals cloud privats per facilitar l'accés al món cloud a qualsevol usuari de la xarxa i així ampliar el mercat de clients per aprofitar al màxim els seus recursos.

Un portal cloud es podria definir una forma de fer servir recursos físics de forma fàcil i ràpida mitjançant les avantatges del IaaS combinat amb el Saas , molts cops amb paquets bàsics als quals es poden afegir extres segons es realitza la compra.

Al final del procés obtindrem un preu segons els serveis que haguem sol·licitat. La particularitat d'aquest model és que gràcies a la gestió dinàmica dels recursos es podria arribar a oferir serveis per un espai de temps reduït, com ara minuts o segons.

La gran diferència es que l'usuari és qui s'auto-aprovisiona els recursos, és a dir, el procés de desplegament i d'escalabilitat el controla l'usuari a través del portal en pocs clics, i si en un futur vol fer modificacions, afegir o treure serveis pot gestionar-ho tot a través de la seva sessió d'usuari només a través d'un navegador web.

Aquest tipus de model és també molt útil de cara a empreses que no volen invertir més en infraestructura pel fet de tenir pics de demanda puntuals, i opten per desplegar màquines en entorns clouds si la demanda supera a la capacitat mentre duri la demanda i alliberar els recursos quan s'estabilitzi. Així poden pagar només pel que han utilitzat.

2.2.2. Portals cloud privats en producció

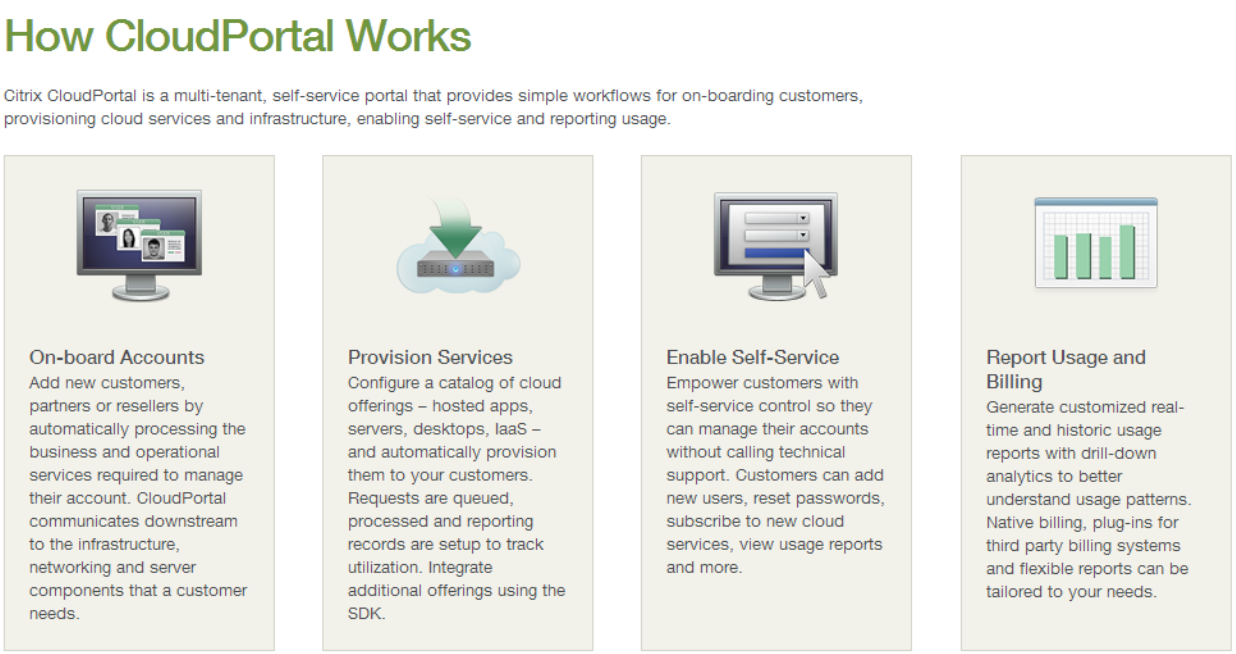
Aquesta secció mostra alguns exemples d'empreses que han creat un portal cloud privat i l'utilitzen per explotar els seus serveis cloud.

- ***Service Provider***

Empreses que ofereixen serveis cloud mitjançant un portal cloud.

Citrix Systems, Inc. (NASDAQ: CTXS)
És una multinacional fundada al 1989, que subministra tecnologies de virtualització de servidors, connexió en xarxa, SaaS i solucions cloud. Un punt a destacar es que va adquirir a XenSource, Inc a l'octubre de 2007, i des de llavors gestiona el projecte del hipervisor Xen de codi obert. Actualment Citrix ofereix servei a unes 230.000 organitzacions de tot el món.

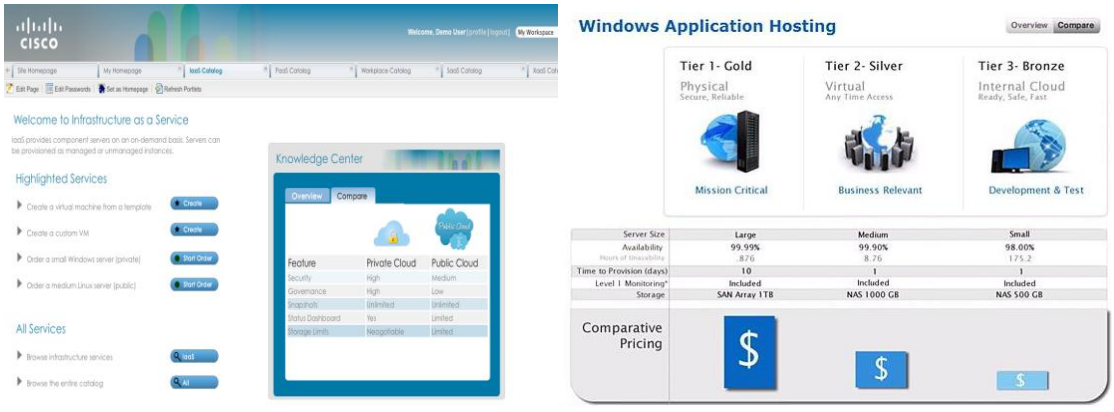
Detall de pàgina de Citrix descrivint el seu Portal Cloud



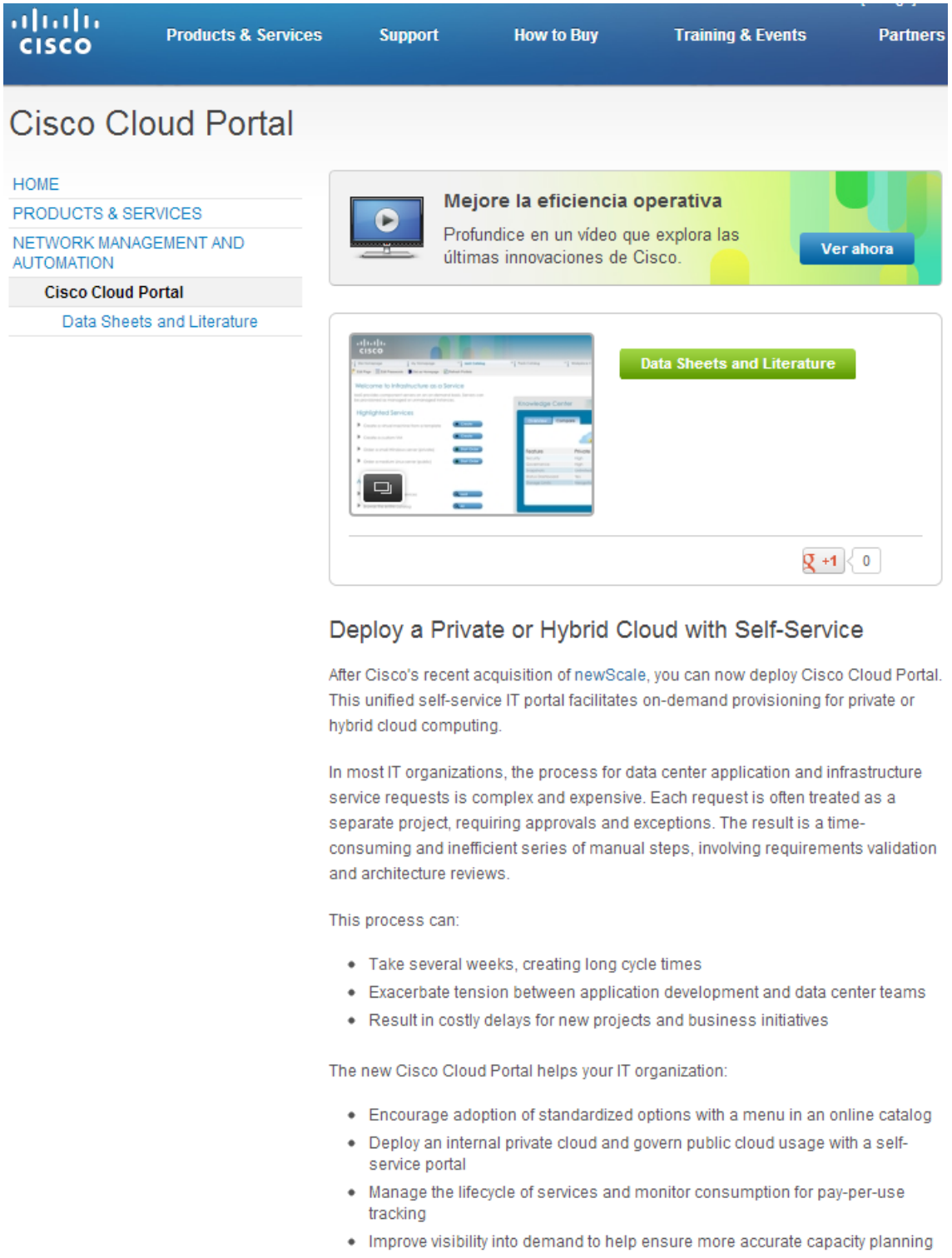
Cisco Systems és una empresa dedicada a la fabricació, venta, manteniment i consultoria de equips de telecomunicacions tals com:

- dispositius de connexió per xarxes informàtiques: *routers*, *switches* i *hubs*
- dispositius de seguretat com *Firewalls* i concentradors per VPN
- Productes de telefonia IP
- Software de gestió de xarxa, com ara CiscoWorks
- Equips per emmagatzematge en xarxes

Fins al 8 de juny de 2009, es considerava una de les grans empreses del sector tecnològic i un important membre del mercat del NASDAQ. Posterior a aquesta data, gràcies a la seva solidesa, ingressa a l'índex d'industrials Dow Jones.



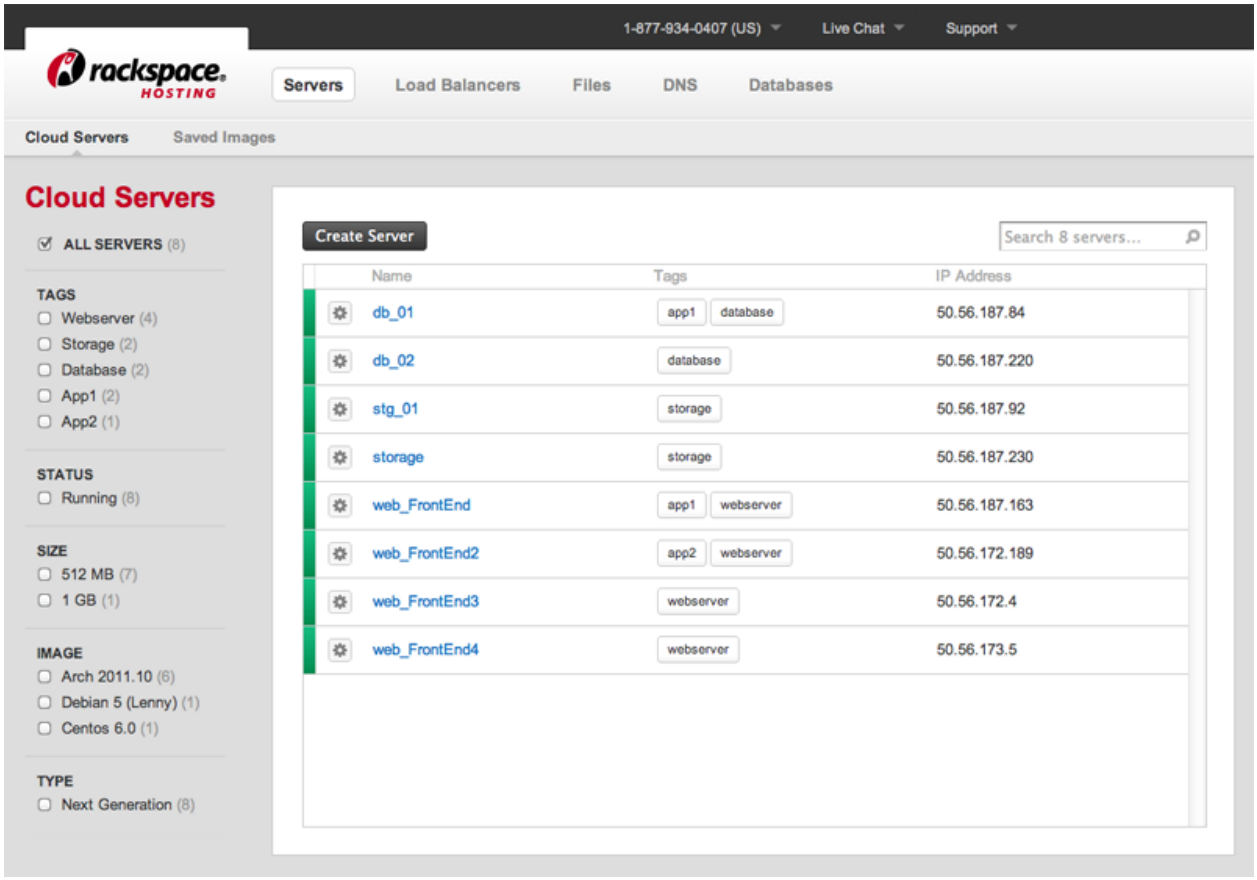
Detall del portal cloud de Cisco i la política de preus



Pàgina de Cisco Systems oferint el Portal Cloud.

Rackspace Us, Inc. (NYSE: RAX) és una empresa dedicada al hosting de serveis per a IT. Té centres de dades operatius a Texas, Illinois, Virginia, Regne Unit i Hong Kong. En la seva cartera de clients en troben el 40% de membres de la llista de Fortune 100, i es va posicionar en el top 50 del Financial Times a regne unit.

http://www.rackspace.com/cloud/managed_cloud/



Vista del portal Cloud utilitzat per Rackspace

- ***Portal Provider***

Empreses que desenvolupen un portal cloud i l'ofereixen per administrar els serveis cloud del client.

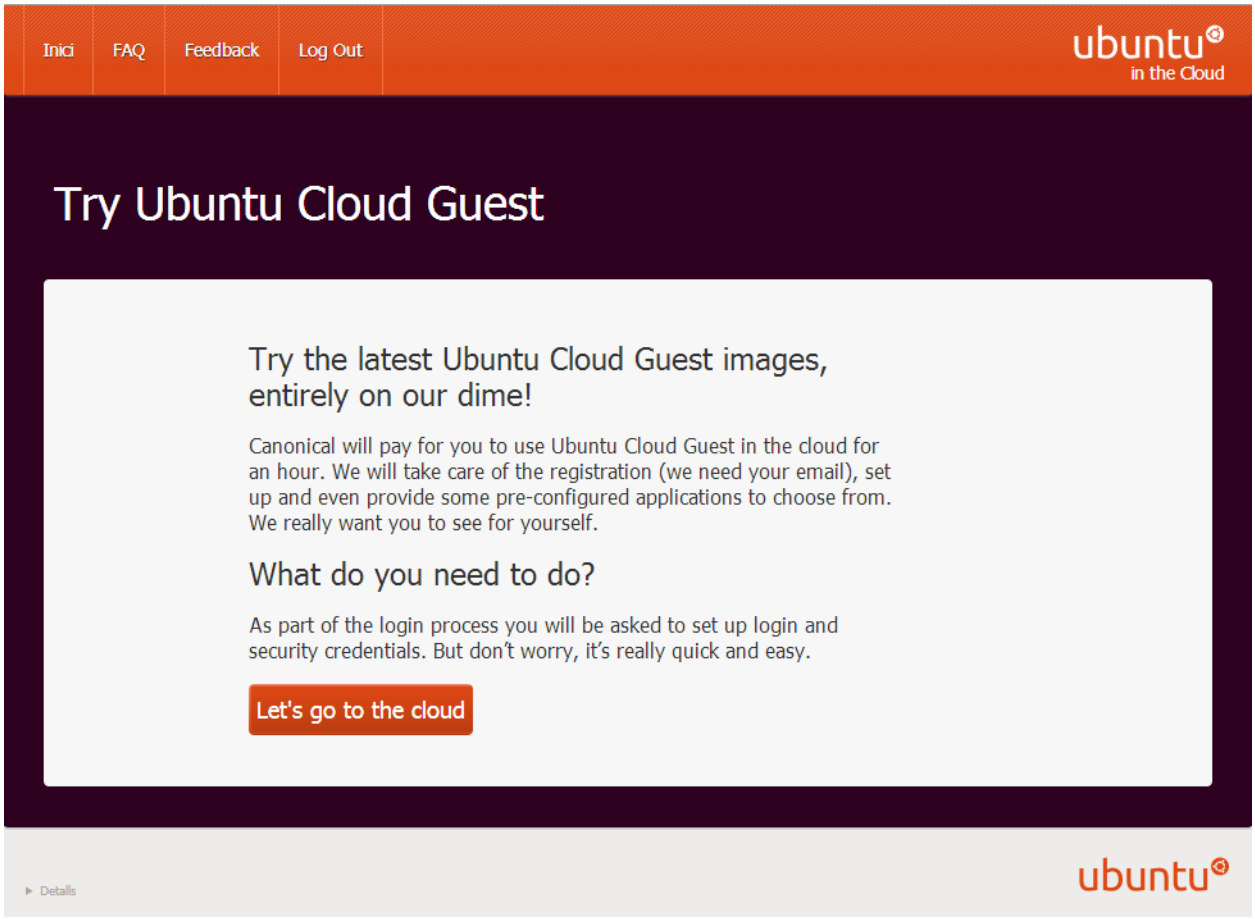
SAP AG és una multinacional alemanya dedicada l disseny de productes informàtics de gestió empresarial, tant per a empreses com per organismes públics. Es competència directa de Oracle, i es calcula que entre el 70 i 80% del mercat de les grans empreses utilitza els seus productes. Va ser fundada al 1972, amb varis productes famosos com ara Sap ERP i Sap Business *warehouse* entre d'altres. La seva capitalització a borsa va ser de 59 milions de USD a l'any 2010.

Vistes del portal cloud de SAP

Ubuntu és un sistema operatiu mantingut per Canonical i la comunitat de desenvolupadors. Utilitza el kernel Linux basat en Debian, i es centra en millorar l'experiència d'usuari. Algunes estadístiques mostren que ocupa el 49% del mercat de distribucions Linux.

Ubuntu Cloud

http://www.ubuntu.com/cloud



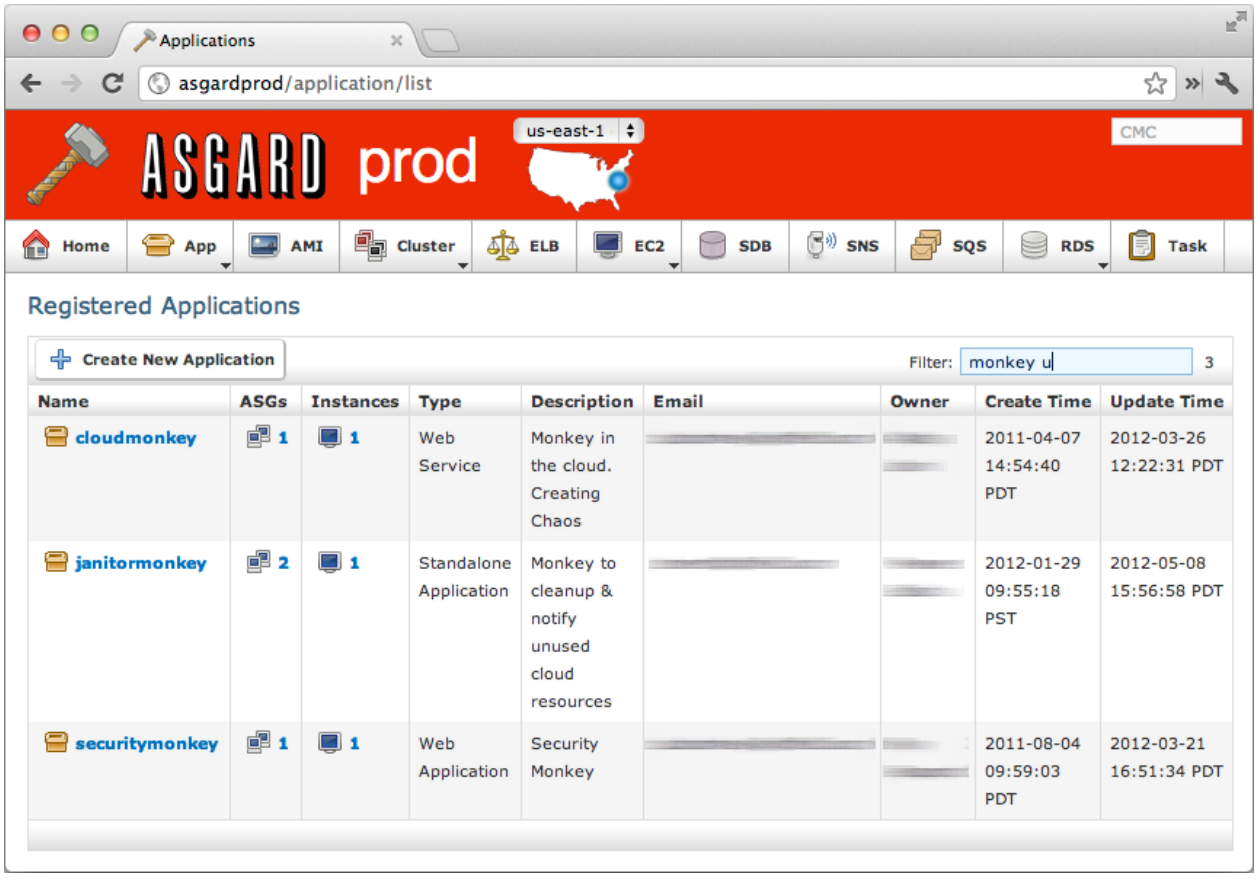
Pàgina de prova de Ubuntu cloud

- ***Solucions Internes***

Empreses que desenvolupen un portal cloud i l'utilitzen internament per administrar els seus serveis cloud.

Netflix és una plataforma de vídeo que ofereix streaming de pel·lícules i sèries de televisió, a canvi d'una quota de subscripció mensual.

Netflix Asgard es un projecte que s'utilitza internament per gestionar els recursos quan tenen més demanda i utilitzen el portal per gestionar les màquines virtuals desplegades a amazon EC2.



Vista del Portal Cloud utilitzat internament per Netflix.

2.3. Estat de l'art sobre tecnologies

Després d'estudiar el mercat de solucions cloud farem una breu explicació sobre quines tecnologies s'han emprat i quins avantatges ens poden aportar cadascuna d'elles.

2.3.1. Llenguatges de programació

La majoria d'empreses del sector es decanten per solucions implementades mitjançant Java2EE, principalment per la gran comunitat de què disposa i per ser codi lliure. També tenim exemples d'implementacions en C#, sobretot en les solucions adoptades per Microsoft.
En altres camps del SaAS, també es sol optar per solucions escrites en ruby o php.

El que queda clar es que totes elles utilitzen un framework per facilitar i garantir la robustesa del producte.

Pel que fa a Abiquo , es troba implementada en Java utilitzant diversos frameworks, on el més important es Spring.

2.3.2. Frameworks a utilitzar

- ***Spring***

Spring és un Framework que facilita el desenvolupament d'aplicacions en Java, ja què permet oblidar-se dels problemes de connexió entre components i capes de serveis, permetent al desenvolupador centrar-se en el model de negoci i no perdre el temps amb tedioses configuracions.

- ***.NET***

.NET és un framework de Microsoft amb independència del hardware, que permet un ràpid desenvolupament d'aplicacions. Gracies a aquest framework, l'empresa intenta desenvolupar una estratègia horitzontal que integri tots els seus productes, des del sistema operatiu fins altres productes de la marca de Microsoft.

- ***Play!***

Framework *open source* escrit en Scala i Java, que segueix el patró Model-Vista-Controlador (MVC). Va sorgir per solucionar els problemes en la creació d'aplicacions web Java, per millorar-ne la productivitat i reduir el temps fins que es trobin en producció. Facilita el desenvolupament a través de gestió de canvis en calent, mostrant errors en format web i permeten una fàcil configuració a través d'una parametrització intuïtiva.

- ***Ruby on Rails***

Ruby on Rails, també conegut com RoR o Rails és un framework d'aplicacions de codi obert, escrit en Ruby, també seguint el paradigma MVC. La característica principal recau en què intenta combinar la simplicitat amb el fet de desenvolupar aplicacions web escrites amb menys codi què amb els altres mètodes.

- ***Symfony***

Symfony és un framework què està dissenyat per optimitzar el desenvolupament d'aplicacions web basades en el model MVC escrites en php. És opensource, i està desenvolupat íntegrament en php 5.3. Ha testejat en nombrosos projectes reals.

- ***Zend***

Zend Framework (ZF) és un framework de codi obert per desenvolupar aplicacions web i serveis web amb PHP 5. Zend Framework és una implementació que utilitza orientació a objectes, on cada component té un acoblament molt baix entre els altres, cosa que crea una arquitectura dèbilment acoblada i que permet utilitzar només algunes parts del framework.

2.3.3. Tecnologies mòbils

A continuació es descriuen quins tipus de tecnologies mòbils es tindran en compte alhora de triar l'arquitectura mòbil.

- ***iOs***

Sistema operatiu d'apple, encarat a dispositius de l'empresa. Actualment totes les gran empreses disposen d'aplicacions en el mercat d'apple. Estan escrites en Objective-c.

- ***Android***

Sistema operatiu de google, basat en el kernell de Linux. Es l'alternativa clara a apple, sorgida per trencar la barrera de preus imposada per apple, la majoria d'aplicacions que es venen són gratuïtes ¿SEGUR?. Les aplicacions es troben escrites en java.

- ***Html5 mobile***

Aplicacions no-natives, sense sistema operatiu definit, ja que s'executen en el navegador del client, i poden garantir que la gran majoria de sistemes operatius amb entorn gràfic disposen de navegador. Es troben escrites en html combinant Ajax (asyncron javascript and xml).

2.3.4. Comunicació amb el IaaS

- ***Mitjançant crides directes (arquitectura SOAP)***

Aquesta solució ens permetria simplificar algunes tasques en comparació a REST, ja què al conèixer les capçaleres es poden personalitzar i adaptar a les crides que s'esperin. El principal problema és el fet de crear un tipus d'arquitectura totalment lligada a la implementació de la plataforma, i en el cas de que es realitzin canvis faria l'aplicació poc sostenible. A més no podria reutilitzar-se per altres serveis Iaas, ja què les funcions serien diferents.

- ***Mitjançant crides estandarditzades (arquitectura REST)***

Tot i que aquest tipus d'arquitectura sigui més costosa d'aplicar, ja què s'ha d'adaptar el producte a crides GET o POST i pot ser que s'hagin de fer més peticions que en el cas anterior, el fet que la resposta estigui estandarditzada millora el manteniment i permet escalar el producte, canviar o afegir noves implementacions dins de la API. A més el fet d'estandarditzar ens permetrà que si en un futur es vol obrir la API a tercers ja estaran familiaritzats amb el seu ús.

- ***Mitjançant un client API***

Clarament si ja es disposa d'una API Rest és un pas necessari si els requeriments per utilitzar la API requereixen un coneixement profund de la plataforma o si la API es molt gran.
En el cas del portal disposem d'una API Rest per comunicar-nos amb el IaaS, però això també implicaria que només es podria usar el portal amb un únic IaaS.

Per solucionar aquest problema s'ha decidit utilitzar jClouds, llibreria cloud escrita en java, del qual Abiquo n'és proveïdor oficial, juntament amb Amazon, entre d'altres. Gràcies a aquest client simplificarem el procés de comunicació amb la plataforma i permetrem que en un futur pugui ser utilitzat amb un altre

servei, segons els requeriments de l'usuari, ja què hem de recordar que el portal cloud és un projecte de software lliure.

2.3.5. Solucions adoptades

- ***Llenguatge de programació: Java2EE***

S'ha triat Java2EE degut a la gran comunitat que existeix, a la robustesa i eficiència del llenguatge (per sobre de php), a la seva portabilitat (millor que la que té C#) i també perquè al ser opensource no ens haurem de preocupar de quines llicències utilitzem.

- ***Framework: Play!***

Després de triar el llenguatge Java, les opcions que teníem eren o bé Spring o Play!. Em triat Play! ja què s'ajusta a les necessitats del portal, no necessitem tanta potència de servei com pot oferir Spring i en tenim prou amb l'arquitectura MVC, ja què utilitzarem una API per realitzar la comunicació amb el IaaS, i ho podem fer des del controlador. A més, simplifica el procés de gestionar les vistes en el portal, i s'integra fàcilment amb html5.

- ***Aplicació mòbil: jquery Mobile***

Tot i que les aplicacions natives representen una millor experiència per part de l'usuari també requereixen d'un manteniment continu, ja que els sistemes operatius més populars (Android i iOs) es troben en continua modificació. Es va preferir un model que perdurés en el temps i que fos compatible amb totes les plataformes, així que es va optar per una aplicació mòbil escrita en HTLM5, la qual amb un únic desenvolupament podia arribar a satisfer en gran part les necessitats de qualsevol client mòbil. Per millorar l'experiència i fer-ho més natiu, es va optar per utilitzar un framework mòbil, i despès d'estudiar el mercat es va decidir utilitzar jquery Mobile, degut a la facilitat d'integrar-se amb els controladors del portal i perquè ha estat desenvolupat pels creadors de jquery, llibreria javascript base que utilitzen tots els *frontends* web actuals, cosa que garanteix el seu continu manteniment i millora.

- ***Comunicació amb el IaaS: jClouds***

Després d'estudiar les opcions quedava clar que si en un futur es vol utilitzar un altre IaaS cal que el portal estigui pensat com a arquitectura distribuïda, i queda clar que l'arquitectura a utilitzar es la REST. Però el fet de configurar el IaaS mitjançant crides REST fa que sigui complicat i poc intuïtiu, així que es va decidir treballar sobre jClouds i implementar les funcionalitats extres que es requerissin pel portal.

CAPÍTOL 3. PLANIFICACIÓ DEL PROJECTE

3.1 Organització temporal

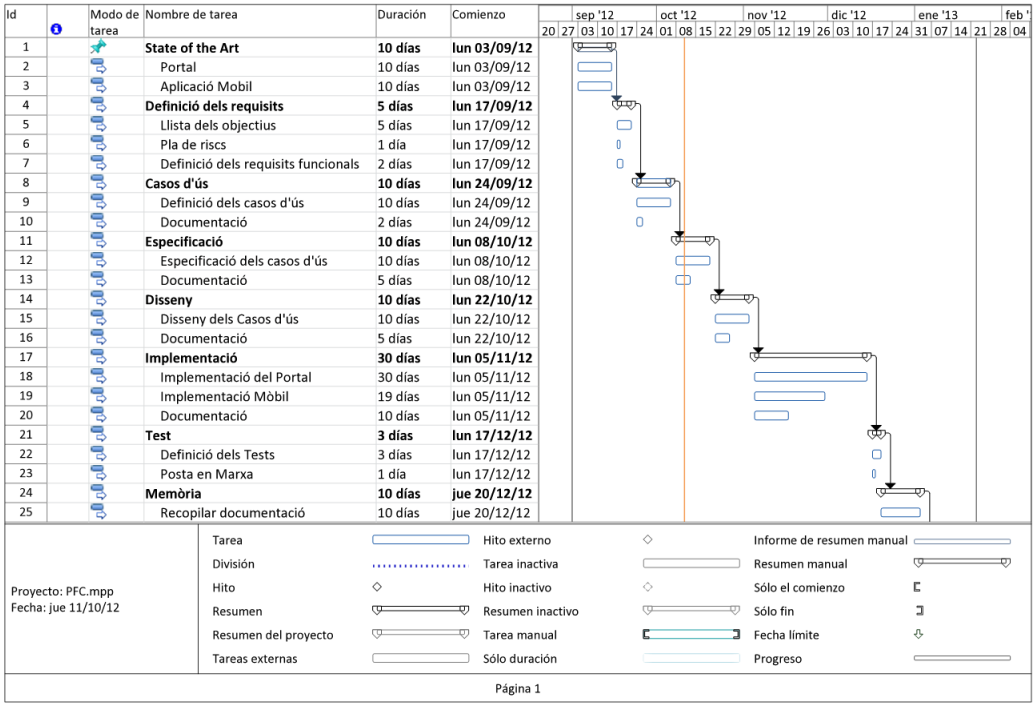
En aquesta secció es mostra l'organització seguida durant el transcurs del projecte, amb el llistat de tasques i les fites més importants que s'han dut a terme.

3.1.1 Planificació de tasques

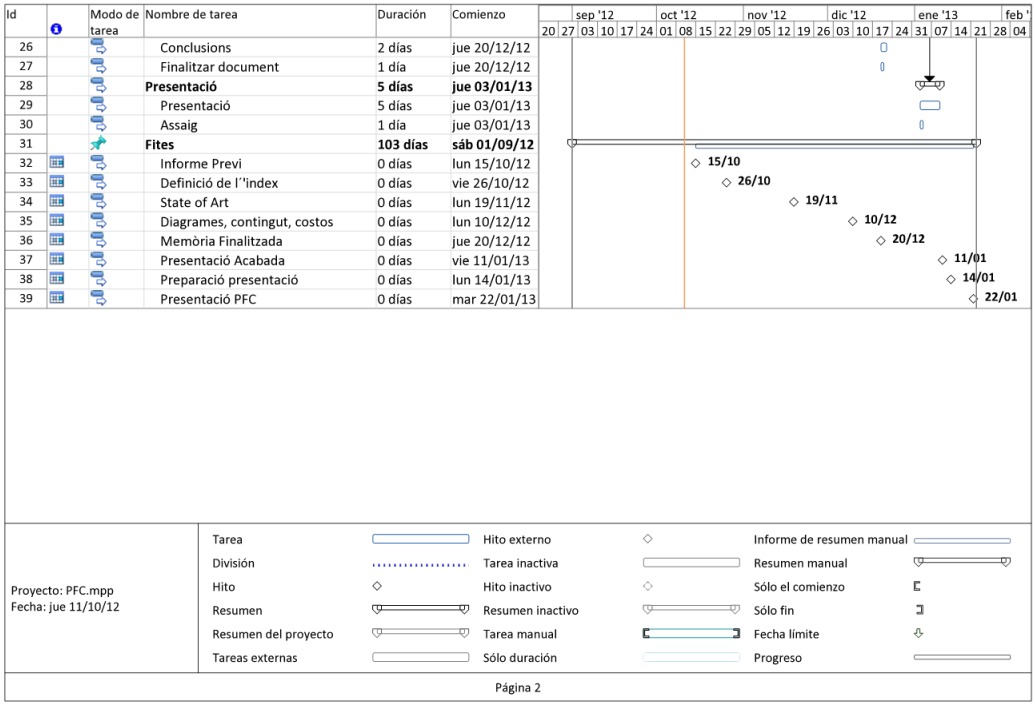
Nom de la tasca	Temps	Comença	Final	Pred.
State of the Art	**10 Dies**	**03/09/12**	**14/09/12**	
Portal	10 dies	03/09/12	14/09/12	
Aplicació Mòbil	10 dies	03/09/12	14/09/12	
Definició dels requisits	**5 dies**	**17/09/12**	**21/09/12**	**1**
Llista dels objectius	5 dies	17/09/12	21/09/12	
Pla de riscs	1 dia	17/09/12	17/09/12	
Definició dels requisits funcionals	2 dies	17/09/12	18/09/12	
Casos d'ús	**10 dies**	**24/09/12**	**05/10/12**	**4**
Definició dels casos d'ús	10 dies	24/09/12	05/10/12	
Documentació	2 dies	24/09/12	25/09/12	
Especificació	**10 dies**	**08/10/12**	**19/10/12**	**8**
Especificació dels casos d'ús	10 dies	08/10/12	19/10/12	
Documentació	5 dies	08/10/12	12/10/12	
Disseny	**10 dies**	**22/10/12**	**02/11/12**	**11**
Disseny dels Casos d'ús	10 dies	22/10/12	02/11/12	
Documentació	5 dies	22/10/12	26/10/12	
Implementació	**30 dies**	**05/11/12**	**14/12/12**	**14**
Implementació del Portal	30 dies	05/11/12	14/12/12	
Implementació Mòbil	19 dies	05/11/12	29/11/12	
Documentació	10 dies	l05/11/12	16/11/12	
Test	**3 dies**	**17/12/12**	**19/12/12**	**17**
Definició dels Tests	3 dies	17/12/12	19/12/12	
Posta en Marxa	1 da	17/12/12	17/12/12	
Memòria	**10 dies**	**20/12/12**	**02/01/13**	**21**
Recopilar documentació	10 dies	20/12/12	02/01/13	
Conclusions	2 dies	20/12/12	21/12/12	
Finalitzar document	1 dia	20/12/12	20/12/12	
Presentació	**5 dies**	**03/01/13**	**09/01/13**	**24**
Presentació	5 dies	03/01/13	09/01/13	
Assaig	1 dia	03/01/13	03/01/13	
Fites	**103 dies**	**01/09/12**	**22/01/13**	

Informe Previ	0 dies	15/10/12	15/10/12	
Definició de l'índex	0 dies	26/10/12	26/10/12	
State of Art	0 dies	19/11/12	19/11/12	
Diagrames, contingut, costos	0 dies	10/12/12	10/12/12	
Memòria Finalitzada	0 dies	20/12/12	20/12/12	
Presentació Acabada	0 dies	11/01/13	11/01/13	
Preparació presentació	0 dies	14/01/13	14/01/13	
Presentació PFC	0 dies	22/01/13	22/01/13	

3.1.2 Diagrama de Gantt



Id	Modo de tarea	Nombre de tarea	Duración	Comienzo
1		**State of the Art**	**10 días**	**lun 03/09/12**
2		Portal	10 días	lun 03/09/12
3		Aplicació Mobil	10 días	lun 03/09/12
4		**Definició dels requisits**	**5 días**	**lun 17/09/12**
5		Llista dels objectius	5 días	lun 17/09/12
6		Pla de riscs	1 día	lun 17/09/12
7		Definició dels requisits funcionals	2 días	lun 17/09/12
8		**Casos d'ús**	**10 días**	**lun 24/09/12**
9		Definició dels casos d'ús	10 días	lun 24/09/12
10		Documentació	2 días	lun 24/09/12
11		**Especificació**	**10 días**	**lun 08/10/12**
12		Especificació dels casos d'ús	10 días	lun 08/10/12
13		Documentació	5 días	lun 08/10/12
14		**Disseny**	**10 días**	**lun 22/10/12**
15		Disseny dels Casos d'ús	10 días	lun 22/10/12
16		Documentació	5 días	lun 22/10/12
17		**Implementació**	**30 días**	**lun 05/11/12**
18		Implementació del Portal	30 días	lun 05/11/12
19		Implementació Mòbil	19 días	lun 05/11/12
20		Documentació	10 días	lun 05/11/12
21		**Test**	**3 días**	**lun 17/12/12**
22		Definició dels Tests	3 días	lun 17/12/12
23		Posta en Marxa	1 día	lun 17/12/12
24		**Memòria**	**10 días**	**jue 20/12/12**
25		Recopilar documentació	10 días	jue 20/12/12



Id	Modo de tarea	Nombre de tarea	Duración	Comienzo
26		Conclusions	2 días	jue 20/12/12
27		Finalitzar document	1 día	jue 20/12/12
28		**Presentació**	**5 días**	**jue 03/01/13**
29		Presentació	5 días	jue 03/01/13
30		Assaig	1 día	jue 03/01/13
31		**Fites**	**103 días**	**sáb 01/09/12**
32		Informe Previ	0 días	lun 15/10/12
33		Definició de l''index	0 días	vie 26/10/12
34		State of Art	0 días	lun 19/11/12
35		Diagrames, contingut, costos	0 días	lun 10/12/12
36		Memòria Finalitzada	0 días	jue 20/12/12
37		Presentació Acabada	0 días	vie 11/01/13
38		Preparació presentació	0 días	lun 14/01/13
39		Presentació PFC	0 días	mar 22/01/13

3.2 Pla de Riscs

Durant el transcurs del projecte poden ocórrer diversos riscs, els qual es detallaran a continuació, juntament amb la mitigació i el pla de contingència que aplicarem en cas que succeeixin:

La comunicació entre el portal cloud i els recursos no es realitza correctament

- ***Mitigació: Realitzaren tests unitaris i d'integració per garantir el correcte funcionament mentre desenvolupem***

- ***Contingència: Prescindir del SDK i utilitzar crides directes a la API de IaaS***

No es possible monitoritzar l'estat de les màquines virtuals un cop desplegades

- ***Mitigació: comprovarem que l'estat recuperat es correspon a l'estat real de les màquines virtuals.***

- ***Contingència: Recuperarem l'estat de les màquines connectant-nos directament.***

Les dades de que es disposa no són correctes

- ***Mitigació: treballarem amb dades falses per garantir que el funcionament es correcte***

- ***Contingència: Crearem un servei per obtenir les dades de forma directa amb el Iaas***

En cas d'error o caiguda de servei les dades deixen de ser consistents

- ***Mitigació: Persistirem els canvis quan estem segurs de que s'han realitzat correctament mitjançant els codis de la resposta.***

- ***Contingència: Mantindrem un backup de la base de dades diari per recuperar la informació.***

El portal provoca condicions de carrera en el desplegament simultani de vàries aplicacions virtuals

- ***Mitigació: Realitzarem la reserva de recursos de forma asíncrona i atendrem les peticions de forma ordenada segons First-come, first-served (FCFS)***

- ***Contingència: No es permetrà el desplegament simultani i es bloquejarà el procés fins que acabi la petició.***

Els recursos no s'alliberen correctament quan s'esborri una oferta

- ***Mitigació: Es comprovarà que els recursos s'hagin alliberat després de l'eliminació a través de la API***

- ***Contingència: S'eliminaran les màquines virtuals del hipervisor de forma directa.***

Els entorns desplegats deixen de ser privats

- ***Mitigació: S'encriptaran les connexions d'usuari i les dades només seran accessibles un cop validat l'accés.***

- ***Contingència: S'haurà de limitar l'accés a través de VPN.***

L'aplicació mòbil no respon amb la usabilitat que s'espera.

- ***Mitigació: es provaran varies alteratives per trobar la que més s'adapti a les necessitats del portal.***

- ***Contingència: Es plantejarà la realització de l'aplicació amb android.***

3.3 Pla de costos

3.3.1 Cost temporal

El cost temporal s'ha calculat tenint en compte el transcurs en què s'ha realitzat el projecte, un total de 8 mesos.

Per tant, tenint en compte que es treballen 4 hores diàries i 5 dies a la setmana, i que es fan 5 hores extres per setmana, fan un total de 800 hores de feina al cap dels 8 mesos comentats.

Si utilitzem com a referència un sou mitjà de 24.000 euros bruts per a un enginyer, amb més d'un any de experiència i en 14 pagues, cada mes cobrarà 1714 euros si fes 40 hores setmanals i, per tant, cobrarà aproximadament 10,70 euros l'hora.

Cost temporal = hores_feina * preu_hora *= 800 * 10,70 =* ***8560 euros***

3.3.2 Cost tecnològic

Aquest cost variarà segons la infraestructura utilitzada si el volem tenir en producció, ja que necessitarem màquines físiques on desplegar les màquines virtuals i un servidor web que gestioni el portal. Per tant, ens centrarem als costos de desenvolupament, i en aquest cas l'empresa no ha invertir diners en recursos tecnològics.

Llicències
Degut a què el portal cloud serà opensource sota llicencia *creative commons* no caldrà afegir costos de llicències addicionals.

3.3.3 Cost total

El cost total del portal serà la suma del cost temporal i del tecnològic:

CT = Cost temporal + Cost tecnològic = 8560 + 1000 = 9560 euros

Aquest cos es veurà incrementat si requerim de infraestructura per realitzar el desplegament dels recursos.

CAPÍTOL 4. ESPECIFICACIÓ

4.1. Generació d'històries d'usuari

Posteriorment es definirà la especificació del projecte, definint els objectius, els requeriments i els riscs associats a les decisions preses. En definitiva, es farà l'anàlisi de requisits per arribar a una correcta especificació del projecte.

4.1.1 Llistat de requeriments

En aquesta secció enumerarem els requeriments mínims que haurà de complir el portal cloud per tal de satisfer totes les necessitats proposades:

- **Auto-aprovisionament de les aplicacions cloud per part dels clients**
  És important que siguin els usuaris de la plataforma qui sol·licitin els recursos, i que aquests es serveixin automàticament, ja què volem eliminar la feina de l'administrador de sistemes relacionada amb la preparació de l'entorn.

- **Auto-gestió de les aplicacions comprades**
  Seguint amb la mateixa filosofia de l'auto-servei, volem que un cop es faci la compra el manteniment bàsic es pugui dur a terme pels propis usuaris, permetent recuperar l'estat en cas que hi hagi error o reiniciant els serveis de forma intuïtiva.

- **Possibilitat d'afegir serveis a les ofertes pre-configurades exposades**
  Volem crear un mercat d'aplicacions cloud on els usuaris puguin personalitzar la seva compra, amb serveix addicionals, com per exemple, la compra de IP's públiques per a poder re-direccionar els seus propis dominis.

- **Accés des de qualsevol navegador o dispositiu al portal**
  És molt important que l'accés no estigui restringit segons els sistema operatiu o el dispositiu. En un Saas, tots els sistemes han de ser el màxim de compatibles sense instal·lar software addicional.

- **Accés a la infraestructura física sense requeriments addicionals**
  Volem proporcionar una interfície amb la que els usuaris puguin accedir a les màquines virtuals de forma remota sense necessitat de requerir més software.

- **Automatitzar el procés de control i alliberació de recursos quan les ofertes expirin**
  El sistema haurà de permetre modificar el període de lloguer de l'oferta, i en cas que cancel·lin la compra s'hauran de alliberar els recursos per tal de mantenir el sistema consistent.

- **Administració de les aplicacions virtuals a través d'un administrador**
  Volem crear un *backoffice* que permeti als administradors crear les ofertes que es publicaran i modificar-les per garantir la qualitat del servei.

4.2. Casos d'ús de negoci

Després de definir els objectius i fer el llistat de requeriments els haurem de detallar per transformar-los en casos d'us de negoci. Per realitzar aquesta tasca utilitzarem el mètode *Volere*, on detallarem per cadascun dels casos d'ús la satisfacció d'implementar-los (o no) en el producte final, en una escala numèrica de 1 a 5.

4.2.1. Llistat de casos d'ús de negoci

Actors: Usuari, Administrador, Sistema

Llistat de casos d'ús de negoci d'Usuari

1. Creació d'usuari
2. Usuari s'autentica
3. Usuari compra una oferta
4. Usuari consulta els detall d'una oferta
5. Usuari consulta les ofertes comprades
6. Usuari consulta els detall d'una oferta comprada
7 . Usuari elimina una oferta comprada
8. .Usuari reinicia una oferta comprada
9. Usuari accedeix a una oferta desplegada

Llistat de casos d'ús de negoci d'Administrador

10. Administrador habilita una oferta
11. Administrador des-habilita una oferta
12 . Administrador configura una oferta

Llistat de casos d'ús de negoci de Sistema

13. El sistema notifica un esdeveniment per correu

4.3. Requisits funcionals

Cas d'ús:

4.3.1. Crear Usuari

Actor Primari: Usuari.
Precondició: Cap.
Trigger: Un usuari s'ha de registrar al sistema.

Escenari Principal:
1. L'usuari proporciona les dades requerides.
2. El sistema valida que les dades són correctes.
3. El sistema crea el client amb les dades introduïdes.
   [→ registrarUsuari]

Extensions:
1.a. L'usuari proporciona els camps no requerits
3.a. Algunes de les dades no són correctes:
   3.a.1. El sistema comunica l'error.
   3.a.2. El cas d'ús continua en l'apartat 1 de l'escenari principal.

Satisfacció: 1	**No satisfacció:** 5

Cas d'ús:

4.3.2. Usuari s'autentica

Actor Primari: Usuari.
Precondició: L'usuari es troba registrat al sistema.
Trigger: Cap.

Escenari Principal:
1. L'usuari proporciona les dades requerides.
2. El sistema valida que les dades són correctes.
3. El sistema mostra la zona d'usuari personalitzada.

Extensions:
2.a. Algunes de les dades no són correctes:
   2.a.1. El sistema comunica l'error.
   2.a.2. El cas d'ús continua en l'apartat 1 de l'escenari principal.

Satisfacció: 1	**No satisfacció:** 5

Cas d'ús:

4.3.3. Usuari compra una oferta

Actor Primari: Usuari.
Precondició: L'oferta ha estat habilitada per un administrador.
Trigger: Un usuari vol comprar una oferta.

Escenari Principal:
1. L'usuari selecciona l'oferta que vol comprar.
2. L'usuari omple els camps requerits
3. L'usuari accepta la compra
4. El sistema valida que les dades són correctes.
5. El sistema desplega les màquines virtuals que s'hagin comprat
6. El sistema crea la nova oferta comprada amb les dades introduïdes.
   [→ compraOferta]

Extensions:
1.a. L'usuari proporciona els camps no requerits
3.a. Algunes de les dades no són correctes:
   3.a.1. El sistema comunica l'error.
   3.a.2. El cas d'ús continua en l'apartat 1 de l'escenari principal.

Satisfacció: 1	**No satisfacció:** 5

Cas d'ús:

4.3.4. Usuari consulta el detall d'una oferta

Actor Primari: Usuari.
Precondició: L'oferta ha estat habilitada per un administrador.
Trigger: cap.

Escenari Principal:
1. L'usuari selecciona l'oferta de la qual vol conèixer els detalls.
2. El sistema mostra la informació referent a l'oferta seleccionada

Extensions:
1.a. L'usuari no es troba autenticat
   1.a.1. El sistema mostra la pantalla d'autenticació
   1.a.2. El cas d'ús continua en l'apartat 1 de l'escenari principal.

Satisfacció: 1	**No satisfacció:** 5

Cas d'ús:

4.3.5. Usuari consulta les ofertes comprades

Actor Primari: Usuari.
Precondició: L'usuari té com a mínim una oferta comprada.
Trigger: Cap.

Escenari Principal:
1. L'usuari selecciona la llista d'ofertes que ha comprat.
2. El sistema mostra la informació referent a les ofertes comprades.

Extensions:
1.a. L'usuari no es troba autenticat
   1.a.1. El sistema mostra la pantalla d'autenticació
   1.a.2. El cas d'ús continua en l'apartat 1 de l'escenari principal.

Satisfacció: 1	**No satisfacció:** 5

Cas d'ús:

4.3.6. Usuari consulta el detalls d'una oferta comprada

Actor Primari: Usuari.
Precondició: L'oferta ha estat comprada per l'usuari.
Trigger: cap.

Escenari Principal:
1. L'usuari selecciona l'oferta comprada de la que vol conèixer els detalls.
2. El sistema mostra la informació referent a l'oferta seleccionada

Extensions:
1.a. L'usuari no es troba autenticat
   1.a.1. El sistema mostra la pantalla d'autenticació
   1.a.2. El cas d'ús continua en l'apartat 1 de l'escenari principal.

Satisfacció: 1	**No satisfacció:** 5

Cas d'ús:

4.3.7. Usuari elimina una oferta comprada

Actor Primari: Usuari.
Precondició: L'oferta pertany a l'usuari.
Trigger: cap.

Escenari Principal:

1. L'usuari selecciona l'oferta que vol eliminar.
2. El sistema mostra un missatge de confirmació.
3. L'usuari accepta que vol eliminar l'oferta
4. El sistema esborra les màquines virtuals que s'havien desplegat
5. El sistema esborra l'oferta comprada.
6. El sistema mostra un missatge amb el resultat obtingut.
   [→ esborraOferta]

Extensions:

4.a. Les màquines virtuals no són accessibles:
   4.a.1. El sistema comunica l'error.
   4.a.2. El cas d'ús continua en l'apartat 1 de l'escenari principal.

Satisfacció: 1	**No satisfacció:** 5

Cas d'ús:

4.3.8. Usuari reinicia una oferta comprada

Actor Primari: Usuari.
Precondició: L'oferta pertany a l'usuari.
Trigger: cap.

Escenari Principal:

1. L'usuari selecciona l'oferta que vol reiniciar.
2. El sistema mostra un missatge de confirmació.
3. L'usuari accepta que vol reiniciar l'oferta
4. El sistema esborra les màquines virtuals que s'havien desplegat
5. El sistema crea les màquines virtuals de nou
6. El sistema actualitza l'estat de l'oferta.
7. El sistema mostra un missatge amb el resultat obtingut.
   [→ ressetejaOferta]

Extensions:

4.a. Les màquines virtuals no són accessibles:
   4.a.1. El sistema comunica l'error.
   4.a.2. El cas d'ús continua en l'apartat 1 de l'escenari principal.

5.a. No es disposa de recursos per crear les màquines virtuals accessibles:
   5.a.1. El sistema comunica l'error.
   5.a.2. El cas d'ús continua en l'apartat 1 de l'escenari principal.

Satisfacció: 1	**No satisfacció:** 5

Cas d'ús:

4.3.9. Usuari accedeix a una oferta desplegada

Actor Primari: Usuari.
Precondició: L'oferta pertany a l'usuari.
Trigger: cap.

Escenari Principal:
1. L'usuari selecciona l'oferta a la que vol accedir.
2. El sistema mostra les màquines virtuals de l'oferta.
3. L'usuari selecciona la maquina virtual a la que vol accedir
4. El sistema inicia la sessió vnc per interactuar amb la màquina virtual

Extensions:
4.a. Les màquines virtuals no són accessibles:
   4.a.1. El sistema comunica l'error.
   4.a.2. El cas d'ús continua en l'apartat 1 de l'escenari principal.

Satisfacció: 1	**No satisfacció:** 5

Cas d'ús:

4.3.10. Administrador habilita una oferta

Actor Primari: Administrador.
Precondició: L'oferta es troba configurada.
Trigger: Cap.

Escenari Principal:
1. L'administrador selecciona l'oferta que vol habilitar.
2. El sistema registra l'oferta com apta per a ser comprada.

Extensions:
1.a. L'administrador no es troba autenticat
   1.a.1. El sistema mostra la pantalla d'autenticació
   1.a.2. El cas d'ús continua en l'apartat 1 de l'escenari principal.

Satisfacció: 1	**No satisfacció:** 5

Cas d'ús:

4.3.11. Administrador des-habilita una oferta

Actor Primari: Administrador.
Precondició: L'oferta es troba habilitada.
Trigger: Cap.

Escenari Principal:
1. L'administrador selecciona l'oferta que vol des-habilitar.
2. El sistema registra l'oferta com oculta.

Extensions:
1.a. L'administrador no es troba autenticat
   1.a.1. El sistema mostra la pantalla d'autenticació
   1.a.2. El cas d'ús continua en l'apartat 1 de l'escenari principal.

Satisfacció: 1	**No satisfacció:** 5

Cas d'ús:

4.3.12. Administrador configura una oferta

Actor Primari: Administrador.
Precondició: L'oferta es troba creada.
Trigger: Cap.

Escenari Principal:
1. L'administrador selecciona l'oferta que vol configurar.
2. El sistema mostra el formulari de configuració d'ofertes.
3. L'administrador omple els camps requerits.
4. L'administrador accepta crear la nova oferta.
5. El sistema marca l'oferta com a activa.

Extensions:
1.a. L'administrador no es troba autenticat
   1.a.1. El sistema mostra la pantalla d'autenticació
   1.a.2. El cas d'ús continua en l'apartat 1 de l'escenari principal.

3.a. L'administrador omple els camps no requerits

4.a. Els camps introduïts són incorrectes
   4.a.1. El sistema comunica l'error.
   4.a.2. El cas d'ús continua en l'apartat 3 de l'escenari principal.

Satisfacció: 1	**No satisfacció:** 5

Cas d'ús:

4.3.13. El sistema notifica un event per correu

Actor Primari: Sistema.
Precondició: El sistema es troba executant-se.
Trigger: ressetejaOferta || compraOferta || esborraOferta || registraUsuari.

Escenari Principal:

1. El sistema consulta les dades de l'usuari a qui enviar el correu
2. El sistema crea un missatge segons el trigger rebut.
3. El sistema envia el correu.

Extensions:

3.a. L'usuari té un correu incorrecte
   3.a.1. El sistema envia l'error a l'administrador

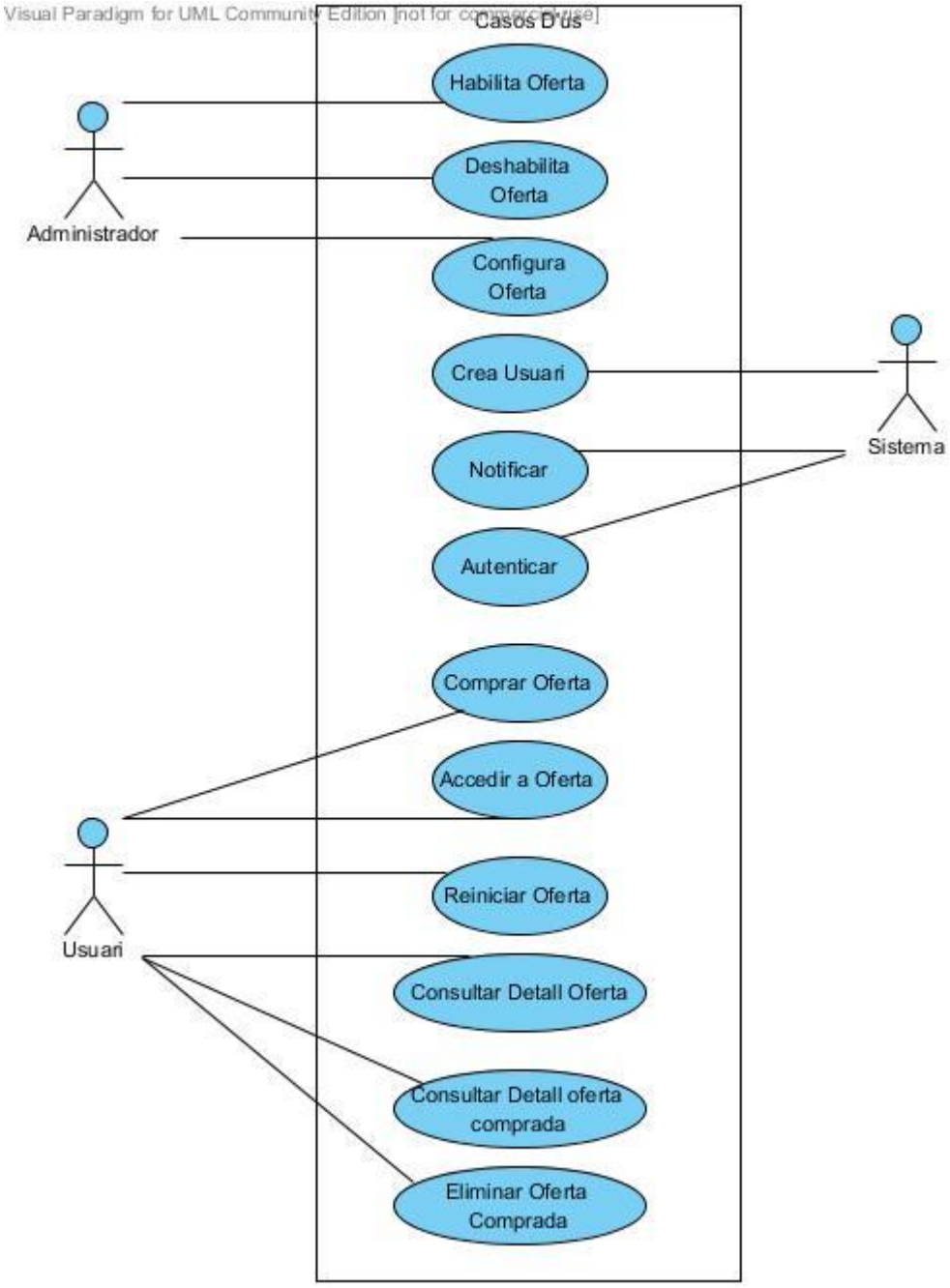
Satisfacció: 1	**No satisfacció:** 5

CAPÍTOL 5. DISSENY

5.1. Casos d'ús de sistema

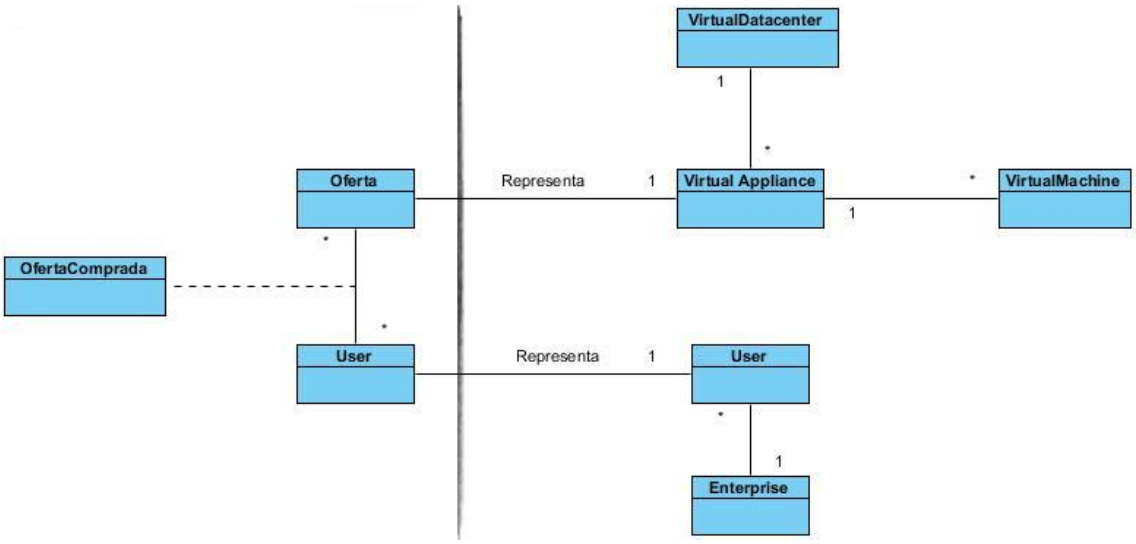
A continuació, començarà el disseny de la plataforma, formalitzant els casos d'ús per a definir l'arquitectura del sistema.

5.1.1. Diagrama de casos d'ús de sistema



5.1.2. Diagrama de classes

En aquesta secció es mostrarà el diagrama de les classes utilitzades en el portal per representar les diverses entitats del sistema i la relació amb Abiquo.



Com es pot comprovar algunes de les entitats es troben en el portal i altres a la plataforma d 'Abiquo (separades amb una línia vertical). D'aquesta manera no caldrà que les entitats remotes siguin persistents en el portal, ja què la informació es recuperarà sota demanda des de Abiquo, aprofitant els avantatges que ens ofereix l'API Rest. D'aquesta manera podrem garantir la fiabilitat en les dades, ja què sempre que es mostrin representaran el que s'ha obtingut directament dels hipervisors.

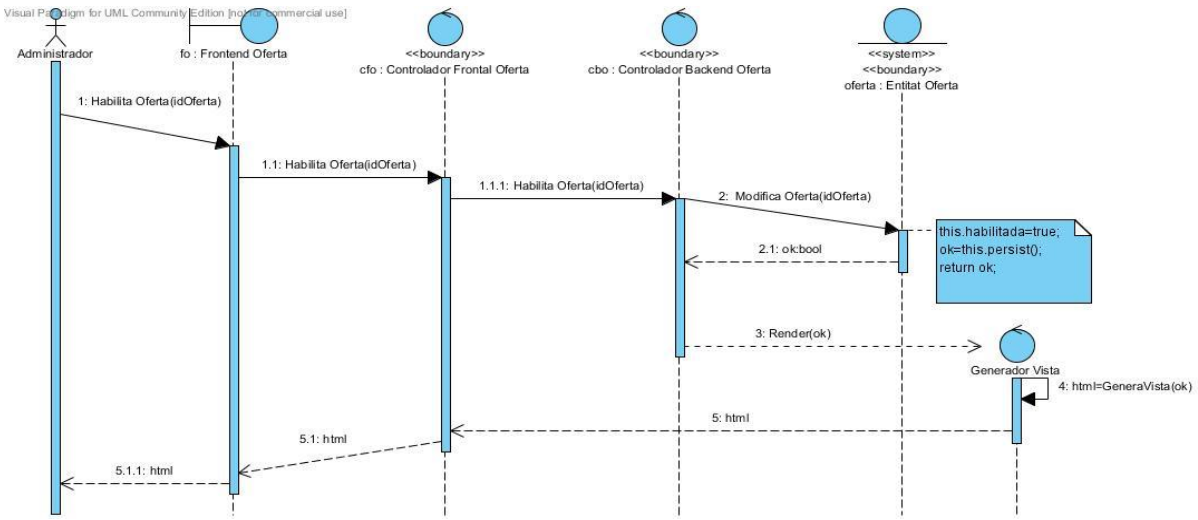
Es va plantejar la redundància en la base de dades per garantir la recuperació d'aquestes en cas de fallada de la connexió, tot i què si no es disposa de connexió tampoc es podrà accedir a les màquines virtuals desplegades, i només representaria una informació que no hauria de ser fiable. Per tant, es va optar per una solució completament distribuïda recuperant les dades quan es necessitin.

5.1.3. Diagrama d'activitat

En aquesta secció mostrem els diagrames d'activitat més rellevants de la plataforma.

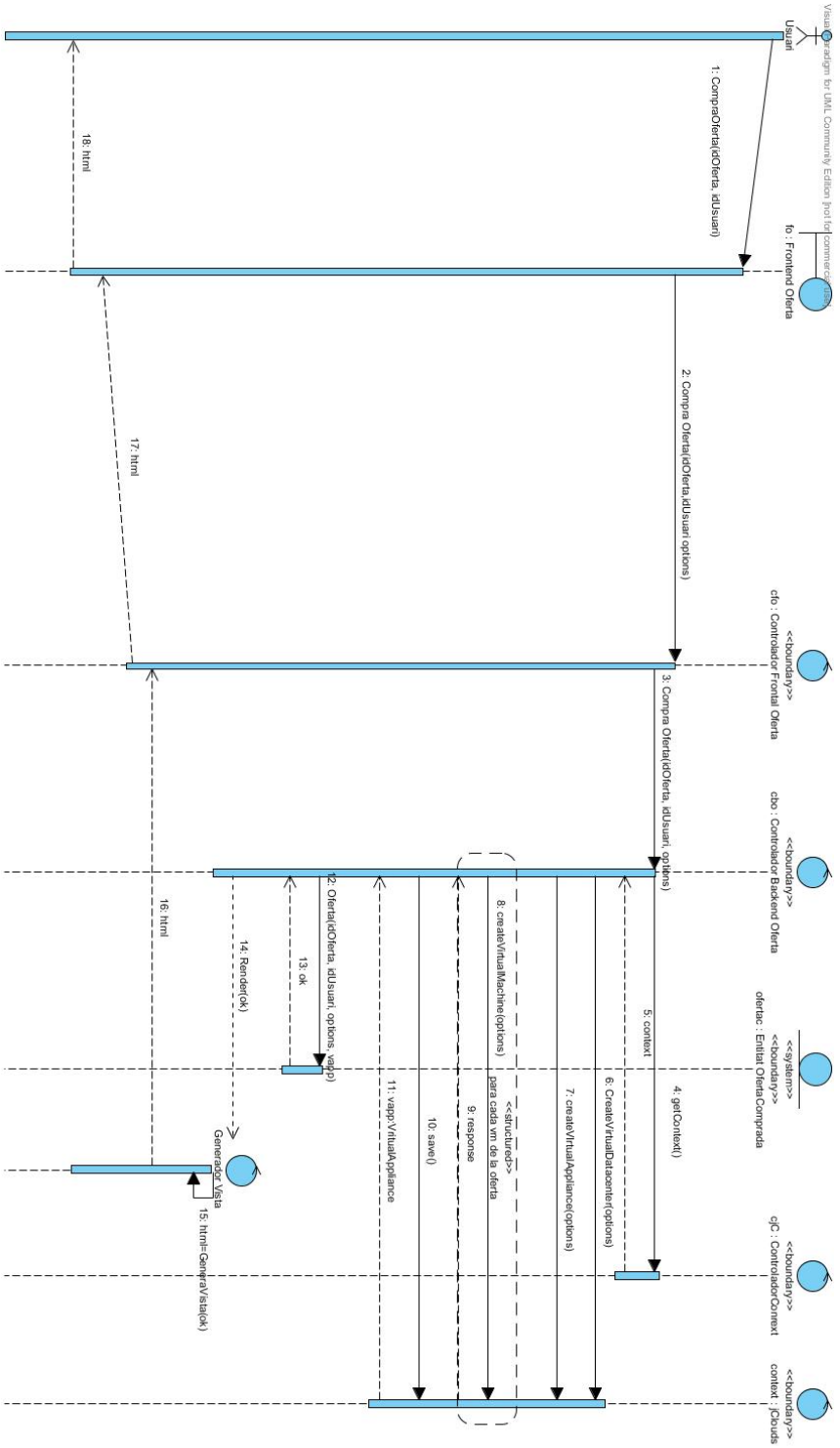
5.1.3.1. Habilitar una oferta

En aquest diagrama d'activitat l'administrador habilita una oferta en el portal, i aquesta passa a ser visible pels usuaris que vulguin comprar-la, seguint el patró MVC de play.



5.1.3.2. Comprar una oferta

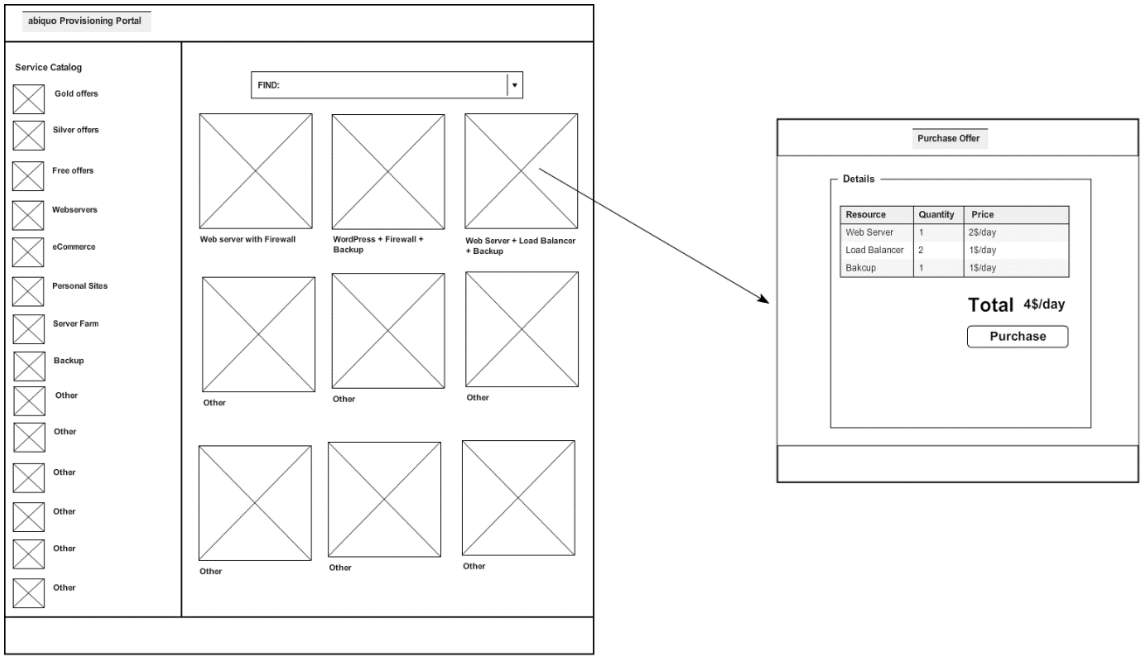
Aquest diagrama mostra el flux d'un usuari que desitgi realitzar una compra, amb la creació del centre de dades virtual, l'aplicació virtual i les màquines virtuals corresponents a l'oferta comprada a Abiquo, mitjançant jClouds. El procés de creació del context i de la comunicació amb la API Rest es troba descrit en la secció 6.3.1


Usuari
fo : Frontend Oferta
cfo : Controlador Frontal Oferta
cbo : Controlador Backend Oferta
ofertac : Entitat OfertaComprada
cjC : ControladorConnext
context : jClouds
Generador Vista
1: CompraOferta(idOferta, idUsuari)
2: Compra Oferta(idOferta,idUsuari options)
3: Compra Oferta(idOferta, idUsuari, options)
4: getContext()
5: context
6: CreateVirtualDatacenter(options)
7: createVirtualAppliance(options)
8: createVirtualMachine(options)
para cada vm de la oferta
9: response
10: save()
11: vapp.VirtualAppliance
12: Oferta(idOferta, idUsuari, options, vapp)
13: ok
14: Render(ok)
15: html=GenerarVista(ok)
16: html
17: html
18: html
<<boundary>>
<<system>>
<<structured>>

5.2. Wireframes

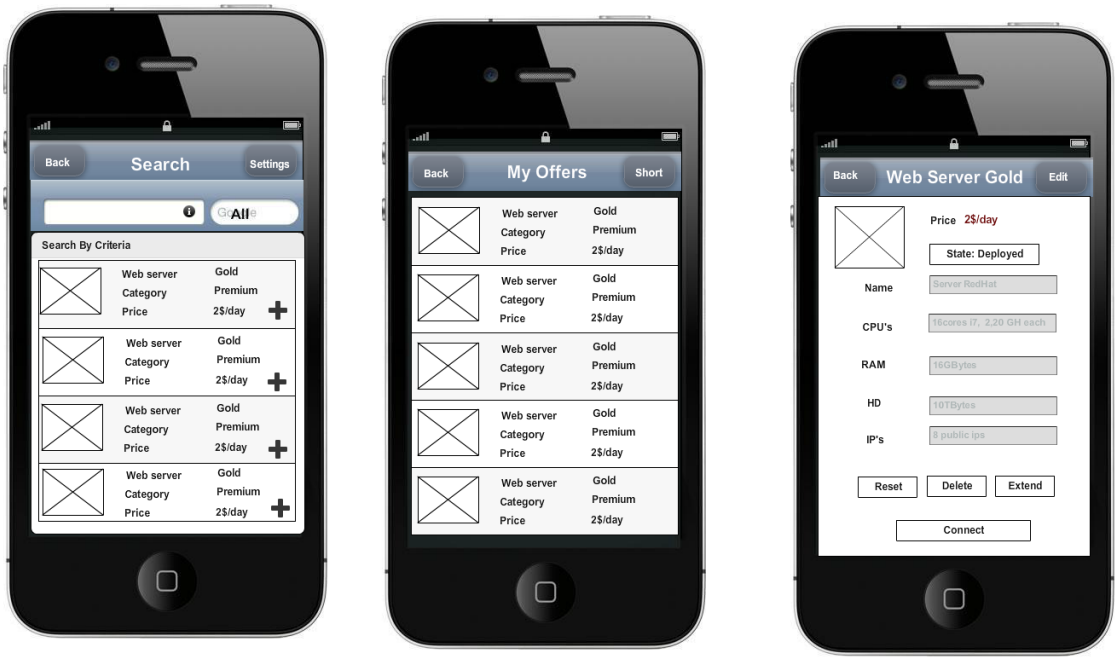
5.2.1. Wireframes portal web

Pel que fa a l'aplicació web, mostrem el wireframe del cas d'ús més important de la plataforma: el que involucra la compra d'una oferta i el corresponent desplegament de les màquines virtuals que continguin l'oferta.
La pàgina d'inici d'usuaris (un cop es trobin identificats) serà com la que es mostra en la primera imatge. Una llista de tipus de serveis amb ofertes (aplicacions cloud) organitzades per categories. L'usuari podrà buscar des d'un servidor web fins a un servei de backup o de prova.
Després d'escollir l'aplicació que es vulgui comprar, sortiran els detalls de l' oferta, mostrant quines màquines virtuals es desplegaran i els recursos d'aquestes (CPU, RAM, HD, IP's, etc), a més del preu de l'oferta en qüestió. Si l'usuari decideix realitzar la compra, el sistema afegiria la nova aplicació a l'àrea personal de l'usuari i desplegaria els recursos pertinents en el IaaS per tal de crear les màquines virtuals requerides i donar accés al servei contractat.



5.2.2. Wireframes portal mobile

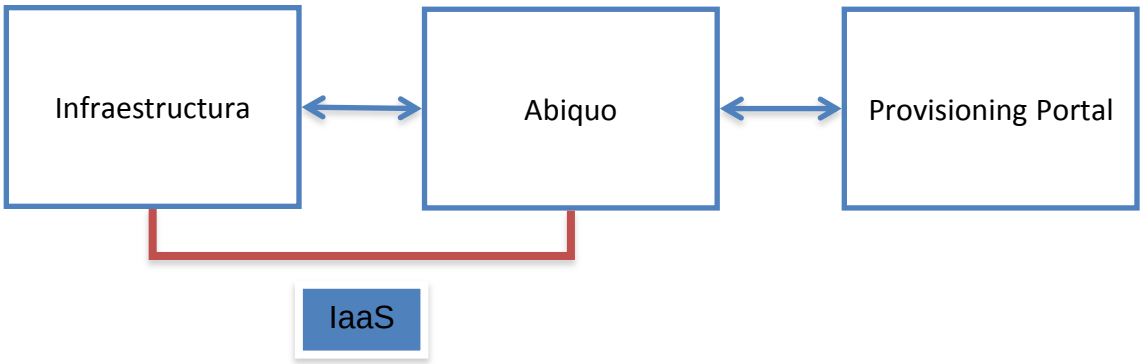
La versió mòbil compleix gairebé tots els casos d'ús de l'aplicació, excepte la administració i habilitat de les ofertes, ja què es va decidir que l'administració fos només web. No obstant això, l'administrador té l'opció de llistar les ofertes per veure en l'estat que es troben.
L'usuari, en canvi, té control absolut de les aplicacions cloud que compri, és més, s'implementarà un servei per facilitar la connexió amb les màquines virtuals a través de l'aplicació web.



CAPÍTOL 6. IMPLEMENTACIÓ

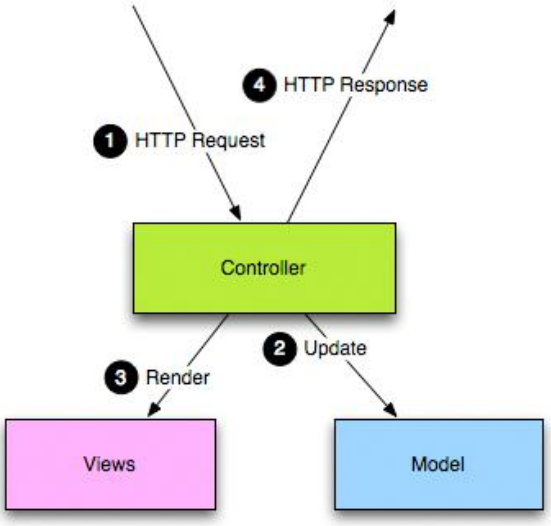
6.1. Arquitectura provisioning portal

6.1.1. Diagrama de tecnologies



6.1.2. El model MVC

Una aplicació desenvolupada en Play segueix el patró per a aplicacions web conegut com MVC (Model-Vista-Controlador).
Aquest patró organitza l'aplicació en capes separades: la de presentació i la de model. A més, la capa de presentació es separa en una capa de vista i una altra de controlador.
En el Model es representen les entitats del sistema, implementades com a classes Java. En el model de MVC no es parla de la persistència de dades, ja què es considera que es troba per sota o bé encapsulada dins del model.
La vista desplega la informació del model de la forma apropiada perquè l'usuari interactuï amb aquesta. Es poden crear múltiples vistes per un model segons convingui, i tot i que comunament en formats web s'espera HTML, JSON o XML hi haurà casos on el tipus sigui personalitzat, com per exemple renderitzar un binari.
El controlador es qui atén els esdeveniments (normalment accions generades per l'usuari) i els processa. Típicament les peticions són del tipus HTTP, d'on s'extreu la informació rellevant com podrien ser paràmetres o les capçaleres i aplica els canvis al model segons convingui. També s'encarrega de generar la resposta segons els resultats obtinguts i informar a l'usuari.



En una aplicació de Play aquestes tres capes es defineixen en el directori app, cadascuna en un paquet de Java diferent:

app/controllers

Un controlador és una classe de java en la que cada mètode públic i estàtic representa una acció o "action method". Una acció és un punt d'entrada a una aplicació Java que s'invoca quan arriba una petició HTTP. El codi Java del controlador en realitat no està orientat a objectes, és meta codi. L'action method extreu la informació rellevant de la petició HTTP, llegeix o actualitza el model d'objectes i retorna el resulta creant una resposta HTTP.

app/models

Representa la capa del model d'objectes del domini. Són un conjunt de classes de Java que utilitzen l'orientació a objectes que proporciona Java. Conté estructures de dades (estat) i operacions (comportament) sobre les que opera la aplicació. En cas que el model d'objectes requereixi persistència de dades pot contenir directives, com ara notacions JPA o sentències SQL.

app/views

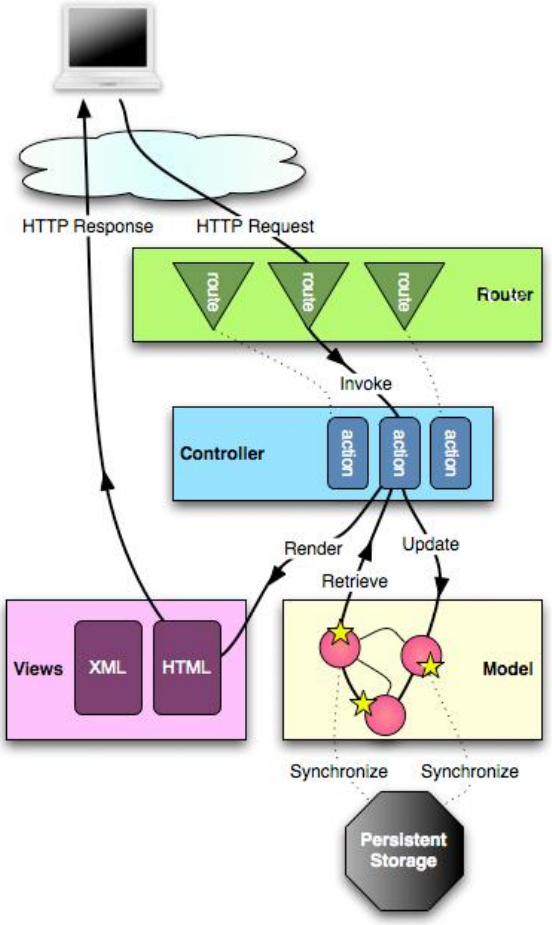
La majoria de les vistes de l'aplicació són generades per un motor de "templating" proveït per Play. El controlador obté la informació consultant a la capa de dades (si es necessari) i posteriorment aplica un "template" per decorar aquests objectes. Aquesta capa conté HTML, XML, JSON i altres fitxers de "templates" pensats per generar dinàmicament la representació del model

El cicle de vida de la petició

El framework de Play és completament “stateless” (sense estat), orientat exclusivament a petició/resposta. Totes les peticions HTTP segueixen el mateix camí:

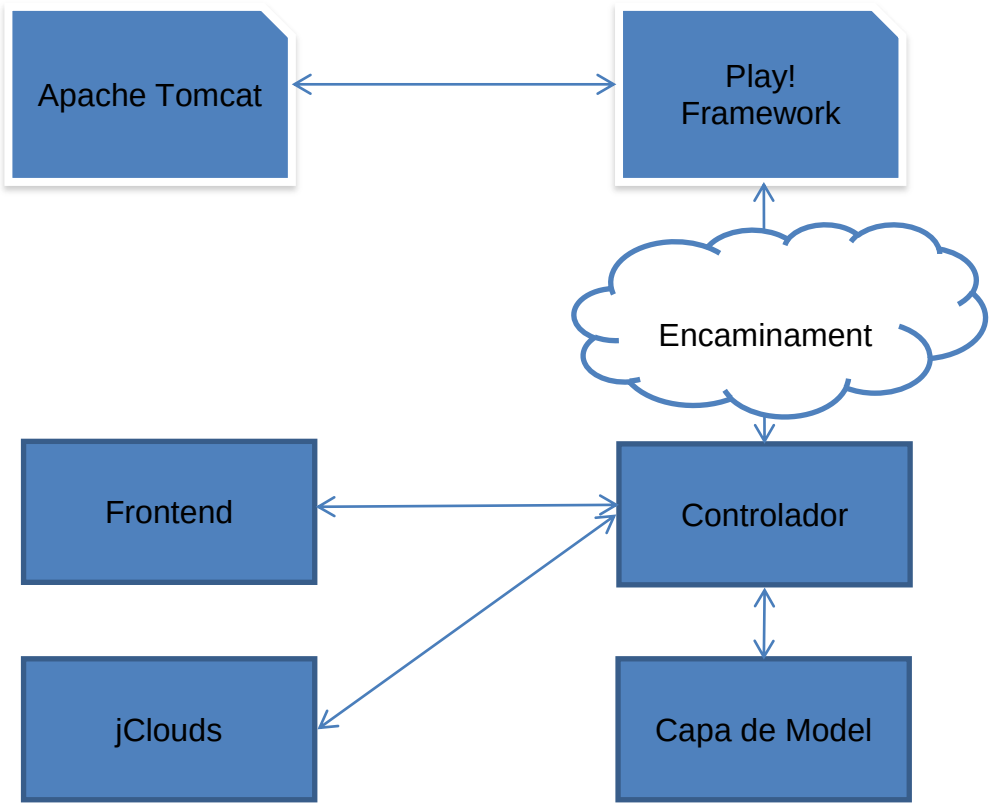
- Es registra una petició HTTP en el framework.
- El *router* (enrutador) intenta trobar la millor ruta per atendre la petició, i invoca al action method corresponent.
- S’executa el codi de la aplicació
- Si es necessita generar una vista complexa, s’avalua un template encarregat de generar-la
- El resultat del action method es retorna com una resposta HTTP.

El diagrama següent mostra el cicle de vida d’una petició HTTP:



6.1.3. Comunicació amb el IaaS

Després de veure el cicle d'una petició bàsica en play, fa falta definir com es farà la comunicació amb el IaaS i quina persistència de dades utilitzarem en el portal cloud. A més, haurem de definir com integrar l'aplicació mòbil en tot aquest procés.
Les comunicacions que falten definir són les següents



Apache Tomcat

Contenidor dels CGI's encarregats d'atendre i gestionar les peticions al portal cloud. És una implementació del popular servei web Apache adaptat a aplicacions Java.

Frontend

És el controlador que gestiona la capa de presentació. En aquest punt, segons des de quin dispositiu s'hagi realitzat la petició es renderitzarà el template corresponent a l'aplicació d'escriptori o a la aplicació mòbil. És degut a aquesta versatilitat alhora de triar la presentació que ens vam decantar per utilitzar jquery Mobile, ja què les plantilles es defineixen en html amb *tags* especials.

jClouds

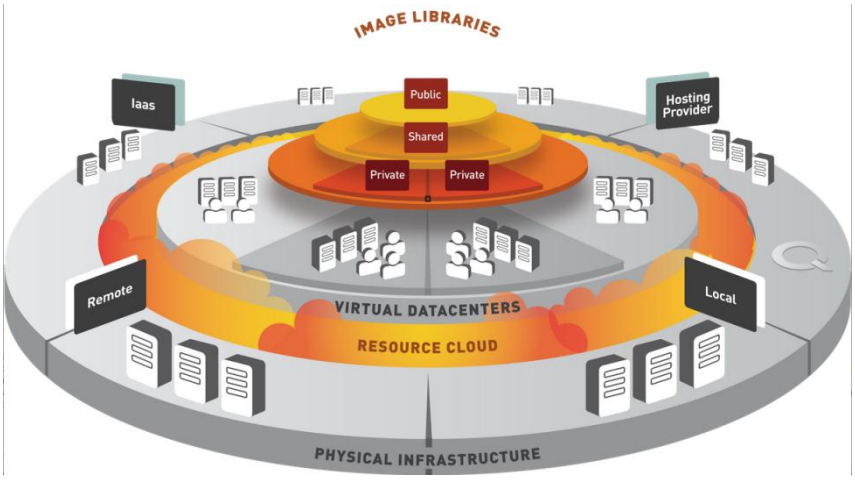
Utilitzarem les llibreries de jClouds per comunicar-nos amb la API Rest de Abiquo i realitzar les peticions al IaaS. El fet d'utilitzar jClouds ens permetrà encapsular les peticions i millorar la reutilització de codi.

Capa de Model

En aquesta capa definirem i representarem les entitats que representaran el model del provisioning portal. Per fer-ho, utilitzarem JPA en la definició, que ens permetrà realitzar canvis durant el desenvolupament, i treballar amb un ORM gestionant les entitats com a objectes. El ORM que utilitzarem serà Hibernate, degut a la gran comunitat que l'utilitza i la robustesa del producte, y la base de dades serà mySQL, ja que es opensource i s'ajusta a les necessitats del portal.

6.2. Abiquo

6.2.1. El recurs cloud



Abiquo integrat en la infraestructura física

La visió de Abiquo sobre aprovisionar infraestructura física es troba totalment separada de la aplicació virtual per un recurs cloud. La infraestructura física, normalment gestionada pel departament de IT, aprovisiona els recursos que utilitzaran posteriorment les empreses que contenen centres de dades, aplicacions i màquines virtuals.

Actualment, parlem de recursos com a nuclis de CPU, memòria, emmagatzematge i connectivitat, tot i que a mesura que evolucioni el cloud computing parlarem de temes més universals.

El principal gran canvi es que les organitzacions de TI deleguen el aprovisionament dels recursos a les empreses virtuals, les quals són gestionades per un administrador i tenen limitats els recursos físics als que poden accedir, permetent que els administradors de sistemes es puguin dedicar a altres tasques de manteniment i que els propis treballadors administrin els recursos que necessitin de forma controlada.

6.2.1.1. Administració del a càrrega de treball

El segon gran canvi és la decisió d'on es desplegarà la màquina virtual. Aquesta es fa de forma automàtica, segons la política que es defineixi, permetent polítiques de aïllament entre xarxes o d'ocupació de recursos de forma balancejada.

6.2.1.2. Elasticitat

Els recursos físics poden ser proveïts per un centre de dades local, remot o per tercers com ara clouds públics. Combinant aquests recursos amb les polítiques definides es poden utilitzar els recursos de forma escalada i segons les necessitats dels clients.

6.2.1.3. Beneficis

Gràcies a aquesta organització es redueix la càrrega de treball en el departament de TI, delegant responsabilitats de forma segura i controlada. Donant accés a les empreses virtuals no es tindrà accés a la infraestructura física, i els recursos de que disposin es trobaran restringits. Això redueix dràsticament el temps de preparació dels entorns de treball, millorant la productivitat de l'equip.

6.2.2. Arquitectura a Abiquo

El software de Abiquo transforma els servidors, l'emmagatzematge i la xarxa existent en una Infraestructura com a servei. A més es pot integrar amb el sistema intern de IT a traves de la API Rest. Tota la plataforma es troba dividida en mòduls per millorar la escalabilitat del sistema.

A continuació es descriuran els components més importants de la plataforma i la posició que ocupa en el desenvolupament del IaaS.

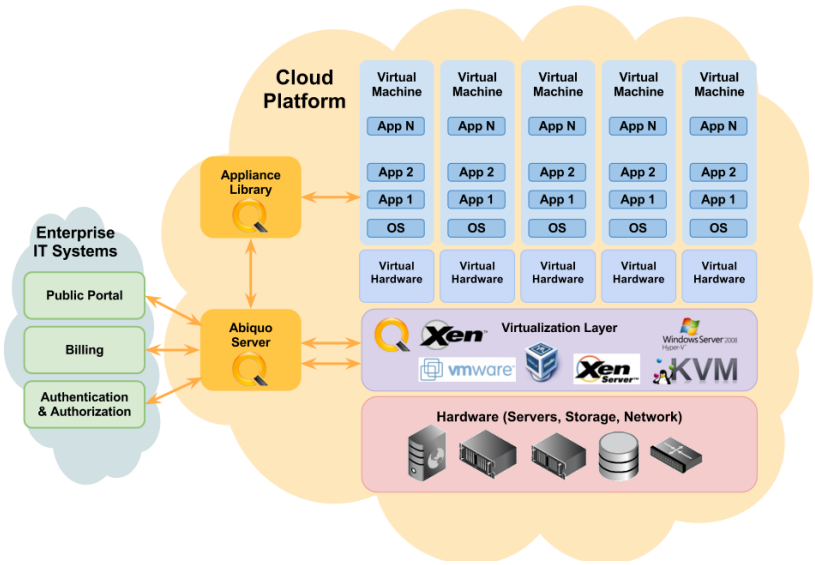


Figura 1. Posició de Abiquo en l'arquitectura d'infraestructura cloud.

En la imatge (*Figura 1*) es mostra com Abiquo es comunica amb els Hipervisors de diverses marques i amb serveis externs, entre ells, el portal públic de l'empresa, el qual representa el provisioning portal descrit en aquest document.

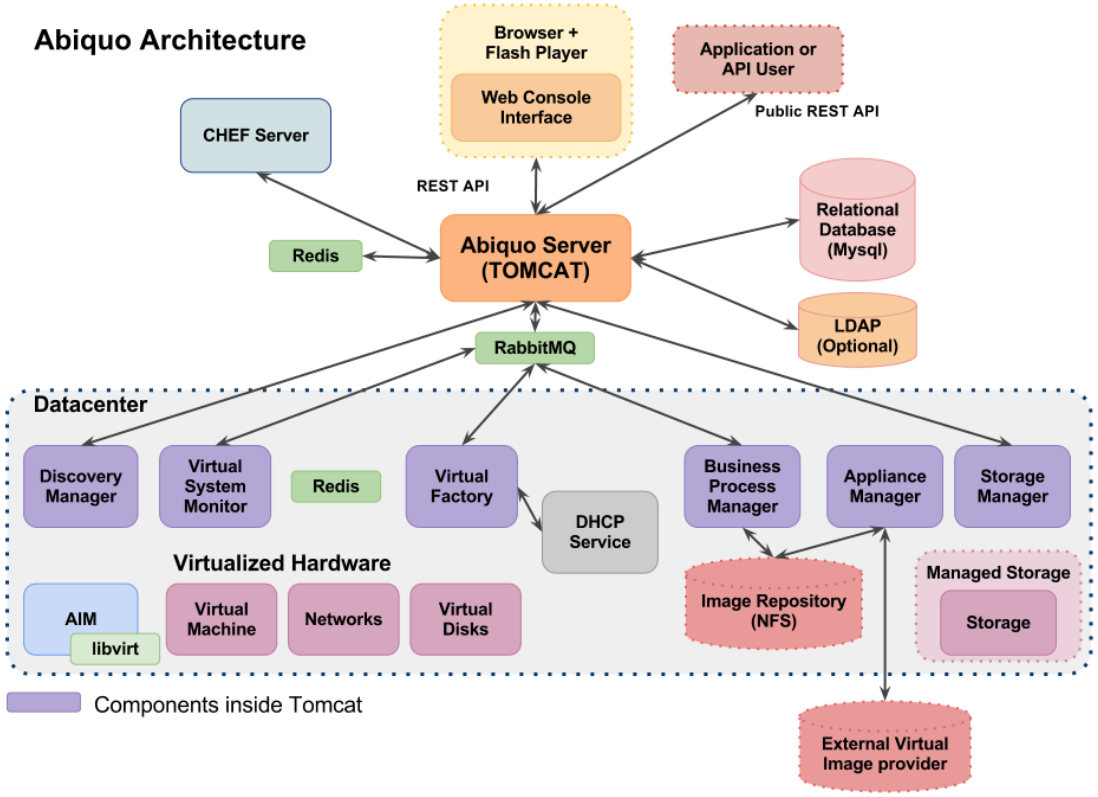


Figura 2. Descripció de l'arquitectura d'Abiquo.

En la imatge superior es mostren els components de la plataforma i la connexió entre ells. Cal destacar el mòdul API User, el qual facilita la API REST de la plataforma perquè els clients puguin utilitzar-la. És a través d'aquest mòdul com el provisioning portal es comunicarà amb Abiquo, ja què les peticions través de jClouds s'atendran en aquest punt. La comunicació del portal i jClouds es definirà en punt 6.3.1. Connexió entre el portal i el Iaas.

A continuació, es farà una breu descripció dels mòduls més importants del IaaS:

6.2.2.1. Datacenters

Un centre de dades (*datacenter*) està format per:

- ***Un conjunt de hipervisors***
- ***Serveis d'emmagatzematge***
- ***Sistema de fitxers en xarxa (NFS)***
- ***Servidor DHCP***
- ***Serveis remots de Abiquo***

6.2.2.2. Hipervisors

Abiquo treballa amb múltiples hipervisors i conversions entre ells. Els classifiquem a continuació:

Gestionats a través de la API que ofereixen

- ***ESXi***
- ***XenServer***
- ***Hyper-V***
- ***VirtualBox***

Gestionats a través del servei Abiquo AIM (agent basat en libvirt)

- ***KVM***
- ***Xen***

6.2.2.3. Aim

Es el servei d 'Abiquo encarregat de la gestió dels hipervisors que no disposen de API per realitzar tasques.

Per KVM i Xen permet:

- ***Copia de discs***
- ***Configurar Vlans***
- ***Monitoritzar les màquines***

6.2.2.4. Abiquo remote services

La plataforma d 'Abiquo utilitza els següents serveis remots per a gestionar els hipervisors del centre de dades:

- **Appliance Manager**

Permet pujar i descarregar plantilles de maquines virtuals al *repositori* del centre de dades, les quals s'utilitzaran en la creació de les màquines virtuals.

- **Business Process Manager**

S'encarrega de la conversió d'imatges entre diferents formats de disc suportats pels hipervisors.

- **Discovery Manager**

Permet investigar al hipervisor i descobrir les dades referents a:
- Tipus d'hipervisor
- Màquines virtuals desplegades
- Capacitats físiques de la màquina i dels recursos

- **Virtualization Manager**

És la capa encarregada d'unificar i gestionar les capacitats de virtualització per cada hipervisor. Gestiona el cicle de vida de les maquines virtuals,
Les capacitats de xarxa ai d'emmagatzematge.
Les peticions de treballs i les respostes es gestionen a través de cues de missatges implementades en RabbitMQ, basat en un model d'actors utilitzant Akka.

- **Storage System Manager**

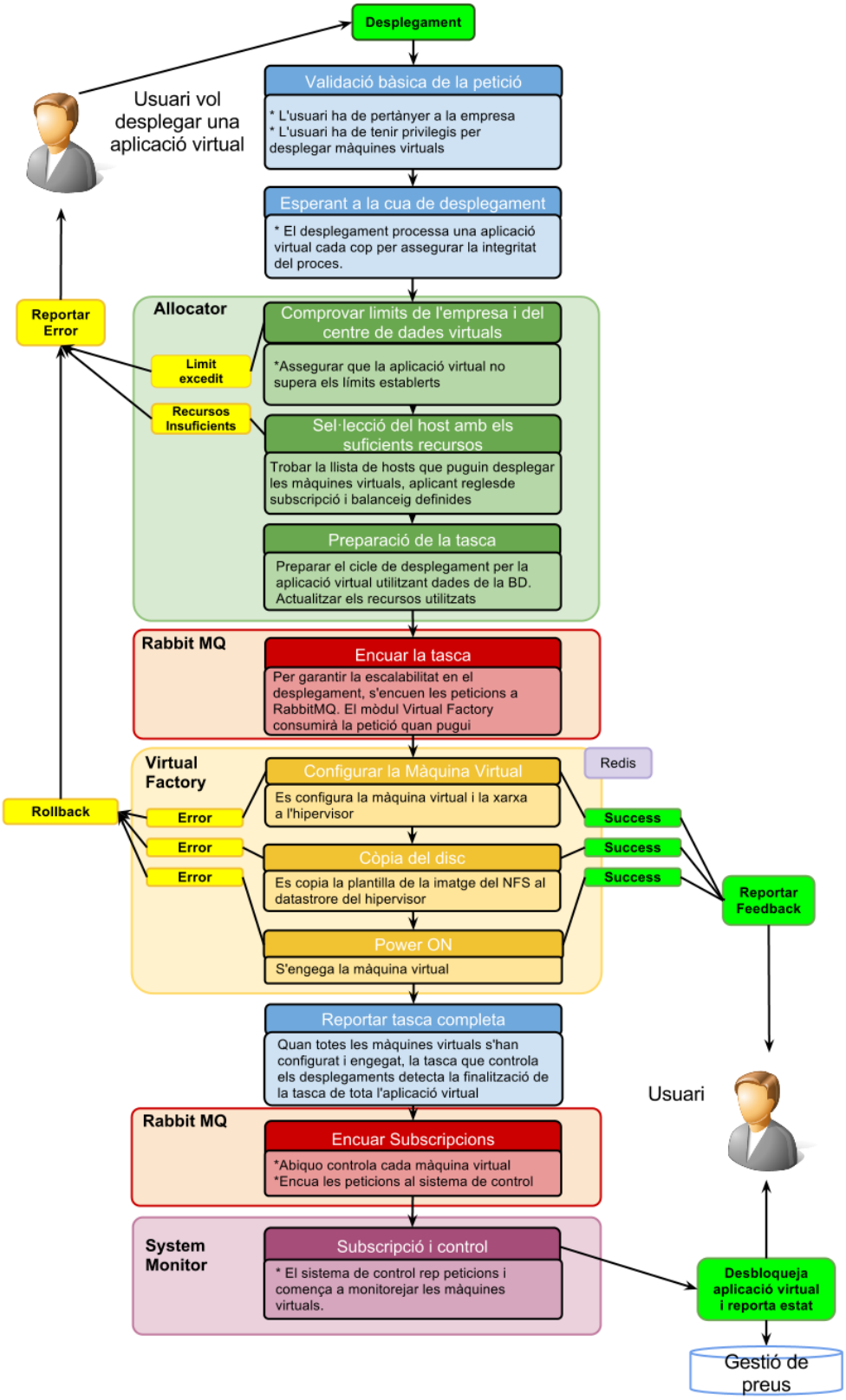
És la capa encarregada d'unificar i gestionar les capacitats dels servidors d'emmagatzematge a través de volums iSCSI.

- **Virtual System Monitor**

Gestiona un conjunt de monitors. Cadascun d'ells controla l'estat de les màquines virtuals i notifica els canvis. Utilitza Redis per gestionar les subscripcions i el mecanisme de publicació-subscripció pels events de cada monitor. Els canvis d'estat es notifiquen mitjançant cues implementades en RabbitMQ.

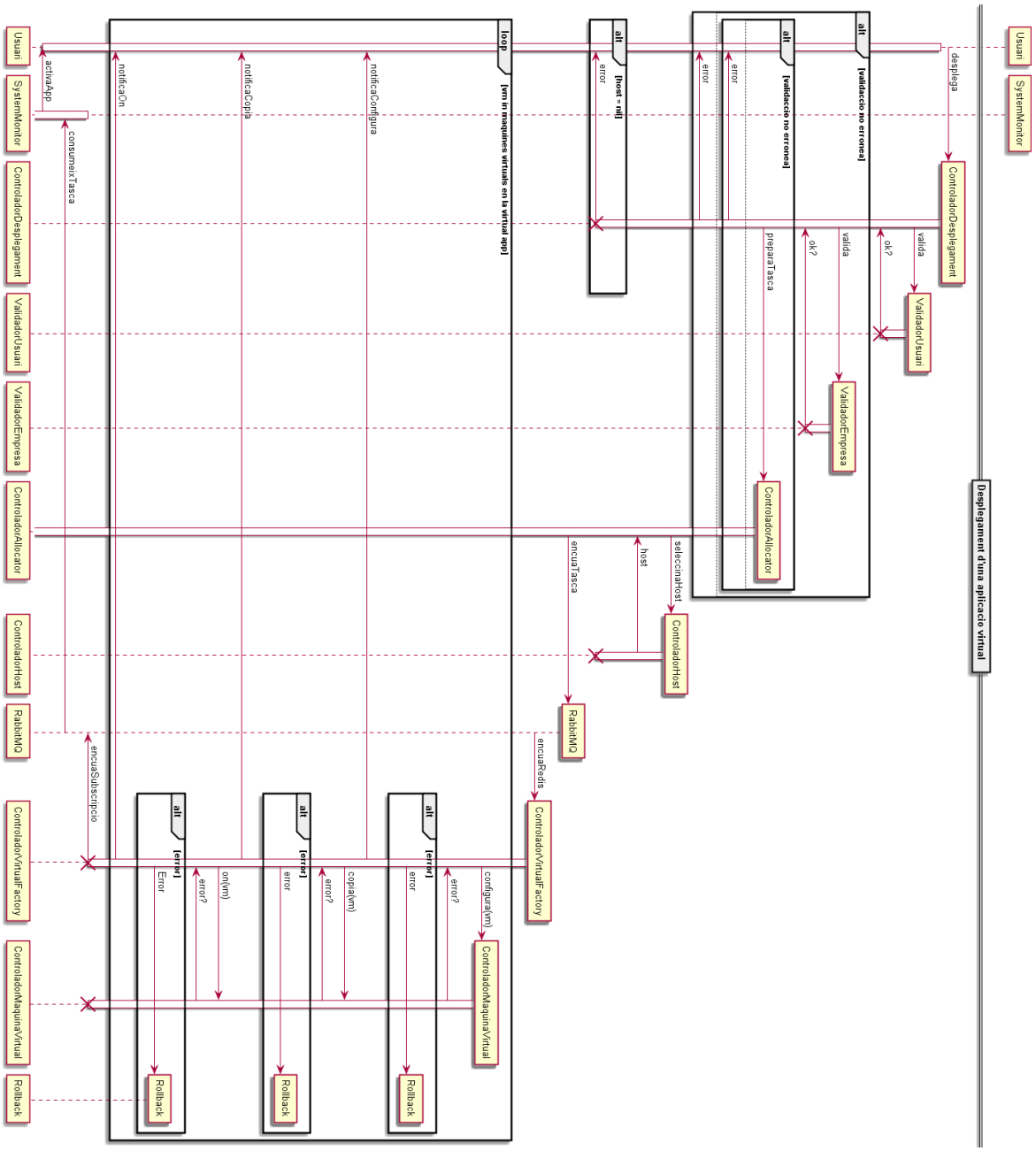
6.2.3. El procés de desplegament

El procés de desplegament de màquines virtuals en la infraestructura física es fa seguint l'esquema següent. Quan un usuari genera la petició de desplegar una aplicació virtual comença el procés de desplegament.



6.2.4. Diagrama de desplegament

Aquest diagrama mostra el procés de desplegament d'una aplicació virtual a Abiquo.



6.3. Implementació del portal

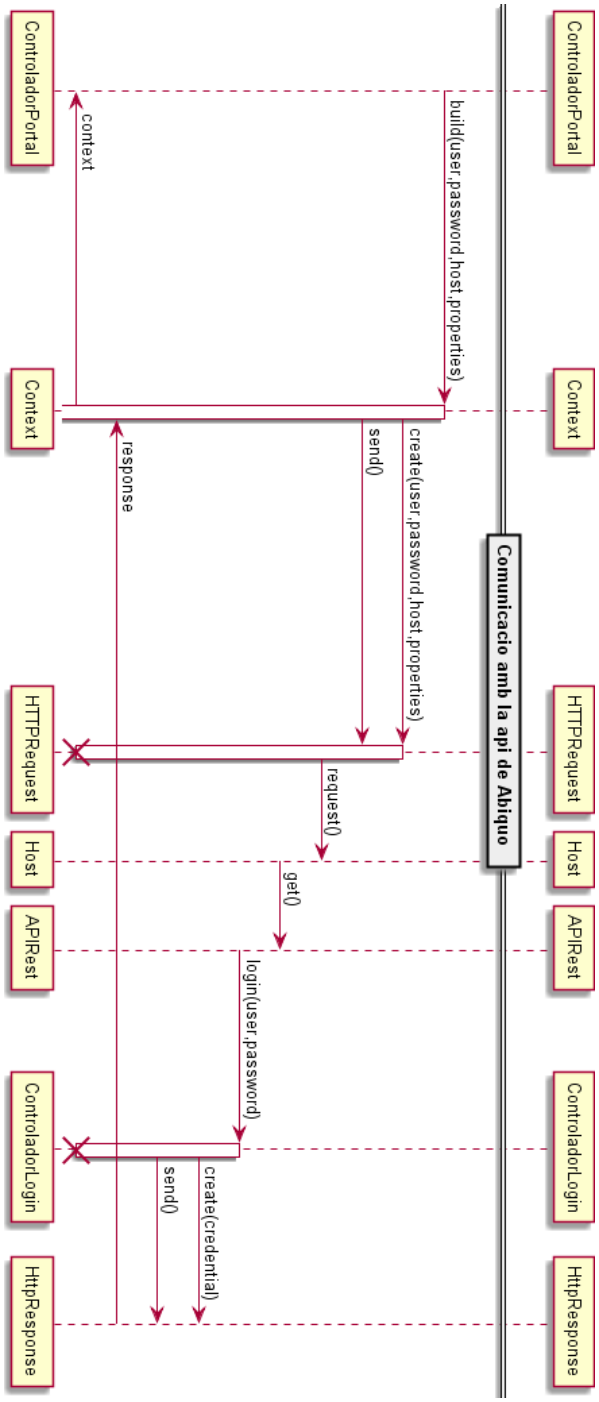
En aquesta secció es detallarà la implementació del portal segons les tecnologies escollides i el disseny detallat anteriorment.

6.3.1. Connexió entre el portal i el IaaS

Abans de començar a descriure la implementació dels casos d'ús més rellevants, descriurem els conceptes relacionats amb la comunicació entre el portal i el IaaS a través de la plataforma de Abiquo i jClouds, mostrant exemples de comunicació i com simplificar la gestió de peticions de la API REST.

6.3.1.1. Diagrama de treball

El diagrama següent mostra la connexió entre els controladors del portal i la API de Abiquo mitjançant jClouds.



Com es pot veure els controladors del portal creen un objecte Context, el qual pertany a jClouds, a partir de les credencials d'usuari. Aquest es crea mitjançant una petició a la API Rest que es troba escoltant al host que s'hagi instal·lat. Un cop creat el context podrem treballar amb aquest i realitzar les peticions directament a jClouds

6.3.1.2. Construint el context

A jClouds el context es qui s'encarrega de gestionar les peticions a la API que s'hagi especificat aprofitant les avantatges de la orientació a objectes. D'aquesta manera, un cop disposem del context no haurem de tornar a autenticar-nos, i les peticions que es facin seran sempre a través d'objectes, simplificant la feina de *serialitzar* i construir els Dto's per a cada petició.

```
public static AbiquoContext getApiClient(final String username,
        final String password) {
    if (username != null && password != null) {
        Properties props = new Properties();
        // load a properties file
        try {
            props.load(new FileInputStream(Play.getFile("conf/config.properties")));
            props.put("abiquo.endpoint", props.getProperty("api"));
            String token = generateToken(userSession);

            //We will use token based authentication
            props.setProperty(AbiquoProperties.CREDENTIAL_IS_TOKEN,"true");
```

En el codi anterior mostrem el mètode de la classe Context encarregat de realitzar la tasca definida anteriorment. En aquesta porció de codi obtenim la propietat definida en el fitxer *config.properties* que conté la direcció de la API REST on es connectarà el context per realitzar les peticions. En aquest cas, veiem que obtenim la propietat *api* i la guardem en les propietats com a *abiquo.endpoint*. A més, generarem un *token* utilitzant la sessió d'usuari per realitzar les connexions.

```
// Do not retry methods that fail with 5xx error codes
props.put("jclouds.max-retries", "0");
// Custom timeouts in ms
// External storage operations take a while in some storage
// devices
props.put("jclouds.timeouts.CloudClient.createVolume", "90000");
props.put("jclouds.timeouts.CloudClient.updateVolume", "90000");
props.put("jclouds.timeouts.CloudClient.replaceVolumes","90000");
props.put("jclouds.timeouts.CloudClient.deleteVolume", "90000");
props.put("jclouds.timeouts.CloudClient.makePersistentVirtualMachine",
        "300000");

context = ContextBuilder.newBuilder(new AbiquoApiMetadata()) //
        .endpoint(props.getProperty("api")) //
        .credentials(username, password) //
        .modules(ImmutableSet.<Module> of(new SLF4JLoggingModule())) //
        .overrides(props) //
        .build(AbiquoContext.class);
```

En aquesta part es defineixen vàries propietats sobre el temps d'espera abans de considerar error en operacions concretes. Finalment, es crea el context, a partir de les propietats definides, les credencials d'usuari i informació referent a Abiquo.

Cal destacar que jClouds es un client que suporta una gran varietat de proveïdors cloud, i per això hem d'especificar que el context que crearem es per realitzar peticions a una api REST que controla una distribució de Abiquo.
A partir d'aquest moment, ja podrem utilitzar el context en els controladors del provisioning portal.

6.3.1.3. API de Abiquo

Com s'ha vist anteriorment, jClouds realitzarà peticions a la API REST que se li indiqui, en aquest cas de Abiquo. Això facilitarà el desenvolupament, ajudant-nos a adquirir un estàndard dins de l'especificació de cloud computing. Però també es podrien realitzar les peticions a traves d'una petició tradicional (com fa jClouds internament).

Petició general autenticada:
curl –X GET –u "user:password"
http://X.X.X.X/api

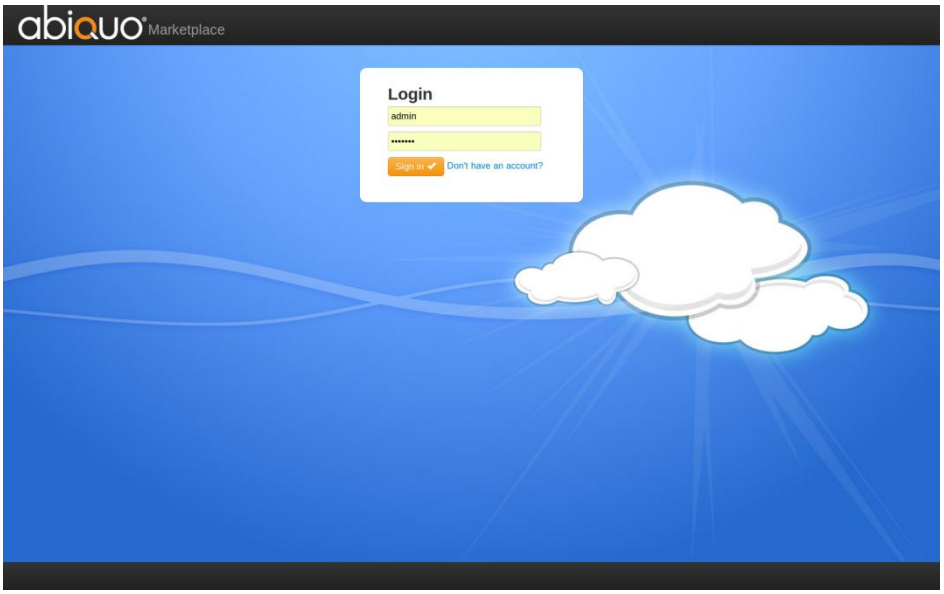
Obtenció centres virtuals de un usuari en concret:
curl –X GET –u "user:password"
http://X.X.X.X/api/cloud/virtualdatacenters

La API de Abiquo és molt extensa i permet controlar el IaaS a través d'aquesta. Gràcies al client Java implementat per jClouds simplifiquem el procés de realitzar aquestes crides i processar les respostes.

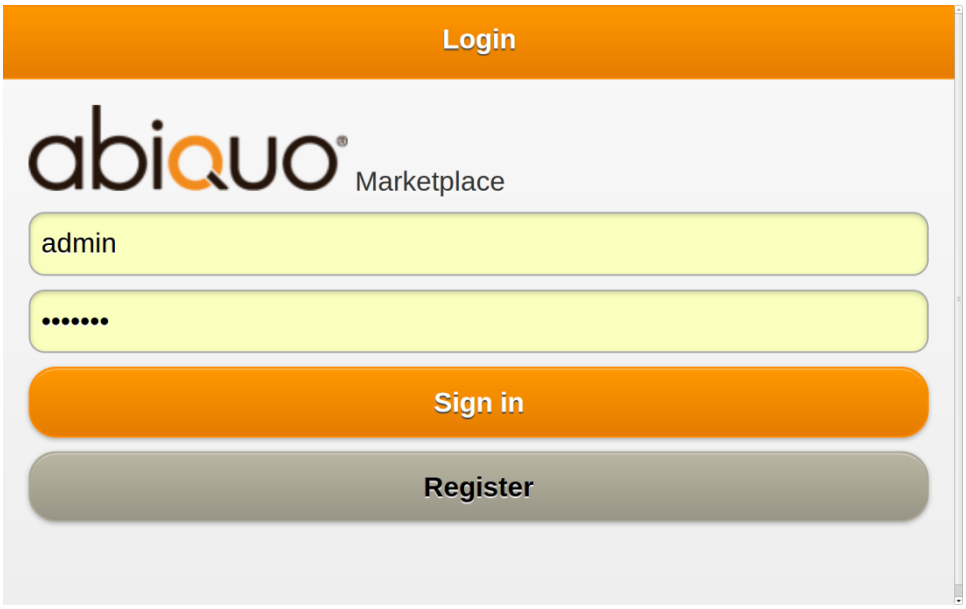
6.3.2. Implementació usuari s'autentica

En aquest cas d'ús un usuari que es troba prèviament registrat al sistema accedeix al portal i introdueix les seves credencials. Aquest cas d'ús utilitza el procés definit en el diagrama d'activitat anterior per creat el context i autenticar l'usuari.

A continuació es mostra la vista principal del portal web i de l'aplicació mòbil.



Formulari d'autenticació versió escriptori



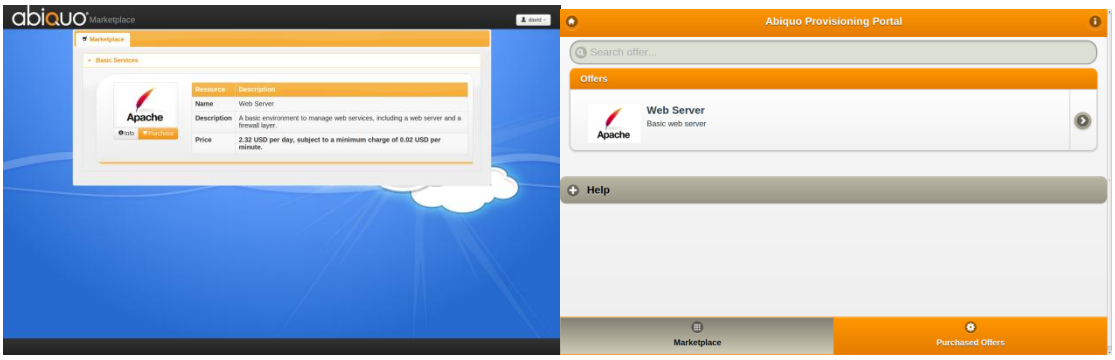
Formulari d'autenticació versió mòbil

En els següents casos d'ús es detallarà la implementació dels controladors i de les rutes utilitzades

6.3.3. Un usuari compra una oferta

Aquest és el cas d'ús més rellevant de la plataforma, ja que involucra el procés de configuració de la màquina virtual amb els serveis extra que es vulguin afegir i les peticions necessàries per a la creació del centre de dades virtual, aplicació virtual i màquines virtuals que representen l'oferta al IaaS.

Prèviament a aquest cas d'ús l'administrador haurà de configurar i habilitar l'oferta perquè l'usuari la pugui comprar, així com definir la política de preus si ho desitja.



Vista de una oferta habilitada per ser comprada

Encaminador Llistar Ofertes

GET /consumer/servicecatalog Consumer.ServiceCatalog

Aquest encaminador mostra que quan es rebi una petició a la url *host/consumer/servicecatalog* creï una instància del controlador *Consumer,* que representa a l'usuari, i invoqui al mètode que s'hi troba definit *ServiceCatalog.* El definim a continuació:

Codi controlador llistar ofertes

Aquest codi mostra com es recupera la informació del context de l'usuari que es troba autenticat, i a través del servei recuperem el nombre d'ofertes habilitades per comprar, D'aquesta manera, aprofitant la reescriptura de play, poden definir elements html a través de la resposta d'una petició realitzada. Ho comprovarem en la vista generada.

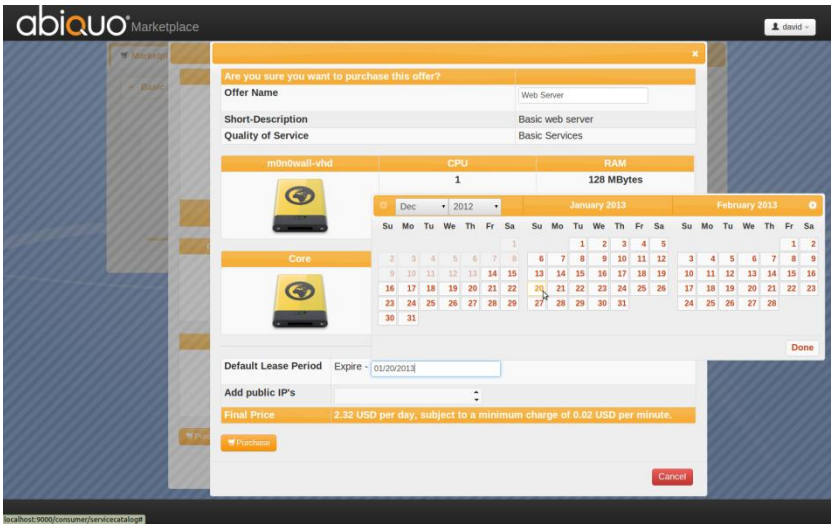
```
public static void ServiceCatalog() {
    String user = session.get("username");
    String password = session.get("password");
    if (user != null) {
        AbiquoContext context = Context.getApiClient(user, password);
        if (contextt != null) {
            AbiquoUtils.setAbiquoUtilsContext(context);
            final User userAbiquo = context.getAdministrationService()
                                           .getCurrentUser();
            final Integer numOffers = ProducerDAO
                                           .getOffersPurchasedFromEnterpriseId
                                           (userAbiquo.getEnterprise().getId())
                                           .size();
            render(user, numOffers);
        } else {
            flash.error("You are not connected.Please Login");
            Login.login_page();
        }
    }
}
```

Vista generada

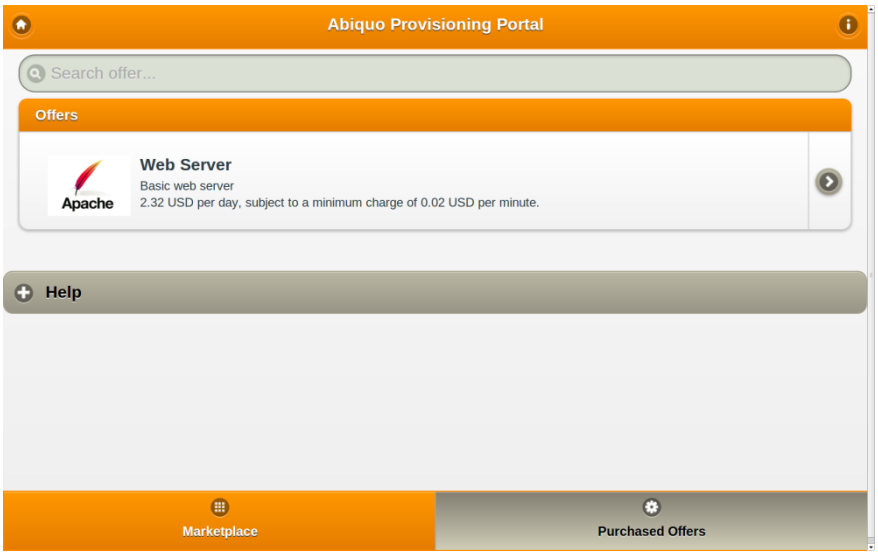
En aquesta vista es pot observar com, utilitzant l'herència de plantilles de vista, s'estén l'esquelet html general de la plataforma, i afegim el nou codi, que mostrarà les ofertes que estan habilitades i les que ja han sigut comprades de l'usuari en qüestió. Això es possible gràcies a la notació *@{Consumer.availableOffersAll(enterpriseID)}*, la qual generarà la pertinent petició de play i crearà la resposta, mostrant totes les ofertes disponibles per la empresa on es trobi l'usuari autenticat. La segona petició recuperarà les ofertes que ja han estat comprades per algun usuari de l'empresa a la que pertany l'usuari autenticat.

```
#{extends 'wrapper.html' /}
    <div id="tabs">
        <ul>
            <li><a href="@{Consumer.availableOffersAll(enterpriseID)}"><i
                    class="icon-shopping-cart"></i> Marketplace</a></li> #{if numOffers > 0
            }
            <li><a href="@{Consumer.purchasedOffers(enterpriseID)}"><i
                    class="icon-briefcase"></i> Purchased Offers</a></li> #{/if}
        </ul>
    </div>
</div>
```

Un cop llistades les ofertes, si l'usuari decideix comprar-ne una, en particular es mostrarà el formulari següent:



Detall de vista de compra d'una oferta des del portal web



Detall de vista de compra d'una oferta des del portal mòbil

Encaminador confirmar compra
GET
/consumer/purchaseconfirmation Consumer.purchaseConfirmation

Aquest enrutador mostra que quan es rebi una petició a la url *host/consumer/purchaseconfirmation* creï una instància del controlador *Consumer,* que representa a l'usuari, i invoqui al mètode que s'hi troba definit *purchaseConfirmation.* El definim a continuació:

Codi controlador confirma compra

En aquest cas només cal recuperar la informació de l'oferta en qüestió, ja què abans de realitzar el procés de desplegament hem de mostrar un resum de la compra i validar que no hi ha errors.

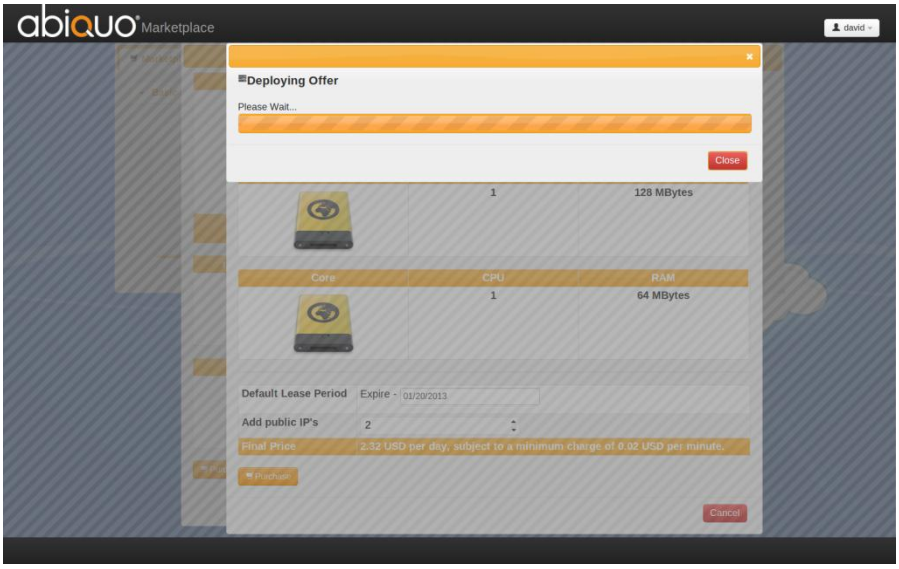
```
public static void purchaseConfirmation(final Integer offer_id) {
    String user = session.get("username");
    if (user != null) {
        Offer offer = Offer.findById(offer_id);
        render(offer, user);
    } else {
        flash.error("You are not connected.Please Login");
        Login.login_page();
    }
}
```

Vista generada

La vista generada mostra informació sobre tots els nodes que conté l'oferta (nom, CPU i ram de cadascun d'ells). Tot i què disposem de més informació aquesta és la més rellevant per l'usuari objectiu del portal.

```
#{list offer.nodes, as:'node'}
<table
    class="table table-bordered table-condensed table-striped middle">
    <thead class="ui-widget-header">
        <tr>
            <th width="200px"><strong><center>${node.node_name}</center></strong></th>
            <td width="200px"><center>CPU</center></td>
            <td width="200px"><center>RAM</center></td>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td><center>
                    <img src="@{'/public/images/user/1341575776_Yellow_Server.png'}"
                        width="100">
                </center></td>
            <td><strong><center>${node.cpu}</center></strong></td>
            <td><strong><center>${node.ram} MBytes</center></strong></td>
        </tr>
    </tbody>
</table>
#{/list}
```

Un cop introduïda la informació sobre la data fins quan es vol mantenir l'oferta activa i sobre IPs extra que es vulguin afegir, es procedirà al desplegament de l'oferta. Respecte a afegir IP's externes es molt interessant en ofertes que han d'exposar un servei web públic i que es pugui afegir a les màquines virtuals un punt d'accés extern.



Detall de vista mentre es despleguen els recursos

Encaminador desplegar recursos

GET /consumer/deploy Consumer.deploy

Aquest enrutador mostra que quan es rebi una petició a l'url *host/* **consumer/deploy** creï una instància del controlador *Consumer,* que representa a l'usuari, i invoqui al mètode que s'hi troba definit *deploy.* El definim a continuació:

Codi controlador desplegar oferta

En aquesta secció es mostra el procés de creació de una nova oferta personalitzada per a l'usuari.

```
virtualDC = AbiquoUtils.getMarketplaceDetails(vdc_id_param);
vdc_name = virtualDC.getName();
HypervisorType hypervisor = virtualDC.getHypervisorType();

Datacenter datacenter = virtualDC.getDatacenter();

PrivateNetwork network = PrivateNetwork.builder(context.getApiContext())
        .name("192.168.0.0").gateway("192.168.0.1")
        .address("192.168.0.0").mask(22).build();

vdc_toDeploy = VirtualDatacenter
        .builder(context.getApiContext(), datacenter, enterprise).name(vdcname)
        .cpuCountLimits(0, 0).hdLimitsInMb(0, 0)
        .publicIpsLimits(0, 0).ramLimits(0, 0)
        .storageLimits(0, 0).vlansLimits(0, 0)
        .hypervisorType(hypervisor).network(network).build();
vdc_toDeploy.save();

virtualapp_todeploy = VirtualAppliance
        .builder(context.getApiContext(), vdc_toDeploy)
        .name(va_param).build();
virtualapp_todeploy.save();
```

Les parts més importants són les següents:

Consultem el centre de dades virtuals per obtenir el tipus de hipervisor, ja què alhora de crear-ne un de nou hem de triar de quin tipus serà.
Consultem el centre de dades físic on desplegarem l'oferta.
Creem una xarxa privada estàndard per poder comunicar-nos amb la màquina desplegada.
Construïm el centre de dades virtual amb els paràmetres obtinguts.
Creem una aplicació virtual que contindrà l'oferta i que es trobarà dins del centre de dades virtual.

```
for (Nodes aVM : vmlist_todeploy) {
    String vmName = aVM.getNode_name();
    VirtualMachineTemplate vm_template_todeploy = virtualDC.getAvailableTemplate(aVM.getIdImage());
    int cpu = aVM.getCpu();
    int ram = aVM.getRam();
    // String description = aVM.getDescription();

    vm_todeploy = VirtualMachine
            .builder(context.getApiContext(), virtualapp_todeploy,
                    vm_template_todeploy).nameLabel(vmName)
            .cpu(cpu).ram(ram).password("vmpassword")
            .build();

    vm_todeploy.save();
```

Posteriorment consultem per a cadascuna de les màquines virtuals que s'hagin de crear (nodes) la informació de la CPU, RAM i la plantilla de la imatge , entre d'altres paràmetres. Mitjançant el constructor de màquina virtual creem una per l'usuari amb les dades obtingudes i salvem persistim els canvis.

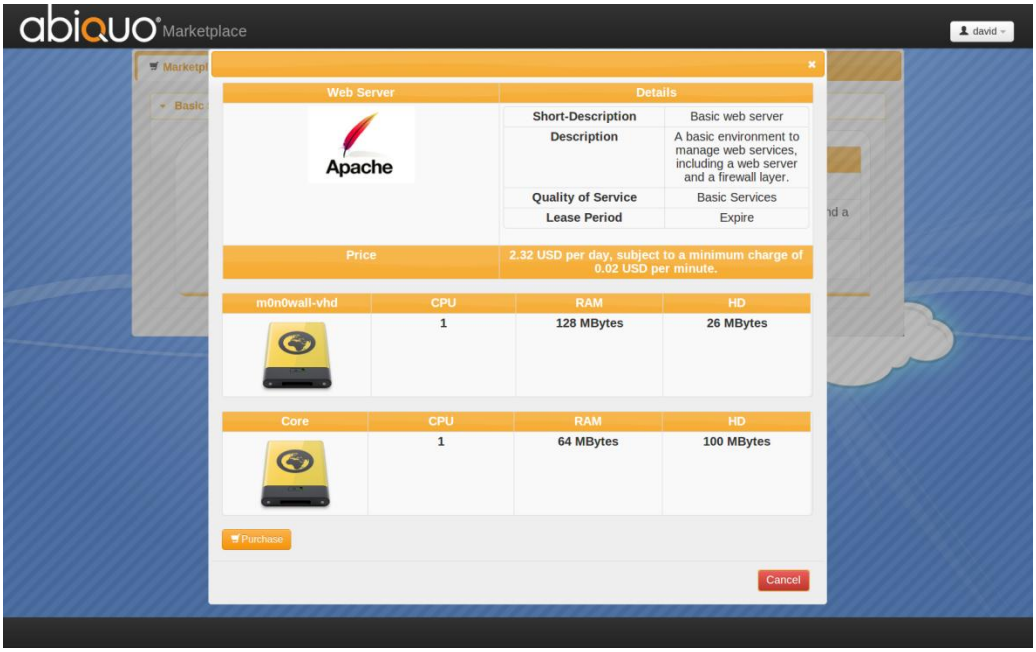
```
// Register the handler so it starts to listen to events
VirtualApplianceMonitor monitor = context.getMonitoringService().getVirtualApplianceMonitor();
VappEventHandler handler = new VappEventHandler(monitor);
monitor.register(handler);
// Monitor the task and call the callback when it completes
monitor.monitorDeploy(virtualapp_todeploy);
// The 'monitor' method will not block and the program execution will continue
// normally. Events will be dispatched to handlers when monitor completes, fails
// or reaches timeout.
```

Un cop s'ha realitzat la petició de desplegament utilitzem el monitor d'aplicacions virtuals per subscriure'ns als events que es generin i processar-los quan la tasca estigui finalitzada, evitant d'aquesta forma bloquejar la petició del portal i generar la resposta de forma asíncrona.

Els constructors que s'utilitzen per definir les màquines virtuals formen part de les definicions de jClouds. Un cop s'hagin definit les entitats amb els atributs corresponents, s'esperarà fins que es persisteixin els canvis (crida al mètode *save())* per generaran les peticions necessàries que realitzin les tasques desitjades a Abiquo.

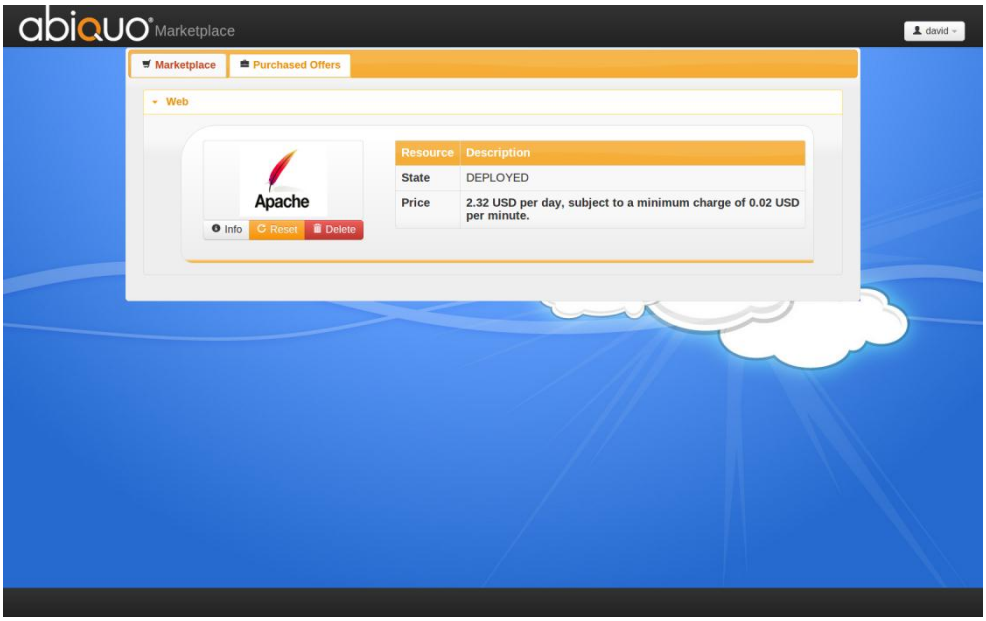
En l'apartat anterior s'ha definit el procés que realitza la petició des del controlador del portal fins que es realitza el desplegament de les màquines virtuals a Abiquo.

6.3.4. Un usuari consulta els detalls d'una oferta



Aquest cas d'ús mostra els detalls d'una oferta que ha estat habilitada per ser comprada. El sistema recupera la informació de la base de dades de portal i de Abiquo a traves de jClouds.

6.3.5. Un usuari consulta les ofertes comprades



Aquest cas d'ús mostra la llista d'ofertes comprades per un usuari. El sistema recupera la informació de la base de dades de portal i de Abiquo a través de jClouds.

6.3.6. Un usuari consulta una oferta comprada

Enrutador desplegar recursos
GET /consumer/offerdetailspurchased Consumer.offerDetailsPurchased

Aquest enrutador mostra que quan es rebi una petició a la url *host/*consumer/ offerdetailspurchased creï una instància del controlador *Consumer,* que representa a l'usuari, i invoqui al mètode que s'hi troba definit *offersDetailsPurchased.* El definim a continuació:

Codi controlador detalls oferta comprada

```
public static void offerDetailsPurchased(final Integer offerPurchasedId) {
    String user = session.get("username");
    String password = session.get("password");
    if (user != null) {
        Set<Nodes> nodes_list = null;
        Set<Nodes_Resources> nodes_resources = null;

        OfferPurchased offerPurchased = OfferPurchased.findById(offerPurchasedId);
        nodes_list = offerPurchased.getOffer().getNodes();
        AbiquoContext contextt = Context.getApiClient(user, password);
        if (context != null) {
            AbiquoUtils.setAbiquoUtilsContext(context);
```

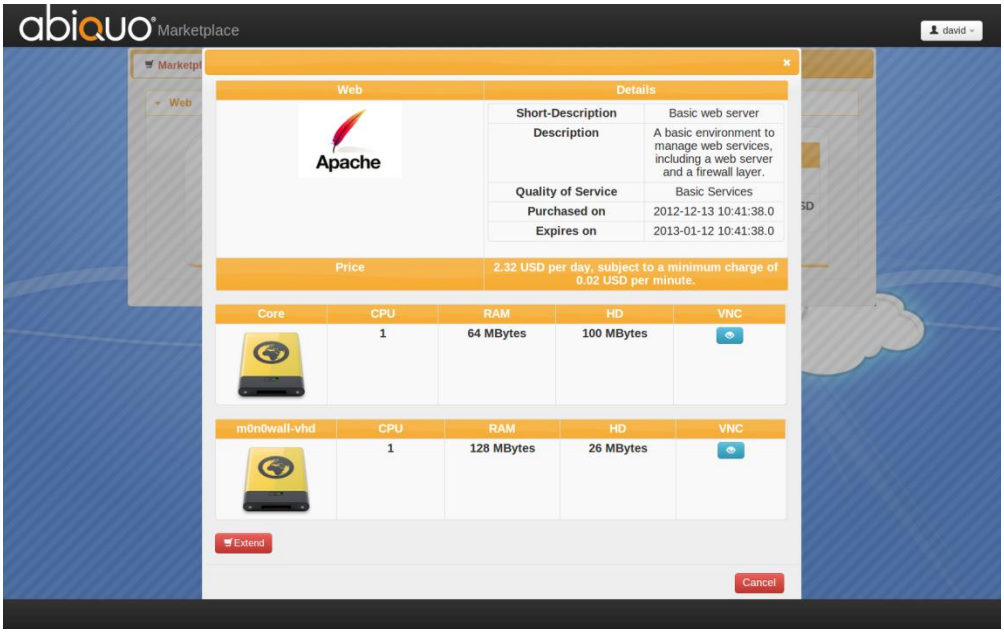
En aquesta secció es mostra el procés que s'utilitza per obrir un context (o connexió) entre el portal i el IaaS a través de jClouds. Els paràmetres que s'utilitzen son l'usuari i la contrasenya , els quals s'obtenen de la sessió per automatitzar el procés. El context representa una instància de l'objecte encarregat de gestionar els canvis en el Iaas i la comunicació amb aquest utilitzant les credencials amb les que s'ha creat. D'aquesta manera els permisos i els rols de seguretat seran els que tingui l'usuari que representa les credencials utilitzades.

```
    final CloudService cloudService = context.getCloudService();

    VirtualDatacenter vdc = cloudService.getVirtualDatacenter(offerPurchased
                    .getIdVirtualDatacenterUser());
    VirtualAppliance vapp = vdc.getVirtualAppliance(offerPurchased.getIdVirtualApplianceUser());
    List<VirtualMachine> listVM = vapp.listVirtualMachines();
```

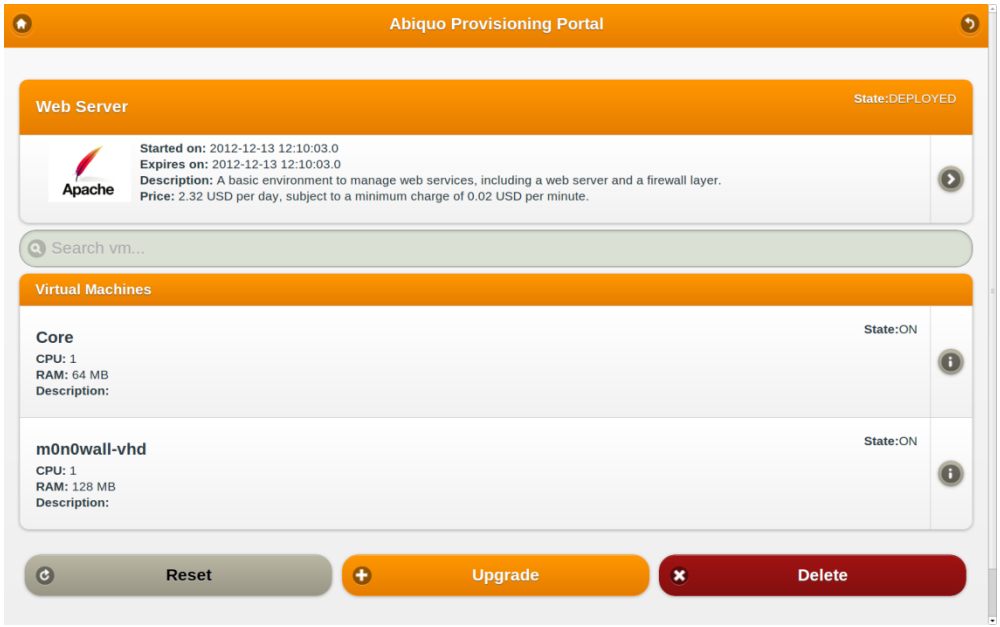
En el codi anterior podem comprovar com es crea l'objecte Cloud Service del context prèviament obert. Aquest objecte representa la resposta, després de ser *deserialitzada*, de la petició a la API següent:

“curl –X GET –u ‘user:password’ http://host/api/cloud”
Un cop obtenim en CloudService l’utilitzarem per llistar la informació de l’oferta (VirtualAppliance en el IaaS) i la llista de màquines virtuals que conté (objectes VirtualMachine) que es troben dins del centre de dades virtual abans creat en la compra de l’oferta, el qual pertany únicament a un usuari.

Vista Generada



Detall de vista amb la informació d’una oferta comprada



Detall de vista amb la informació d’una oferta comprada en l’aplicació mòbil

6.3.7. Un usuari accedeix a una oferta desplegada

En aquest cas d'ús s'obté la informació necessària per connectar-nos a les màquines virtuals a través d'una connexió vnc aprofitant els avantatges dels websockets per a HTML5.

Enrutador connexió vnc

GET /consumer/vnc Consumer.vncConnection

Codi controlador connexió vnc

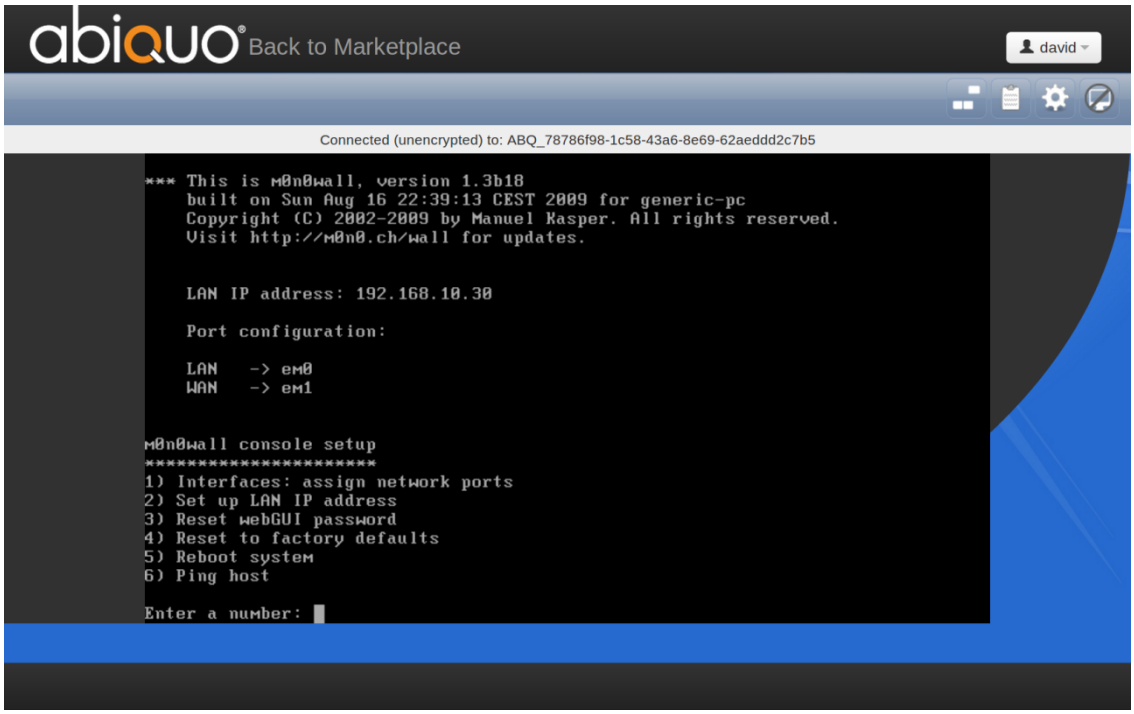
```
public static void vncConnection(final String vncAddress,
        final String vncPort, final String vncPassword) {
    String user = session.get("username");
    if (user != null) {
        try {
            Properties props = new Properties();
            // load a properties file
            props.load(new FileInputStream(Play.getFile("conf/config.properties")));

            final String noVNCPath = props.getProperty("noVNC");
            final String noVNCPort = props.getProperty("noVNCPort");
            final String noVNCServer = props.getProperty("noVNCServer");
```

En aquesta porció de codi obtenim les dades d'usuari i carreguem del fitxer de propietats les dades referents al servidor VNC. Per evitar utilitzar recursos innecessaris, només es crearà el servei quan es necessiti crear una nova connexió VNC amb el hipervisor que contingui la màquina virtual.

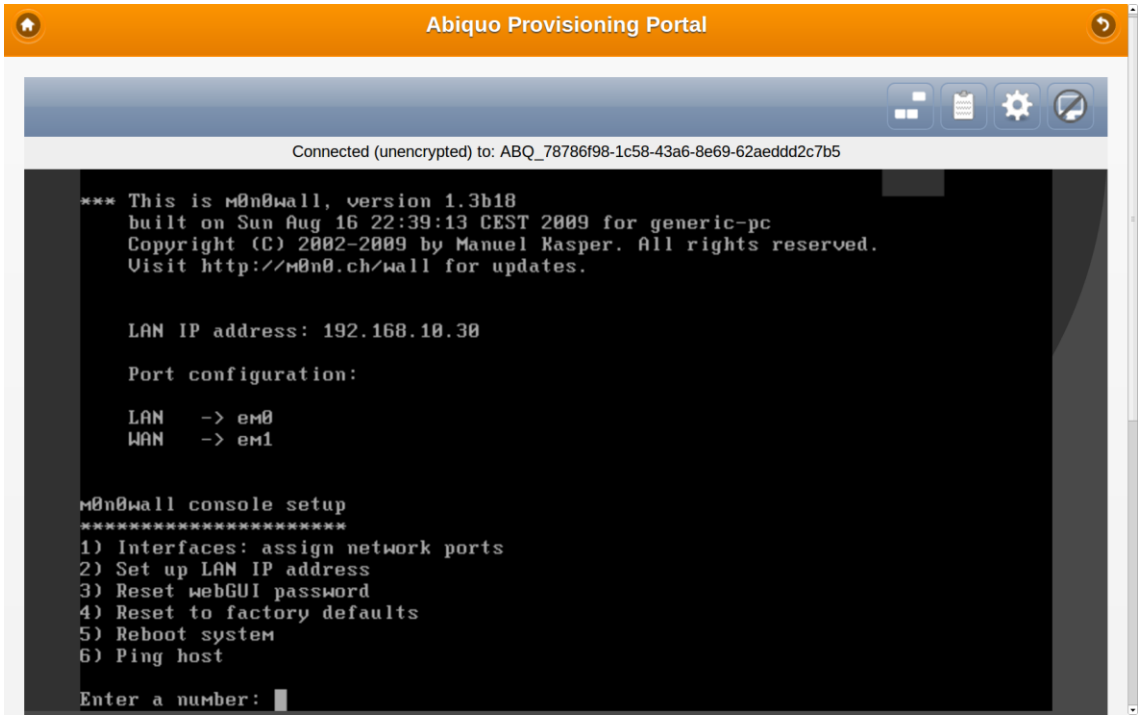
```
ProcessBuilder pb = new ProcessBuilder(
        "public/noVNC/utils/websockify", "--web",
        "public/noVNC/", noVNCPort, vncAddress + ":" + vncPort);
pb.redirectErrorStream(); // redirect stderr to stdout
Process process = pb.start();
play.mvc.Http.Request current = play.mvc.Http.Request.current();
String url = current.url;
String domain = current.domain;
```

El servei VNC crearà un *socket* web que connectarà la màquina local (la qual té hostatjada el portal cloud i el client de VNC) i el hipervisor, el qual s'encarregarà d'instal·lar el servidor VNC a la màquina virtual.

Vista Generada



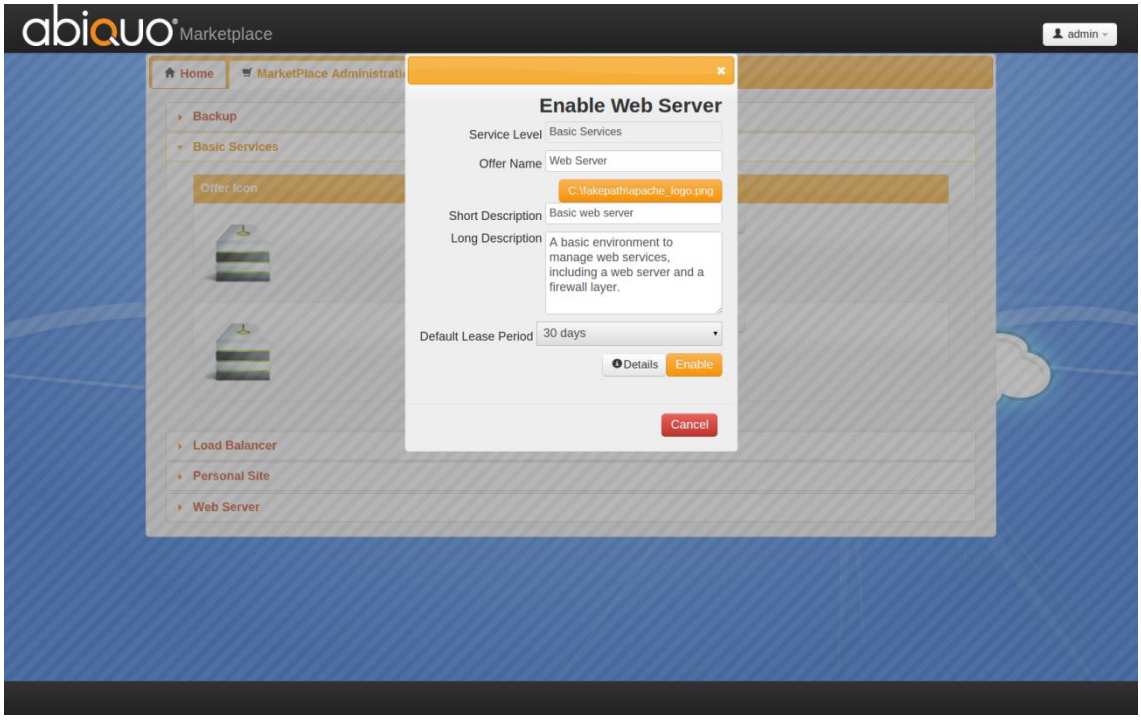
Detall vista conexió vnc portal web

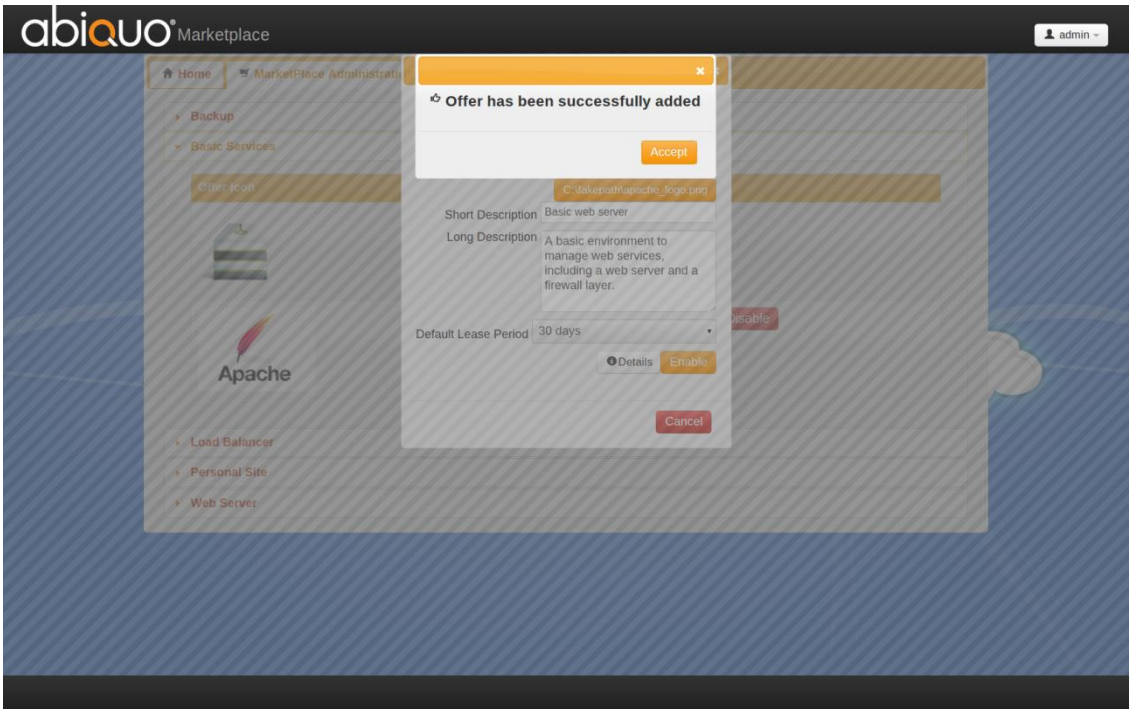


Detall vista conexió vnc portal web

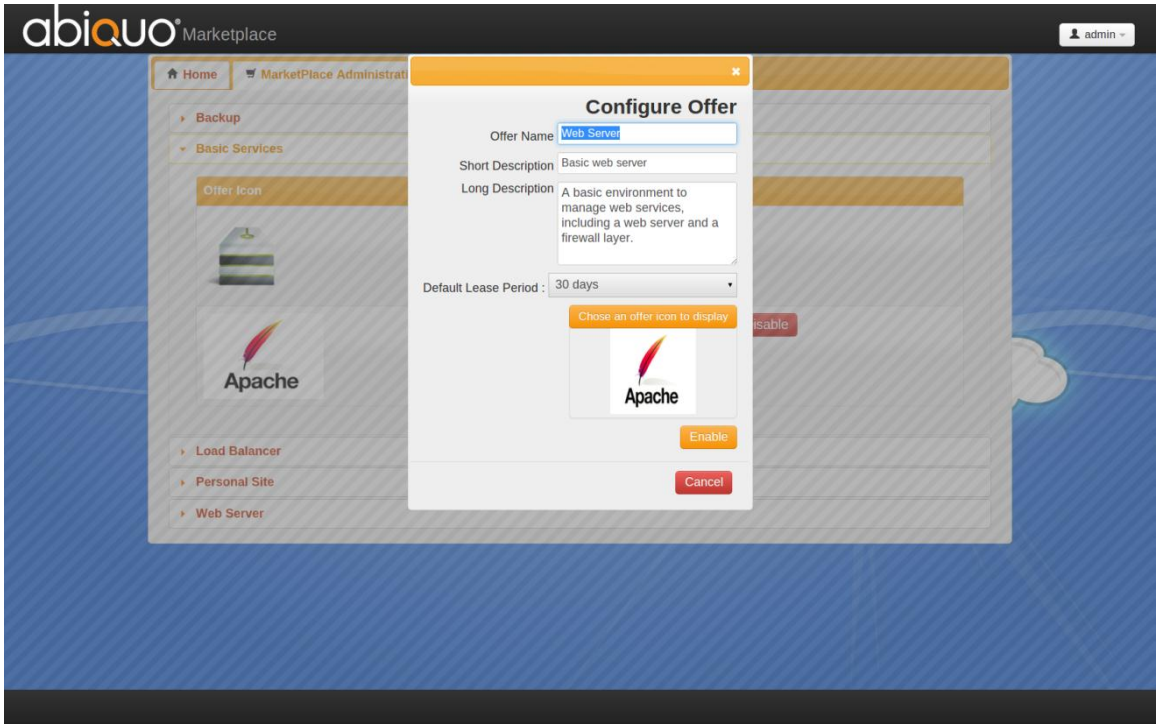
6.3.8. Administració d'ofertes

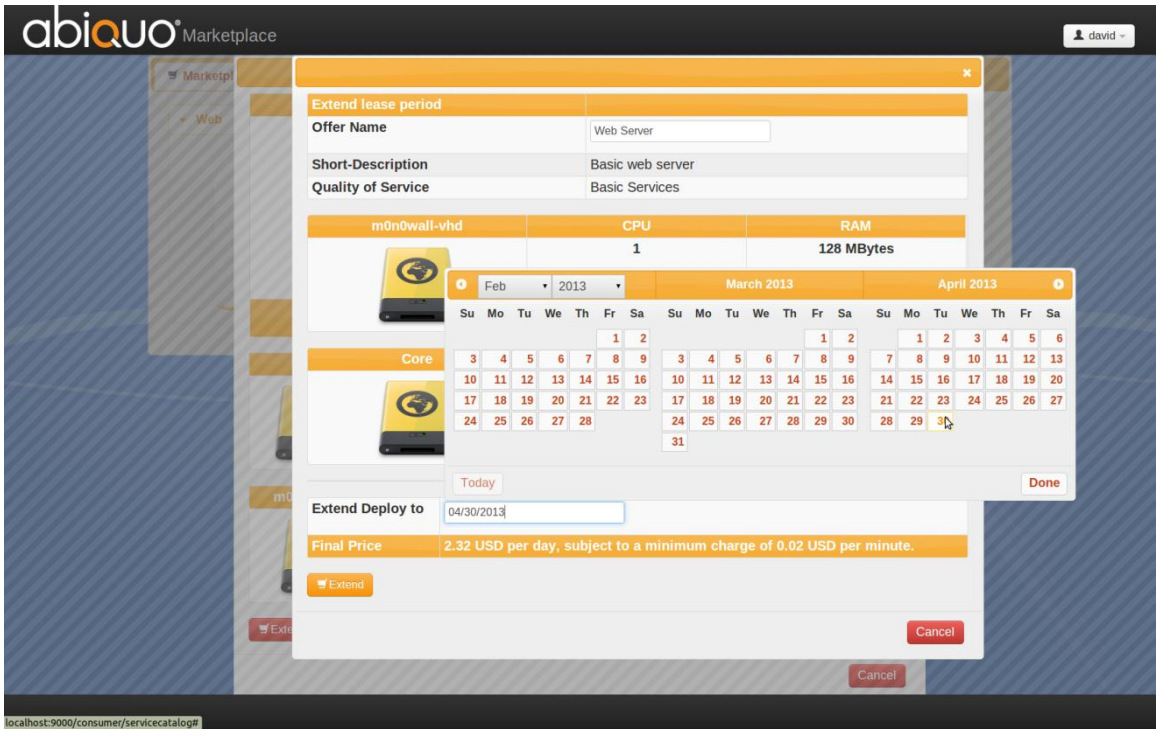
6.3.8.1. Administrador habilita una oferta





6.3.8.2. Administrador configura una oferta





6.3.9. Control del sistema

6.3.9.1. Gestió de la expiració i reaprofitament de recursos

En el provisioning portal s'han definit serveis que comproven cada cert temps si hi ha alguna oferta que hagi expirat, i ho comuniquen a l'usuari propietari de la compra a través de correu.

```
@OnApplicationStart
@Every("day")
public class UndeployScheduler extends Job {

    @Override
    public void doJob() {
        Query query = JPA.em().createQuery("select op from OfferPurchased as op");
        List<OfferPurchased> result = query.getResultList();
        for (OfferPurchased record : result) {
            final Collection<Integer> daysToSendMail = Lists.newArrayList(1, 2,3, 4, 5, 7, 15, 30);
            if (expDate.before(currentDate)) {
                Logger.info("Lease has expired.");
                deleteOffer(record);
                Mails.sendExpiredEmail(record);
            } else if (daysToSendMail.contains(days)) {
                Logger.info("Lease is almost expired.");
                Mails.sendExtendEmail(record);
            }
        }
    }
```

Si l'usuari ho desitja pot allargar el període de compra i allargar més el període de lloguer de recursos. En cas que el període de compra hagi expirat, es procedirà al alliberament de recursos en el corresponent hipervisor, eliminant les màquines virtuals i la informació de la base de dades del portal. D'aquesta manera es garanteix que els recursos puguin ser reciclats i reaprofitats al màxim.

```
public static void deleteOffer(OfferPurchased offerPurchased) {
    Properties props = new Properties();
    // load a properties file
    try {
        VirtualDatacenter vdc = context
                .getCloudService()
                .getVirtualDatacenter(
                        offerPurchased.getIdVirtualDatacenterUser());
        VirtualAppliance vapp = vdc
                .getVirtualAppliance(offerPurchased
                        .getIdVirtualApplianceUser());

        VirtualApplianceMonitor monitorVapp = context
                .getMonitoringService()
                .getVirtualApplianceMonitor();
        AsyncTask[] undeployTasks = vapp.undeploy();
        monitorVapp.awaitCompletionUndeploy(vapp);

        if (vapp.getState() == VirtualApplianceState.NOT_DEPLOYED) {
            vapp.delete();
            vdc.delete();
            offerPurchased.delete();
        } else {
            AbiquoUtils.checkErrorsInTasks(undeployTasks);
            Logger.info("Tasks Checked");
        }
```

En la imatge anterior es mostra el mètode encarregat del procés de borrat. Donada una oferta que ha de ser borrada es consulten les dades relatives al centre de dades i a la apliació virtual on es troba. Un cop es coneixen, es crida al mètode *undeploy()* de la aplicació virtual, i ens esperem a que acabi. Aquest procés es síncron, ja que volem assegurar que els recursos quedin consistents abans d'alliberar els recursos per a un nou ús. Un cop s'hagin alliberat els recursos de la aplicació virtual en el centre de dades eliminarem la informació relativa a Abiquo (referent al centre de dades i a la aplicació virtual) i al portal (referent a la entitat que representa l'oferta comprada).

6.3.9.2. Notificacions

A més del control d'expiració, també es notifica per correu a l'usuari quan després d'una compra les màquines virtual s'hagin desplegat correctament, ja que al ser un procés asíncron no obtindrem resposta immediata. No bloquejarem el procés del portal, i tenint en compte que el desplegament pot requerir minuts en alguns casos (segons la mida de la imatge) es una bona pràctica el realitzar la petició seguint la metodologia de consumidor-subscriptor. Les imatges següents mostren com s'envia el correu corresponent segons la finalització de la tasca.

```
@Subscribe
public void onComplete(final CompletedEvent<VirtualAppliance> event) {
    // Handle completion here
    VirtualAppliance vapp = event.getTarget();
    try {
        JPAPlugin.startTx(Boolean.TRUE);
        List<OfferPurchased> resultSet1 = ProducerDAO
                .getOffersPurchasedFromVappId(vapp.getId());
        OfferPurchased op = resultSet1.get(0);
        Mails.sendEmail(op, vapp);
```

```
@Subscribe
public void onFailure(final FailedEvent<VirtualAppliance> event) {
    // Handle failures here
    VirtualAppliance vapp = event.getTarget();
    try {
        JPAPlugin.startTx(Boolean.TRUE);
        List<OfferPurchased> resultSet1 = ProducerDAO
                .getOffersPurchasedFromVappId(vapp.getId());
        OfferPurchased op = resultSet1.get(0);
        //Mails.sendEmail(op, vapp);
        Mails.sendFailureEmail(op.getOffer().getName(), op.getUser().getNick(), op.getUser().getEmail());
```

```
@Subscribe
public void onTimeout(final TimeoutEvent<VirtualAppliance> event) {
    // Handle timeout here
    VirtualAppliance vapp = event.getTarget();
    try {
        JPAPlugin.startTx(Boolean.TRUE);
        List<OfferPurchased> resultSet1 = ProducerDAO
                .getOffersPurchasedFromVappId(vapp.getId());
        OfferPurchased op = resultSet1.get(0);
        //Mails.sendEmail(op, vapp);
        Mails.sendFailureEmail(op.getOffer().getName(), op.getUser().getNick(), op.getUser().getEmail());
```

Controlador de correu

```
public static void sendEmail(OfferPurchased op, VirtualAppliance vapp) {
    setSubject("Abiquo Confirmation");
    addRecipient(op.getUser().getEmail());
    setFrom("Admin <provisioning-portal@abiquo.com>");
    List<VirtualMachine> vms = vapp.listVirtualMachines();
    send(op, vms);
}
```

Vista generada de correu

```
#{extends '/Mails/sendEmailWrapper.html' /}
<layout label="Text only">
<title>Cloud Services Confirmation</title>
    <table class="w580" width="580" cellpadding="0" cellspacing="0" border="0">
        <tbody><tr>
            <td class="w580" width="580">
                <p align="left" class="article-title"><singleline label="Title">${message} done<singleline></p>
                <div align="left" class="article-content">
                    <multiline label="Description">Thank you for using Cloud Services, your order is ready. Please review the order information below.
                    For assistance contact your cloud administrator
                        <p>User:  <b>${useremail}</b></p>p>
                        <p>Offer name: <b>${offerName} </b></p>
                        <p>Offer expiration date: <b>${exp_date} </b></p>
                        <p>You can use any VNC utility to access the Virtual Machine and enter above details</p>
                        <p>TightVNC can be downloaded from http://www.tightvnc.com/download.php</p><br>
                        <p>UltraVNC can be downloaded from http://www.uvnc.com/downloads/ultravnc.html</p><br>
                    </multiline>
                </div>
            </td>
        </tr>
        <tr><td class="w580" width="580" height="10"></td></tr>
    </tbody></table>
</layout>
```

CAPÍTOL 7. PROVES

7.1. Test

Durant la implementació s'han realitzat una sèrie de tests unitaris, funcionals i d'integració per garantir que les funcionalitats fan el que s'espera realment.
A més, també s'han realitzat proves d'interacció de client web mitjançant Selenium. Aquestes simulen la interacció d'un client de navegador per comprovar la correcta interacció dels elements del portal.

7.2. Metodologia de desenvolupament

El codi s'ha versionat mitjançant git, i es troba compartit en la pàgina de Github

https://github.com/abiquo/provisioning-portal

Com s'ha comentat durant el transcurs del document el projecte es de codi lliure, així que qualsevol persona es pot descarregar el codi i adaptar-lo a les seves necessitats, així com col·laborar a millorar el projecte a traves de les eines que proporciona github.
A més s'ha creat una wiki accessible des de l'enllaç anterior que resumeix els objectius que s'han exposat i mostra una guia de configuració de l'entorn necessari per utilitzar el portal cloud privat.
També s'explica com treballar amb Heroku, plataforma com a servei que ens permet desplegar entorns de treball i que es configurin segons els nostres requeriments. En el nostre cas, ens ha servit per desplegar directament l'entorn de play als seus servidors i fer-lo públic sense haver de configurar un servei web amb accés a internet i de forma gratuïta. Així que simplement ens podríem descarregar el codi del portal i pujar-lo a una compta personal de Heroku i funcionaria correctament, sempre que modifiquéssim la IP de la API a la que consulta el portal.
Cal remarcar que en aquest context estaríem treballant en les tres grans capes del cloud computing, ja que el portal (Saas) estaria gestionant per Heroku (PaaS) i que de forma distribuïda controla una infraestructura mitjançant Abiquo (IaaS)

CAPÍTOL 8. CONCLUSIONS

8.1. Resum

Gràcies a la realització d'aquest projecte s'han pogut obrir noves línies de negoci a la empresa, reinterpretant el funcionament de la plataforma per adaptar-lo a les necessitats dels clients finals. En l'actualitat, els portals cloud per auto-gestionar infraestructures virtuals i aprovisionar software es troben en expansió, i aquesta funcionalitat podrà ser coberta directament per Abiquo, juntament amb la plataforma comercial. D'aquesta manera, les empreses que disposis d'un IaaS podran gestionar els recursos de forma ràpida i senzilla, permetent que els usuaris finals auto-gestionin els seus entorns cloud de forma personalitzada, oferint l'aprovisionament de infraestructures, configurades i llestes per treballar en producció, i del software que es requereixi.

8.2. Objectius assolits

Fent balanç sobre el desenvolupament global els principals objectius que es van planificar s'han complert, ja que el portal privat permet auto-aprovisionar de recursos als usuaris finals de forma personalitzada, es pot accedir i controlar les màquines virtuals que s'hagin desplegat i s'alliberen correctament els recursos un cop una oferta ha expirat o s'ha eliminat.

8.3. Ampliacions futures

Els passos següent a realitzar serien afegir més personalització a les ofertes, permetent configurar xarxes complexes, afegir discs o emmagatzematge extra a les màquines desplegades, crear una zona de control i monitoratge de l'estat de les màquines desplegades i permetre afegir més recursos a les màquines virtuals a través dels hipervisors, fet que faria els entorns altament escalables.

8.4. Aptituds assolides

El punt més important ha estat el de treballar en un projecte on hi ha hagut integració de varis productes fets en la empresa, com la plataforma de Abiquo i el client de jClouds, cosa que m'ha ensenyat com treballar en equips grans amb control, planificació, verisionat de codi i testeig de la qualitat.

He aprés com funciona un entorn de cloud computing real, com és el disseny i arquitectura d'un IaaS, com es gestionen els recursos dinàmicament i les necessitats que tenen els clients sobre la gestió dels centres de dades cloud, en entorns privats, públics i híbrids en producció, i sobre el funcionament i disseny de xarxes virtuals, i les solucions que existeixen en l'emmagatzematge dinàmic.

També he assolit molts coneixements sobre virtualització i els grans avantatges que suposa el fet de treballar en infraestructures com a servei, les quals poden

estalviar grans costos en compra de recursos físics i que faciliten el creixement i la escalabilitat dels entorns sense comprometre el producte, fet que obre nous mercats i models de negoci més rentables.

A més, he treballat amb hipervisors de diversos tipus, com ara ESX, KVM, XEN, XEN Server, Microsoft Hyper-V o Virtualbox, inclús alguns de forma virtualitzada, fet que permet adaptar al màxim les necessitats dels clients.

També he aprés a dissenyar i desenvolupar un software com a servei que pugui treballar amb nodes remots de forma distribuïda, i a integrar-lo amb altres nivells de cloud, com ara amb el IaaS, per crear infraestructures virtuals auto-configurables o bé per aprovisionar qualsevol software sota demanda i pagar per l'ús que se n'està fent.

He gestionat el provisioning portal públicament a github, on qualsevol persona interessada pot col·laborar en el seu desenvolupament o a solucionar errors, els quals es troben reportats mitjançant Jira, una eina de control i gestió dels errors utilitzada per grans equips de desenvolupament arreu del món. A més he treballat amb vàries plataformes com a servei, com ara Heroku, per poder publicar l'entorn durant el desenvolupament de forma automatitzada.

També he aprés a gestionar un projecte en el marc empresarial, amb uns objectius i fites que complir, els quals requereixen un esforç extra en la organització i dedicació pel treball diari.

Pel que fa a la implementació, he aprés a utilitzar diversos frameworks de java i mòbil, amb els que he pogut distribuir les capes de presentació i lògica del negoci, de tal forma que la vista es processa en un servidor que no ha d'estar necessàriament en la màquina que gestiona les accions del portal.

La vista s'ha realitzat mitjançant HTML5 tant per la part d'escriptori com per la part mòbil, cosa que m'ha ensenyat a reaprofitar la lògica del negoci i utilitzar la vista que s'adapti millor a les necessitats del client, gestionant-la de forma web sensible al dispositiu que l'utilitzi, mostrant l'aplicació d'escriptori en cas de sobretaula o la versió mòbil en *smartphone* i *tablets*. A més, la gestió d'events es asíncrona, utilitzant tecnologies de javascript (com ara jquery i jquery Mobile), cosa que crea una experiència i usabilitat més elevada que en les pàgines tradicionals.

CAPÍTOL 9. BIBLIOGRAFIA

- https://github.com/abiquo/provisioning-portal
- http://wiki.abiquo.com
- http://blog.stevensanderson.com/2012/08/01/rich-javascript-applications-the-seven-frameworks-throne-of-js-2012/
- http://www.slideshare.net/raulfraile/symfony2-interaccin-con-css-js-y-html5

  http://coding.smashingmagazine.com/2012/07/27/journey-through-the-javascript-mvc-jungle/
- http://addyosmani.com/resources/essentialjsdesignpatterns/book/#detailmvcmvp
- http://todomvc.com/
- http://docs.openstack.org/diablo/openstack-compute/starter/content/Scheduler_nova-scheduler_-d1e262.html
- http://docs.openstack.org/trunk/openstack-compute/admin/content/ch_scheduling.html
- http://forums.openstack.org/viewtopic.php?f=16&t=1068
- http://www.citrix.com/products/cloudportal/overview.html
- http://scn.sap.com/community/netweaver-portal/cloud
- http://www.cisco.com/en/US/products/ps11927/index.html
- http://www.cisco.com/en/US/products/ps11927/prod_literature.html
- http://techblog.netflix.com/2012/06/asgard-web-based-cloud-management-and.html

Facultat d'Informàtica de Barcelona

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

ANNEXOS

Títol del TFC: Abiquo Provisioning Portal

Titulació: Enginyeria superior en Informàtica (pla 2003)

Autor: David López Padilla

Director: Xavier Fernández

Ponent: Jordi Nin Guerrero

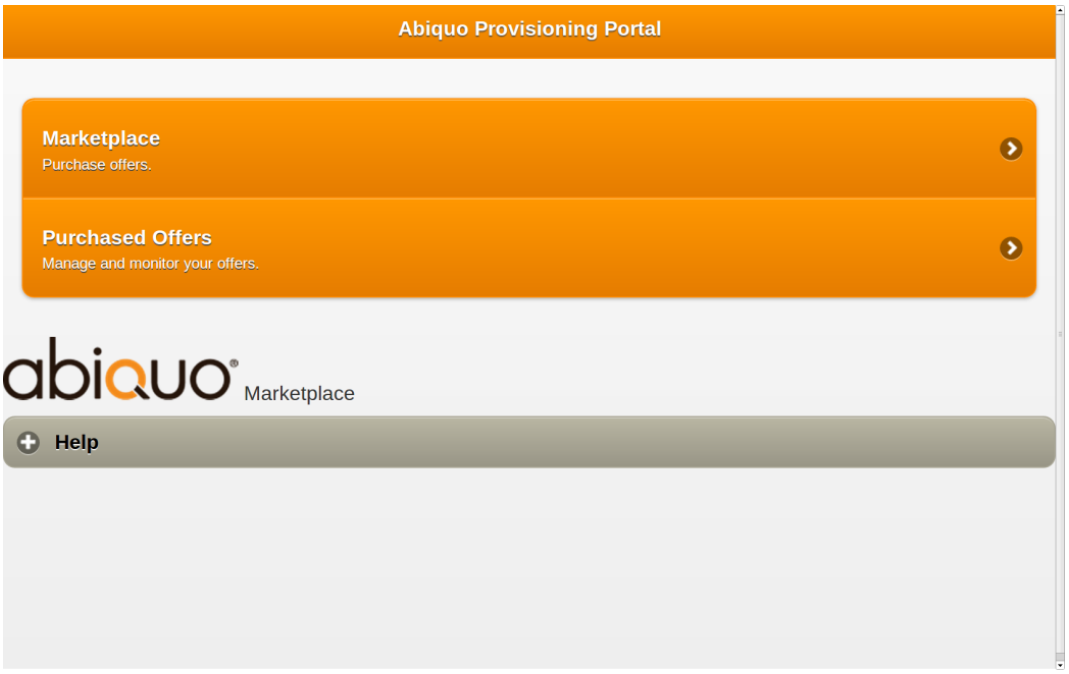
Data: 21 de Gener de 2013

Glossari

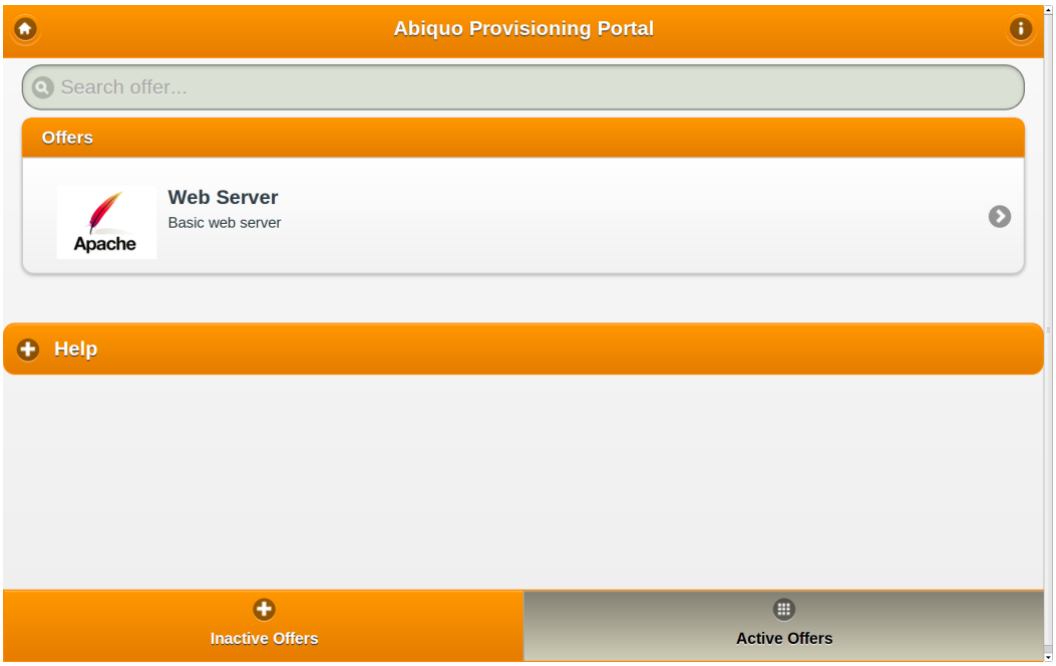
Aquesta secció descriu els diferents conceptes que involucren al provisioning portal i a la plataforma d 'Abiquo

Concepte	Descripció
Cloud	Concepte d 'Abiquo que defineix un grup de *datacenters* que ofereix escalabilitat i altres propòsits als usuaris finals.
Datacenter	Grup de màquines físiques en la mateixa LAN (àrea de xarxa local). Aquestes maquines es troben en el mateix lloc físic, i comparteixen xarxa i recursos.
Enterprise	Una Enterprise representa una empresa, un grup de treball, un departament, etc. que té accés a la infraestructura virtual.
Enterprise Repository	Cada Empresa té la seva llibreria d'imatges i plantilles per desplegar les màquines virtuals.
Hipervisor	És una plataforma de virtualització que permet la creació de diverses màquina virtuals en la mateixa màquina física.
Network	La plataforma permet gestionar les IP's públiques i privades de la infraestructura i de les màquines virtuals.
Oferta	Representa una virtual appliance de Abiquo en el portal, afegint informació addicional necessària en el portal, com ara dades descripció de l'oferta, recursos extra definits, període de temps de la compra, etc.
Physical Machine or **Cloud Node**	Un *host* o màquina física conté un hipervisor executant-se per proveir la infraestructura de virtualització.
Rack	Representa un grup de màquines físiques en el centre de dades que comparteixen el mateix switch, com es fa típicament en un centre de dades físic.
Remote Service	Les diferents aplicacions Java desplegades en cada centre de dades per mostrar la virtualització, la xarxa i l'emmagatzematge de les màquines físiques.
Remote Service List	La llista de remote services que permeten a Abiquo gestionar la infraestructura amb un centre de dades. Són: **Virtualization Manager**: Gestiona el cicle de vida de la virtual appliance. Es connecta als nodes cloud per realitzar tasques a les màquines virtuals. **Monitor Manager**: És el responsable del monitoratge de la virtual appliance. Escolta els events dels nodes cloud i actualitza l'estat de la virtual appliance.

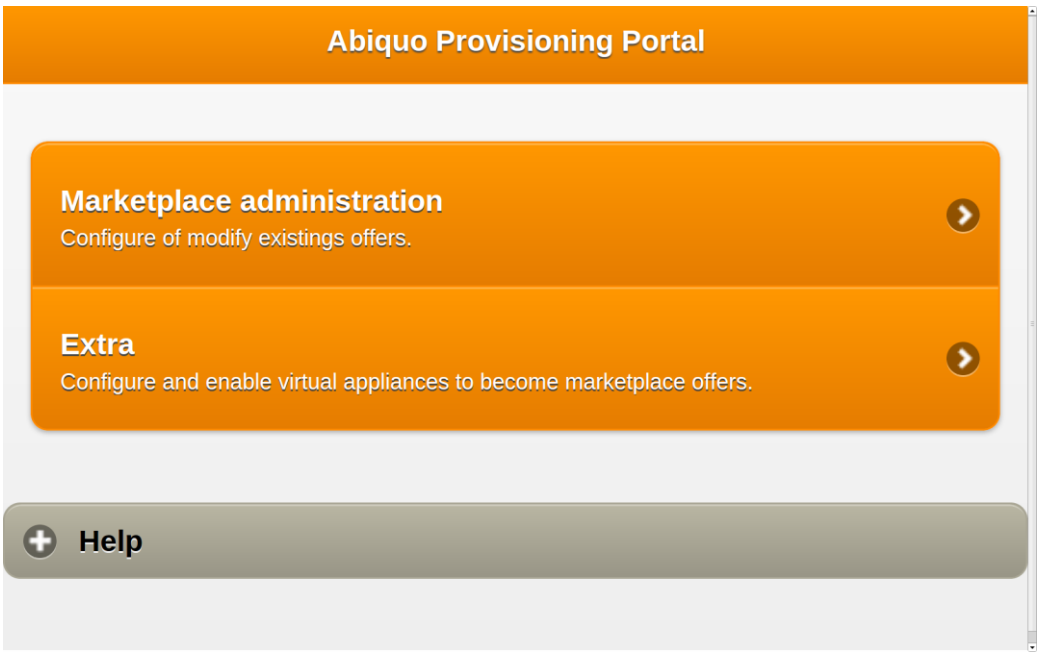
Concepte	Descripció
	Appliance Manager: Gestiona la llibreria d'imatges i els repositoris. **Discovery Manager**: S'utilitza per auto-descobrir màquines físiques. Pot obtenir informació sobre el Hardware, tipus de Hipervisor i informació sobre màquines virtuals ja existents. **DHCP Service**: El servei de DHCP s'utilitza per assignar adreces IP a les màquines virtuals que gestiona Abiquo.
Storage Manager	El virtual Storage Manager es el servei remot que gestiona els diversos sistemes d'emmagatzematge.
Storage Pool	És una infraestructura de storage centralitzada que permet als usuaris crear volums iSCSI i afegir-los a les màquines virtuals.
Template Repository	És un espai públic o privat de plantilles de màquines virtuals. Les descripcions i els fitxers de disc se serveixen per HTTP.
Users	Els usuaris són la representació de persones que s'agrupa per objectiu, geolocalització, etc. Cada usuari té rols i privilegis diversos.
Virtual Appliance	Conjunt de màquines virtuals desplegades en el mateix centre de dades virtual.
Virtual Datacenter	Centre de dades virtual que agrupa un conjunt de virtual appliances en el mateix centre de dades físic, utilitzant la mateixa tecnologia.
Virtual Machine	Màquina virtual, instància d'una imatge virtual amb configuracions de xarxa i emmagatzematge. Es pot entendre com una instància d'un sistema operatiu. Una màquina física amb un hipervisor instal·lat pot contenir vaires màquines virtuals.
Virtual Machine Template	Disc virtual amb els requeriments de desplegament: recursos (CPU, RAM and HD) i la tecnologia del hipervisor associat amb el format del disc.



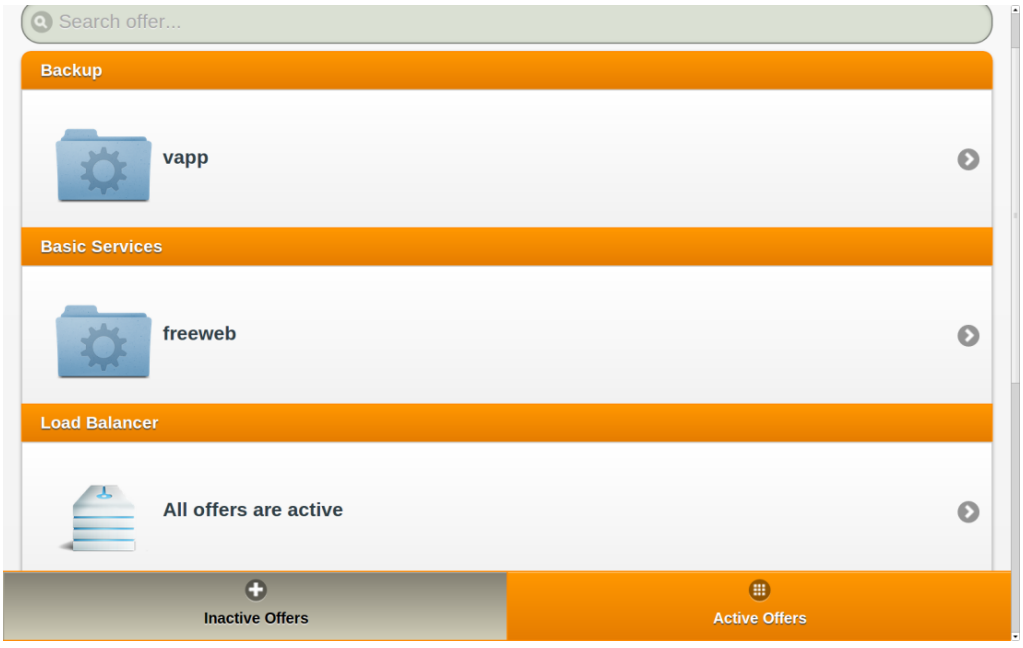
Vista del menú principal d'usuari en mòbils



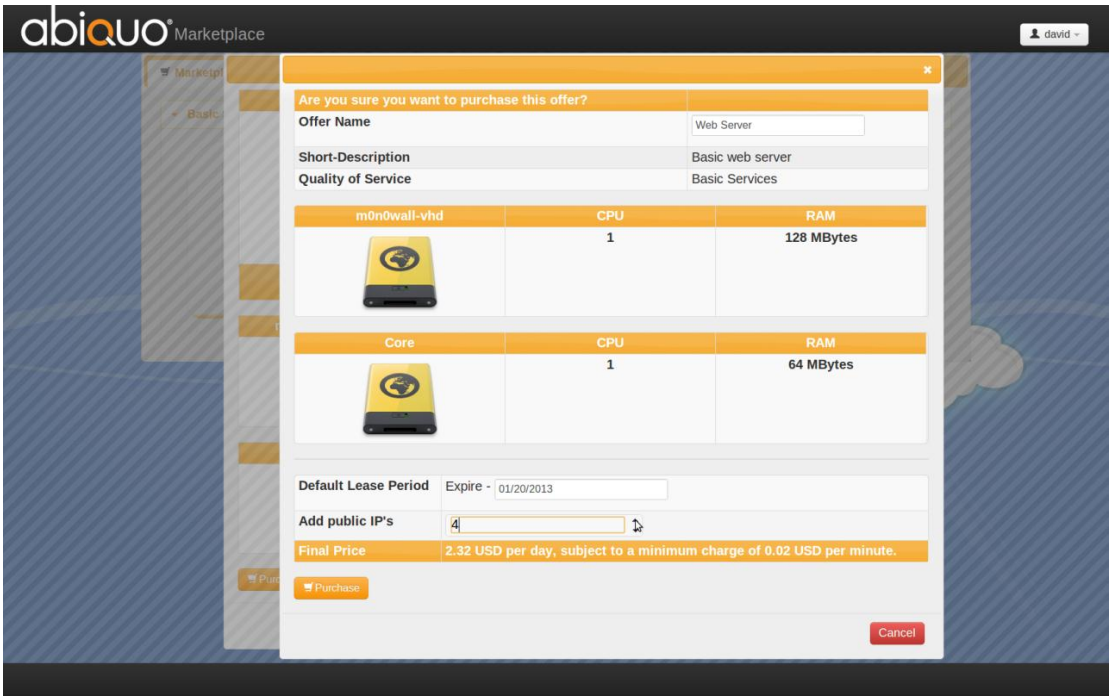
Vista del llistat d'ofertes comprades per l'usuari en mòbils



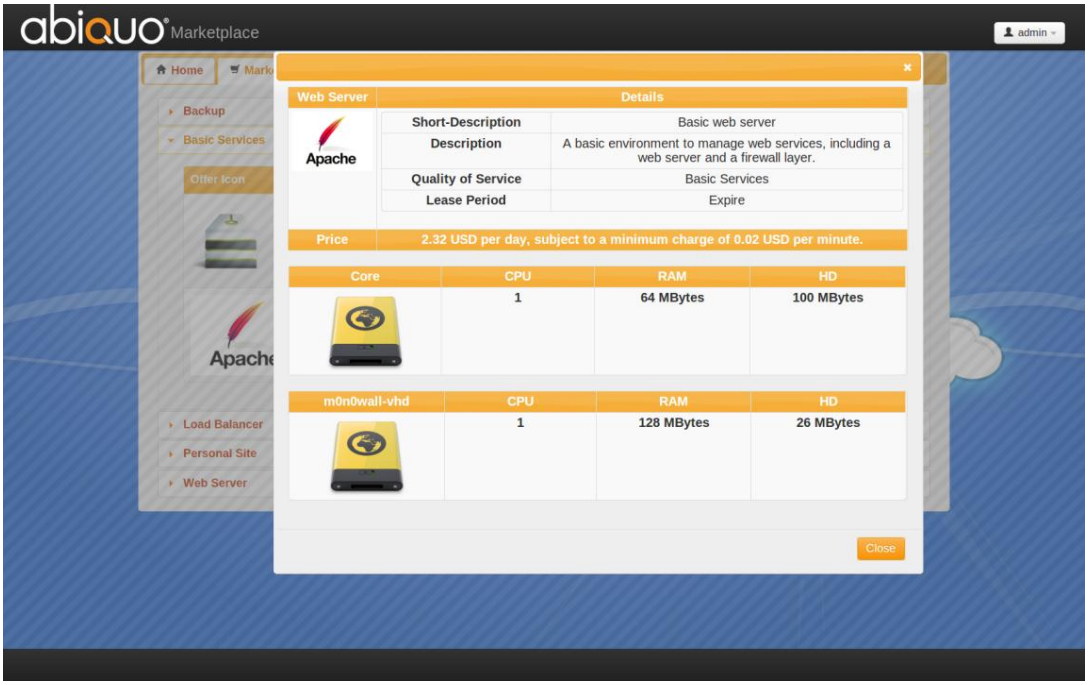
Vista del menú principal d'administradors en mòbils



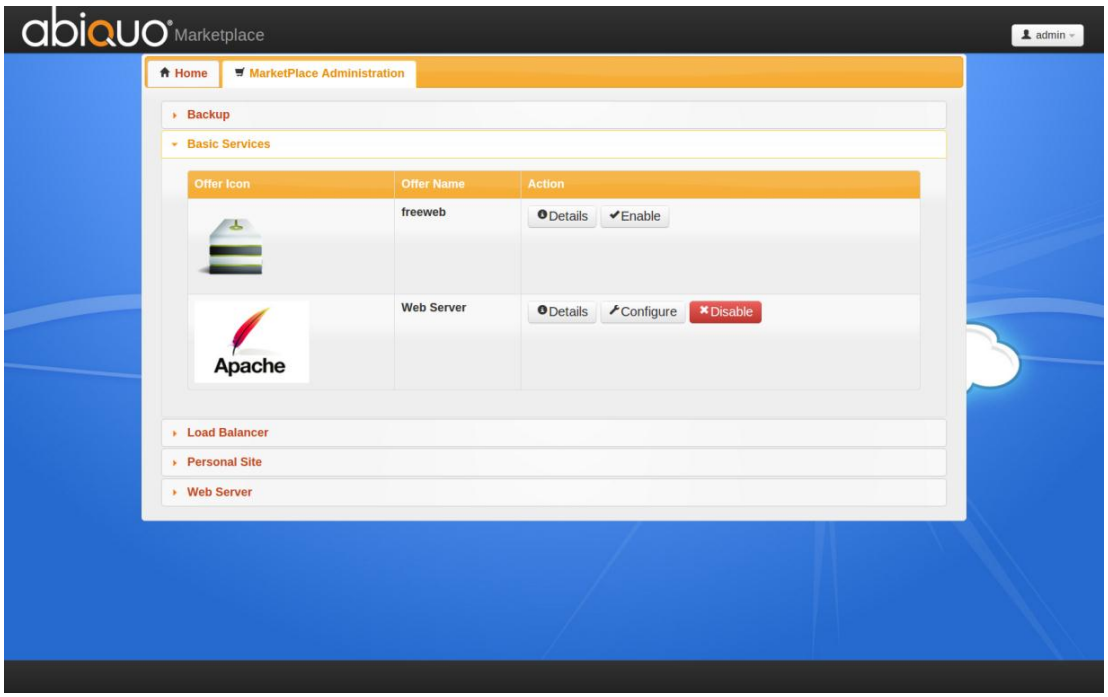
Vista de l'administració d'ofertes en mòbils



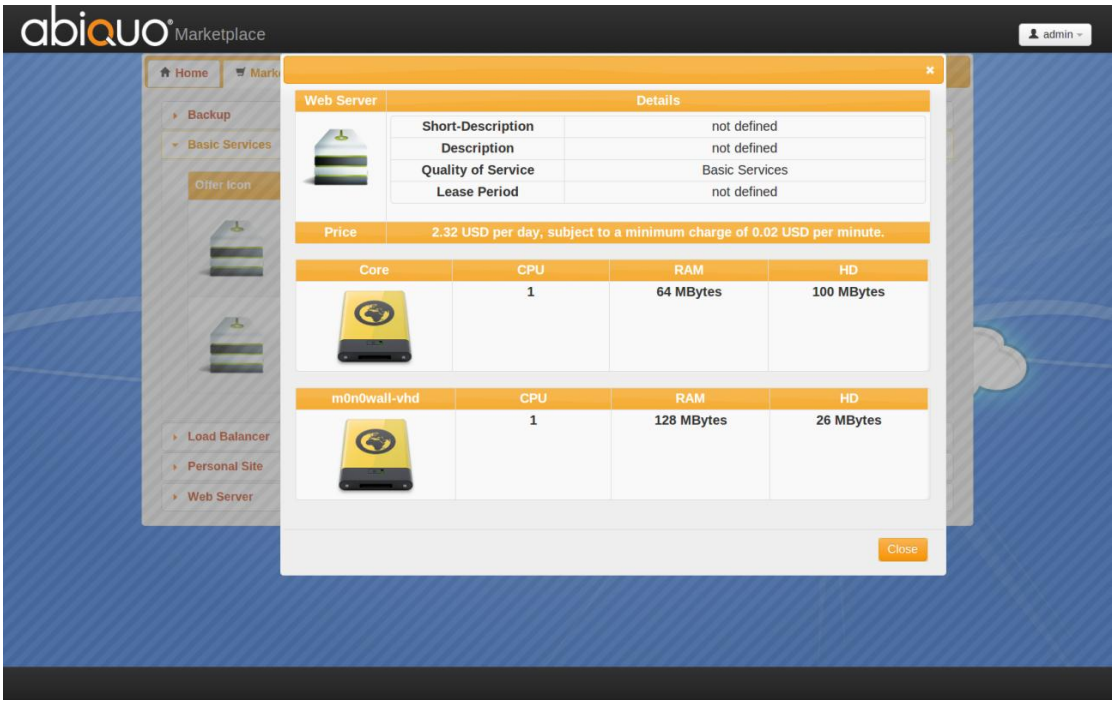
Vista de la configuració d'una oferta durant la compra



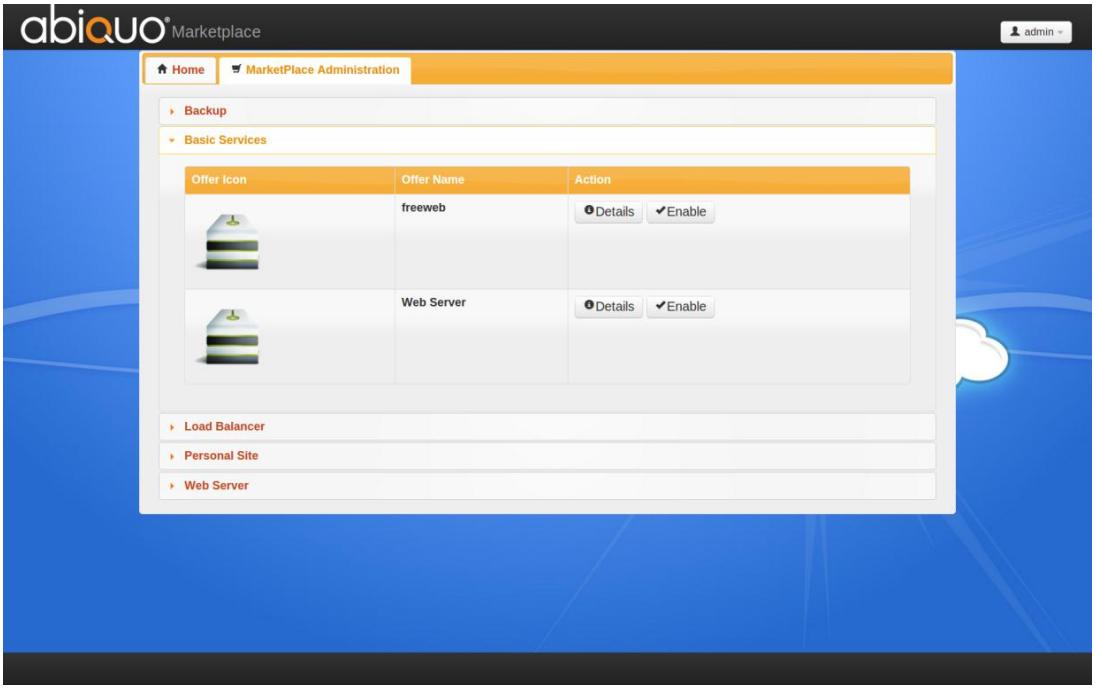
Vista de la informació sobre una oferta habilitada



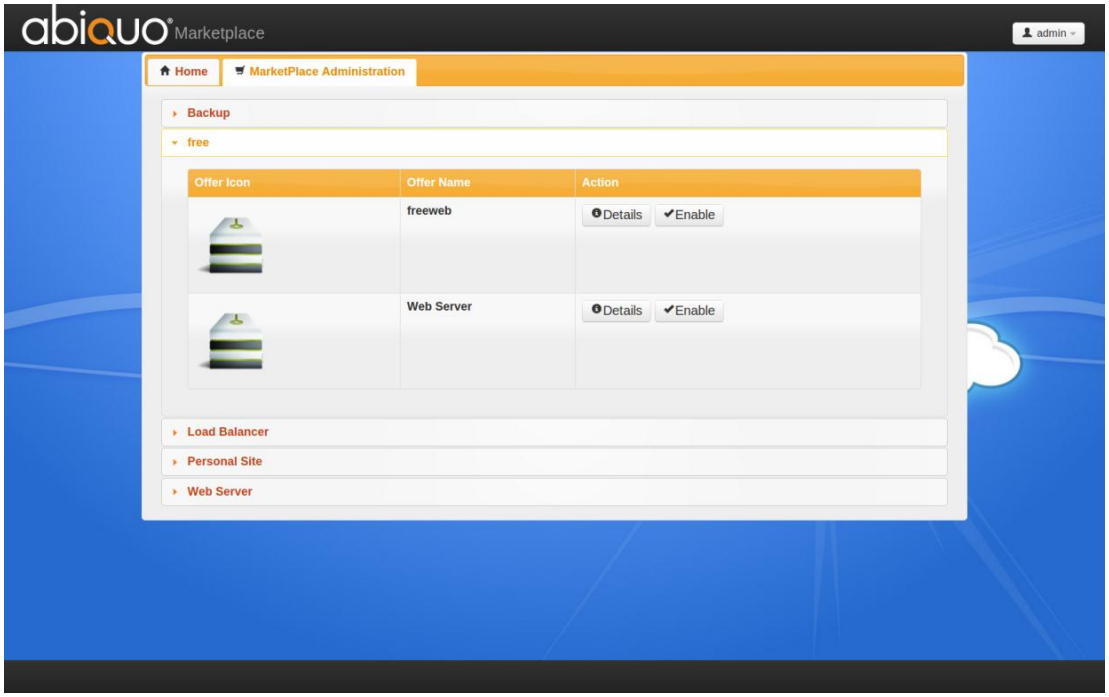
Vista de la administració d'ofertes



Vista de la informació sobre una oferta no configurada



Vista de la administració d'ofertes



Vista de la administració d'ofertes