Trabajo de Fin de Grado

Grado en Ingeniería Informática

ULL-CloudIDE: Plataforma de entornos de desarrollo para la docencia

ULL-CloudIDE: Platform of development environments for teaching

Alberto Gabriel Ruiz Pérez

La Laguna, 1 de junio de 2018

D. **Alberto Hamilton Castro**, con N.I.F. 43.773.884-P profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor.

D. **Vicente José Blanco Pérez** con N.I.F. 42.171.808-C profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor.

C E R T I F I C A (N)

Que la presente memoria titulada:

“ULL-CloudIDE: Plataforma de entornos de desarrollo para la docencia”

ha sido realizada bajo su dirección por D. **Alberto Gabriel Ruiz Pérez**, con N.I.F. 77.724.177-R.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 1 de junio de 2018

Agradecimientos

A mi tutor, Alberto Hamilton Castro, y a mi cotutor, Vicente José Blanco Pérez, por darme la posibilidad de realizar este proyecto y por el constante apoyo e interés durante la realización del mismo.

A todas aquellas personas importantes en mi vida que me han acompañado en esta estapa.

Licencia

Resumen

El desarrollo y uso de servicios en la nube se está convirtiendo en uno de los propósitos principales del panorama tecnológico actual. Existen numerosas aplicaciones que se están trasladando a sistemas externos en Internet, lo que implica una multitud de beneficios tanto para los usuarios finales como para los desarrolladores.

En nuestro caso, dada la demanda y los problemas de los estudiantes en los últimos años, para tener un lugar donde llevar a cabo y desplegar sus proyectos informáticos, este Trabajo Final de Grado (TFG) nació con el objetivo principal de implementar una plataforma interna para la Universidad de La Laguna (ULL), que ofrece tanto a profesores como a estudiantes Entornos de Desarrollo Integrado (Entorno de Desarrollo Integrado) (IDE), en la nube utilizando un navegador web.

Por un lado, los profesores pueden otorgar a los usuarios el uso de un IDE (principalmente Eclipse Che), obteniendo al mismo tiempo información continua del estado de los mismos y teniendo el poder de tomar cualquier acción sobre ellos, de ahí viene el término enfoque hacia la enseñanza. De esta manera, los estudiantes / usuarios tendrán la posibilidad de utilizar su IDE correspondiente y realizar bien cualquier tipo de proyecto (prácticas, exámenes, etc.).

La gran ventaja de que su implementación sea interna, es que tanto la plataforma desarrollada, los servicios de los que hace uso y los datos que se generan quedan dentro de las instalaciones de la ULL.

Para obtener el control y la privacidad que se requieren en éste tipo de aplicaciones, ha sido necesario diseñar un esquema de operación y hacer uso de numerosas tecnologías (plataforma de virtualización oVirt, autenticación de usuarios centralizada, firewall, etc), de tal manera que podamos construir un servicio seguro y escalable que cumpla con las principales necesidades.

Palabras clave: Docencia, IDE, Eclipse Che, seguridad, privacidad y escalabilidad.

Abstract

The development and use of services in the cloud is becoming one of the main purposes of the current technological landscape. There are numerous applications that are moving to external systems on the Internet, which entails a multitude of benefits for both end users and developers.

In our case, given the demand and problems in the last years of the students to have a place in which to carry out and deploy their computer projects, this Final Degree Project (TFG) was born with the main objective of implementing an internal platform to the University of La Laguna (ULL), which offers both teachers and students, Integrated Development Environments (Integrated Development Environment) (IDE), in the cloud using a web browser.

On the one hand, teachers can grant users the use of an IDE (mainly Eclipse Che), obtaining at the same time continuous information of the state of the themselves and having the power to take any action over them, from here comes the term focus towards teaching In this way the students / users will have the possibility to use your corresponding IDE and perform well any type of project in it (practices, exams, etc).

The great advantage is that the implementation is internal, the developed platform, the services of those who do use and the data that is generated remain within the ULL facilities.

To obtain the control and privacy that is required in this type of applications, it has been necessary to devise a scheme of operation and make use of numerous technologies (virtualization platform oVirt, user authentication centralized, firewall, etc), in such a way that we can build a safe and scalable service that lives up to the main needs.

Keywords: Teaching, IDE, Eclipse Che, security, privacy and scalability.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

1.1 Introducción

En los últimos años uno de los términos más interesantes y que cobra cada día mayor importancia es el de ***Cloud Computing***, y es que gracias a la evolución constante en todos los aspectos tecnológicos en lo que se refiere a portabilidad, escalabilidad y convergencia tanto de hardware como de software ha dado lugar a una nueva forma de negocio y de prestación de servicios.

Esta nueva tendencia tecnológica nace con el concepto de desplazar servidores y aplicaciónes a internet, de ésta forma se pretende dar un cambio en la forma habitual de trabajar, ahora pasarán a ser servicios accesibles donde y cuando se quiera, sin necesidad de tener un equipo específico, ahorrando en inversiones tecnológicas y formándo así un nuevo modelo de negocio.

Dentro de éste ámbito podemos dividir los tipos de servicios:

- **IaaS** (Infraestructure as a Service): Se trata de una infraestructura que permite proveer máquinas virtuales, además también se contempla la posibilidad de proveer más funcionalidades (escalabilidad, balanceadores de carga, etc). Actualmente existe un servicio de éste tipo en la Universidad de La Laguna, que nos será de gran ayuda como base para dar la prestación de éste proyecto.

- **PaaS** (Platform as a Service): Además de un entorno de virtualización se le proporciona un entorno para el desarrollo y hospedaje de una aplicación (servidores, acceso a red, SO, base de datos, servidor web, etc).

- **SaaS** (Software as a Service): En este tipo de servicios se brinda acceso a una aplicación determinada que se ejecuta en internet, tal vez éste término es el que más se asocia a *Cloud Computing* (Google Apps, Dropbox, Office 365, etc).

En cuanto a nuestra herramienta, principalmente lo que se pretende es dar un servicio enfocado a profesores y alumnos, donde éstos puedan disponer de un entorno de desarrollo en la nube y alojado en la ULL, de ésta manera logramos proporcionar una plataforma interna a la universidad, en el que tanto la privacidad, seguridad y datos quedan controlados por las instalaciones y personal de la ULL. A esta herramienta la hemos denominado ULL-CloudIDE, haciendo referencia a un servicio perteneciente a la ULL, el cual brinda IDEs (entornos de desarrollo) en la nube, alojados en instalaciones internas que serán accesibles a través de internet, especialmente a través de un navegador.

Figura 1.1: Logo ULL-CloudIDE

1.2 Antecedentes y estado actual

En este apartado se comentará el estado actual del ámbito en el que estamos trabajando, además se nombrarán aquellas aplicaciones que hayan perseguido los mismos objetivos que nuestra herramienta, así podremos comparar la calidad del servicio que se pretende ofertar, además de visualizar aquellas características que puedan ser útiles y beneficiosas para añadir en un futuro a nuestro software.

1.2.1 Cloud9 IDE

Cloud9 IDE es un entorno de desarrollo integrado en línea que actualmente está bajo el dominio de Amazon Web Services (AWS). Soporta cientos de lenguajes de programación, incluyendo C, C ++, PHP, Ruby, Perl, Python, JavaScript con Node.js y Go. Podriamos decir que es uno de los IDEs en la nube más conocido y usado, permite obtener workspaces preconfigurados según las necesidades, colaborar con otros usuarios y testear las aplicaciones que se estén desarrollando.

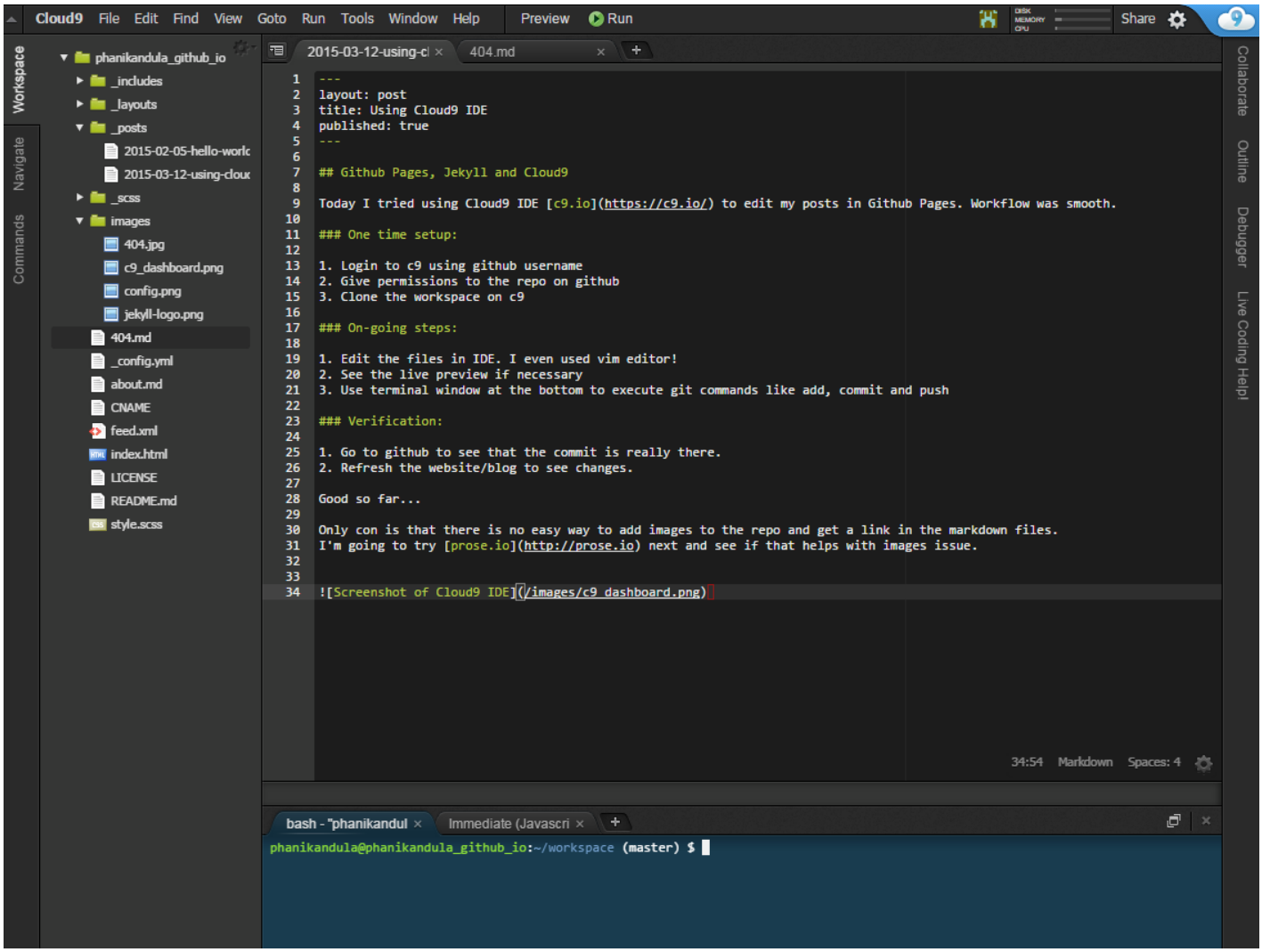

Figura 1.2.1: Cloud9 IDE (obtenida de [1])

Quizás la característica más negativa para los usuarios de ésta plataforma es que actualmente para obtener una cuenta gratis se deberá adjuntar una tarjeta de crédito, ésto en gran medida es perjudicial, ya que puede comprometer tanto la seguridad como la privacidad de tus datos personales. A su vez, como casi todas las plataformas que se nombran en éste documento, nos ofrecen una tarifa gratuita que tal vez nos pueda resultar limitada a nuestros intereses.

1.2.2 Koding

Se trata de una plataforma de desarrollo en la nube creada por *Devrim Yasar y Sinan Yasar* y que a día de hoy lleva una andadura de 5 años de continuo crecimiento.

Quizás lo más destacable es que además de proporcionar un entorno de máquinas virtuales para el desarrollo y despliegue de proyectos, incluye aplicaciones propias que hacen más accesibles el control del ámbito en el que se encuentra el usuario, a su vez han tratado de realizar una plataforma enfocada a la colaboración entre programadores como si de una red social se tratara, en la que puedes invitar, ayudar y seguir a otros usuarios.

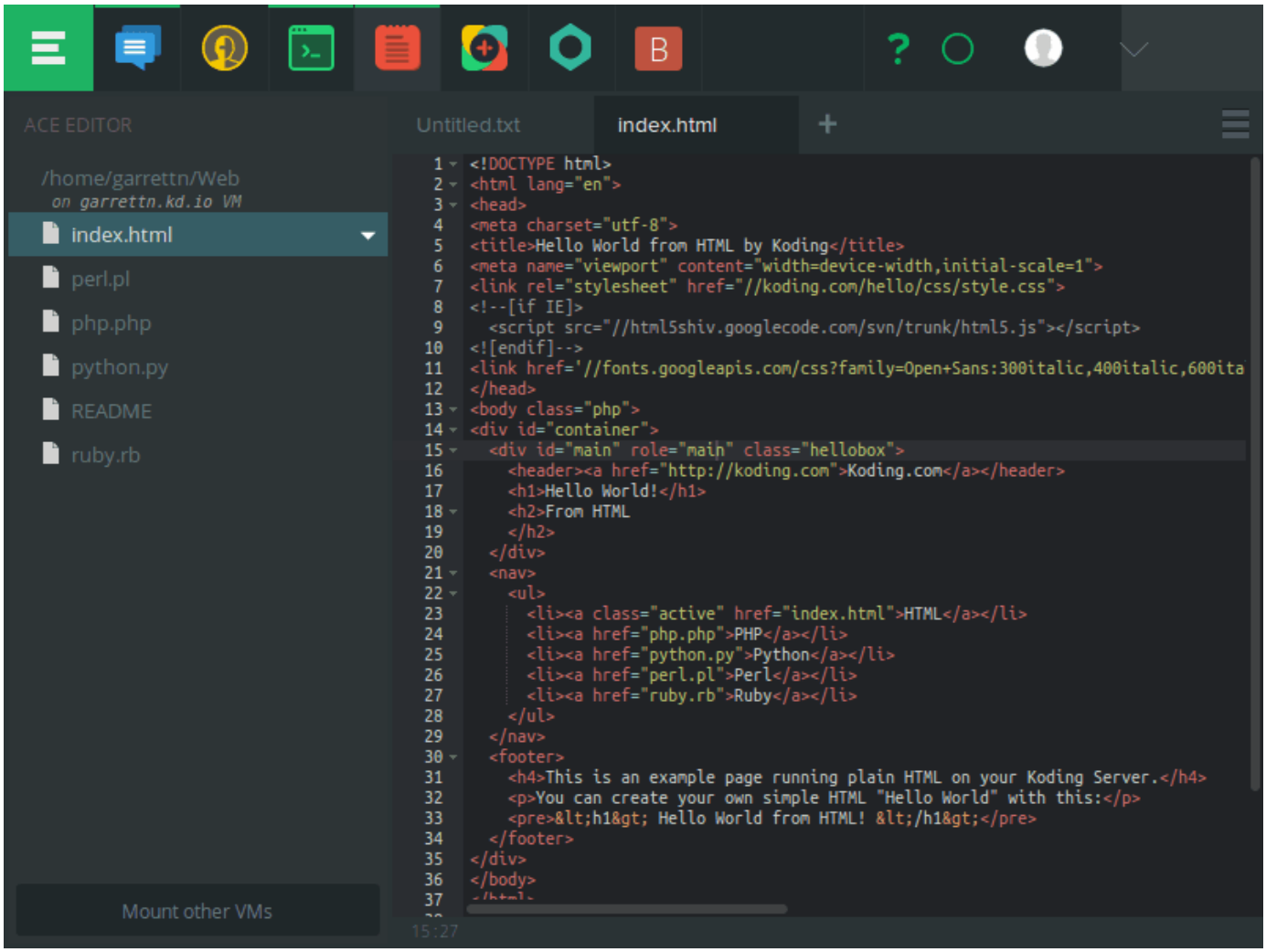

Figura 1.2.2: Koding (obtenida de [2])

Después de haber analizado sus funcionalidades se puede decir que estamos ante una completa herramienta, pero que no se ajusta a lo que tenemos como objetivo.

Por nombrar algunos de los aspectos negativos desde mi punto de vista, es que no tienen una gran personalización en cuanto a los recursos de las máquinas virtuales que ofrecen, nombramos aquí el tipo de sistema operativo, capacidad, banda ancha, etc. Por otro lado, y tal vez uno de los puntos más importantes, es que no es una herramienta de código abierto, es decir, para poder hacer uso de ella tenemos que elegir una de las tarifas que nos ofrecen en su página web, teniéndo así un costo monetario por usuario.

1.2.3 CodeAnywhere

CodeAnywhere es una herramienta de desarrollo online fundada por los croatas *Ivan Burazin y Vedran Jukic*.

Ésta plataforma además del acceso mediante un navegador web ofrece diversas aplicaciones para acceder a través de diferentes dispositivos (Android, iOS, Chrome, etc). Como la anterior plataforma ésta también ofrece una tarifa gratuita aunque personalmente bastante limitada en cuanto a recursos y funcionalidades, aún así puedes colaborar con otros programadores accediendo a sus entornos.

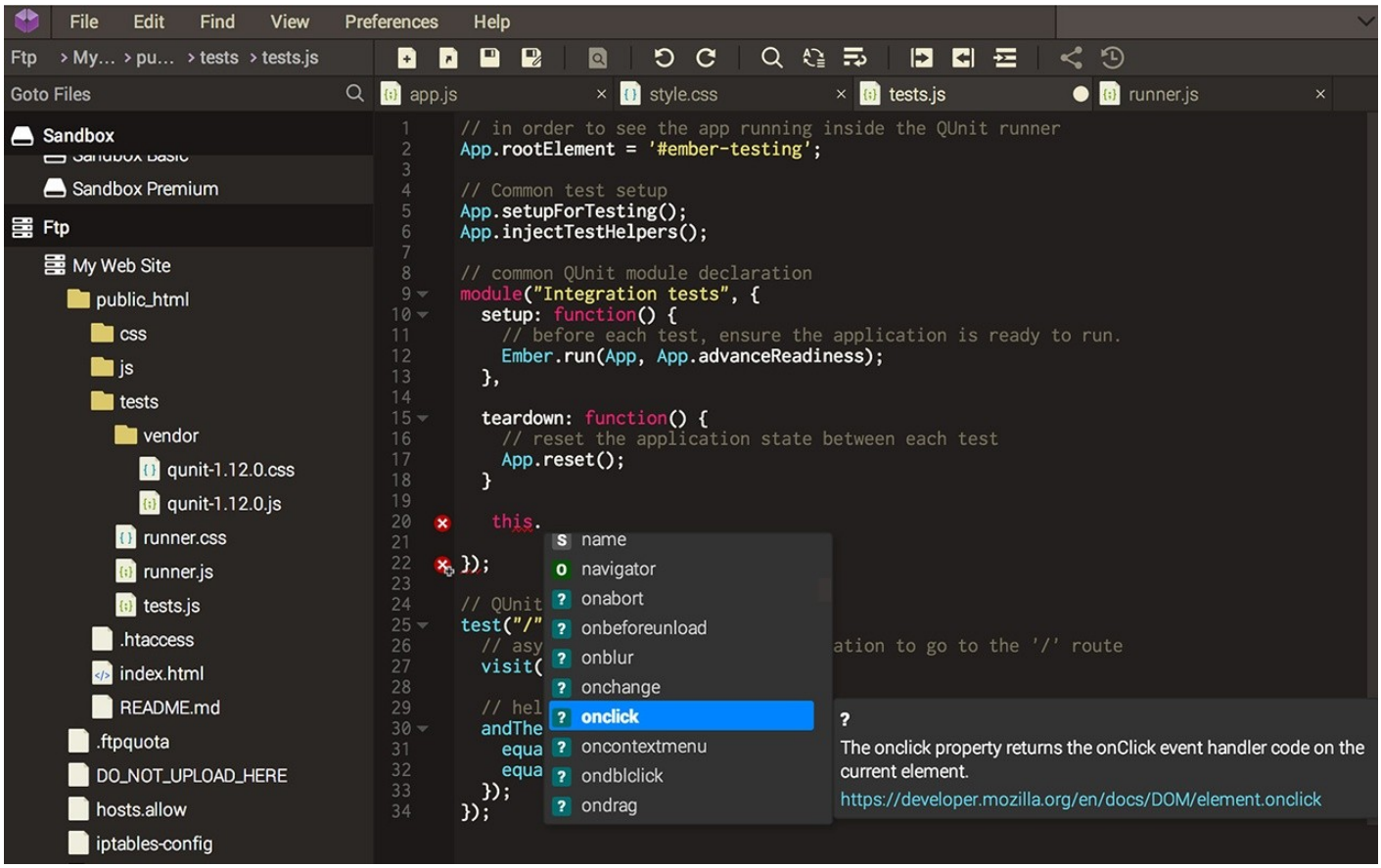

Figura 1.2.3: CodeAnywhere (obtenida de [3])

Una de las funcionalidades destacadas es la amplia variedad de servicios a los que puedes vincularte, como por ejemplo Dropbox, Google Drive, Github, servidores FTP, etc. Tal vez una de las limitaciones encontradas es el bajo número de lenguajes de programación que puedes desarrollar en ella, ya que se ciñen a los más usuales entre la comunidad.

1.3 Objetivos

El objetivo principal de este TFG consiste en desarrollar e implantar una plataforma que permita tanto a profesores y alumnos asignar y acceder a entornos de desarrollos a medida para la realización de proyectos informáticos en el ámbito educativo.

Entre los hitos fundamentales que la herramienta pretende alcanzar se podrían nombrar los siguientes:

- Proporcionar privacidad y un control de usuarios para el acceso a los entornos de desarrollo.
- Brindar un nuevo servicio de la ULL para que los alumnos no tengan que acudir a herramientas externas más inseguras.
- Desarrollar una plataforma estable y escalable, que pueda alojar a todos los usuarios que se requieran en cada instante.

1.4 Planificación

A continuación se detallarán de forma ordenada las actividades que se han ido realizando durante el desarrollo del proyecto hasta alcanzar los objetivos indicados.

1. **Definición de los requisitos**, que marcarán el objetivo principal de éste proyecto.

2. **Estudio, análisis y pruebas de las diferentes herramientas** posibles que se puedan emplear, de ésta forma tendremos claro cual es el camino a seguir.

3. **Familiarizarse con el entorno establecido**, donde deberemos de adquirir numerosos nuevos conocimientos de las herramientas que se utilizarán, en el capítulo 2 se especificarán algunos de ellos.

4. **Desarrollo y corrección de errores de la plataforma**. Actividad que conllevará la mayor parte del tiempo establecido, en el que se tratará de cumplir los requerimientos nombrados en un principio. A su vez la detección y corrección de errores se hará de forma paralela al desarrollo, tratando de mejorar la estabilidad de la plataforma con el tiempo.

5. **Desarrollo de la memoria y tutoriales**. Por último se procederá a realizar la memoria del TFG, como también tutoriales que expliquen y simplifiquen la utilización de la plataforma de los usuarios finales.

Capítulo 2
Tecnologías

En éste capítulo se enumerarán tanto las tecnologías como herramientas utilizadas durante el desarrollo de la plataforma, que a su vez ayudarán a conseguir los objetivos de este TFG.

2.1 Atom

Atom[4] es un editor de código de fuente abierto para macOS, Linux, y Windows con soporte para plug-ins escrito en NodeJS, incrustando Git Control y desarrollado por GitHub.

El IDE consta de una aplicación de escritorio construida utilizando tecnologías web. La mayor parte de los paquetes tienen licencias de software libre y es construido y mantenido por su comunidad. Atom está basado en Electrón, un framework que permite aplicaciones de escritorio multiplataforma usando Chromium y NodeJS. Está escrito en CoffeeScript y Less. También puede ser utilizado como un entorno de desarrollo integrado (IDE).

Figura 2.1: Logo Atom

Entre las características principales de este IDE nos podemos encontrar con las siguientes:

- Facilidad de instalación de un amplio abanico de paquetes modificables que hacen más personalizable esta herramienta.
- Gran variedad de lenguajes de programación soportados.
- Integración completa con Git, ya que al ser una herramienta desarrollada por integrantes de Github, tendrá funcionalidades muy útiles, como por ejemplo la sincronización automática y el muestreo de información adicional.

2.2 Eclipse Che

Eclipse Che[5][6], desarrollado por *Eclipse Foundation* y cuya publicación inicial fue en Septiembre de 2016, es un espacio de trabajo de código abierto potente y personalizable que ofrece al usuario un entorno de desarrollo

integrado en la nube. Está basado en java, y cada servidor ofrece un servicio web RESTful, proporcionando así una gran flexibilidad. A su vez contiene un SDK con el que se pueden desarrollar nuevos complementos o herramientas.

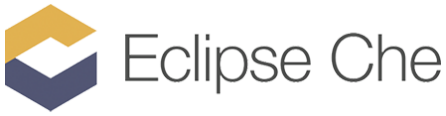

Figura 2.2.1: Logo Eclipse Che

Eclipse Che nos ofrece características atractivas para el desarrollo sin excluir los tiempos de ejecución y producción. Además nos ofrece espacios de trabajo “Dev Mode”, servidores de área de trabajo, varios frameworks y compatibilidad con diferentes lenguajes de programación.

Ésta herramienta se puede decir que será la principal de ésta plataforma, ya que al fin y al cabo es el servicio que se está intentando dar a los usuarios finales. Aunque durante la presente memoria se nombren continuamente características de ella, a continuación se enumerarán varias que se consideran las más descriptivas:

- **Software libre**: Gratis para que todos puedan descargar su código fuente y utilizarlo.
- **Multiplataforma**: Todos los usuarios de Windows, Gnu/Linux y Mac pueden hacer uso de la herramienta.
- **Software de código abierto**, donde cualquiera puede contribuir con el código fuente[7].
- El IDE es **personalizable**, utilizando pilas, plantillas, extensiones IDE, API RESTful, etc.
- Compatibilidad total con **Docker**.
- Espacios de trabajo compartibles.
- Posibilidad de conexión mediante **SSH**.
- Múltiples complementos personalizables (autocompletación de sintaxis, comprobación de errores, depurador, etc).
- Incluyen **Language Server Protocol** y Terminal.
- Capacidad de **pre-visualizar** y **depurar** proyectos.
- **No** tiene soporte multiusuario para el caso de una instalación **single server**.

Por último a continuación se muestra un ejemplo de la apariencia que tiene Eclipse Che en un navegador web (*Figura 2.2.2*).

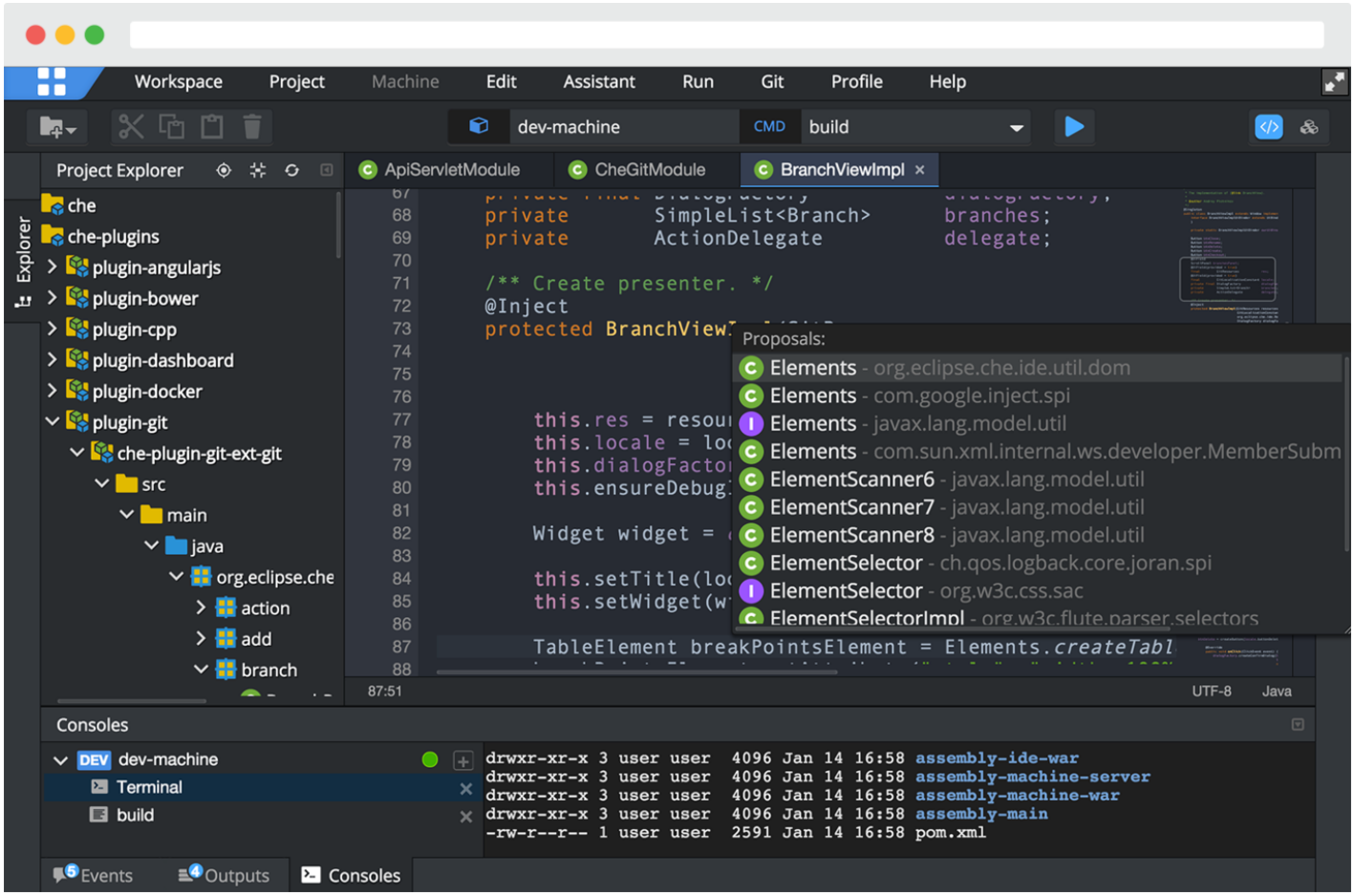

Figura 2.2.2: Captura workspace Eclipse Che (obtenida de [8])

2.3 Docker

Docker[9][10][11] es un proyecto de código abierto desarrollado por *Solomon Hykes* cuyo lanzamiento inicial fue en el año 2013. Esta herramienta automatiza el despliegue de aplicaciones dentro de contenedores software, proporcionando una capa adicional de abstracción y automatización de virtualización a nivel de sistema operativo, ya que utiliza características de aislamiento de recursos del kernel de linux .

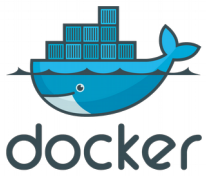

Figura 2.3: Logo Docker

La principal intención de ésta tecnología es la de tener la capacidad de ejecutar varios procesos y aplicaciones separados unos de los otros para hacer un mejor uso de su infraestructura, y mantener la seguridad que tendría con sistemas separados. Las herramientas del contenedor, como Docker, ofrecen un modelo de implementación basado en imágenes. Esto permite compartir una aplicación o un conjunto de servicios, con todas sus dependencias en varios entornos.

2.4 NodeJs

Node[12][13] es un entorno de Javascript del lado del servidor cuya característica principal es que está basado en eventos. Node ejecuta Javascript utilizando el motor V8, desarrollado por Google para uso de su navegador Chrome. La ventaja de éste lenguaje es su velocidad, ya que dicho código javascript es compilado a código máquina nativo, en lugar de interpretarlo o ejecutarlo como bytecode.

Figura 2.4.1: Logo NodeJS

Además de la alta velocidad de ejecución de Javascript, la verdadera característica que lo hace único es el llamado **Bucle de Eventos** (Event Loop). Para escalar grandes volúmenes de clientes, todas las operaciones intensivas I/O se llevan a cabo de forma asíncrona. Cuando una aplicación necesita realizar una operación de bloqueo (lectura de archivos, peticiones http, etc) envía una tarea asíncrona al event loop, junto con un callback para que cuando se termine de ejecutar siga donde le indiquemos. A pesar de ser una de las grandes ventajas de Node, es necesario controlar de manera efectiva el asincronismo, ya que si queremos un orden debemos de cerciorarnos de que todo se ejecuta conforme a nuestras necesidades.

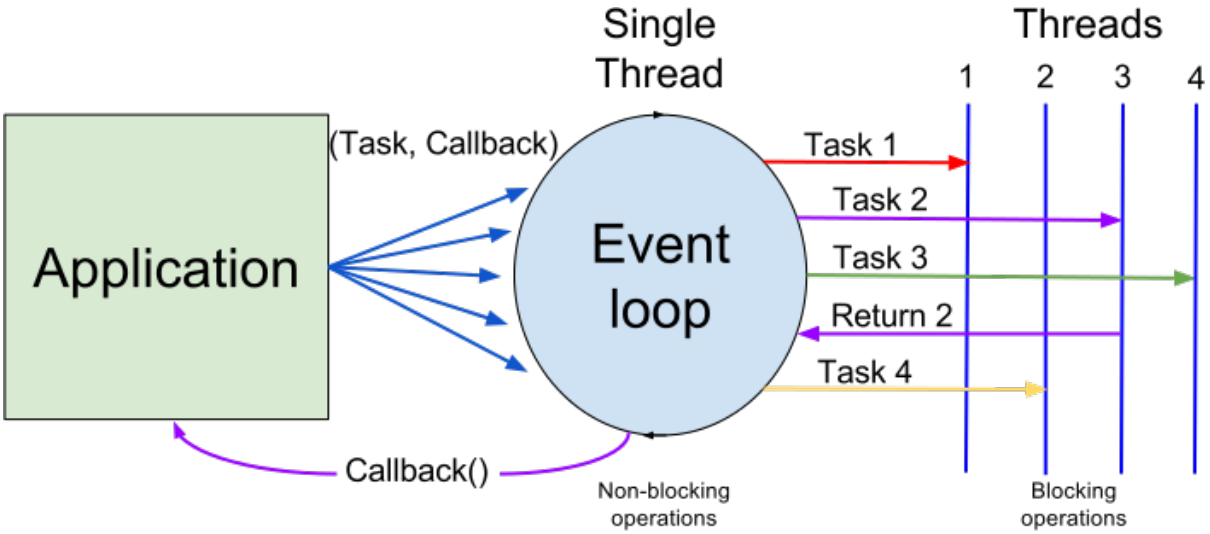

Figura 2.4.2: NodeJS Event Loop (obtenida de [14])

Por último, cabe destacar que a pesar de que se trata de una tecnología bastante reciente y en continuo despliegue, consta de una amplia comunidad que aporta cada vez más módulos que hacen potente y atractivo este lenguaje de programación.

2.5 oVirt

Ovirt[15], cuyo autor original es Red Hat, es una plataforma de virtualización de código abierto por la cual podemos administrar por medio de una aplicación web dicha infraestructura. Podríamos decir que actualmente oVirt es la versión open source que compite con Vmware Vsphere a nivel de soluciones de virtualización en datacenters.

Ovirt utiliza libvirt, lo cual permite manejar diferentes máquinas virtuales corriendo en diferentes backend, incluyendo KVM, Xen y Virtualbox.

Figura 2.5.1: Logo Ovirt

A continuación (*Figura 2.5.2*) se mostrará un modelo sobre la arquitectura que tiene:

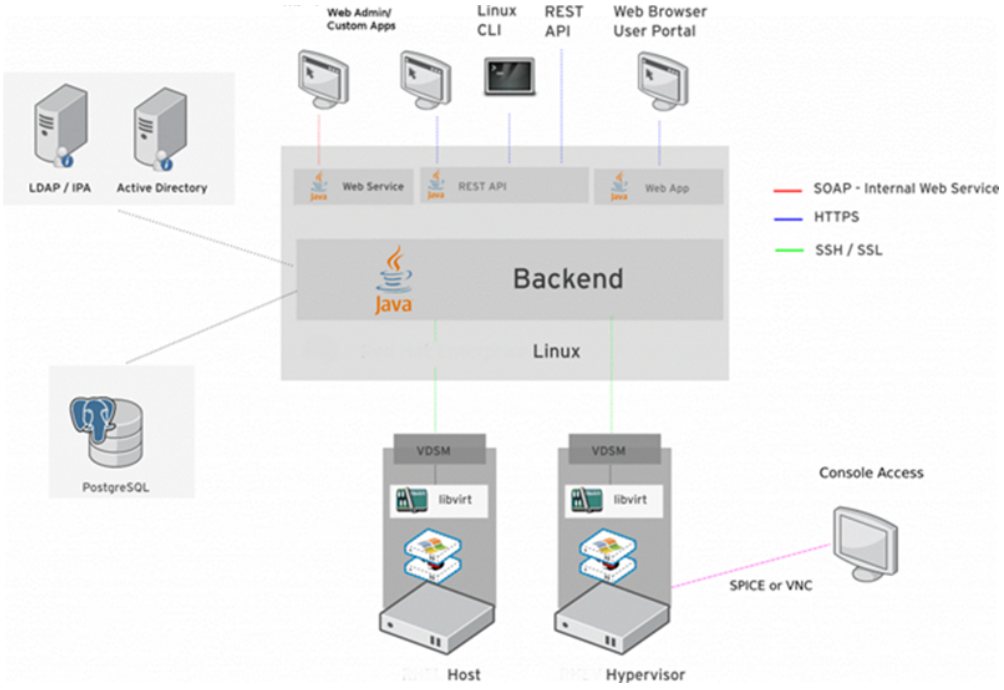

Figura 2.5.2: Arquitectura oVirt (obtenida de [16])

- **oVirt-engine:** el cual es usado para desplegar, monitorizar, mover, parar y crear imágenes de VM, así como configurar almacenamientos, redes, etc.
- Uno o más **hosts** (nodes), en donde se ejecutarán las máquinas virtuales (VM).
- Uno o más nodos **Storage** (almacenamiento), en donde estarán las imágenes ISO que corresponden a cada VM.

Además se necesita un servicio de autenticación para manejar todo el sistema de administración de usuarios para ovirt-engine, para ello se puede hacer uso de Active Directory, LDAP o IPA.

Por último, se podría nombrar diversas características de ésta plataforma, como por ejemplo:

- **Alta disponibilidad**: Reiniciar los Guests Vms de un host con errores automáticamente a otro host.
- **Migración en caliente de VM y Storage:** Poder mover máquinas de un disco a otro sin downtime.
- Balanceo continuo de los recursos de las VM.
- **Administrador de mantenimiento y de imágenes**: Creación de plantillas para VM, snapshots, etc.
- **Monitorización e informes**.
- **Importar y exportar VM**.

2.6 MySQL

MySQL[17][18] es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual (Licencia pública general / Licencia comercial) por Oracle Corporation y está considerada como la base de datos de código abierto más popular del mundo.

Figura 2.6: Logo MySQL

A continuación nombramos varias de las características más reconocidas de MySQL:

- Amplio subconjunto del lenguaje SQL, además de la adición de diversas extensiones.
- Alta disponibilidad en cuanto a plataformas y sistemas.
- Posibilidad de personalización en cuanto a velocidades, transacciones, soporte, etc.
- Transacciones y claves foráneas.
- Conectividad segura.
- Replicación.
- Búsqueda e indexación de campos de texto.
- Permite escoger entre múltiples motores de almacenamiento para cada tabla.
- Agrupación de transacciones, reuniendo múltiples transacciones de varias

conexiones para incrementar la velocidad de cómputo de éstas.

2.7 Servicio de Auntenticación Centralizado (CAS)

CAS (**Central Authentication Service**) es una aplicación web que nos permite implementar el conocido **SSO** (**Single Sign On**), que es un procedimiento de autenticación que habilita a un usuario para acceder a distintas aplicaciones web (en distintos dominios y en distintos servidores) con hacer login una única vez.

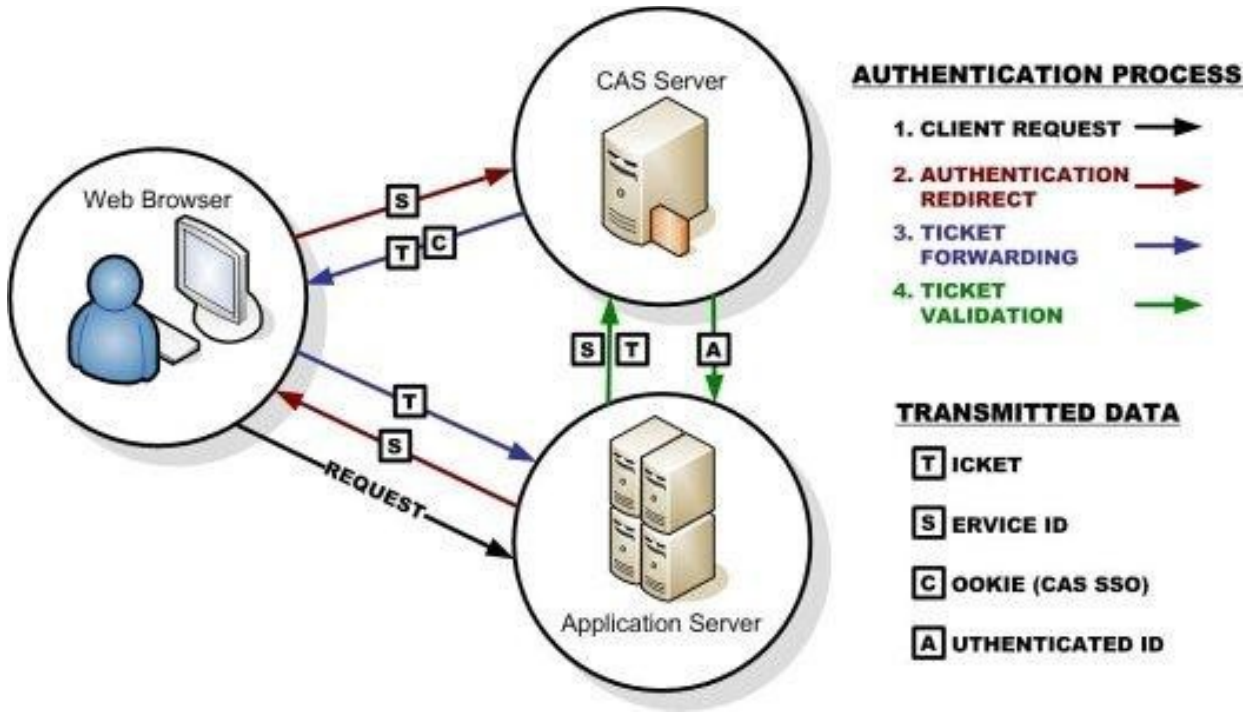

Figura 2.7: Servicio de Autenticación Centralizado (CAS) (obtenida de [19])

En general, cuando un usuario se conecta a una de estas aplicaciones el sistema comprueba si está autenticado y si no lo está, lo redirige a la pantalla del servidor de autenticación. Si la autenticación es correcta, el servidor CAS le vuelve a redirigir al usuario a la página que quería acceder en el primer momento.

Éste sistema es una herramienta muy útil para los desarrolladores, ya que no se deben preocupar en hacer una autenticación de usuarios segura, sino que se centraliza en un único método para todas las páginas que queramos reunir.

Actualmente se hará uso del servidor CAS implementado en la ULL, por el que se podrá acceder a la plataforma mediante las credenciales institucionales asignadas para los demás servicios que utilizamos (Campus Virtual, Portal, Sede Electrónica, etc).

2.8 Redis

Redis[20][21] es un motor de base de datos en memoria, desarrollado por *Salvatore Sanfilippo* y lanzado en 2009, basado en el almacenamiento en tablas de hashes (clave/valor) pero que opcionalmente puede ser usada como una base de datos persistente.

Figura 2.8: Logo Redis

Su funcionamiento como ya se nombró anteriormente es de tipo diccionario (clave/valor), siendo la principal diferencia con otros sistemas similares es que los valores no están limitados a ser de tipo string, sino que permite valores de tipo Listas, Set o hashes de strings.

A su vez desde la version 2.6 es capaz de ejecutar scripts en el servidor Redis escritos en lenguaje Lua.

2.9 Iptables

Un firewall (cortafuegos) es una medida de protección básica en un servidor. Es utilizado para filtrar o restringir el tráfico de red en un sistema, tanto entrante como saliente. En los sistemas operativos Linux, **iptables**[22] es la herramienta más ampliamente utilizada para implementar firewalls.

Figura 2.9: Iptables

Iptables gestiona, mantiene e inspecciona las reglas de filtrado de paquetes Ipv4 a través de tablas. Estas tablas clasifican y organizan las reglas de acuerdo al tipo de decisiones que se deben tomar sobre los paquetes. A su vez, dentro de cada tabla las reglas se organizan en cadenas (chains), que representan el tipo de evento que disparan o inician cada regla.

- **Prerouting**: Tráfico entrante, son procesadas antes de tomar cualquier decisión de ruteo.
- **Input**: Tráfico entrante, luego de haber sido ruteado y destinado al sistema local.
- **Forward**: Tráfico entrante, luego de haber sido ruteado y destinado hacia otro host (reenviado).
- **Output**: Tráfico saliente originado en el sistema local.

- **Postrouting**: Tráfico saliente orginado en el sistema local o reenviado, luego de haber sido ruteado.

2.10 WebSocket

Websocket[23] es un protocolo nuevo para la web bajo TCP, por el cual, a diferencia de la conexión que venimos usando bajo HTTP, este es bi-direccional, es decir, las dos partes de la conexión pueden enviar información a la otra, al contrario de la mentalidad anterior en la que uno hace de cliente y otro de servidor.

El funcionamiento de éste protocolo puede verse a continuación:

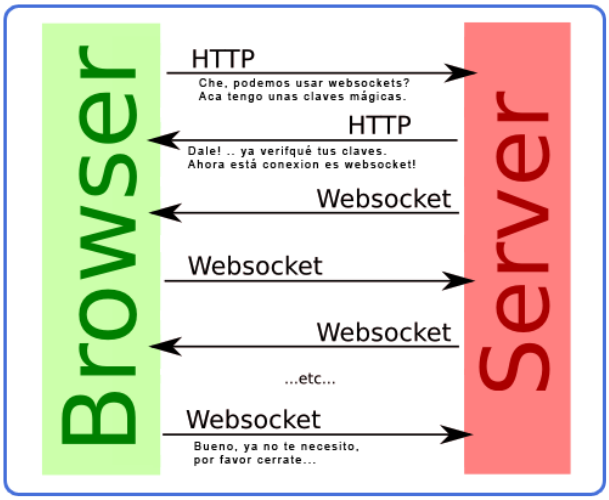

Figura 2.10.1: Funcionamiento WeBsocket (obtenida de [24])

Para iniciar la conexión con el protocolo WS primero el cliente le pide al servidor que quiere iniciar esta conexión (handshake), y éste le responde aceptándola. Es a partir de aquí cuando se deja de usar HTTP y pasan a WS.

Las ventajas de utilizar éste protocolo son varias:

- Mucho más rapido que HTTP.
- Reduce el uso de la red, ya que tras la conexión no se utilizan cabeceras innecesarias.
- Minimiza la latencia en las conexiones.
- Facilita una mayor escalabilidad en la web.

Figura 2.10.2: Logo socket.io

Socket.io[25] es una librería de open source que nos permite manejar eventos en tiempo real mediante el protocolo de websockets. Dicha librería está soportada para un gran número de lenguajes de programación, facilitándo así la comunicación entre todo tipo de aplicaciones escritas. Por otro lado, está enfocada tanto a cliente como servidor.

Por último, como ya dijimos, su simplicidad y organización mediante eventos es la característica que la hace más atractiva, ya que se puede dividir tareas de manera ordenada y poder controlar la transferencia de información eficazmente.

2.11 HTML, CSS y JS

Para el desarrollo del frontend de la plataforma se hará uso de los lenguajes web estándar que se muestran en la ilustración.

Por un lado se utilizará el lenguaje de marcado HTML para mostrar el contenido y la estructura de la página web en cuestión, por otro lado para darle una apariencia personalizada utilizaremos el lenguaje de diseño gráfico CSS, mientras que por último utilizaremos Javascript para añadir funcionalidades a la plataforma.

Figura 2.11.1: Logos HTML, CSS, JS

Para la implementación del frontend con el lenguaje NodeJS anteriormente nombrado, se utilizará el módulo de **ExpressJS** añadiendo a su vez el motor de vistas denominado **EJS**, ésto nos da la posibilidad de formar una web que aloje a distintos usuarios con datos dinámicos.

Capítulo 3
Desarrollo e implementación

Durante este capítulo se abordarán, de manera resumida dadas las limitaciones de espacio, las tareas más destacables que se han tenido que desarrollar para el cumplimento de diversos objetivos de este TFG.

3.1 Idea principal y características a solucionar

En primer lugar, partimos de la necesidad de proporcionar a los alumnos un entorno de programación asociado a la ULL, para ello evaluamos diversas alternativas siendo la mejor opción el IDE Eclipse Che[17][26], básicamente porque se trata de código libre y que actualmente tiene un gran auge en el mundo de los entornos en la nube, ya que está en continuo desarrollo ofreciendo nuevas características y soporte.

Al escoger ésta plataforma lo que primero se hizo fue comprobar tanto sus funcionalidades como la forma en la que se ejecuta y trabaja, con esto pudimos ver que nos encontramos ante una herramienta potente que puede estar a la altura de muchas otras y de los objetivos que teníamos al principio. Por otro lado también nos encontrarnos con diversos inconvenientes que de cierta manera indicaban que tareas debían desarrollarse para solventarlos y dar un servicio medianamente acorde a lo esperado.

La primera característica destacable es que estábamos ante un IDE que **no tiene control de usuarios**, por lo que si se ejecuta en una máquina todos los que estén dentro de la red pueden acceder a la plataforma, lo cual no es lo que se pretende, ya que se quiere un servicio con seguridad en el que cada usuario tenga sus **entornos privados** independientemente de los demás. Para solucionar este problema se ideó un portal intermedio y un esquema de red que veremos en las siguientes secciones.

Por otro lado, la forma en la que ésta plataforma trabaja es importante, ya que dependiendo de ella tuvimos que buscar una solución escalable para soportar a todos los usuarios. A continuación (*Figura 3.1.1*) se muestra una ilustración que lo explica:

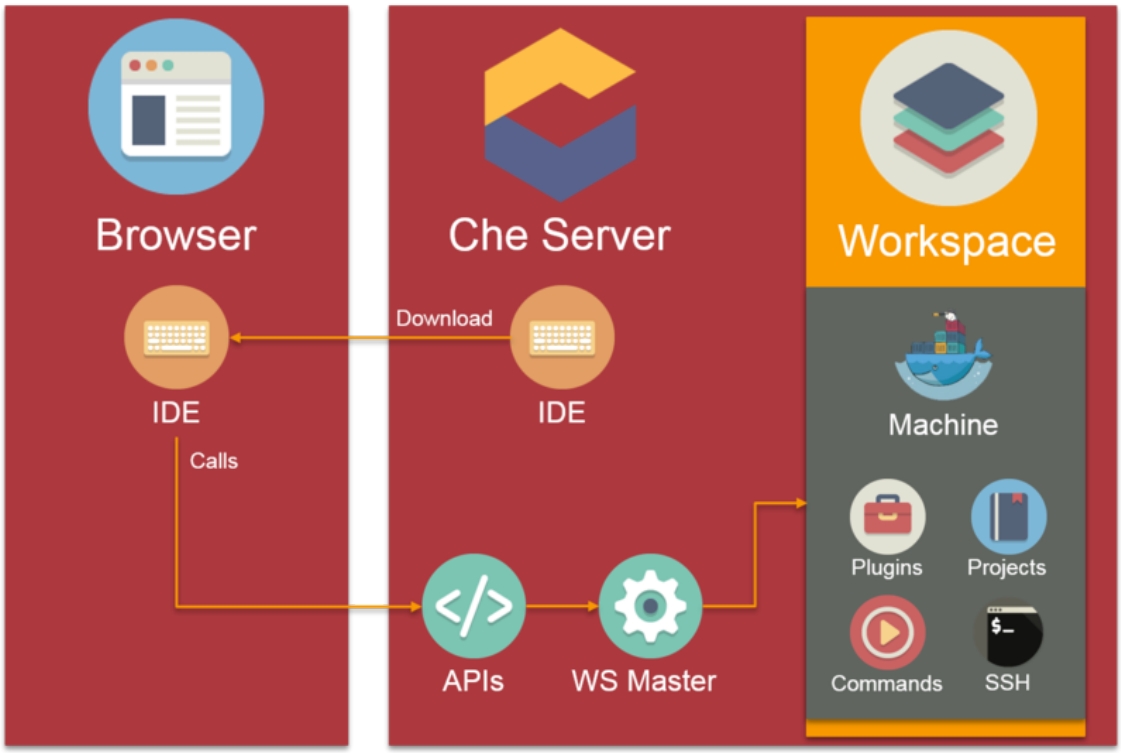

Figura 3.1.1: Modo de funcionamiento de Eclipse Che (obtenida de [26])

El servidor se ejecuta sobre un contenedor de docker, a su vez a través de éste se pueden crear diferentes workspaces, que no son más que máquinas con unas características y paquetes instalados determinados, dichos workspaces se ejecutan sobre contenedores de docker.

Port »»»»»»»»»»»	Service »»»»»»»»»»»	Notes
8080	Tomcat Port	Che server default port
8000	Server Debug Port	Users developing Che extensions and custom assemblies would use this debug port to connect a remote debugger to Che server.
32768-65535	Docker and Che Agents	Users who launch servers in their workspace bind to ephemeral ports in this range. This range can be limited.

Figura 3.1.2: Puertos necesarios en Eclipse Che (obtenida de [26])

Cuando un servidor Che está en funcionamiento, para que éste pueda ser alcanzado por los clientes (navegadores web), deberán comunicarse a través de unos puertos TCP específicos, los más importantes son los siguientes:

- **Puerto servidor Che**: Cuando levantamos un servidor Che le indicamos el puerto (de la máquina en la que está) que queramos que escuche (ej:8080).

- **Puertos efímeros**: La documentación de Eclipse Che nos indica que debemos dejar estos puertos (32768-65535) abiertos a la comunicación, ya que es a través de ellos donde se realiza la comunicación con los agentes de los workspaces que puedan existir. Cuando un workspace es creado, se escogen aleatoriamente puertos que no estén en uso.

Otro de los objetivos principales que se proponía era darle a la plataforma un enfoque educativo, es decir, un lugar donde profesores y alumnos sean los primordiales destinatarios de éste servicio.

Aunque las funcionalidades de cada uno de los roles se nombrarán más detenidamente en el capítulo 4, aclararemos ahora el motivo de esta elección. Hasta el momento, cuando un alumno tenía que realizar cualquier desarrollo, debía de hacer uso de herramientas externas (IDEs, ordenador personal, etc) que escapaban del control docente, es por ello que los profesores son quienes tienen el poder de autorizar el uso de un IDE para cada alumno.

El funcionamiento se asemeja al sistema de asignación de máquinas virtuales del IaaS[27], donde los profesores tienen un pool de recursos, pudiendo asignarlos a los usuarios que crea convenientes. De manera análoga, el profesorado podrá conceder un entorno de desarrollo a sus alumnos por un motivo específico (asignaturas, exámenes, prácticas, etc), obteniendo a su vez información continua sobre el estado del mismo y teniendo la potestad de realizar acciones sobre él, como por ejemplo determinar la desmatriculación de usuarios y por consiguiente la eliminación de los datos generados por el IDE.

Vistas las características anteriores, concluimos que lo correcto sería que se le asignara un servidor Che a cada alumno por cada servicio que le hayan proporcionado. Para darle la seguridad, privacidad y escalabilidad que se requiere decidimos que la idea que más se ajustaba era la de elaborar una granja de servidores Che, un sistema de máquinas virtuales que se ejecutaran según la demanda de los usuarios.

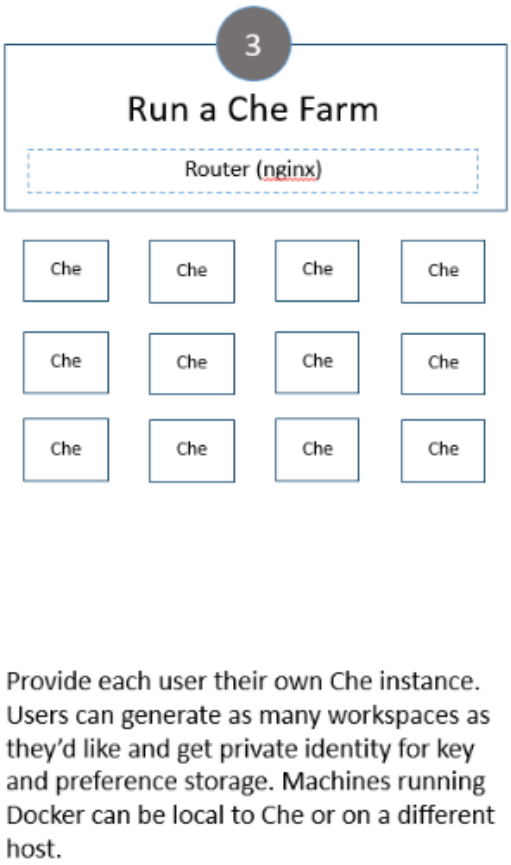

Figura 3.1.3: Che Farm (obtenida de [26])

Llegamos a la conclusión que para su correcto funcionamiento se debían cumplir estas definiciones:

- Todos los IDEs (servicios asignados por un profesor) pertenecientes a un alumno/usuario se debían alojar en la misma máquina, ya que debido a características inherentes del propio Eclipse Che (puertos que necesita) y networking posterior es la única solución que se ha encontrado.
- IDEs (servidores Che) de varios usuarios pueden compartir la misma máquina en la que se ejecuta, compartirán puertos efímeros. Ésta característica será modificable en la plataforma desarrollada, se podrá elegir tanto el número de usuarios por máquina virtual (Nºusuarios/VM) como el número de servidores Che corriendo al mismo tiempo por usuario (NºChe/usuario).
- El usuario final cuando se conecta a un IDE no tiene que saber en qué máquina está alojado ni con quien comparte recursos, la plataforma debe hacer invisible a los demás el sistema interno que tiene.

3.2 Esquema de red

Como se comentó anteriormente, para poder proporcionar un control de usuarios y privacidad a la plataforma se elaboró un esquema de red específico (*Figura 3.2.1*). En este apartado se explicarán las razones por las cuales se ha tomado dicha solución.

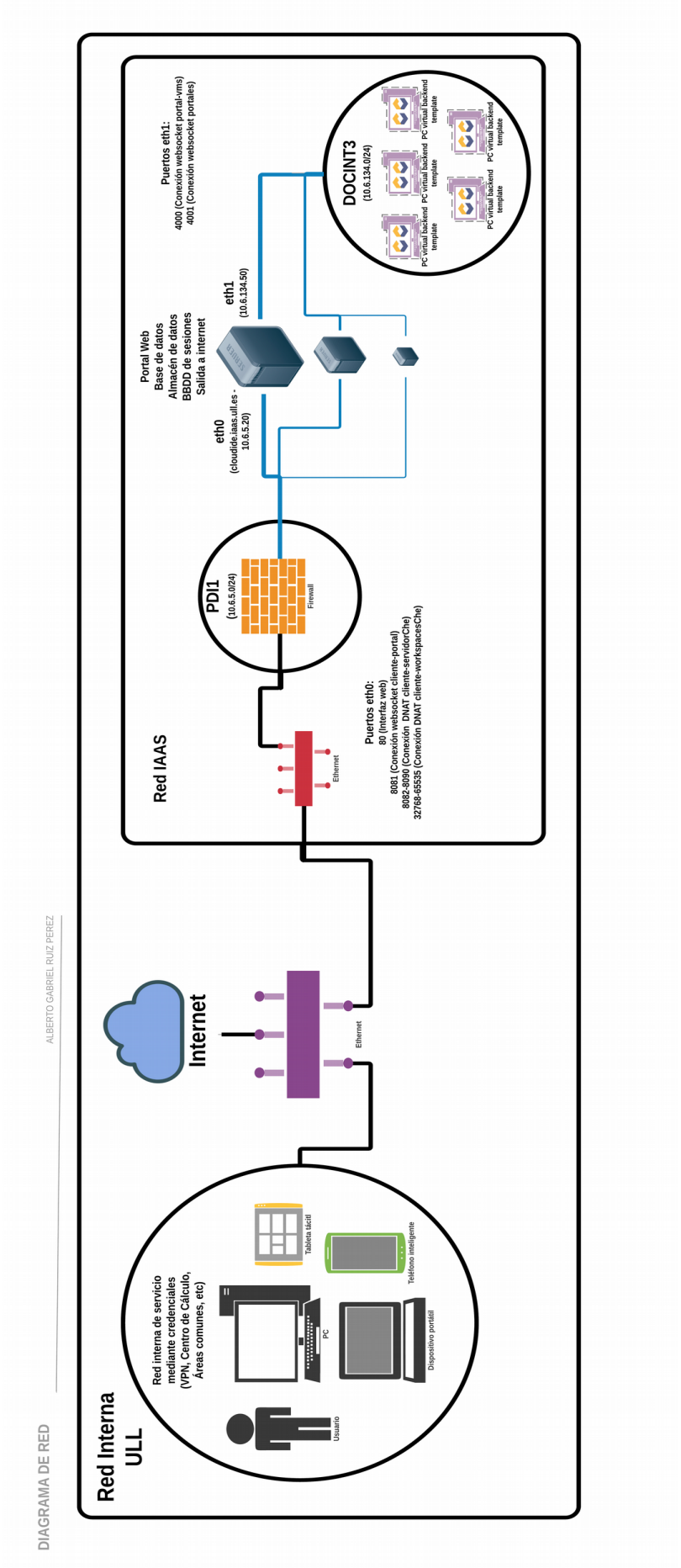

Figura 3.2.1: Esquema de red

Como podemos ver, dentro de la ULL coexisten varias redes, aunque en la imagen solo se muestran las que son relativas a nuestro proyecto.

- **Red de servicio**: Es la red a la que se conectan todos los dispositivos que están físicamente en las instalaciones de la ULL o aquellos que se han conectado mediante VPN. Para tener acceso a esta red es necesario un credencial institucional, de tal forma que se pueda emparejar a un usuario con un dispositivo y una dirección interna única. Será desde ésta red donde se podrá acceder a la plataforma.
- **Red IAAS**: El Servicio de Tecnologías de la Información y Comunicación (STIC) de la Universidad, como bien nombramos anteriormente también da soporte IaaS con una infraestructura de máquinas virtuales soportadas mediante la plataforma de oVirt. Dentro de ésta se administran diferentes subredes con determinados rangos de direcciones, ésto es de gran beneficio, ya que de ésta manera existe la posibilidad deconfigurar redes con unas características específicas que sirven de ayuda a numerosos proyectos como el de éste TFG. Se pueden nombrar varias redes que existen actualmente (DOCINT1, PDI1, DOCINT2, DOCINT3, etc), pero ahora nos centraremos en las dos principales que tienen repercusión en la plataforma:
  - **PDI1**: Ésta red de rango de direcciones 10.6.5.0/24 se caracteriza por tener acceso a internet y comunicación directa con la red de servicio, ésto significa que si un servidor está localizado aquí, los usuarios de la ULL tendrán acceso mediante su dirección.
  - **DOCINT3**: Ésta red de rango 10.6.134.0/24 se caracteriza por **no tener acceso a internet y a otras redes**, se trata de una red aislada. Para éste proyecto será la clave para proporcionar la privacidad y seguridad necesaria en los IDEs de cada usuario, además de albergar a las VMs backend que se creen dinámicamente.

Ya descritas las características más señalables de cada red, nombramos ahora el servidor central, el portal, la máquina donde se alojará tanto la plataforma web como los datos de almacenamiento de cada IDE de los usuarios, a su vez también un servidor MySQL y de Redis.

A continuación se muestra una tabla (tabla 3.2) que contiene las especificaciones de dicha máquina:

SO	RAM	SWAP	Almacenamiento	CPU	Arquitectura	I. Red
Ubuntu 16.04.4 LTS	4GB	1GB	20GB	4 x Intel Core i7 9xx	64bits	2

Tabla 3.2: Tabla de especificaciones máquina portal

Esta tendrá dos interfaces, la eth0 (cloudide.iaas.ull.es – 10.6.5.20) y la eth1 (10.6.134.50), por lo que con dicha configuración se abre un amplio esquema de posibilidades. Por un lado al ser la única conexión entre ambas subredes, será el enlace por el que pasen todos los datos tanto de entrada como de

salida, por lo que se podrá controlar mediante firewall el flujo de información. A su vez dotamos a las máquinas backend de la red DOCINT3 de acceso a internet, ya que es necesario para la descarga de datos de los servidores Che.

Por otro lado existe unos puertos determinados que han de estar disponibles a los clientes, para ello se han tenido que añadir una serie de excepciones en el firewall institucional, ya que por defecto solamente admite una cantidad de puertos insuficientes para el correcto funcionamiento:

- **Puerto 80**: Puerto por el que escuchará el servidor desarrollado, que proporcionará tanto la interfaz de usuario como servicio para determinadas peticiones POST.
- **Puerto 8081**: Conexión WebSocket desde la interfaz hacia el servidor portal, será a través de dicho puerto por el que se envíen y reciban diversos eventos asíncronos (encendido y apagado de servidores).
- **Puertos 8082-8090**: Rango de puertos por los que el cliente se conectará al servidor Che que se le haya asignado, para que dicha conexión tenga éxito hacia las máquinas internas se deberá realizar una redirección de puertos determinada a la necesidad del momento, más adelante se comentará específicamente éste caso.
- **Puertos 32768-65535**: Rango de puertos por los que el cliente se conecta a los workspaces que haya creado y ejecutado, dichos puertos son efímeros y aleatorios, es el servidor Che quien decide cual utilizar, como en el caso anterior, es necesario redireccionar el rango completo para asegurar la conexión.
- **Puerto 4000**: Conexión WebSocket desde el portal hasta las máquinas virtuales backend, a través de ésta conexión irán los eventos de encendido y apagado de servidores Che, como también el control del estado de las máquinas.
- **Puerto 4001**: Conexión WebSocket entre portales, por si hiciera falta en un futuro trabajar con varios servidores balanceando la carga, para ello haría falta una comunicación de datos continua para el sincronismo de información.

Por último, se puede ver en la imagen (*Figura 3.2.1*) que se han añadido varios servidores portal, esto quiere decir que aunque desde un principio y actualmente la plataforma se ejecuta con uno solo, en un futuro puede funcionar conjuntamente con varios y un servidor HA proxy balanceando la carga. Por ello se decidió usar Redis para almacenar las sesiones de los usuarios, de ésta manera serían accesibles entre todos los portales ya que deben contener la misma información. También existe una conexión directa entre servidores a través del puerto 4001 como se comentó antes, básicamente sincroniza información entre ellos, útil para aplicar reglas firewall.

3.3 Backend VMs

En este TFG denominamos *backend vms* al proceso o configuración que se ejecuta en las máquinas virtuales que se crean y que albergan los servidores que ejecutan Eclipse Che de un usuario cualquiera.

Como se muestra en el esquema de red (*Figura 3.2.1*) y tal y como acabamos de decir, las máquinas backend son las máquinas virtuales que se crean dinámicamente según la necesidad y carga de usuarios. En este apartado nos centramos en la configuración que se ha tenido que hacer y en el proceso desarrollado para la plataforma.

Para un mejor entendimiento de esta parte del desarrollo, lo dividiremos en necesidades y soluciones propuestas, que cronológicamente han ido consiguiendo una serie de objetivos requeridos.

- **Creación de la imagen base como plantilla**: Desde el primer momento se empezará a crear una imagen en oVirt de una máquina virtual que se utilizará como plantilla para crear las demás. En dichas máquinas será necesario la inclusión de una serie de paquetes, entre los más destacados podemos nombrar a Docker y NodeJS, éstos permitirán ejecutar el proceso desarrollado, que a su vez hará uso de scripts para el levantamiento de servidores de Che. A continuación se muestra una tabla (*tabla 3.3*) que contiene las especificaciones de dichas máquinas:

SO	RAM	SWAP	Almacenamiento	CPU	Arquitectura	I. Red
Ubuntu 16.04.4 LTS	4GB	1GB	20GB	2 x Intel Core i7 9xx	64bits	1

Tabla 3.3: Tabla de especificaciones máquinas backend

- **Proporcionar conexión a internet**: Como bien hemos nombrado, estas máquinas se sitúan en una subred aislada sin conexión a internet (DOCINT3), pero como es lógico éstas necesitan descargar una serie de datos de internet. Para solucionar ésto decidimos que tanto el tráfico entrante y saliente debía pasar por el enlace de la máquina portal. Desde las máquinas backend tan solo debemos poner una regla que diga que todo el tráfico por defecto debe ir hacia la ip del servidor central:

  ```
  $ sudo route add default gw 10.6.134.50
  ```

  Y en la máquina portal habrá que activar el bit de *ip_forward* para que acepte los paquetes y los rediriga hacia su destino.

- **Asignación de IPs a Vms**: Para este proyecto se necesitará que cada máquina virtual tenga una dirección ip única y determinada por nosotros, para ello será necesario poder asignarle una ip a una vm **antes de su encendido**. ¿Cómo hacemos este proceso?, la clave está en la herramienta **cloud-init**, ésta es utilizada en ámbitos de virtualización que permite inicializar varios aspectos de una máquina virtual antes de su encendido.

Se deberá realizar una instalación en la máquina con el siguiente comando:

$ sudo apt-get install cloud-init

Con ello quedará instalada la herramienta. En nuestro caso la queríamos para la asignación de una dirección, para ello hay que modificar el fichero */etc/network/interfaces*, quedando tal que así:

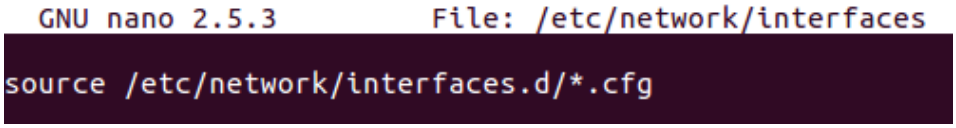

Figura 3.3.1: Cloud-Init para asignar dirección ip

Con ésta configuración será posible la asignación de ip mediante cloud-init, en capitulos posteriores veremos como se realiza éste proceso a través de scripts en python con la api de oVirt.

- **Montaje de directorio remoto de almacenamiento centralizado**: Toda la información de los entornos de desarrollo asociados a los usuarios se guardarán en la máquina portal de manera centralizada, con esto conseguimos que todas las máquinas puedan acceder a la misma información, haciendo así una plataforma independiente de los datos.

  Para ello se ha hecho uso del protocolo **NFS**, el cual nos permite la compartición de datos entre redes. La máquina portal es la que hará de servidor, por lo que se debió instalar los paquetes adecuados nfs.

  A continuación se procede a compartir el directorio en cuestión, con ello se tiene que modificar el archivo */etc/exports* (en la máquina servidor) y escribir lo siguiente:

  /mnt/cloudIDE 10.6.134.0/24 (rw,no_subtree_check,sync,no_root_squash)

  De esta forma nos aseguramos que se comparte ese directorio con la subred indicada y con los permisos establecidos (lectura y escritura).

  Para el caso de los clientes (máquinas backend) se instalarán los paquetes cliente de nfs y se realizará otra configuración:

  $ sudo apt-get install nfs-common

  Y para que el directorio remoto se monte nada más iniciar la máquina deberemos de editar el fichero */etc/fstab:*

  10.6.134.50:/mnt/cloudIDE /mnt/cloudIDE auto -netdev,defaults 0 0

  Así quedará configurado el directorio centralizado accesible desde todas las máquinas backend.

- **Desarrollo del programa backend**: En éste capítulo se detalla la que puede ser una de las partes más importantes de la plataforma, básicamente se ha desarrollado un programa en NodeJS que estará en continua comunicación con la máquina central (portal), y que se encargará de recibir información, levantar o apagar servidores de Eclipse Che y enviar información hacia sus respectivos clientes.

Para visualizar de mejor manera éste proceso, a continuación se adjunta una figura que refleja un supuesto caso de levantamiento de un servidor Che (*Figura 3.3.2*):

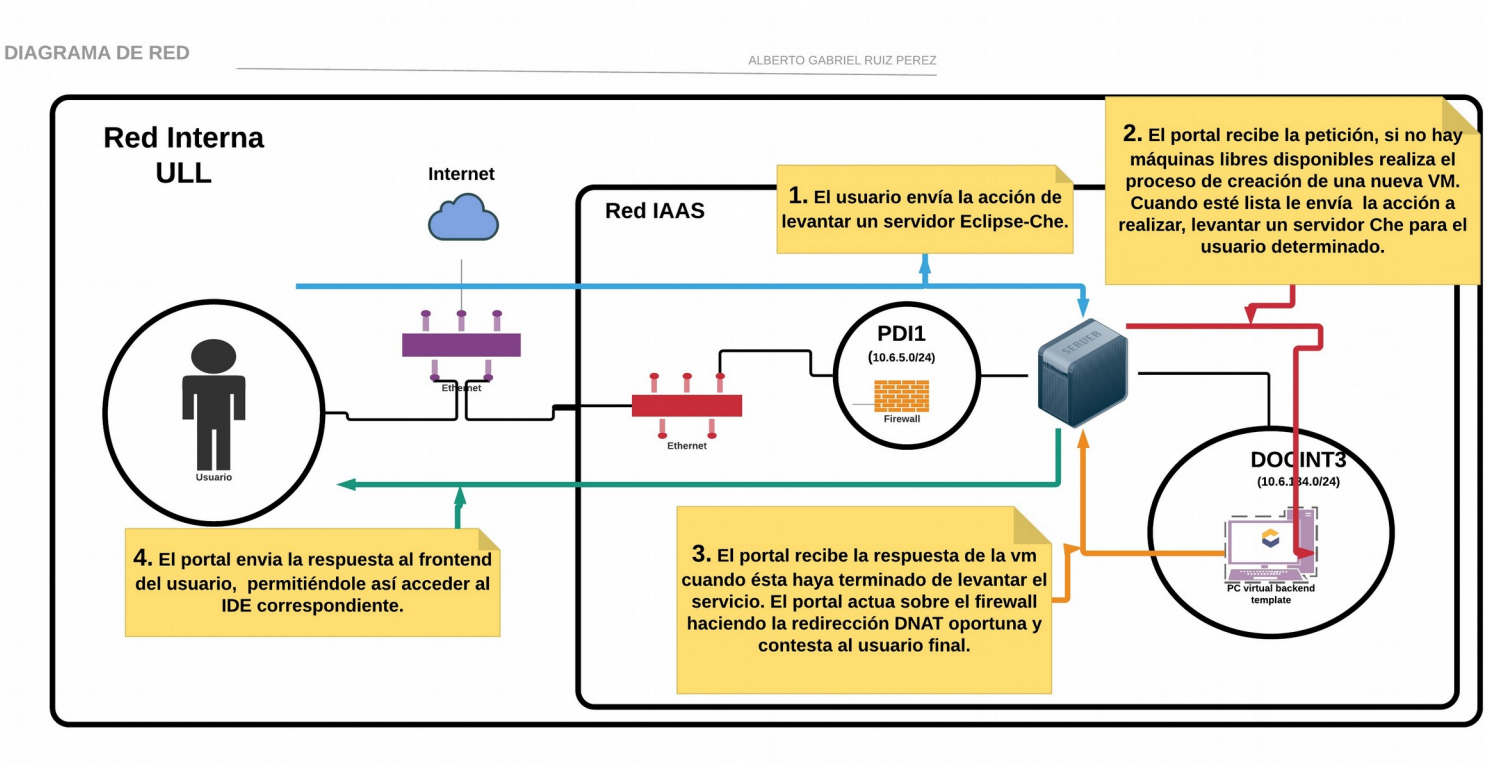

Figura 3.3.2: Procedimiento de actuación

Como se ha podido observar, el procedimiento de actuación no es mas que una transmisión de acciones entre distintos actores en el que cada uno realiza una determinada acción. En el caso de las VMs, nada más arrancar se ejecutará el programa que hemos desarrollado, esto creará una conexión permanente a través de websocket con la máquina portal, a través de ésta conexión se esperará cualquier acción a ejecutar, como también el envío del resultado cuando alguna se haya completado.

Para ahondar más en el detalle de éste procedimiento, a continuación se muestra un diagrama de flujo explicando resumidamente lo que sucede cuando una petición llega al servidor central.

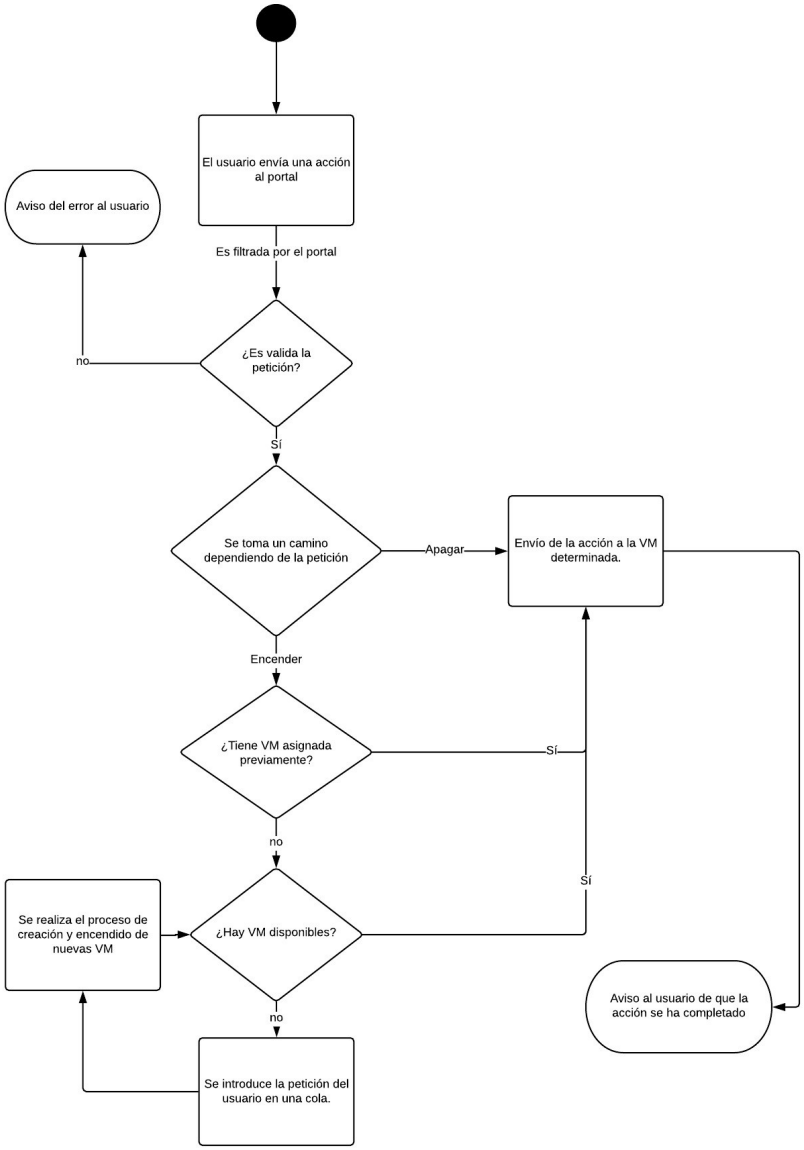

Figura 3.3.3: Diagrama de flujo del procedimiento de actuación

La máquina escuchará peticiones formando una cola de acciones que ejecutará una tras otra. Las instrucciones que podrá hacer se resumen al levantamiento y apagado de servidores de Eclipse Che en un puerto determinado y con los datos del usuario indicado (respaldados de forma centralizada). Como ejemplo podemos mostrar el script en bash que realiza el encendido y apagado de un servidor Che versión 6.0.0-M4 con docker (*Figura 3.3.3*):

```
if [ $1 -eq 1 ]
then


sudo docker run --rm -e CHE_CONTAINER_PREFIX='ULLcloudIDE' -e
CHE_WORKSPACE_AGENT_DEV_INACTIVE__STOP__TIMEOUT__MS=2592000000 -v /var/run/docker.sock:/var/run/
docker.sock -v $7$2:/data -e CHE_PORT=$3 -e CHE_HOST=$5 -e CHE_DOCKER_IP_EXTERNAL=$6 eclipse/
che:6.0.0-M4 start --skip:preflight


a=$(sudo docker ps -qf "name=ULLcloudIDE-$3")
sudo docker update --restart no $a

fi

if [ $1 -eq 0 ]
then
sudo docker run --rm -e CHE_CONTAINER_PREFIX='ULLcloudIDE' -e
CHE_WORKSPACE_AGENT_DEV_INACTIVE__STOP__TIMEOUT__MS=2592000000 -v /var/run/docker.sock:/var/run/
docker.sock -v $7$2:/data -e CHE_PORT=$3 -e CHE_HOST=$5 -e CHE_DOCKER_IP_EXTERNAL=$6 eclipse/
che:6.0.0-M4 stop --skip:preflight


fi
```

Figura 3.3.4: Script en bash para encender y apagar servidores Che

Por último, una de las especificaciones que tiene ésta máquina y que limita en mayor o menor medida es el almacenamiento, actualmente tienen un disco de 20GB donde se almacenará tanto el SO como los datos que genere la herramienta. Ésto es una característica importante a nombrar, ya que debido a la forma en la que trabaja Eclipse Che es necesario tener un control continuo. Los workspaces de Che son montados sobre imágenes personalizadas de docker, éstas imágenes pueden tener un tamaño variable (50MB - 2GB), pero que tras el paso del tiempo y de la continua creación de nuevos workspaces, las imágenes se almacenarán innecesariamente, llegando incluso a impedir el funcionamiento correcto del IDE. Para mitigar esto, se ha desarrollado un pequeño script (*Figura 3.3.4*), que se ejecuta en unos determinados momentos y que hace una limpieza de las imágenes que menos se utilizan:

```
sudo docker rmi $(sudo docker images | awk '$1 !~ /(alpine)|(eclipse\/che$)|(eclipse\/che-
action)|(eclipse\/che-test)|(eclipse\/che-server)|(eclipse\/che-init)|(eclipse\/che-dir)|
(eclipse\/che-mount)|(eclipse\/che-ip)|(docker\/compose)/ {print $3}')
```

Figura 3.3.5: Script en bash para limpiar imágenes Docker

- **Aceptar imágenes docker desde registros http**: Como se ha nombrado anteriormente, los workspaces de Che se ejecutan sobre imágenes personalizadas de docker, por lo que cualquiera puede construir la suya a medida (ej: profesor crea una imágen para los alumnos). Éstas deben ser guardadas en un registro de Docker, en el caso de que queramos alojarlas en el repositorio público oficial disponemos el propio de Docker[28], pero por otro lado podemos crearnos nuestro propio registro interno a la ULL.

  Para que dichas máquinas acepten imágenes desde registros no oficiales http es necesario añadir una configuración (*Figura 3.3.6*) a docker:

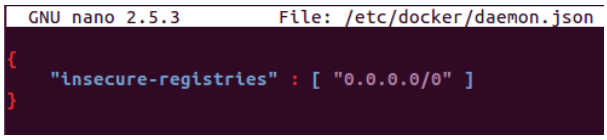

Figura 3.3.6: Configuración registros http docker

De esta manera aceptamos imágenes de cualquier repositorio, pudiendo así crear uno privado con las imágenes que se quieran[29].

- **Inicio automático del proceso al arrancar:** Como es lógico se hace necesario la ejecución del programa NodeJS desarrollado nada más arrancar la máquina, para ello haremos uso de dos herramientas bien conocidas:
  - **Forever**: Es un paquete NodeJS con el que se lanzará la aplicación en cuestión. Su objetivo es la de estar constantemente atento al estado del programa, por lo que si ocurre cualquier error inesperado volverá a iniciar la aplicación, manteniéndola siempre en funcionamiento.
  - **Crontab**: Es una herramienta muy utilizada en Unix con la que se pueden programar la ejecución de cualquier comando. Con ella indicaremos el inicio de nuestro programa al arrancar la máquina, veamos el siguiente comando en el archivo de configuración *crontab -l*:

@reboot /usr/local/bin/forever start /home/usuario/daemon/index.js

Cuando arranque la máquina forever ejecutará nuestro programa.

3.4 Autenticación de usuarios

Para el acceso a la plataforma, puesto que únicamente los usuarios de la ULL serán los que la utilicen, se definió que lo ideal sería hacer uso del servicio de autenticación centralizada (CAS) que hay ya implantado.

Para utilizarla se tuvo que dar acceso a nuestra aplicación [30], por lo que cuando un usuario entra el servidor CAS nos proporciona una información determinada, en éste caso el *username*, *email* y *apellidos*.

La plataforma controla dos tipos de usuarios, **profesores** y **alumnos**. Dado que la información de un usuario recibida por el CAS no nos señala su rol (profesor o usuario), se tuvo que idear una manera para identificarlos, y es tan sencillo como basarse en el email. A continuación se muestra un sencillo código donde realizamos la identificación (*Figura 3.4.1*):

```
// alumno -> aluxxxxxxxxxx@...
// profesor -> nombreapellidos@...

var regexp = /\balu\d{10}\b/;

if(regexp.test(req.session.user) == true){
 req.session.rol = "alumno";
}
else{
 req.session.rol = "profesor";
}
```

Figura 3.4.1: Identificación de rol de usuario

Gracias a la expresión regular podemos identificar si es o no un alumno, ya que por defecto los email de los alumnos tienen la norma *aluxxxxxxxxxx@ull.edu.es*.

Cuando el usuario a superado la autenticación se crea una sesión activa con sus datos.

3.5 Orden y control de peticiones

Una tarea importante durante el desarrollo e implantación de la plataforma sin ninguna duda ha sido el tener que comprender y controlar tanto el ámbito en el que va a ser utilizada como la naturaleza propia de las herramientas empleadas, especialmente en los caso de NodeJS y MySQL, que serán la columna de éste apartado.

En primer lugar, cualquier persona que haya tenido que desarrollar en NodeJS se habrá dado cuenta de la importancia del asincronismo, y es que cuando se requiere de un orden y control de ejecución se deberá hacer uso de unas técnicas específicas.

En esta plataforma es de vital importancia tener un control de peticiones, es decir, aunque el servidor actue asíncronamente se deberá tener un orden interno. Por ejemplo, podemos ponernos en el supuesto caso de que dos usuarios quieran levantar un IDE, como bien sabemos hasta este punto se le deberá asignar una VM, y en el caso de que esa máquina solo pueda albergar a un usuario, se deberá poner en cola de espera.

A continuación se muestra una ilustración de las tablas existentes en la base de datos (*Figura 3.5.1*). Muchas de ellas ayudan a tener un control de peticiones, independientemente del estado de los servidores que puedan estar en funcionamiento:

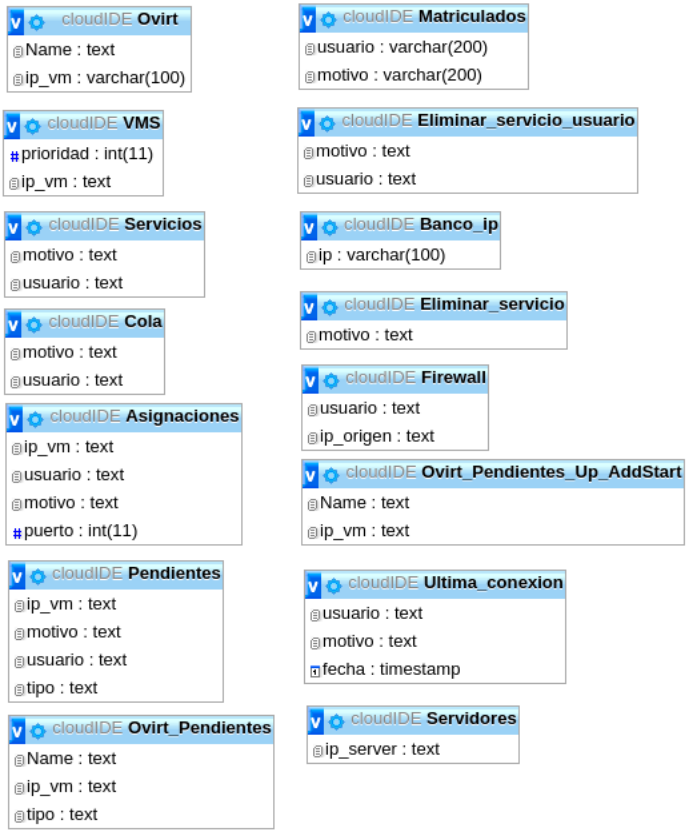

Figura 3.5.1: Tablas base de datos

Por decirlo de alguna forma, con NodeJS podemos controlar el asincronismo del propio proceso, pero, ¿cómo controlamos el asincronismo de peticiones a la base de datos?, como sabemos pueden haber numerosas peticiones de manera concurrente, y en nuestro caso puede ocasionar irregularidades en los datos (ej:Dos procesos cogen al mismo usuario en cola). Por lo que también necesitamos en ocasiones que el acceso a la base de datos sea único por un proceso, es decir, que hasta que el proceso no haya terminado, ninguna otra petición podrá alterar el resultado, formando así una cola de peticiones MySQL.

Para solucionar este problema optamos por el **bloqueo de tablas**, el proceso que vaya a entrar en la base de datos bloqueará las tablas que se consideren oportunas, y hasta que éste no acabe y las libere ningún otro proceso podrá entrar y modificarlas, esperando en una cola de peticiones. Veamos a continuación un código de ejemplo para este caso (*Figura 3.5.2*):

```
var bloqueo_tablas = "LOCK TABLES VMS WRITE, VMS as v1 READ, Ovirt_Pendientes_Up_AddStart WRITE, Ovirt_Pendientes_Up_AddStart as
ovpuas READ, Ultima_conexion WRITE, Ultima_conexion as uc READ, Eliminar_servicio_usuario WRITE, Eliminar_servicio_usuario as esu
READ, Eliminar_servicio WRITE, Eliminar_servicio as es READ, Servicios WRITE, Servicios as s1 READ, Matriculados WRITE, Matriculados
as m1 READ, Ovirt WRITE, Ovirt as ov READ, Ovirt_Pendientes WRITE, Ovirt_Pendientes as ovp READ, Banco_ip WRITE, Banco_ip as bip
READ, Firewall WRITE, Firewall as f1 READ, Pendientes WRITE, Pendientes as p1 READ, Asignaciones WRITE, Asignaciones as a1 READ, Cola
WRITE, Cola as c1 READ";


pool.getConnection(function(err, connection) {
      var conexion = connection;
      conexion.query(bloqueo_tablas,function(error, results, fields) {
        conexion.query("DELETE FROM Servidores WHERE ip_server='"+ip+"'",function(error, result, fields) {
          console.log('server disconnected');
          conexion.query("UNLOCK TABLES",function(error, results, fields) {
            console.log("liberando tablas");
            conexion.release();
          });
        });
      });
    });
```

Figura 3.5.2: Bloqueo de tablas en MySQL

Éste tipo de técnicas controlará el acceso a los datos asegurándose que solamente un proceso puede modificarlos. Es una forma de emular a las transacciones en una base de datos.

3.6 Redirección de tráfico

Como bien hemos nombrado en numerosas ocasiones, éste proyecto hace uso de varias subredes privadas para conseguir diversos objetivos, ya sea por necesidad de la infraestructura de máquinas virtuales o para obtener características fundamentales como es la privacidad en cuanto a IDEs de usuarios se refiere.

La plataforma está enfocada al uso de multitud de usuarios de forma concurrente, conectados desde las redes de servicio de la ULL con una dirección ip única. Todo el tráfico (puertos indicados) de ese usuario desde el dispositivo donde inició sesión será redirigido a la máquina virtual interna que se le haya asignado, más específicamente hacia los servidores de Eclipse Che que tenga en funcionamiento. Éste proceso se deberá hacer dinámicamente haciendo uso del potente firewall integrado en el kernel de linux, **iptables**[22].

En primer lugar, antes de adentrarnos en la redirección de tráfico de usuarios debemos comentar el proceso para proporcionar conexión a internet a las máquinas de la red aislada *DOCINT3*. Todo lo que se comenta en éste apartado es en base a la máquina portal, siendo en ella donde se tendrá que incluir la siguiente configuración:

Primero activamos *ip forward*, con ello indicamos que todos los paquetes que entren en la máquina y cuyo destino no sea el suyo, los reenvíe y no los descarte:

$ echo 1 > /proc/sys/net/ipv4/ip_forward

De manera adicional también tenemos que enmascarar dichos paquetes, ya que no pueden salir al exterior de la red con su ip original, debemos hacer el proceso de NAT:

$ sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

Con la regla anterior estamos indicando que todos los paquetes que salgan por la interfaz *eth0* deberán ser enmascarados con dicha ip (ip de la interfaz eth0).

Por otro lado, en cuanto a la redirección del tráfico perteneciente a Eclipse Che, debemos tener varios aspectos en cuenta. Para un mejor entendimiento se ha elaborado una ilustración (*Figura 3.6.1*) de un ejemplo de la plataforma en un momento dado:

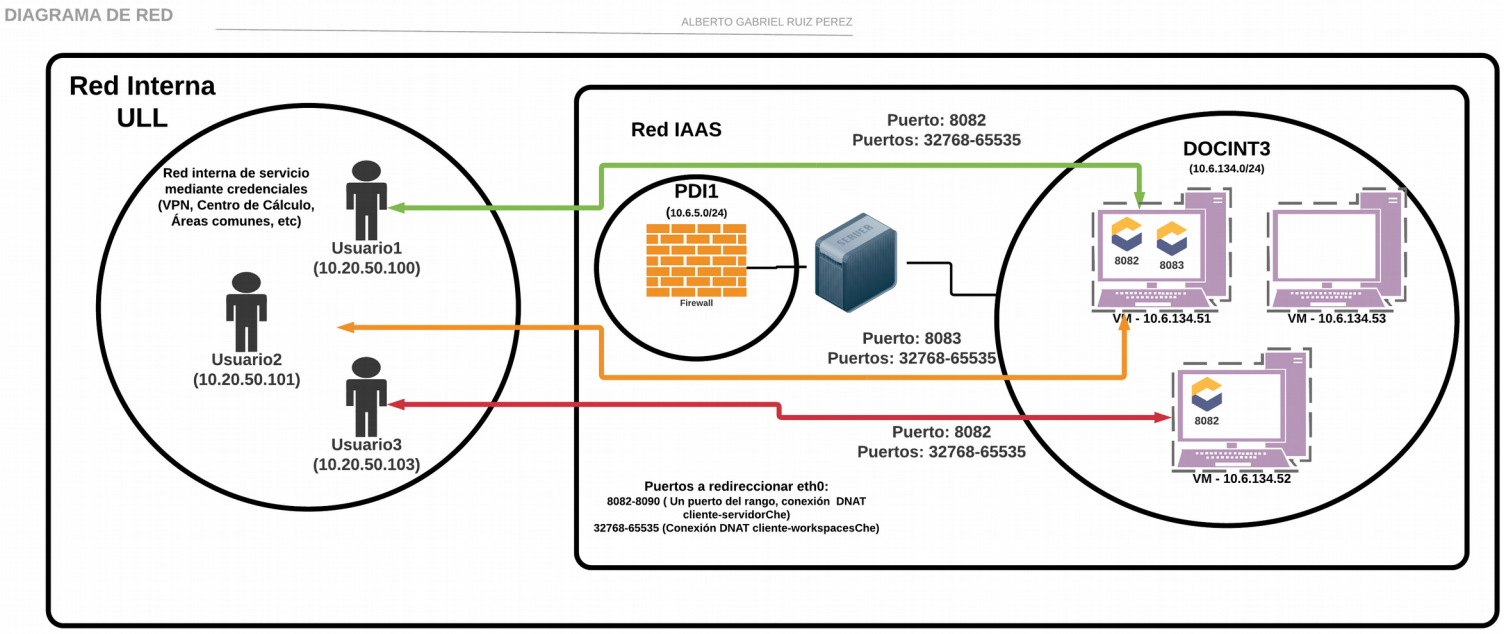

Figura 3.6.1: Ejemplo sobre redirección DNAT

Como se observa, en la red de servicio existen 3 usuarios activos, éstos tienen asignada una dirección ip única, es de vital importancia ya que para hacer una gobernación de tráfico efectiva debemos tener dos atributos fundamentales, una ip de origen y otra de destino.

Por otro lado en la red interna (*DOCINT3*) hay 3 VMs en funcionamiento. En dos de ellas hay varios servidores Che ejecutándose (de un usuario y en un puerto específico), por lo que de manera automática la máquina portal deberá redirigir todo el tráfico, desde la máquina del usuario hasta la máquina virtual que le corresponde. El usuario final no tiene por qué saber a qué VM está conectado, creyendo que solamente se está comunicando con la máquina portal que hace de intermediario. Esta característica que hace invisible toda la orquestación interna es gracias a reglas iptables DNAT. Un esquema de regla correcto sería el siguiente:

$ sudo iptables -w -t nat -p tcp -A PREROUTING -s <ip-origen> --match multiport --dports 32768:65535 -j DNAT --to-destination <ip-destino>

$ sudo iptables -w -t nat -p tcp -A PREROUTING -s <ip-origen> --dport <puerto-Che> -j DNAT --to-destination <ip-destino>

Aplicándola al esquema anterior para el caso del *usuario1* podríamos ponerla tal que así:

$ sudo iptables -w -t nat -p tcp -A PREROUTING -s 10.20.50.100 --match multiport --dports 32768:65535 -j DNAT --to-destination 10.6.134.51

$ sudo iptables -w -t nat -p tcp -A PREROUTING -s 10.20.50.100 --dport 8082 -j DNAT --to-destination 10.6.134.51

De esta manera estamos redireccionando todo el tráfico (puerto 8082 de Che y puertos efímeros) que nos viene de la dirección 10.20.50.100 hacia la máquina

interna 10.6.134.51. El usuario creerá que se está comunicando con la máquina portal, pero ésta administra el tráfico según cree conveniente. En éste proyecto se deberá tener en cuenta múltiples aspectos como son los siguientes:

- Desde una misma ip solo se podrá tener una sesión activa de un único usuario, aunque por otro lado un usuario puede estar activo desde varios dispositivos (direcciones). Esto es debido a la necesidad de identificar a un usuario con una ip única para dirigir el tráfico.
- Se deberá aplicar reglas dinámicamente, ya que el usuario puede ejecutar nuevos servidores cuando se desee, por otro lado se deberá hacer limpieza de reglas cuando un usuario cierre sesión.
- También se debe tener en cuenta posibles errores de desconexión, ya que en éste caso (desconexión y conexión) el servidor DHCP institucional asignará al usuario una nueva ip, habrá que identificarlo para hacer una limpieza y redireccionar el tráfico correctamente.
- Cuando se modifican unas reglas iptables hay que estar pendiente de las **conexiones TCP establecidas**. Nombro este caso ya que por la naturaleza de tcp, si previamente hay conexiones establecidas entre los dos puntos y dirigimos posteriormente el tráfico, éstas conexiones siguen su camino inicial, ignorando de alguna forma las reglas que se han impuesto y pudiendo interferir en el correcto funcionamiento de los servidores con el frontend de las máquinas cliente.

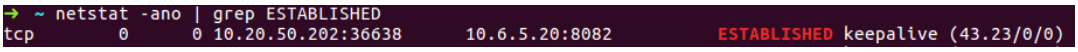

Figura 3.6.2: Conexiones TCP establecidas

Para solucionar este aspecto tenemos varias posibilidades, por un lado está la inclusión de reglas iptables que eliminen los paquetes de conexiones establecidas produciendo la deseada desconexión, pero negativamente este tipo de reglas impediría el correcto funcionamiento de la plataforma, ya que además es bastante dependiente del tiempo produciendo retardo en el proceso, por lo que se necesitan herramientas más eficaces. Una de las herramientas que podía conseguir éste objetivo es la llamada **tcpkill**[31], una aplicación en línea de comandos incluida en el paquete *dsniff* cuyo objetivo principal es la terminación de conexiones tcp. Su ejecución es bastante sencilla, por ejemplo (*Figura 3.6.3*):

```
sudo tcpkill host $1 and portrange 8082-8089 and portrange 32768-65535
sleep 20
sudo kill -9 $!
```

Figura 3.6.3: Script tcpkill

Dicho script ejecuta la herramienta, terminando las conexiones con un origen *host* y el rango de puertos como destino especificado. Su funcionamiento consiste en la escucha de dicho tráfico hasta poder enviar paquetes con el flag RST activado para terminar las conexiones. Se ejecutará durante 20 segundos y luego se finalizará el proceso que lo ejecuta. Éste método se usará conjuntamente con iptables durante el cierre de sesión de un usuario, a su vez se han incluido una serie de

retardos para asegurarnos el cierre de las conexiones establecidas antes de que otro nuevo usuario intente utilizarlas.

3.7 SDK oVirt

Tras haber comentado en los anteriores subapartados las distintas tareas que se han tenido que ejecutar, para acabar el capítulo de desarrollo comentaremos una de las herramientas vitales en el proyecto, y que sin ninguna duda es la que nos brinda poder desplegar dicha infraestructura en las instalaciones de la ULL.

Como bien sabemos el STIC de la universidad ofrece una plataforma de virtualización IaaS, actualmente se emplea el software libre oVirt. Se trata de una plataforma de virtualización sobre KVM basada en permisos, que permite asignar a cada usuario privilegios concretos haciéndolo extremadamente versátil en cuanto a necesidades.

Lo que hace beneficioso esta herramienta a nuestra plataforma es el SDK que proporciona, de esta manera podemos hacer uso de numerosas funcionalidades a través de una gran variedad de lenguajes de programación. En nuestro caso usaremos el SDK versión 4 de python[32], ésto nos posibilitará crear nuevas máquinas virtuales basándonos en la plantilla que habíamos diseñado, eliminarlas, modificarlas, etc.

El uso que le daremos a esta API se hará desde la máquina orquestador, la que montará la principal infraestructura de la plataforma.

En primer lugar lo que haremos es una instalación de los paquetes *ovirt-engine-sdk-python*, una vez incluidos se podrán elaborar los diferentes scripts que harán su uso. Básicamente el empleo que le daremos será el de crear nuevas máquinas virtuales en base a la plantilla *ULL-CloudIDE-backend-tpl*, a la que le podremos asignar una determinada dirección ip por medio de *cloud-init*, como también apagarlas y eliminarlas cuando creamos conveniente.

Por ejemplo a continuación mostramos una serie de ilustraciones de un script (*Figuras 3.7.1 y 3.7.2*) que crea una nueva vm en base a la plantilla y la enciende:

```
import logging
import ovirtsdk4 as sdk
import ovirtsdk4.types as types

def createconnection():

        connection = sdk.Connection(
         url='https://iaas.ull.es/ovirt-engine/api',
         username='alu0100825510@ULL',
         password='*********',
         ca_file='./ovirtpython/cert',
         debug=True,
         insecure=True,
         timeout=30,
         headers={'filter':True}
         )

        print "Connection created"

        return connection
```

Figura 3.7.1: oVirt SDK, inicializar conexión

Dicha función será la que cree una conexión a la API de oVirt, se hará a través de nuestro usuario y contraseña, haciendo uso además del archivo del

certificado SSL.

```
vm = vms_service.add(
    types.Vm(
        name=sys.argv[1],
        cluster=types.Cluster(
            name='Cluster-Rojo'
        ),
        template=types.Template(
            id=template_id
        ),
    )
)

vm_service = vms_service.vm_service(vm.id)

while True:
    time.sleep(5)
    vm = vm_service.get()
    if vm.status == types.VmStatus.DOWN:
        break

print "VM created"

print "Starting VM"
vm_service.start(
    use_cloud_init=True,
    vm=types.Vm(
        initialization=types.Initialization(
            nic_configurations=[
                types.NicConfiguration(
                    name='eth0',
                    on_boot=True,
                    boot_protocol=types.BootProtocol.STATIC,
                    ip=types.Ip(
                        version=types.IpVersion.V4,
                        address=sys.argv[2],
                        netmask='255.255.255.0'
                    )
                )
            ]
        )
    )
)

print "VM started"

# Close the connection to the server:
connection.close()
```

Figura 3.7.2: oVirt SDK, creación y arranque de una vm

El script de la imagen anterior tiene la finalidad de crear una nueva máquina virtual en el primer paso, y por medio de su id (todas las máquinas tienen un id único) arrancarla con *cloud-init* con unos determinados parámetros, en éste caso le indicamos que dirección ip tendrá la interfaz *eth0*. Más scripts como el anterior se pueden encontrar en el código fuente de la plataforma.

Para dotar al servicio de una mayor configuración se decidió darle un enfoque personalizable en cuanto a las máquinas virtuales se refiere. Como se comentó en otras ocasiones se puede configurar cuantos usuarios debe alojar cada máquina virtual, pero en éste apartado hablaremos del indicador de **máquinas de reserva**.

Éste valor y característica se decidió para darle una mayor velocidad a la plataforma, y es que podemos indicar el número de VMs que queremos que haya siempre encendidas de reserva esperando a que un usuario quiera utilizarlas, de tal forma que no haya que esperar el tiempo de creación y puesta en marcha de una nueva. Siempre se ha de tener máquinas sobre ese rango, por lo que constantemente se vigilará cuantas hay disponibles, y tomar decisiones en cuanto al estado, encender más máquinas si está por debajo del valor indicado o apagar y eliminar si está por encima.

Como es lógico también se puede indicar que no haya máquinas de reserva, sino que se vayan arrancando según la demanda, aunque esto hace una plataforma más lenta en cuanto a tiempo de despliegue de servidores Che (tiempo de creación y arranque vm + tiempo de arranque Che).

Capítulo 4
ULL-CloudIDE

Durante el presente capítulo se describirán las principales funcionalidades de la plataforma desarrollada, dentro de lo posible también trataremos de relacionar y visualizar dichas capacidades con los procesos internos que ocurren.

4.1 Inicio de sesión

En primer lugar, en el caso de que no tengamos una sesión activa (no hayamos iniciado sesión previamente en la plataforma), cuando se acceda a la dirección del servicio [30] lo que se encontrará será una página de inicio.

En ella y por medio de un simple botón podremos dirigirnos hacia la autenticación, recordemos en éste punto que será a través del servicio de autenticación centralizado ya implantado en la universidad, pudiendo acceder así con las credenciales institucionales asignadas.

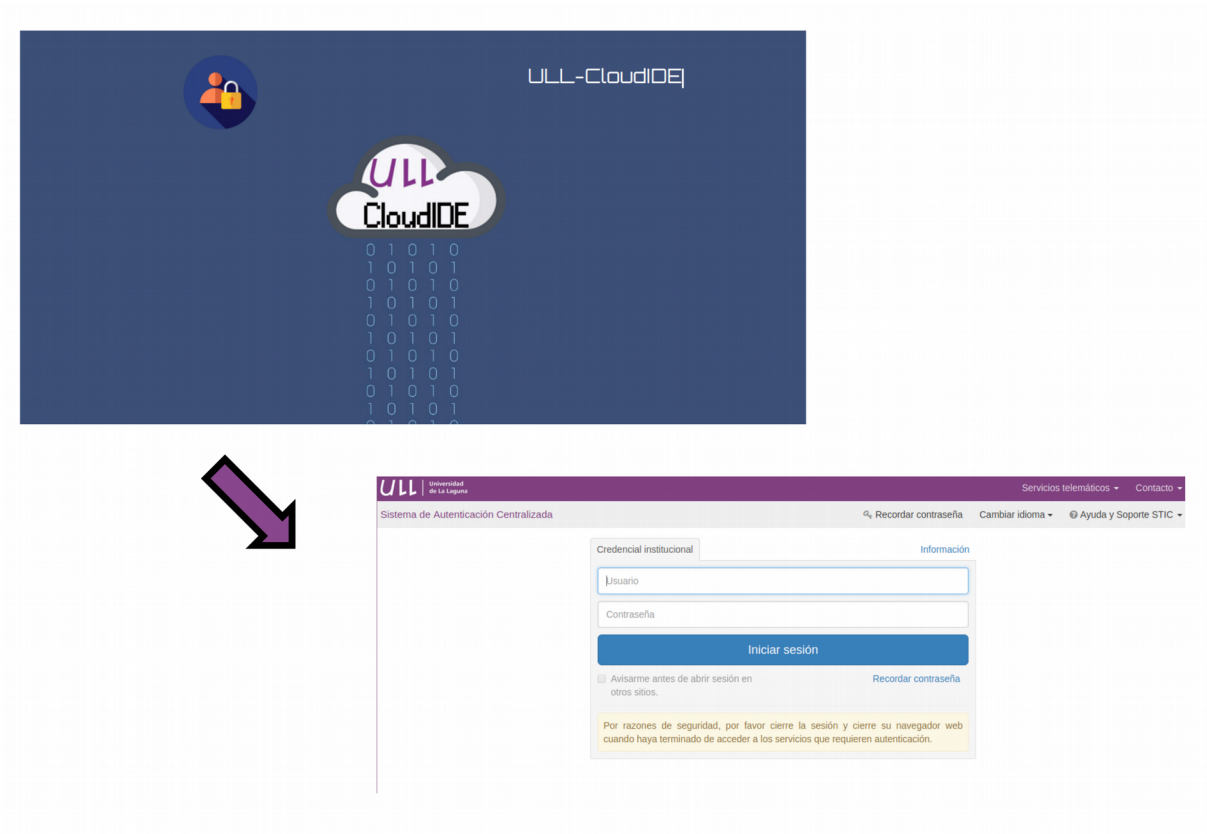

Figura 4.1: ULL-CloudIDE, inicio de sesión

Cuando se haya redireccionado hacia el CAS, pueden ocurrir dos estados:

- **Ya se ha iniciado sesión previamente en CAS**: Ésto significa que previamente tenemos una sesión activa de autenticación CAS debido a que se ha accedido a otros servicios de la ULL. Por lo que no se nos volverá a pedir el ingreso de credenciales, sino que se introducirá directamente en la plataforma.
- **No se ha iniciado sesión previa en CAS**: Por otro lado, si es la primera vez que se accede al CAS se deberá hacer uso de las credenciales. En caso satisfactorio ya habrá una sesión CAS activa, por lo que para otros servicios ULL no será necesaria una nueva autenticación.

Cuando el inicio de sesión se ha efectuado, inmediatamente se redirecciona todo el tráfico perteneciente a los IDEs de ese usuario desde la dirección ip desde la que está conectado hasta la máquina virtual que lo aloja. Al final de éste proceso se le dirige hacia su panel de control donde podrá hacer uso de diversas funcionalidades.

4.2 Alumno(a)

Como bien hemos nombrado existen dos tipos de roles en la plataforma, alumnos y profesores. Aún así ambos tienen funcionalidades comunes, es el caso del muestreo de los servicios asignados y su utilización.

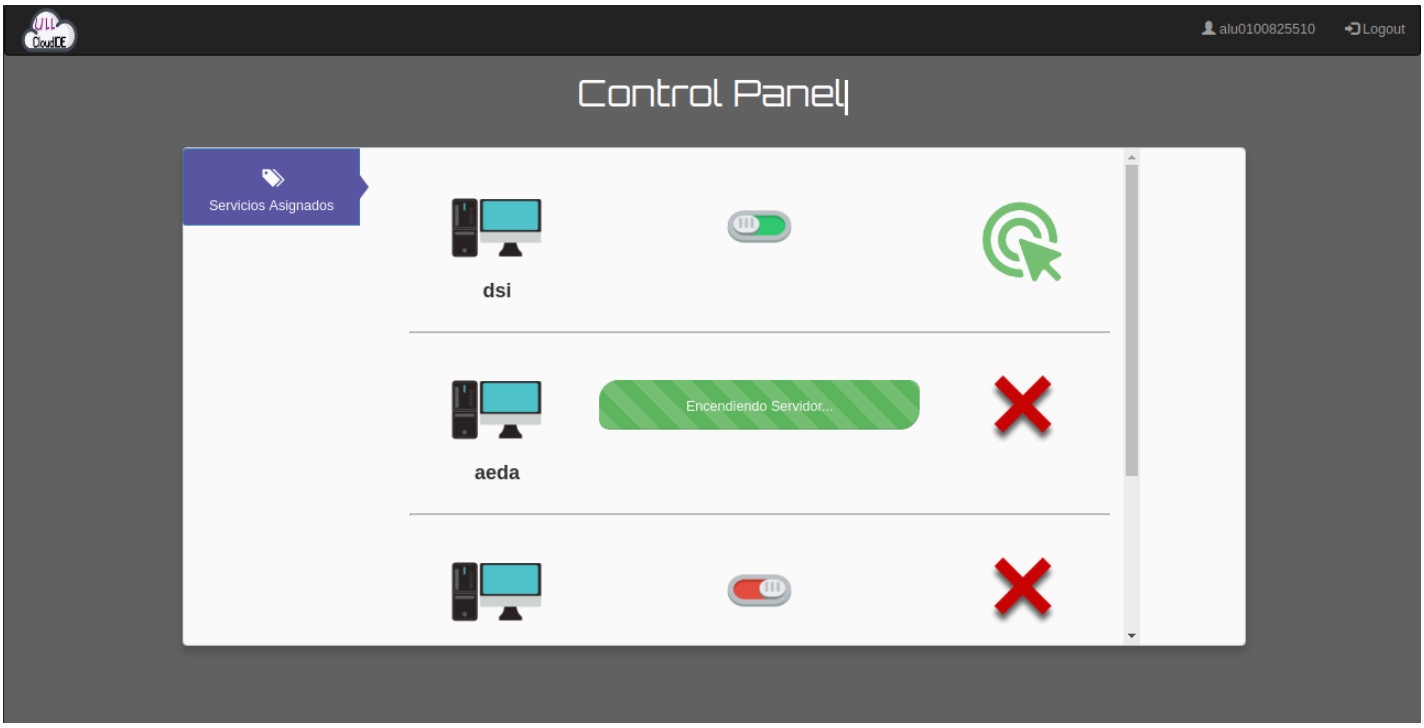

Figura 4.2.1: ULL-CloudIDE, servicios asignados

Un profesor puede asignar servicios a otros usuarios, pero a él también pueden asignarle entornos de desarrollo.

El control de los servicios se lleva enteramente a través de la conexión websocket establecida con el servidor, de tal manera que el usuario pueda indicar el encendido y el apagado de manera dinámica, y cuando esté

disponible recibir una notificación en tiempo real para mostrarlo adecuadamente en el frontend, siendo así una gran ventaja el no tener que cambiar constantemente de páginas.

Una vez que el IDE está disponible, el frontend dispondrá el enlace para ser accedido y desarrollar en él, el cual siempre sigue la regla *http://cloudide.iaas.ull.es/cloud/<servicio>.*

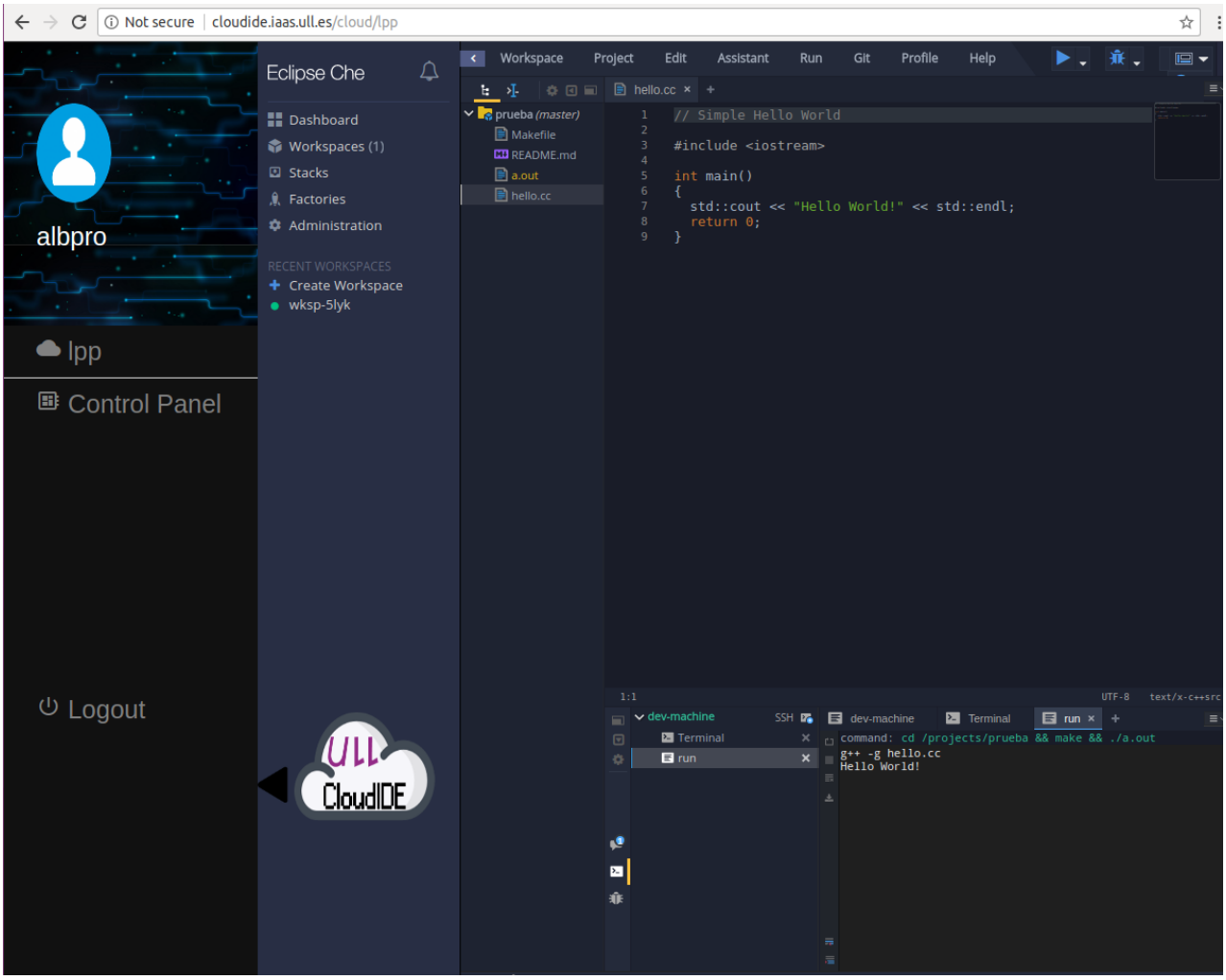

Figura 4.2.2: ULL-CloudIDE, ejemplo de servicio IDE

Dicha página incrusta (utilizando *iframe*)[33] el IDE dentro de una interfaz de usuario desplegable, desde el que se indica el servicio actual, acceso al panel de control o cerrar sesión si se desea.

Por otro lado, como se ha mencionado, existe un almacenamiento centralizado para guardar todos los datos relativos a los proyectos y configuraciones que hace cada usuario en los workspaces de los IDEs asignados. Éstos están ubicados en la máquina central en el directorio */mnt/cloudIDE*.

```
usuario@cloudide:~$ ls /mnt/cloudIDE/
albham-Empotrados             albpro-ssi1819       alu0100825510-lpp
albham-InformáticaIndustrial  alu0100825510-aeda   alu0100825510-ssi1819
albpro-dsi                    alu0100825510-dsi
```

Figura 4.2.3: ULL-CloudIDE, estructura de datos centralizados

Para seguir una regla hemos decidiido que cada directorio de cada IDE tiene

que ser *<usuario>-<servicio>*. De esta manera se hace más fácil la identificación a la hora de realizar diversas acciones, como por ejemplo el borrado de datos cuando se quiera.

Por último, nombrar que ya una vez dentro del ámbito de un workspace, el funcionamiento y limitaciones del mismo se remiten a la documentación propia de Eclipse Che[26].

4.3 Profesor(a)

Se ha creído conveniente la creación de un apartado enfocado a las funcionalidades que posee el rol de profesor en la plataforma, ya que en gran medida es el que **genera y controla** los accesos a los servicios que crea. A continuación explicaremos brevemente dichas características con ilustraciones de ejemplos de uso.

4.3.1 Añadir nuevo servicio

El funcionamiento de la plataforma funciona de manera similar al campus virtual con las asignaturas, un profesor puede crear un nuevo servicio (símil de asignatura), y éste será asignado a una serie de usuarios (alumnos y/o profesores).

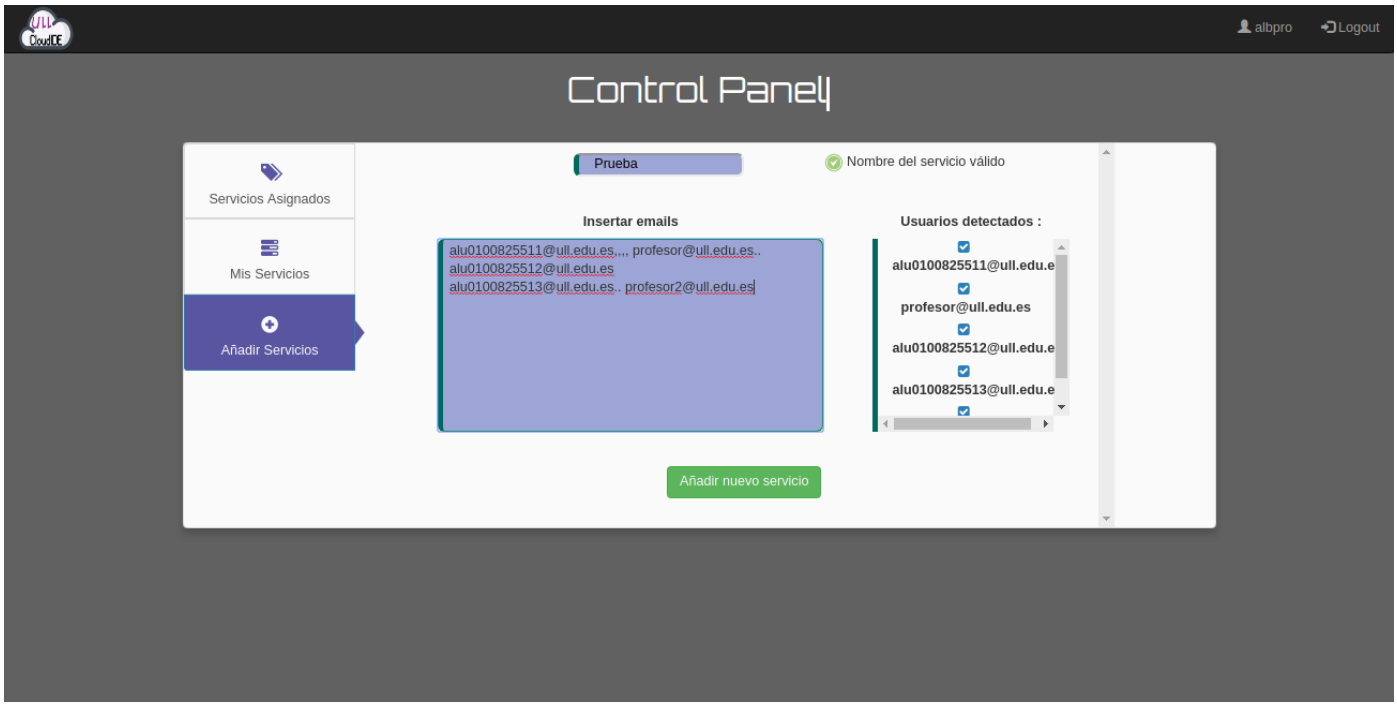

Figura 4.3.1: ULL-CloudIDE, añadir nuevo servicio

En la creación de un servicio se debe elegir un nombre único, es decir, un servicio que no esté en uso. Para su comprobación, cada vez que se escriba dentro del cuadro de texto se asegurará dinámica y asíncronamente de la existencia de ese servicio, indicándolo en ambos casos y habilitando el botón de creación cuando sea verificado.

A la hora de introducir usuarios, una de las características fundamentales era hacer sencilla esta tarea a los profesores. Como lo más fácil (para los

profesores) es copiar la lista de correos institucionales de los alumnos, se decidió incorporar una caja de texto donde incluir dicha lista. Una vez pegada, la plataforma se encargará de identificar a todos los usuarios, para ello se ha hecho uso de una expresión regular a medida para limpiar todo tipo de caracteres indeseables y quedarnos con el objetivo, el nombre de que les identificará. Así mismo se podrá seleccionar y deseleccionar aquellos que se crea conveniente. Una vez se añada el servicio, éste estará disponible para todos los usuarios adjuntados.

4.3.2 Mis servicios

Después de haber añadido algún servicio, éste aparecerá dentro de *Mis Servicios*, es aquí donde el dueño podrá hacer uso de diversas funciones. Visualmente se mostrarán como menús desplegables tal y como se muestra en la siguiente ilustración (*Figura 4.3.2*).

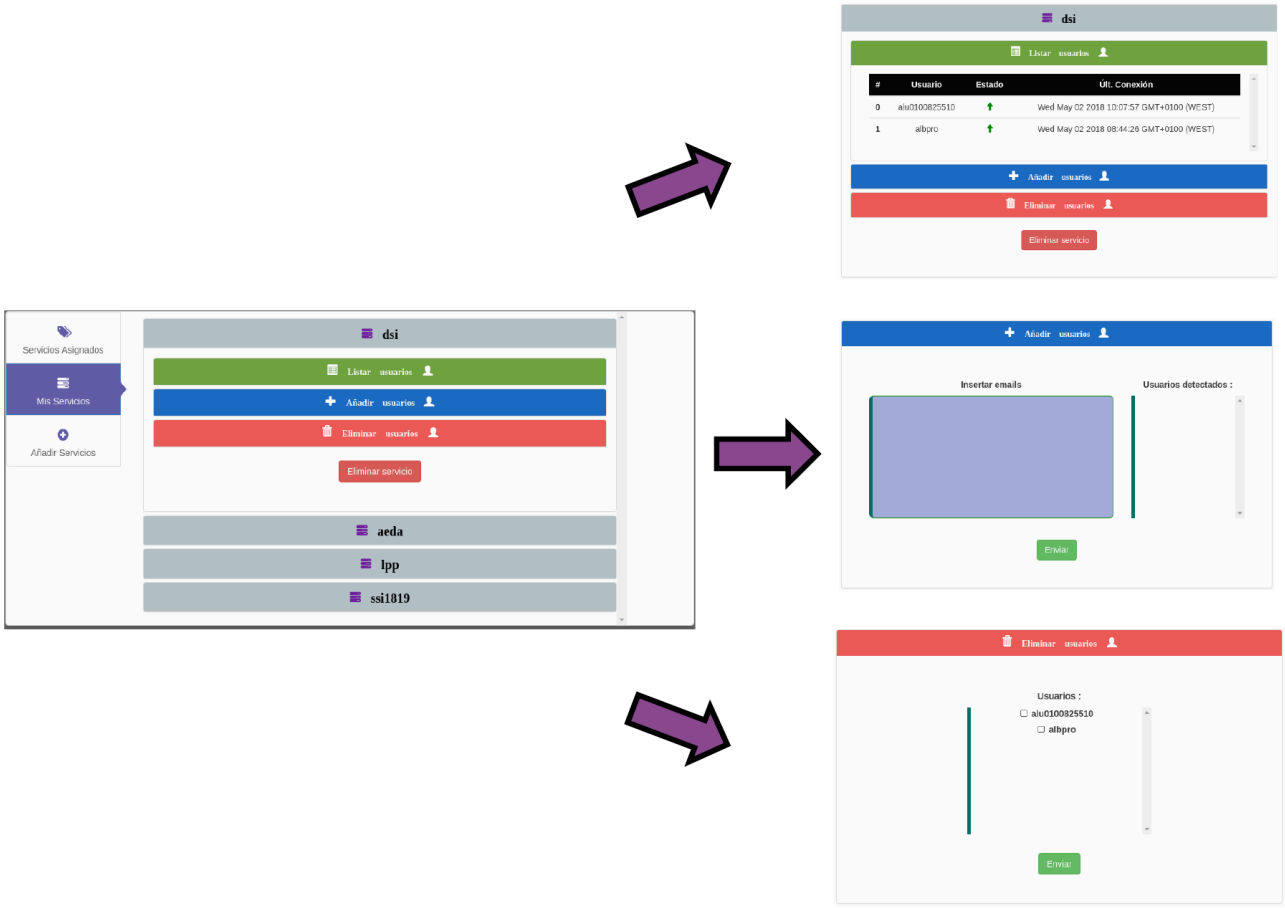

Figura 4.3.2: ULL-CloudIDE, mis servicios

En cuanto a las funcionalidades que se pueden hacer uso dentro de un servicio son las siguientes:

- **Listar usuarios**: En el primer submenú se podrán listar los usuarios adscritos al servicio en cuestión. A su vez se mostrarán datos relativos a cada uno de ellos, como puede ser el estado de su servidor (apagado o encendido) o la fecha del último acceso a su IDE.
- **Añadir usuarios**: Se podrán añadir nuevos usuarios, útil en el caso de

que no hayan sido añadidos cuando se creó el servicio.

- **Eliminar usuarios**: Se puede eliminar usuarios del servicio. Ésto conllevará multitud de procesos internos, ya que antes se deberá impedir el acceso a dichos usuarios, se apagarán los servidores en el caso de que estén en funcionamiento, se modificarán reglas firewall y por último se borrará el directorio de datos de ese IDE.
- **Eliminar servicio**: De forma análoga a la anterior acción, también se puede eliminar el servicio por completo, ésto llevará a cabo los procesos anteriormente nombrados pero con todos los usuarios adjuntos. Por último el servicio se libera y podrá volver a utilizarse su nombre.

Las funciones de eliminar requieren de una doble confirmación, ya que por seguridad el propietario del servicio debe estar seguro de las acciones que va a ejecutar.

4.4 Cierre de sesión

Cuando un usuario cierra sesión, aparte de no permitir el acceso a sus datos visuales también se deberán modificar las reglas firewall que se le hayan asignado previamente (en el caso de que tenga IDEs activos). Éste proceso básicamente consiste en **destruir** las reglas DNAT que le dirigen hacia sus servidores desde la ip en la que se logueó y cerró sesión. De forma paralela también tendrá lugar la terminación de conexiones tcp establecidas mediante la herramienta *tcpkill* nombrada en capítulos anteriores.

4.5 Mantenimiento

Como se ha podido observar, a lo largo de este documento se han nombrado multitud de herramientas y servicios de los cuales la plataforma final hace uso.

Estas circunstancias nos puede ofrecer infinidad de beneficios, pero también nos hace aumentar tanto la posibilidad de encontrar errores como la inestabilidad de la propia herramienta. Esto es debido a que si uno de esos servicios de los que hacemos uso no funciona como debería en un momento dado, puede llegar a hacer que la plataforma no funcione de la manera esperada.

Por poner ejemplos, podemos nombrar el caso en el que nos quedemos sin almacenamiento (tanto el centralizado como en las máquinas virtuales), surja un problema de red que haga que los nodos no puedan comunicarse, que el servidor de autenticación no funcione correctamente o que el servicio de máquinas virtuales experimente algún problema (lógico ya que también ellos dependen de muchos factores variables).

Durante el desarrollo de este proyecto constantemente se ha intentado mejorar la corrección, detección y el tratamiento de errores, pero como es lógico es imposible ponernos en todos los subcasos posibles que puedan aparecer. Por ello se ha decidido desarrollar una serie de scripts que podrán ser usados en

caso de alguna desconfiguración o mal funcionamiento de la plataforma.

1) Primero nos aseguraremos de parar el proceso que ejecuta la plataforma en la máquina portal, para ello podemos hacerlo con el siguiente comando:

   $ sudo killall node

2) Una vez que la plataforma esté detenida, procedemos a ejecutar el script desarrollado, ubicado en el directorio raiz del código fuente:

   $ sudo node restaurarservicio.js

   Dicho script tiene como finalidad hacer una limpieza de las tablas de la base de datos que puedan contener datos inestables y/o corruptos. A su vez, gracias a la api de oVirt se hace una limpieza de las máquinas virtuales, es decir, apaga y elimina todas las que han estado en funcionamiento. Ésto quiere decir que los servidores Che también se habrán detenido, pero como bien sabemos los datos alojados en */projects* de cada workspace siempre estarán a salvo en el directorio centralizado.

   Este script se ha de ejecutar constantemente hasta que finalice sin ningún error aparente, eso querrá decir que el servicio de las VMs funciona correctamente y puede volver a iniciarse la plataforma.

3) Por último, podremos volver a poner en funcionamiento el servidor, para dejarlo en segundo plano lo hacemos tal que así:

   $ nohup sudo node index.js > out.log &

Aún así de cara al futuro uno de los principales objetivos será incluir técnicas para mejorar el control y tratamiento de errores, así obtendríamos una plataforma mucho más estable a los cambios que puedan aparecer.

4.6 Código fuente

En el siguiente enlace [34] se puede visitar el repositorio GitHub donde están alojados los ficheros desarrollados para éste proyecto. A su vez también puede encontrar información sobre su uso y otras funcionalidades.

Capítulo 5
Conclusiones y líneas futuras

Durante este capítulo se nombrarán tanto las conclusiones que se han podido reflejar en el proyecto como las características o el camino a seguir para mejorar dicha herramienta.

5.1 Conclusiones

- Se ha realizado un estudio previo sobre los requerimientos principales y las características de Eclipse Che, que ha dado lugar al diseño de un esquema de red determinado, haciendo uso de las diferentes subredes existentes en el IaaS.
- Se ha implementado un servidor web principal, que ofrece tanto a profesores como alumnos un sistema enfocado a la docencia que permite administrar y acceder a IDEs.
- Se han asignado dos interfaces de red al servidor central, haciendo así de intermediario entre los usuarios y las máquinas virtuales que ejecutan IDEs de Eclipse Che y que están alojadas en una subred aislada.
- Se ha utilizado una base de datos relacional MySQL, en la que se almacena toda la información perteneciente a la plataforma y que nos ayuda a realizar un control ordenado de las peticiones recibidas por el portal.
- Se ha diseñado un sistema de redireccionamiento mediante el firewall iptables, de ésta forma se consigue que cada usuario pueda tener sus entornos privados y accesibles desde cualquier dispositivo.
- Se ha configurado una plantilla de máquina virtual, que ejecuta un servicio que atiende las órdendes del servidor principal, cuya función es la de encender o apagar IDEs dinámicamente.
- Se ha desarrollado un procedimiento de gestión de máquinas virtuales, que haciendo uso de scripts con la api de oVirt en python, se consigue escalar la plataforma en base a la carga de usuarios en cada momento.
- Se ha incorporado el servicio de autenticación centralizado de la ULL, de tal manera que se pueda acceder a la plataforma con las credenciales institucionales.

5.2 Líneas futuras

A pesar de que se han completado la mayoría de los hitos propuestos y que se ha podido desarrollar e implementar una plataforma funcional y estable, existen varios aspectos a tener en cuenta de cara al futuro para poder ofrecer un mejor servicio. A continuación se enumeran varias que se consideran fundamentales:

- En primer lugar estaría la mejora de detección y tratamiento de errores. Como se ha comentado en varias ocasiones, esta plataforma hace uso de muchos servicios externos, y como es lógico pueden fallar en cualquier momento. Esos errores generados externamente pueden afectar directamente a la herramienta, por lo que habría que estar constantemente al tanto de su correcto funcionamiento. Por eso también se debería mejorar lo desarrollado enfocándolo a todos los errores que puedan ocurrir, y en caso de que así sea escoger el camino más adecuado para que se solucione sin necesitar soporte activamente.
- Proporcionar más personalización a los profesores, donde ellos puedan escoger características que tengan los servidores Che de sus alumnos. Ésto viene limitado por Eclipse Che, ya que ellos son los encargados de añadir funcionalidades al IDE para hacerlo editable a nuestro gusto.
- Mejorar el entendimiento y la estructura del código desarrollado. Este proyecto se ha desarrollado utilizando varios lenguajes de programación, y existen muchos procedimientos que los mezclan, haciéndo bastante difícil su comprensión para quien lo ve por primera vez. Nombro este aspecto de cara a futuras mejoras de la plataforma, hacer una base sólida y que de ella se ofrezcan todas las características que mejoran el servicio.
- Una funcionalidad que haría aún más potente la plataforma sería la posibilidad de que los profesores pudieran visitar los IDEs de los alumnos, de ésta manera podrían visualizar los proyectos desarrollados y/o desplegados con mayor facilidad. Ésto conllevaría tener un alto control de las conexiones establecidas y de la redirección de tráfico.
- Una característica adicional sería incluir avisos por correo electrónico. Así se podría avisar a los usuarios de cualquier cambio que haya en su plataforma (encendido y apagado de servidores, eliminación y/o asignación de servicios, etc).
- Como es lógico, también se incluyen como líneas futuras aquellos avances que vayan aportando en las nuevas versiones de Eclipse Che, en éste caso nos remitiremos a la documentación oficial[26].

Capítulo 6
Summary and Conclusions

During this chapter, the conclusions of the project will be developed and the characteristics will be listed to improve this tool.

6.1 Conclusions

- A preliminary study has been carried out on the main requirements and characteristics of Eclipse Che, which has led to the design of a determined network scheme, making use of the different subnets in the IaaS.
- A main web server was implemented, which offers both teachers and students a system focused on teaching that allows them to manage and access IDEs.
- Two network interfaces have been assigned to the central server, thus acting as an intermediary between the users and the virtual machines running the Eclipse Che IDE hosted on an isolated subnet.
- A MySQL relational database has been used, in which all the information pertaining to the platform is stored and which helps us to carry out an orderly control of the requests received by the portal.
- A redirection system has been designed through the iptables firewall, in this way it is possible for each user to have their private environments accessible from any device.
- A virtual machine template has been configured, which executes a service that handles the orders of the main server, whose function is to activate or deactivate the IDEs dynamically.
- A virtual machine management procedure has been developed, which through the use of scripts with the Ovirt API in python, can scale the platform according to the load of users at each moment.
- The centralized authentication service of the ULL has been incorporated, in such a way that the platform can be accessed with institutional credentials.

6.2 Future work

Although most of the milestones have been completed and a functional and stable platform has been developed and implemented, there are several aspects to consider in the future in order to offer a better service. Exist several things that are considered fundamental:

- First, be in improving the detection and treatment of errors. As has been mentioned several times, this platform has many external services, and of course it can fail at any time. These errors generated externally can directly affect the tool, so you should be aware of its correct operation. That's why you could improve the development by focusing on the errors that may arise, and chose the right solution without needing support actively.
- Provide more personalization to teachers, when there are. This is limited by Eclipse Che, since they are responsible for adding functionalities to the IDE to make it editable to our liking.
- Improve the knowledge and structure of the developed code. This project has been developed using several programming languages, and there are many procedures that explain them, making it difficult for the first time to see it. I mention this aspect in the face of the improvements of the platform, to make a solid base and to repaired all the features that improve the service.
- A functionality that could be more powerful is the possibility for teachers to visit student IDEs, in an easy way. This would entail having a high control of the established connections and the redirection of traffic.
- A second additional reason is email notifications. This way, users can be notified of any change in their platform (transfer of services, elimination and assignment of services, etc.).
- As is logical, they are included as the last advances that will contribute in the new versions of Eclipse Che, in this case we refer to the official documentation.

Capítulo 7
Presupuesto

7.1 Presupuesto

Recursos	Presupuesto
Nº de horas de trabajo: 400 (20€/hora)	8000€
Equipo de desarrollo	500€

TOTAL	8500€

Tabla 7.1: Tabla de presupuestos

Bibligrafía

[1] Captura de pantalla Cloud9.
https://ull-esit-sytw-1617.github.io/presentaciones-todos/cloud-9/
[2] Captura de pantalla Koding.
http://garrettn.github.io/blog/2014/07/07/battle-of-the-clouds-koding-vs-codio/
[3] Captura de pantalla CodeAnywhere.
https://codeanywhere.en.softonic.com/
[4] Página oficial IDE atom.
https://atom.io/
[5] Información sobre MySQL en Wikipedia.
https://es.wikipedia.org/wiki/MySQL
[6] Página de descarga Eclipse Che.
https://www.eclipse.org/che/getting-started/
[7] Repositorio Github Eclipse Che.
https://github.com/eclipse/che/
[8] Captura de pantalla Eclipse Che.
https://marketplace.atlassian.com/apps/1214702/codenvy-agile-plugin-for-jira
[9] Información sobre Docker en Wikipedia.
https://es.wikipedia.org/wiki/Docker
[10] ¿Qué es Docker?
https://www.redhat.com/es/topics/containers/what-is-docker
[11] Página de descarga Docker.
https://www.docker.com/get-docker
[12] ¿Qué es y para que sirve NodeJS?
https://www.netconsulting.es/blog/nodejs/
[13] Página oficial NodeJS.
https://nodejs.org/es/
[14] NodeJS, general callback-queue order.
https://stackoverflow.com/questions/47145255/node-js-general-callback-queue-order
[15] Virtualizando gratuitamente con oVirt.
https://alexismarin.wordpress.com/2015/01/15/ovirt-en-fedora/
[16] Arquitectura de oVirt.
https://www.ovirt.org/documentation/architecture/architecture/
[17] Eclipse Che, un IDE y área de trabajo de próxima generación en la nube.
https://ubunlog.com/eclipse-che-ide-area-trabajo-proxima-generacion-la-nube/
[18] Página de descarga MySQL.
https://www.mysql.com/downloads/
[19] Cas Authentication.
https://medium.com/@adiletmaratov/central-authentication-service-cas-implementation-using-django-microservices-70c4c50d5b6f
[20] Información sobre Redis en Wikipedia.
https://es.wikipedia.org/wiki/Redis

[21] Página de descarga Redis.
https://redis.io/download
[22] Tutorial básico de iptables en Linux.
https://www.linuxito.com/seguridad/793-tutorial-basico-de-iptables-en-linux
[23] WebScokets y SocketIO.
https://fernetjs.com/2012/11/websockets-y-socketio/
[24] Funcionamiento websocket.
https://ollyxar.com/websockets
[25] Página de descarga librería socket.io.
https://socket.io/
[26] Información oficial sobre Eclipse Che.
https://www.eclipse.org/che/docs/
[27] Manual de administración de pool de recursos iaas.
https://docs.google.com/document/d/1eKdwXsAfZJ9HJ92iLcj2VVieCujfagt9T2dXf-Gr_3s/edit
[28] Repositorio oficial de imágenes Docker.
https://hub.docker.com/
[29] Ejemplo de despliegue de un repositorio privado de Docker.
https://solidgeargroup.com/desplegando-un-registro-de-docker-privado-y-seguro?lang=es
[30] Página de la plataforma ULL-CloudIDE.
http://cloudide.iaas.ull.es
[31] Tcpkill
https://tournasdimitrios1.wordpress.com/2011/02/12/tcpkill-the-emergency-tcp-connection-terminator-utility/
[32] Guía y documentación sobre el sdk4 de oVirt en python.
https://www.ovirt.org/develop/release-management/features/infra/python-sdk/
[33] HTML <iframe> tag
https://www.w3schools.com/tags/tag_iframe.asp
[34] Repositorio Github de la plataforma ULL-CloudIDE.
https://github.com/alber7rp/TFG-ULL-CloudIDE