Gestión y desarrollo del proyecto de una aplicación web para la gestión de máquinas virtuales desde una perspectiva ágil

Tesis del Máster en Ingeniería del Software, Métodos Formales y Sistemas de Información

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Salvador Marí Alarcón
Dirigida por Agustín Rafael Espinosa Minguet

Agradecimiento

Quiero expresar mi sincero agradecimiento a Agustín Espinosa, Javier Salavert y Fernando Ferrer por haber confiado en mí y haberme animado a emprender la elaboración de este proyecto.

Agradezco también a mi compañero Marcos todos sus consejos y apoyo. Así como mis amigos y familiares que me han ayudado y apoyado durante este largo camino.

Y por último, gracias a mi prometida Sofía por su paciencia y ayuda a lo largo de estos meses.

Muchas gracias a todos.

“Any fool can write code that a computer can understand.
Good programmers write code that humans can understand.”

Martin Fowler
Refactoring: Improving the Design of Existing Code, 1999

Resumen

En la actualidad el departamento del DSIC cuenta con una página web a través de la cual el usuario accede a sus máquinas virtuales. Con el surgir de nuevas necesidades por parte del usuario, esta plataforma ha perdido funcionalidad. El objeto de nuestro proyecto será desarrollar una aplicación web para la gestión de máquinas virtuales que desde una perspectiva profesional logre adaptarse a las nuevas necesidades del usuario de una manera práctica y funcional. Este proyecto abarcará el seguimiento íntegro de la aplicación web, desde la elección de tecnologías hasta la implantación de estas y contará con un equipo de trabajo integrado por seis miembros en el que realizaré labores de dirección técnica. Para la planificación del proyecto se emplearán metodologías ágiles que permitirán facilitar la toma de decisiones.

Abstract

Currently, the DSIC department has a website through which the user can access their virtual machines. With the emergence of new needs on the part of the user, this platform has lost some functionality. The purpose of our project is to develop a web application for the management of virtual machines which, from a professional perspective, will effectively adapt to the user's new needs in a practical and functional manner. This project will cover the complete follow-up of the web application, from the choice of technologies to their implementation, and will have a work team consisting of six members, where I will carry out the task of technical direction. The methodologies used to plan the project will be agile, in order to facilitate decision making.

Resum

En l'actualitat el departament del DSIC compta amb una pàgina web a través de la qual l'usuari accedix a les seues màquines virtuals. Amb el sorgir de noves necessitats per part de l'usuari, esta plataforma ha perdut funcionalitat. L'objecte del nostre projecte serà desenrotllar una aplicació web per a la gestió de màquines virtuals que des d'una perspectiva professional aconseguisca adaptar-se a les noves necessitats de l'usuari d'una manera pràctica i funcional. Este projecte comprendrà el seguiment íntegre de l'aplicació web, des de l'elecció de tecnologies fins a la implantació d'estes i comptarà amb un equip de treball integrat per sis membres en què realitzaré labors de direcció tècnica. Per a la planificació del projecte s'empraran metodologies àgils que permetran facilitar la presa de decisions.

Índice

1. Introducción

1.1 Motivación

En la actualidad, el departamento del DSIC cuenta con una plataforma web que permite a los usuarios acceder a sus máquinas virtuales. Este portal, aunque funcional, ha quedado obsoleto, dejando de manifiesto la necesidad de actualización a fin de poder garantizar las nuevas necesidades del usuario.

Figura 1. Actual pantalla de login.

La actual plataforma surgió por la necesidad de que cada usuario pudiera acceder a un listado con sus máquinas virtuales asignadas. Además de listar las máquinas virtuales, la aplicación permite realizar las acciones de arrancar y detener una máquina virtual, así como acceder a ella a través de VNC. No obstante, esta aplicación se ha quedado desfasada y requiere de su actualización.
Algunos ejemplos de las carencias que presenta la actual aplicación podrían ser:

- La necesidad de realizar peticiones síncronas para las acciones de arranque y apagado de las máquinas virtuales de forma individual. Esto implica que sea necesario refrescar la pagina para ver los resultados obtenidos, pudiendo no haberse producido ninguna cambio.
- La interfaz no se adapta correctamente a nuevos dispositivos como smartphones.
- Se muestra muy poca información de cada máquina virtual y debido al diseño en forma de tabla es complicado mostrar más información de una forma ordenada.
- Es complicado incorporar nuevos elementos a la plataforma, ya que está diseñada para cubrir un único cometido que es, listar las máquinas virtuales.
- No cubre la gestión de las máquinas virtuales, recayendo esta tarea en el desarrollo manual por parte de los técnicos.

- ❏ El listado carece de filtro o búsqueda.
- ❏ No hay ningún tipo de paginación, por lo que en cada petición, se devuelve el listado completo.
- ❏ Por último, no existe mucha seguridad sobre el control y ejecución de VNCs por parte de los usuarios.

Son estos y otros factores, los que conllevan la necesidad de plantearnos el desarrollo de una nueva aplicación.

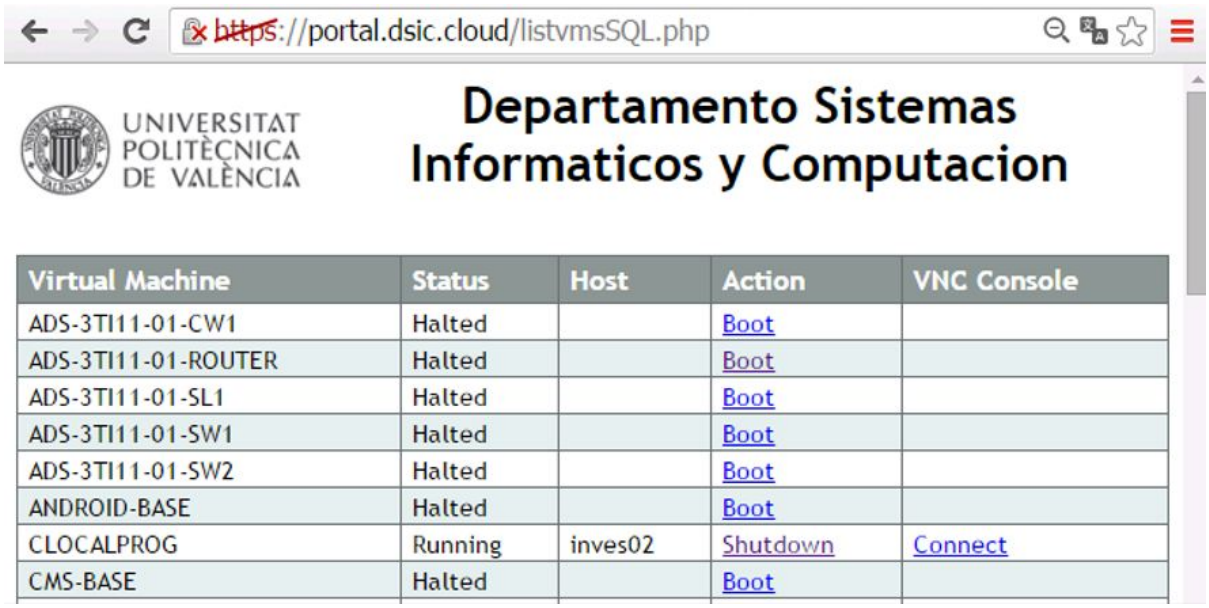


Figura 2. Listado de máquinas virtuales.

1.2 Objetivos

A largo plazo, pretendemos construir una completa infraestructura de gestión de máquinas virtuales en forma de aplicación web, adaptada a las necesidades del DSIC. La primera fase, y en la que se centrará este proyecto, es sentar las bases para la consecución de tal objetivo. Al finalizar esta primera etapa la aplicación deberá cubrir las necesidades que actualmente ofrece el antiguo portal, pero de una forma más dinámica y habiendo añadido ciertas funcionalidades.

No hay que olvidar que se trata de un proyecto de largo recorrido, y que por tanto el equipo técnico variará a lo largo de su desarrollo. Por ello, es importante crear una plataforma intuitiva donde la curva de aprendizaje no sea muy acentuada, y que permita a los nuevos técnicos una incorporación ágil, independientemente del grado de conocimiento en desarrollo web que estos presenten. En definitiva, será importante fomentar el empleo de buenas prácticas en el desarrollo del software que facilite su mantenimiento a largo plazo.

2. Metodología

La metodología de desarrollo de software es el conjunto de herramientas necesarias para planificar, estructurar y controlar el proceso de desarrollo en los sistemas de información. Estas herramientas pueden catalogarse en dos grandes grupos, las metodologías tradicionales y las metodologías ágiles. A continuación exponemos un breve resumen de sus características, ventajas y desventajas[1][2].

2.1 Metodologías tradicionales

Las metodologías tradicionales se caracterizan, principalmente, por seguir una serie de fases secuenciales que son, la definición de requisitos, la planificación, la construcción, las pruebas y el despliegue.

La primera fase, y la más importante, se centra en recoger información por parte del cliente de una forma amplia, detallada y documentada. De la buena ejecución de esta primera fase dependen las restantes, pues a partir de los requisitos marcados deberá quedar configurado claramente el objetivo último que debe garantizar la aplicación, sobre todo porque es posible que el contacto con el cliente no vuelva a producirse hasta la finalización del proyecto.

El carácter normativo y la fuerte dependencia que estas metodologías mantienen con la planificación previa al desarrollo conllevan que sea realmente difícil la adaptación del proyecto a los posibles cambios que se planteen. El entorno adecuado para el uso de esta metodología es aquel donde los requisitos no varíen o estos sean fáciles de predecir, lo que en la práctica no suele producirse.

2.2 Metodologías ágiles

El término ágil asociado a las metodologías de desarrollo de software se acuña por primera vez en Utah en el año 2001, durante una reunión de diecisiete ingenieros de software convocada por Kent Beck. Como consecuencia de esta reunión se desarrollan los principios sobre los que se basarían los métodos alternativos a aquellos denominados formales o tradicionales, de carácter más rígido y exclusivamente dependientes a una primera fase de planificación, como ya se ha explicado anteriormente.

El Manifiesto Ágil sería el resultado práctico de esta reunión. Desarrollado en cuatro postulados en él, se establecen los siguientes principios básicos [3]:

1. Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
2. Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
3. Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
4. Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
7. El software funcionando es la medida principal de progreso.
8. Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
12. A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

Dichos principios pueden agruparse en cuatro valores esenciales:

- **Individuos e interacciones** sobre procesos y herramientas
- **Software funcionando** sobre documentación extensiva
- **Colaboración con el cliente** sobre negociación contractual
- **Respuesta ante el cambio** sobre seguir un plan

Las metodologías ágiles, en el cumplimiento de estos principios, se centran en el desarrollo incremental e iterativo, donde se dividen los requisitos en fases que son ejecutadas y revisadas, constantemente, por grupos reducidos de desarrolladores durante todo el transcurso del proyecto.

2.3 Elección de metodología

La naturaleza impredecible que presenta este proyecto, en el que hemos sido conscientes de las bases del mismo pero no de su evolución, configura la metodología ágil como la mejor alternativa.
Dentro de las metodologías ágiles podemos destacar, Scrum, Kanban y XP (eXtreme Programming).
La metodología Scrum, nos planeta observar el proyecto desde una perspectiva iterativa, proporcionándonos para ello herramientas y roles. Con ello, logramos un mayor seguimiento y una mayor adaptación.
Por su parte, mediante la metodología Kanban, nos centraremos en bloques, no pudiendo avanzar al siguiente hasta que el actual esté finalizado.
Por último, la metodología XP dota al proyecto de una serie de valores como son, simplicidad, comunicación, retroalimentación (feedback), coraje y respeto, habiéndose añadido este último en la segunda edición de Extreme Programming Explained.
Entre estas tres principales metodologías ágiles nos basaremos en la metodología Scrum para el desarrollo de nuestro proyecto, y digo basaremos pues no nos ceñiremos a ella cuál ley, sino que iremos empleando las herramientas que mejor se adapten a las necesidades que se nos planteen en cada fase del proyecto.
Esta elección nos permite abordar las tareas planificadas en un periodo corto de tiempo, pudiendo volver a planificarlas según las necesidades que vaya teniendo el equipo.

2.4 Scrum en nuestro proyecto

Por medio de Scrum, el proyecto queda dividido en bloques temporales de entre una semana y un mes, pudiendo adaptarse este espacio temporal a cada modelo de proyecto. Estos bloques temporales se conocen como “sprints” y aunque su duración dependerá de las preferencias del desarrollador y el cliente, lo que se pretende es que ésta sea la menor posible. Las iteraciones de nuestro proyecto se establecieron en dos semanas, salvo alguna excepción provocada por factores externos.

La metodología Scrum permite que como resultado de cada sprint pueda presentarse al cliente un prototipo funcional, es decir un producto que este podría utilizar. Los sprints se planifican junto al cliente, quien decide los requisitos, su prioridad y valor. Una vez finalizado el sprint se realiza una demostración ante el cliente de los requisitos completados en forma de incremento del producto listo para ser entregado. Tras esta demostración, el cliente plantea las adaptaciones o

cambios que desee y se planifica de nuevo el proyecto. Como vemos, esta metodología permite una adaptación ágil a las nuevas situaciones que surjan a lo largo del proyecto, así como reajustarse tanto a la complejidad del mismo como al coste temporal.

La planificación de los requisitos se desenvuelve a través del Product backlog, esto es una lista de tareas u objetivos que el equipo elabora en la reunión previa a la iteración. Mediante el Product backlog el equipo puede visualizar, de manera fácil, que tareas les supone un mayor esfuerzo o cuales presentan mayor problemática lo que permite tomar decisiones al respecto con mayor rapidez. Es importante advertir, que una regla esencial de esta metodología es que el backlog no podrá modificarse durante el transcurso de la iteración.

Como método de organización de nuestro backlog hemos empleado una pizarra Kanban. En ella puede, visualizarse cómodamente y en todo momento que tareas se encuentran pendientes y que miembros tienen que trabajar en cada una de ellas. Hemos utilizado una herramienta online conocida como Trello[4]. A través de ella, se crea una tarjeta por historia que se asigna al backlog, seguidamente se crean tres columnas, en nuestro caso, READY. WORKING y COMPLETE. Al inicio del “sprint” se mueven las tarjetas que entren dentro de la iteración a READY; cada vez que alguien esté desarrollando una historia, la moverá a WORKING. Una vez finalizada, esta se moverá a COMPLETED.

A su vez, Trello, nos permite asignar personas a cada tarjeta, así como adjuntar imágenes y comentarios a las mismas. De esta forma, toda la información relativa a una historia queda concentrada en un único lugar.

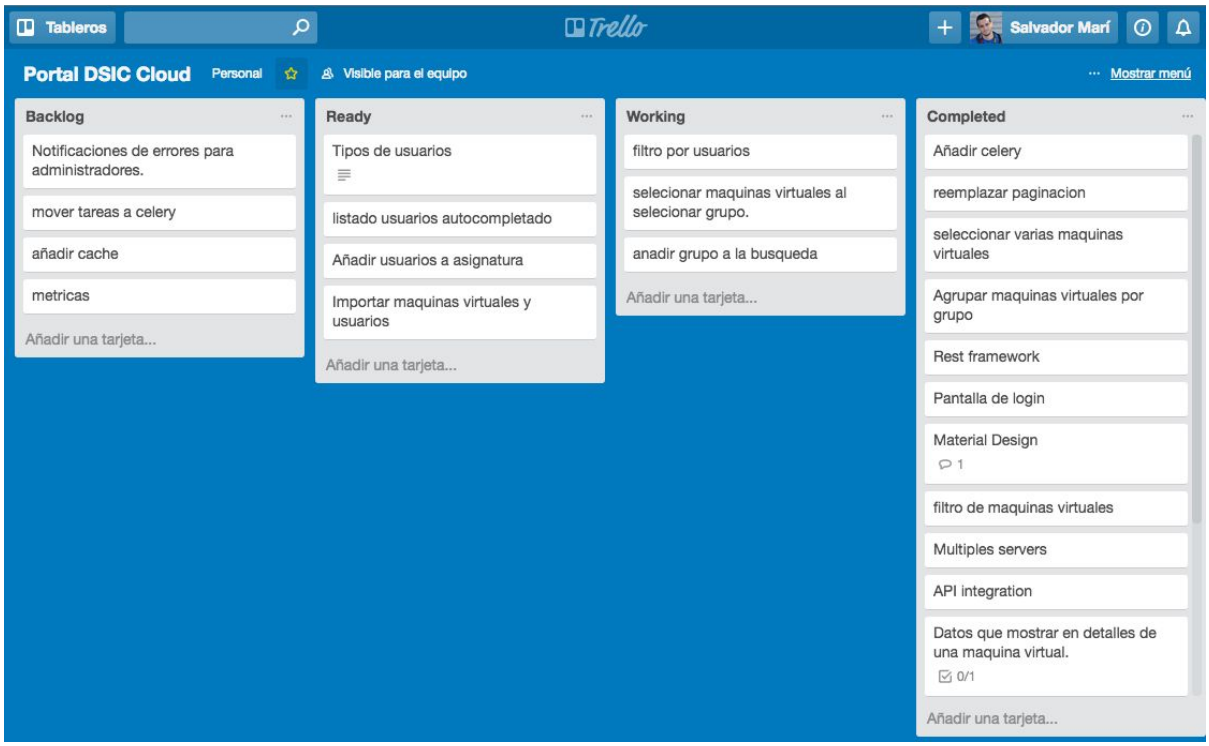


Figura 3. Vista del tablero de Trello.

Durante el sprint, el equipo de desarrollo mantiene reuniones diarias de máximo quince minutos llamadas Daily meeting o reuniones de sincronización. El objetivo de estas reuniones es mantener a todo el equipo al corriente del estado del proyecto y pueda valorar el progreso de la iteración. Cada componente del equipo deberá contestar a las siguientes preguntas:

1. ¿Qué he llevado a cabo desde la última reunión de sincronización?
2. ¿Qué voy a hacer a partir de este momento?
3. ¿Qué impedimentos tengo o puedo tener?

Para que estas reuniones sean fructíferas es importante poder hablar de equipos de desarrollos pequeños y ubicados en un mismo lugar de trabajo. Estas reuniones aunque importantes, tienen un carácter informal, siento preferible que se mantengan estando de pie para evitar una duración excesiva. En nuestro caso, estas reuniones no se han llevado a término debido a que aun siendo un equipo pequeño no se ha cumplido el segundo requisito ya que contábamos con horarios y localizaciones dispares. Aun así, la comunicación se ha mantenido fluida durante todo el proyecto.

2.5 Definición de roles

El equipo de desarrollo de nuestro proyecto se compone de siete miembros.

PRODUCT OWNER:
Agustín Espinosa, quien representa al cliente, asegurándose de que el equipo trabaje de forma adecuada desde la perspectiva del negocio. Es el responsable de fijar las historia de usuario, es decir, los requisitos de cada iteración, priorizarlos y colocarlos en el Product backlog.

SCRUMMASTER O FACILITADOR:
Salvador Marí, cuya principal función es ayudar al equipo a superar los obstáculos que se presenten en el desarrollo de cada sprint, a fin de alcanzar los requisitos fijados en el mismo.

EQUIPO DE DESARROLLO:
Formado por cinco miembros: Javier Salavert, Fernando Ferrer, Juan Miguel Bautista, Sergio Esparcia y Borja Herráez, siendo Sergio y Borja aquellos miembros que, tras su incorporación, se han dedicado de forma exclusiva al desarrollo del proyecto.
Dentro del equipo, adaptamos las tareas a los miembros según su conocimiento. Javier y Fernando se han encargado de la parte de sistemas. Juan Miguel, gracias a su experiencia desarrollando la intranet del DSIC, ha colaborado con tareas tanto de fronted como de backend. Por último, Sergio y Borja se han dedicado a la interacción del backend con XenServer.

3. Requisitos

Las primeras reuniones tuvieron lugar entre los miembros del equipo técnico, Salvador, Javier y Fernando, con el objetivo de establecer los requisitos que debía reunir la plataforma. Fernando es el desarrollador de la plataforma actualmente existente y el miembro del equipo con mayor conocimiento técnico sobre el funcionamiento del servidor de máquinas virtuales. Como facilitador, el primer objetivo que debía abordar era la adquisición de los conocimientos necesarios para la toma de decisiones. Para ello, fue necesario familiarizarse con:

- ❏ XenServer [9]: se trata de un hypervisor esto es, una plataforma que permite la gestión de distintos sistemas operativos en una misma máquina.
- ❏ XenCenter [10]: es una aplicación para windows que nos permite la gestión de XenServer mediante una interfaz gráfica.
- ❏ XenAPI [11]: XenServer proporciona una API con una extensa documentación para acceder a sus servicios. Existen varias versiones para varios lenguajes, aunque no todas están igual de documentadas.

El objetivo de los técnicos se estableció en ofrecer la misma capacidad de gestión de XenCenter, a través del nuevo portal. No obstante, debido al gran número de máquinas virtuales existentes en el sistema supuso que los tiempos de espera en el uso de la aplicación fueran excesivos, y por tanto que el empleo de XenCenter fuera inviable.

Como forma de facilitar nuestro trabajo, en lugar de emplear los servidores del DSIC, creamos un entorno de desarrollo que pudiera ejecutarse en un ordenador personal. Este consistió en instalar un Xenserver sobre VMWare. El Xenserver contenía un Pool con dos máquinas virtuales, una máquina virtual Windows XP y otra Ubuntu. A su vez, añadimos una tercera máquina virtual que consistía en un Windows XP con un XenCenter. Desafortunadamente, esta solución impedía el aprendizaje del funcionamiento de la aplicación de una forma independiente.

Una vez adquirido un mayor conocimiento sobre las tecnologías asociadas al proyecto, decidimos entablar una reunión con los usuarios. Para ello, se propuso una reunión abierta en la que todos los miembros del departamento que lo desearan pudieran dar su opinión. Mediante esta información se pretendió centrar los requisitos y funcionalidades del nuevo servicio.

Esta reunión fue gratamente productiva, pudiendo extraer de ella las siguientes conclusiones:

- ❏ En la actual plataforma existen problemas a la hora de usar una máquina virtual en proyector, al parecer la resolución resultante es demasiado alta y por tanto el tamaño de las fuentes resulta muy pequeño.
- ❏ También existen problemas a la hora de ajustar la resolución, ya se ejecute la máquina virtual por VNC o RDP.
- ❏ El tiempo de respuesta del actual portal es distinto en función del laboratorio del DSIC en el que se esté utilizando.
- ❏ Monitorización del sistema para detectar si alguna máquina virtual está consumiendo recursos excesivos y está impactando en el rendimiento global de la plataforma.
- ❏ Ampliación de la documentación proporcionada a los alumnos para conectarse a las máquinas virtuales.
- ❏ Aprovechamiento del tiempo ocioso para dar servicio a cómputo intensivo.
- ❏ Activación máquinas en un determinado horario para hacer un examen.
- ❏ Simplificación de la forma de acceso externo para que solo se realice a través de Windesktop.
- ❏ Asignación de alumnos a máquinas virtuales usando la aplicación web, incluida la ocultación de la máquina virtual durante la corrección.
- ❏ Uso de nombres de alumnos como identificación, no logins en la plataforma.
- ❏ Posibilidad de realizar acciones sobre un grupo de máquinas virtuales, como pueda ser, desactivar para todo un grupo de usuarios el acceso a las máquinas virtuales.
- ❏ Filtro de máquinas virtuales en base a algún criterio determinado.
- ❏ Posibilidad de añadir más información sobre cada máquina. En particular información adicional: Mac, dispositivo de arranque, métodos de acceso alternativos, archivos RDP, etc.
- ❏ Posibilidad de realizar Backups o snapshots de las máquinas virtuales.
- ❏ Posibilidad de descargarse una imagen de la máquina virtual.
- ❏ Acceso simultáneo a la misma máquina.
- ❏ Uso de ciertos caracteres como @, \ en consola VNC.
- ❏ Acceso a múltiples GPUs para poder ejecutar entornos tipo emulación de Android. En general, ofrecer hw dedicado a la máquina virtual
- ❏ Posibilidad de comunicar tu máquina personal con la máquina virtual remota
- ❏ Herramientas para los técnicos de monitorización y alertas del sistema.
- ❏ Ejecución de la aplicación RMI en laboratorios.
- ❏ Encender los grupos de máquinas virtuales en un host que vayan a caber todas.
- ❏ Posibilidad de que un grupo de máquinas virtuales se ejecuten en un mismo host.
- ❏ Posibilidad de que queden libres los hosts durante las sesiones de prácticas
- ❏ En el actual portal las acciones de encender y apagar son síncronas, es decir, la página no responde hasta que se ha terminado de encender o

apagar una máquina virtual, lo que es especialmente lento en el apagado. Por tanto, se platea la posibilidad de que el encendido y apagado sea asíncrono, tal como hace XenCenter, mostrando la máquina en un estado intermedio de encendiéndose o apagándose y permitiendo la interacción mientras tanto.

Tras la reunión quedó patente la necesidad de aportar ciertas mejoras que permitieran ganar funcionalidad a la actual plataforma, como son la gestión asíncrona de acciones sobres máquinas virtuales, la paginación sobre listados extensos de máquinas virtuales, la gestión de usuarios dentro de la plataforma, la asignación de roles a los distintos tipos de usuarios, la migración de los scripts de los técnicos a funcionalidades dentro de la plataforma, etc.

4. Sprints

Los sprints quedaron establecidos en períodos de quince días, con alguna excepción. A continuación, procederemos a resumir aquellas tareas que se han realizado en cada uno de ellos.

Sprint 1

Las primeras tareas a desarrollar dentro de este primer sprint, radicaron en constituir el entorno de desarrollo. Para el desarrollo del servidor se eligió Django, esto es un framework web escrito en Python. Esta elección era obvia, ya que la intranet del departamento está desarrollada bajo la misma tecnología. No obstante, aunque la solución sencilla y rápida hubiera sido emplear la versión Python 2.7, por ser ésta concretamente la empleada por la intranet del departamento, se impuso como requisito del equipo técnico el empleo de la versión Python 3. Todo ello, porque al tratarse de un proyecto a largo plazo, era recomendable el uso de tecnologías actuales y con proyección de futuro. Además, la interacción de los desarrolladores técnicos con estos incipientes lenguajes de programación conlleva un valor añadido a su experiencia laboral.

En este primer sprint, también se creó un repositorio temporal en bitbucket, con el objetivo de que tras cada iteración el cliente pudiera acceder a su código.

Sprint 2

La segunda iteración de nuestro desarrollo consistió en la creación del modelo de usuarios con vistas para su integración con el usuario “Persona” del DSIC.
Se decidió el uso de Angular para el frontend, y ello debido a que en la actualidad es la librería muy extendida y demanda en el sector. Además, se trata de un framework muy completo que permite hacer ciertas funciones de forma sencilla, como la actualización automática de variables javascript en el html.

Por otro lado, se llevó a cabo la integración del frontend en el proyecto, así como la estructura de éste para nuestra futura aplicación. También, preparamos los scripts necesarios para mimificar y unificar los distintos ficheros javascript y se implementó la vista de login, con su correspondiente validación en Angular.

Sprint 3

En esta tercera iteración, el Product Owner nos proporcionó las guías de estilo que debía tener la aplicación y que seguirán un estilo similar al OneDrive. Al no tener ningún diseñador en el equipo, la mejor opción era buscar un framework que nos proporcionara una apariencia similar y así la adaptación fuera lo menos costosa posible.

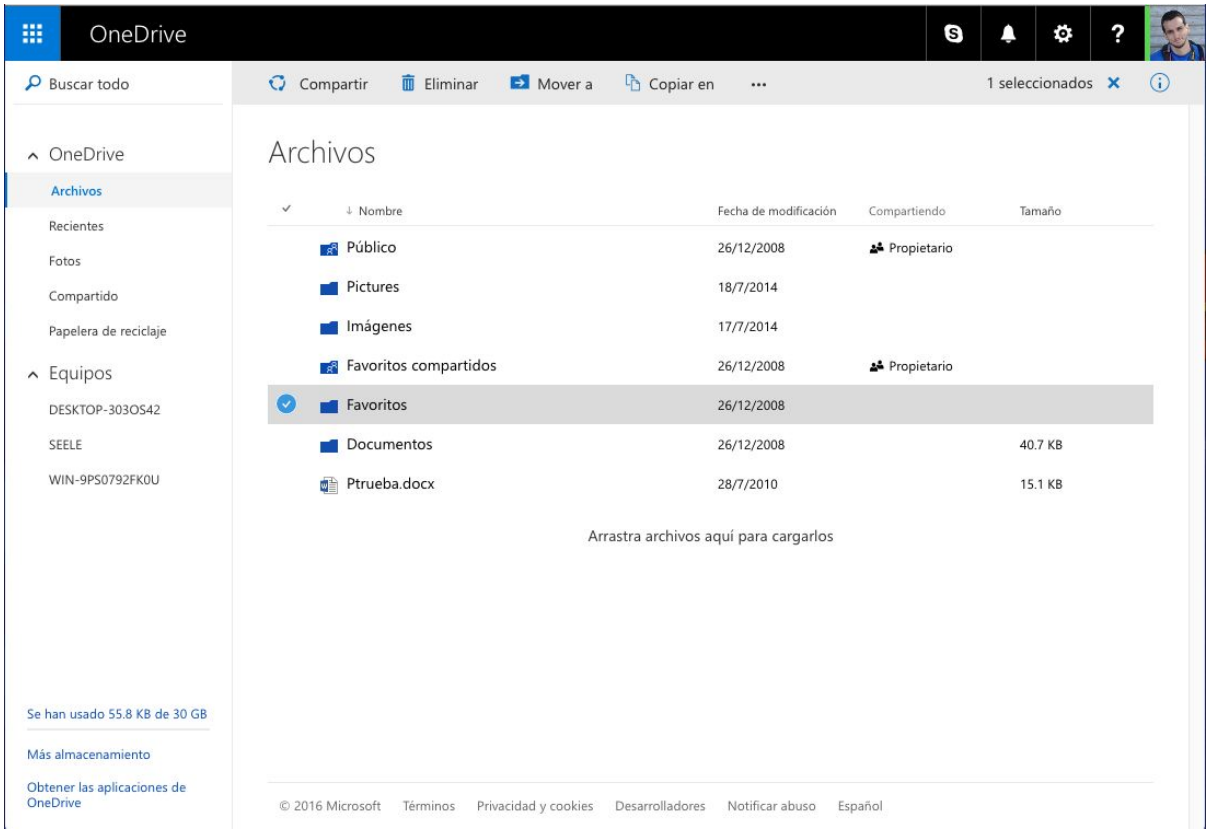


Figura4. Página de inicio de OneDrive.

Después de explorar las distintas opciones, decidimos escoger un framework conocido como Material Design [5] que sigue las líneas de diseño de Google.

La primera opción era Material Design Lite [6], una librería de google que incorpora las estructuras básicas. No obstante, después de implementarlo con nuestro formulario de Login observamos que no nos proporcionaba la infraestructura necesaria para el desarrollo del proyecto. La segundo opción, se centró en Angular Material Design [7]. Esta librería nos proporciona multitud de elementos como modales, alertas, iconos, menús. Además, es fácilmente extensible y modificable. Una vez implementado con el formulario de login comprobamos que era una opción sólida para el desarrollo del proyecto. Con esta opción, nos aseguramos el

progresivo desarrollo y la mejora de esta librería pues su mantenimiento depende de Google.
Una vez seleccionado nuestra librería de estilo, creamos un prototipo de prueba para mostrar al Product Owner como se visualizará el listado de máquinas virtuales después de aplicar los estilos.

Para la organización de los listados, optamos por un patrón master-esclavo, yendo en el master los listados de las máquinas virtuales y en el esclavo los detalles de la máquina virtual seleccionada. En modo móvil, el master pasaría a ser un menú y el esclavo sería la vista principal.

Sprint 4

El cuarto sprint consistió en la implementación de la API para acceder a las máquinas virtuales. Para evitar depender de la librería del hipervisor, decidimos crear nuestra propia librería y que ésta envolviera a la del XenServer. De esta forma, la lógica para interactuar con XenServer queda encapsulada en un único lugar y somos más sólidos frente a cambios de la API, incluso en caso de que se quiera cambiar de hipervisor.
En el momento de implementar los conocimientos obtenidos en la primera etapa del proyecto, nos dimos cuenta que la librería Python que nos proporciona XenServer no es compatible con Python 3, lo que suponía un problema. Ante este dilema, teníamos dos opciones, la migración del proyecto a Python 2.7 o la actualización de la librería que nos proporciona para lograr su compatibilidad con Python 3. Optamos por ésta última. En un primer momento, probamos varias librerías que permiten automatizar el proceso como 2to3, no obstante, al no conseguir resultados procedimos a actualizarla manualmente. Después de actualizar las librerías, realizamos las pruebas pertinentes y compartimos la librería en Github [12] para que cualquiera con el mismo problema pudiera beneficiarse de la experiencia.

Por otro lado, creamos el modelo de Pool con el fin de que tuviera las variables necesarias para poder conectarse a XenServer y acceder así a las funciones del API que habíamos creado de forma automática. Este modelo tiene los parámetros necesarios para conectarnos al servidor, ya sea la IP, URL o el nombre de usuario.

Por último, dentro de nuestra API de hipervisor creamos los métodos para ejecutar las acciones de arrancar y apagar una determinada máquina virtual.

Sprint 5

El siguiente paso era mostrar el listado de máquinas virtuales obteniendo los datos del servidor. Para ello, creamos una función en nuestra API de hipervisor para obtener todas las máquinas virtuales de un determinado Pool. Con esta información, creamos nuestro propio modelo de máquina virtual donde guardamos los datos que más nos interesan de aquella que nos devuelve XenServer. Así pues, empleamos dicha información para actualizar el listado de máquinas virtuales.

Llegados a este punto, necesitábamos crear una API para que Angular pudiera consumir los datos necesarios para renderizar la vista. Para ello, hicimos uso de Django Rest Framework [13]. Se trata de un potente framework destinado a la creación de Web APIs.

Además de modificar el listado para que renderice las máquinas virtuales que recibe desde la API, actualizamos la vista de detalle mostrando la información que guardamos en el modelo de cada máquina virtual.

Sprint 6

Dado que el Pool necesita ser introducido por los técnicos para poder realizar la conexión, la tarea principal de esta iteración consistió en establecer un formulario para la creación y edición de Pools, así como la creación de un listado y una vista de detalle para cada Pool siguiendo los estilos que teníamos en Máquinas Virtuales.

El acceso de este nuevo menú es solo visible para los técnicos y queda disponible en la barra de navegación. El formulario queda integrado en un modal y, la inserción y actualización de Pools se realiza mediante angular y llamadas a la API.

Sprint 7

En este Sprint nos centramos en completar los requisitos relacionados con las máquinas virtuales. Una vez teníamos el listado para máquinas virtuales, era necesario poder realizar acciones sobre ellas. Para ello, dividimos el detalle de una máquina virtual en dos pestañas, siendo una acciones y otra detalles.

Seguidamente, para poder encender y apagar máquinas virtuales, necesitamos exponer dos endpoint de nuestra API rest que mediante un PATH realice la acción. La llamada se realiza mediante Angular y muestra al usuario, que se está realizando

una acción en dicha máquina virtual. Además, el usuario no puede realizar otra acción sobre la máquina virtual mientras no haya terminado la anterior.

Además, establecimos un sistema de comunicación con el usuario que consiste en la implementación de un sistema de alertas que indique, en caso de error, que ha sucedido algo no deseado.

Para finalizar, usamos una librería de iconos, Material Icons [8], para representar las acciones sobre la máquina virtual.

Sprint 8

Los objetivos de esta iteración fueron la búsqueda y desarrollo de un refactoring sobre los listados. En un primer momento, nos percatamos de que los listados de pools, máquinas virtuales y posiblemente cualquier listado nuevo que deseáramos implementar tenía muchos elementos en común. Para la reutilización del código, creamos una serie de templates base, que heredan cualquier listado por lo que únicamente es necesario sobrescribir el detalle que nos gustaría mostrar.

Asimismo, para la búsqueda creamos, en este listado general, un input que pudiera heredar cualquier listado, siendo necesario únicamente modificar la llamada a la API que realizará la búsqueda.

Por último, exploramos la posibilidad de obtener métricas de cada máquina virtual para incorporarlo en posteriores Sprints. No obstante, debido a la escasa documentación y los malos resultados obtenidos en un primer momento, decidimos posponerlo.

Sprint 9

Las tareas a realizar en este Sprint fueron implementar la paginación en los listados y añadir el modelo de asignatura. La paginación fue definida por el Product Owner de forma que pudiera introducir nuevos elementos mientras se avanzaba con el scroll. Para su implementación, hicimos uso de la propia librería de Angular Material, ya que nos proporciona una herramienta para paginar elementos cuando todos comparten la misma altura.

Por otro lado, puesto que no es posible mantener más de dos máquinas virtuales reales en la infraestructura de desarrollo local, creamos una serie de funciones que generase datos falsos sobre máquinas virtuales y así comprobar el funcionamiento y

el tiempo de respuesta de los listados. Además, realizamos las llamadas correspondientes en nuestra API para, así poder paginar listados.

La creación del nuevo modelo de asignatura fue más fácil que en anteriores ocasiones, ya que al tener vistas genéricas, el coste de añadirla fue menor.

Sprint 10

A través de las demostraciones de uso de la aplicación, quedó patente que la interfaz desarrollada no se adecuaba a los requisitos de la aplicación. En dispositivos móviles, la navegación se hacía complicada, teniendo que abrir cada vez el menú para mostrar los listados.

Asimismo, al añadir nuevas opciones como asignaturas, máquinas virtuales o Pools a la cabecera surge la necesidad de crear un nuevo menú.

Después de varias estudiar varias propuestas del equipo de desarrollo, optamos por dotar a nuestra plataforma de un estilo similar al que muestra Inbox de Google. Este estilo, nos ofrece la posibilidad de realizar acciones sobre el listado y evita tener que acceder al detalle para poder ejecutar acciones sobre la máquina virtual.

La migración de una interfaz a otra fue menos complicada de que a primera vista podría parecer, ya que al tener todos los listados centralizados simplemente tuvimos que actualizarlo en un único lugar.

Además del cambio de estilo, en esta iteración incluimos la autentificación de los alumnos contra el LDAP. Este proceso consiste en una validación de usuario y contraseña contra el servidor de LDAP del departamento DSIC. Una vez validado, si no figuran los datos registrados del usuario, se procede a su creación de manera automática de nuestra base de datos local.

A continuación, mostraremos el estilo que tenía la aplicación previamente a los cambios llevados a cabo es esta iteración:

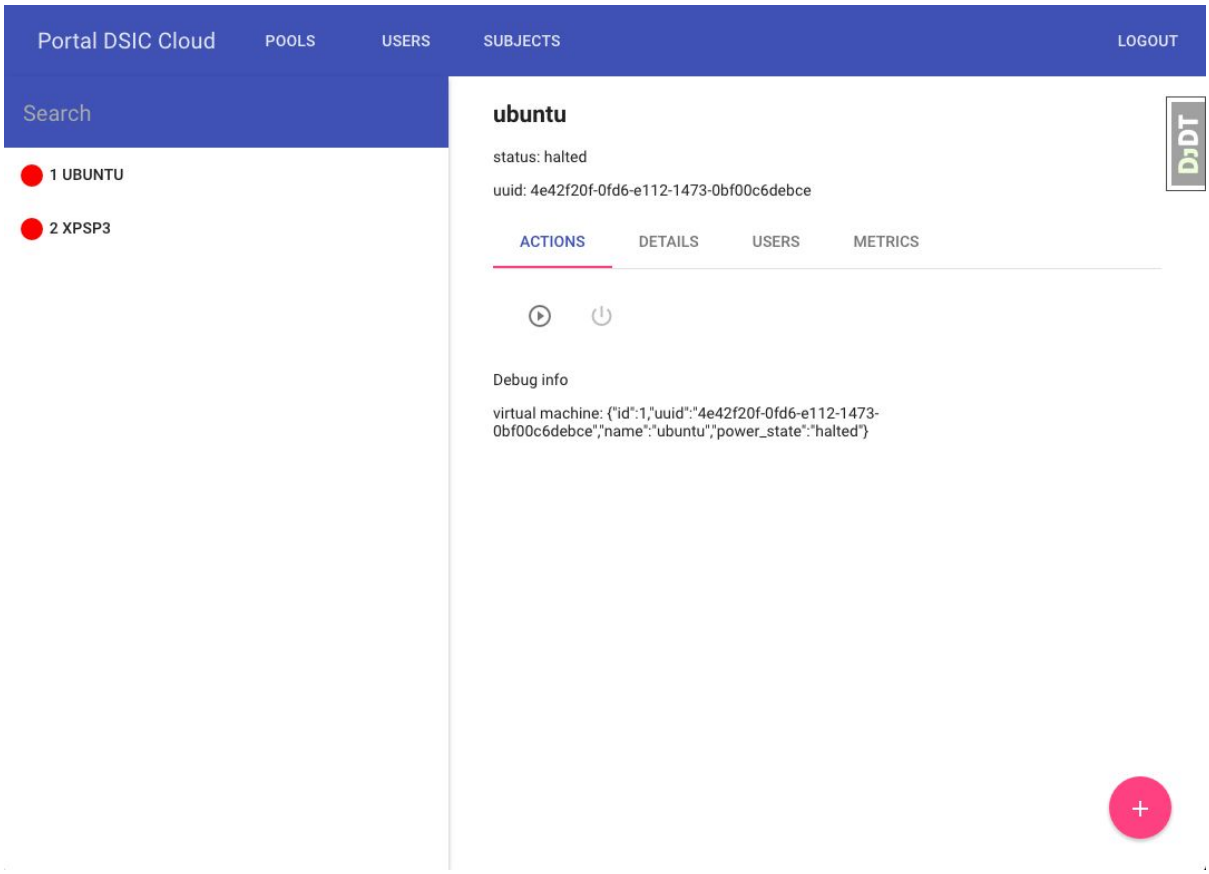


Figura 5. Vista de máquinas virtuales en modo escritorio.

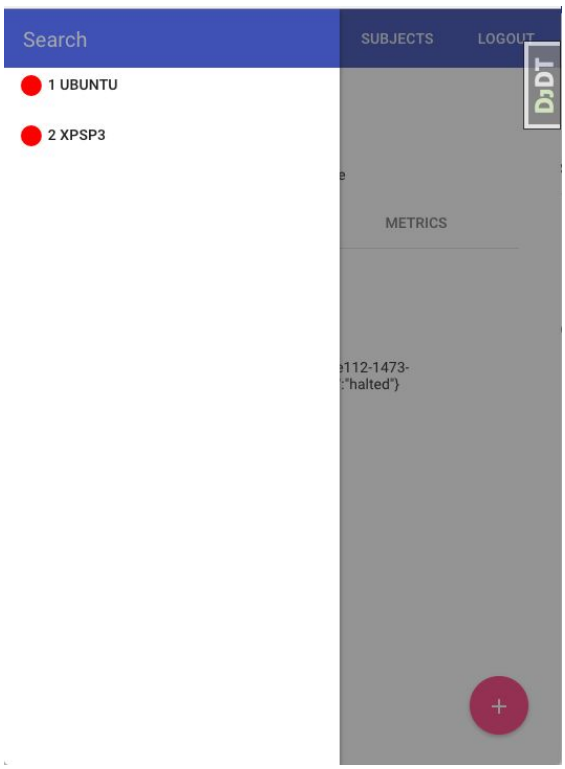


Figura 6. Listado de máquinas virtuales en modo móvil.

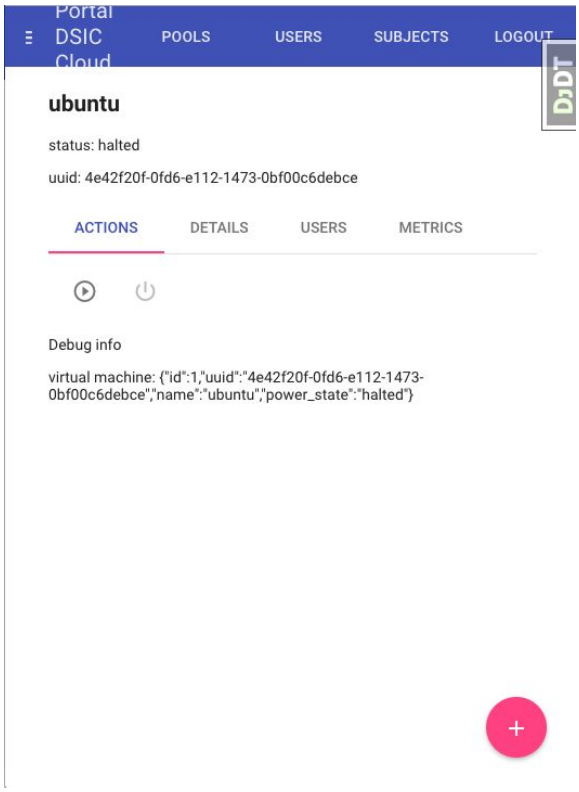


Figura 7. Detalle de máquinas virtuales en modo móvil.

Sprint 11

A lo largo de esta iteración se procedió a determinar las distintas modalidades de acceso según el tipo de usuario. En concreto, existen cuatro tipos de usuarios.

- Técnico
- Responsable
- Profesor
- Alumno

Cada uno de ellos, puede acceder a diferentes funcionalidades de la plataforma. Para ello creamos el modelo de Rol, y así poder añadir nuevos roles en el futuro, si fuera necesario.

Además de la gestión de los roles, así como su listado, modificamos la búsqueda para que fuera común en todas los listados, añadiendo el input de búsqueda a la cabecera.

Sprint 12

Las tareas relativas a este sprint fueron la gestión de los hosts, así como relacionar los usuarios con máquinas virtuales y asignaturas.

Para la gestión de los host, creamos una nueva función en nuestra API de hipervisor para poder importarlos desde el XenServer. Con estos datos, confeccionamos su modelo y su vista correspondiente.

La relación entre usuarios, máquinas virtuales y asignaturas es complicada. En la definición de nuestros requisitos, se estableció que un usuario podía pertenecer a varias asignaturas y una máquina virtual podía tener varios usuarios. Procedimos a crear un modelo Miembro en el que agrupamos los usuarios con su asignatura. En esta relación añadimos el rol, ya que un usuario puede ser profesor en una asignatura y responsable en otra. Para la relación entre usuarios y máquinas virtuales creamos grupos, en los que se asocia una máquina virtual con uno o varios usuarios y una asignatura. Como información adicional, añadimos un nombre al grupo.

La interfaz para la asignación de usuarios a máquinas virtuales dentro de una asignatura requería la creación de grupos, la búsqueda en los usuarios de la base de datos de la Universidad y la asignación de roles. Para ello, dividimos el detalle de una asignatura por roles donde asignaremos los responsables (los cuales solo pueden ser asignados por los técnicos), los profesores (asignables por técnicos y responsables) y los grupos de máquinas.

Sprint 13

En esta iteración nos centramos en la gestión de los grupos. La creación de los grupos está asociada a la creación de máquinas virtuales, ya que primero se crean éstas con un grupo y luego se asocian los alumnos a los grupos.
En este momento, nos centramos en tratar la gestión de usuarios dentro de los grupos, puesto que las máquinas ya vendrían importadas de la actual plataforma con sus grupos y su asignatura asociada.

En el detalle de una asignatura, podemos ver todos los grupos asociados a ésta con un listado de sus usuarios. Además, implementamos un filtro para mostrar solamente los grupos que coincidan con la búsqueda.

En cada grupo del listado, es posible añadir usuarios. Para ello mostramos un modal en el que aparecen los usuarios que actualmente están dentro del grupo y un buscador.
Debemos poder asociar cualquier usuario que exista en la base de datos del departamento a un grupo. De ahí que, las búsquedas las realizaremos sobre esta base datos. Una vez seleccionado el usuario, se añadirá al listado y presionando sobre él, se eliminará del mismo.
Una vez pulsemos sobre el botón guardar, el sistema asociará los usuarios seleccionados al grupo, creando los que todavía nos se encuentren en nuestra base de datos.

Sprint 14

Durante este sprint se decidió gestionar las acciones sobre las máquinas virtuales mediante tareas, para que estas no bloquearán el servidor mientras trabaja XenServer. Hasta el momento, si un usuario apagaba cinco máquinas virtuales, el servidor apagaba una, esperaba a que XenServer terminara y entonces apagaba la siguiente y así sucesivamente, bloqueando la ejecución del hilo e impidiendo que pudieran ejecutarse nuevas peticiones. Esto suponía un problema de rendimiento, ya que el número de peticiones simultáneas es finito.

XenServer nos ofrece la posibilidad de realizar llamadas asíncronas, devolviendo un identificador de la tarea que se ha creado. Esto un primera paso pero no el definitivo, ya que nuestro servidor no sabría cuando ha finalizado.
Para solucionar este problema utilizamos Celery [14] como gestor de tareas y RabbitMQ [14] como gestor de colas, para ello, hay que variar el funcionamiento tanto del frontend como de las llamadas de nuestro API de hypervisor.
Las llamadas para realizar acciones desde el frontend ahora, deben realizar una llamada para ejecutar una acción y posteriormente preguntar por su estado.
En el backend se lanza una tarea a Celery que ejecuta la acción síncrona sobre XenServer y luego pregunta por el estado de la tarea. Una vez finalizada la tarea actualiza el estado.

Sprint 15

Para sustituir el actual portal necesitábamos importar los datos de la base de datos. Para ello, se nos proporciona unos CSV con toda la información relacionada con las asignaturas, grupos y máquinas virtuales.
Además, se decide implementar la última funcionalidad del antiguo portal que aún no soportaba el VNC.

Para implementar el VNC utilizamos la misma librería que empleaba el anterior portal, no VNC. Esta librería, permite usar la interfaz gráfica de una máquina virtual desde el navegador. Para poder hacer funcionar el VNC tuvimos que implementar varias cosas:

- Recuperar el puerto de la máquina virtual que utiliza XenServer para conectarlo.
- Crear un túnel para conectar el puerto de la máquina virtual con el puerto que nos proporciona Xenserver.
- Lanzar un proxy para conectar noVNC [16] con XenServer.
- Una vista renderizada por nuestro servidor que encapsule la que nos proporciona noVNC.

Además, añadimos un modo a pantalla completa para cuando se esté en la consola VNC.

Sprint 16

Las tareas de este sprint están destinadas a evitar al usuario tareas repetitivas permitiendo agrupar éstas. Nuestra primera tarea fue añadir un menú en cada asignatura que permitiera apagar todas las máquinas virtuales asociadas a ésta. Para ello creamos una llamada en nuestra API que al realizar un PATH con un identificador de asignatura, apagase todas las máquinas virtuales asociadas.

El siguiente paso fue permitir seleccionar varias máquinas virtuales en su listado. Añadimos un checkbox que se mostrará cuando un usuario pasa el cursor sobre una máquina virtual. Una vez seleccionada una máquina virtual mostraremos en la cabecera de la página el número de máquinas virtuales seleccionadas, así como, las acciones que se pueden realizar sobre ellas. Además, permitimos que se pueda seleccionar todas las máquinas virtuales de un grupo pulsando sobre el nombre de éste y eliminado la selección si se vuelve a clicar sobre el.

Por último, permitimos que se pueda ocultar a los usuarios de una asignatura las máquinas virtuales de su listado. Para ello, añadimos un botón en el listado de asignaturas donde es posible activar o desactivar esta funcionalidad. De esta forma un profesor puede mostrar y ocultar las máquinas de su asignatura cuando lo necesite.

Sprint 17

En este Sprint, procedimos con la migración de los scripts de creación de máquinas virtuales a la plataforma. En primer lugar, abordamos la creación de máquinas virtuales base. Estas máquinas virtuales son las que editan los profesores para

adaptarlas a su asignatura. Una vez modificadas, se bloquea su edición y pueden ser clonadas para asignarse a los usuarios.
Para ello, en el detalle de asignatura añadimos la seccion correspondiente. Mediante un formulario, el usuario podrá asignar un nombre y seleccionar un template y, a partir de éste, crear su máquina virtual base. Los templates los proporcionan los técnicos y son máquinas “limpias” como por ejemplo, un ubuntu con las aplicaciones imprescindibles instaladas por los técnicos. Una vez creada la máquina base, el usuario podrá acceder a ella para modificarla.

Sprint 18

En esta iteración iniciamos la creación de las máquinas virtuales a las que los usuarios podrían acceder. Como ya podíamos asociar una o más máquinas virtuales base a una asignatura, ahora debíamos clonar éstas para asociarlas a un grupo y posteriormente, asignar qué usuario o usuarios podrían acceder a ellas.
Para ello, lo primero que hicimos fue crear un formulario en el que el usuario pueda seleccionar qué máquinas base clonar de todas aquellas creadas. Una vez seleccionadas, queda la creación de grupos. Este es un proceso lento y costoso, ya que el usuario introduce el número de grupos que quiere crear, y a partir de este se clonan las máquinas virtuales base seleccionadas y se crean ese determinado número de grupos. Una vez terminado este proceso, ya se pueden seleccionar los usuarios que queremos que formen parte del grupo.

4. Interfaz gráfica

En este punto del proyecto, la interfaz gráfica mostraba la siguiente apariencia:

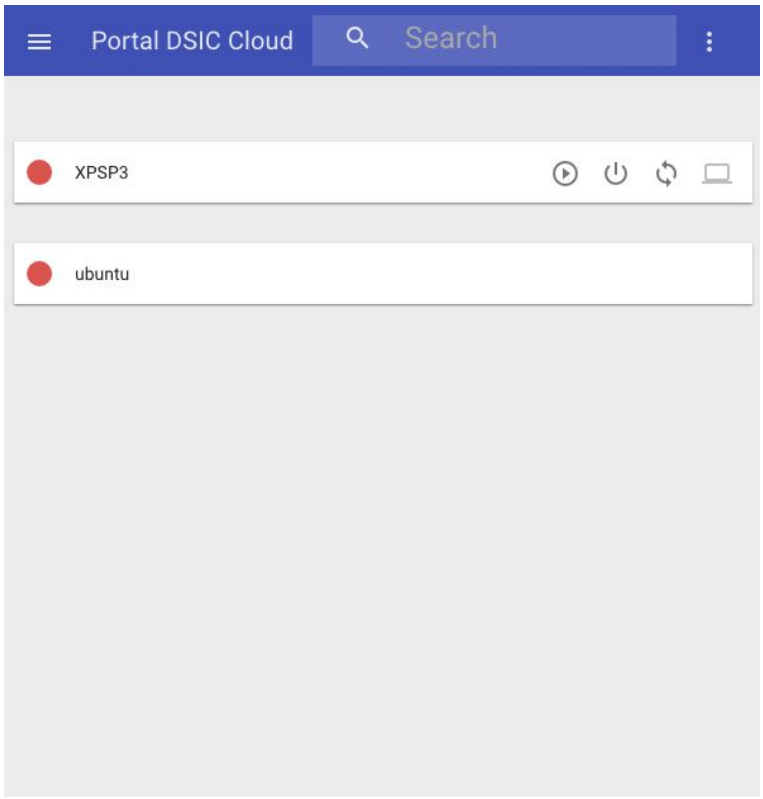


Figura 8. Listado de máquinas virtuales en modo móvil.

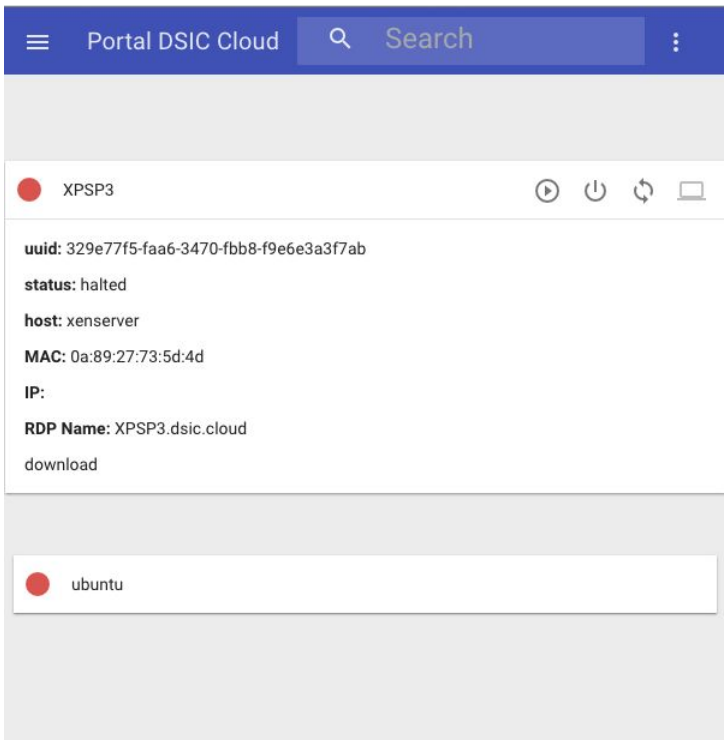


Figura 9. Detalle de una máquina virtual en modo móvil.

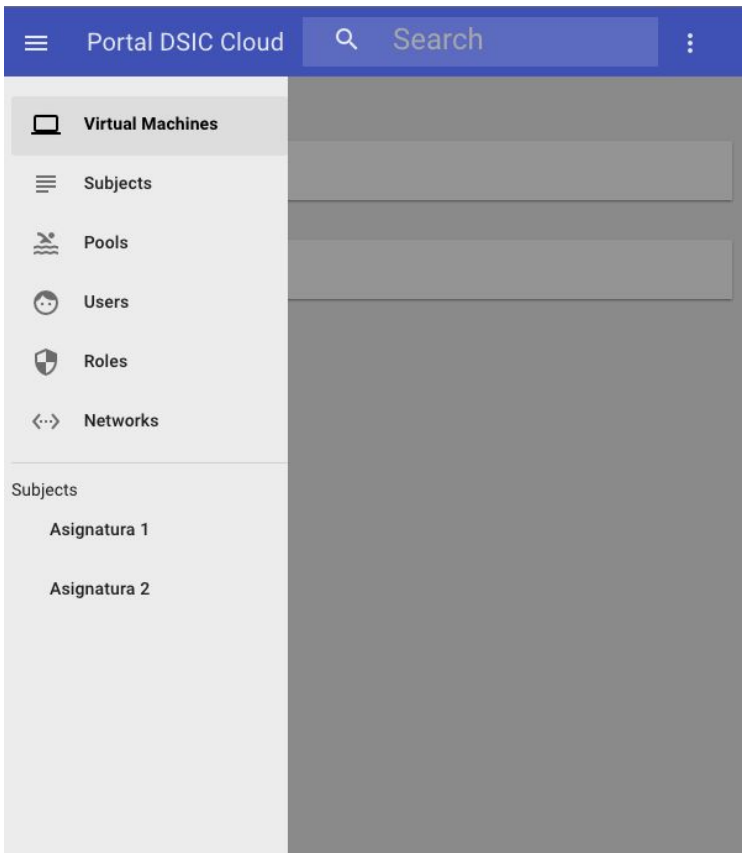


Figura 10. Menú en modo móvil.

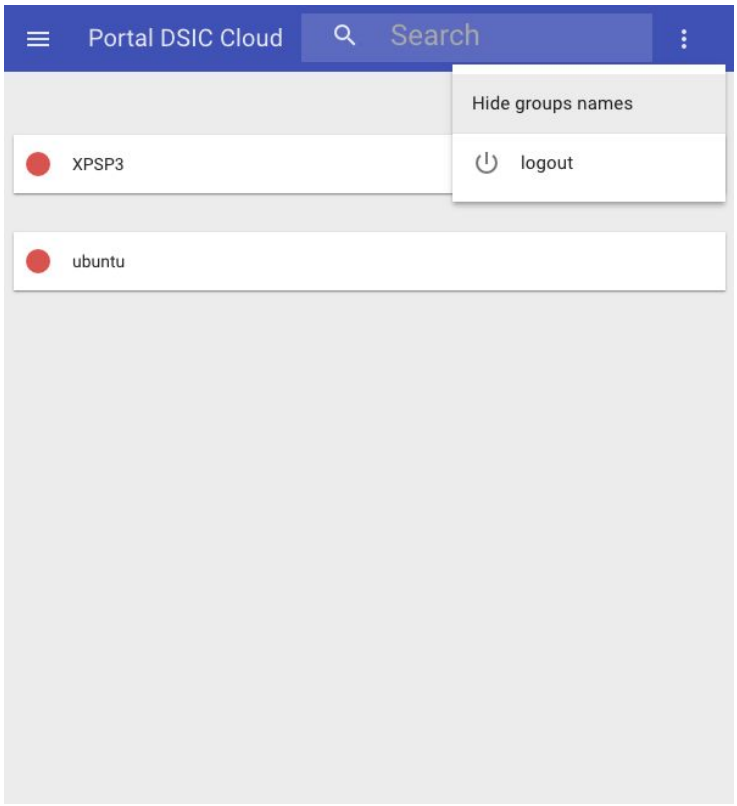


Figura 11. Menú con las opciones de usuario.

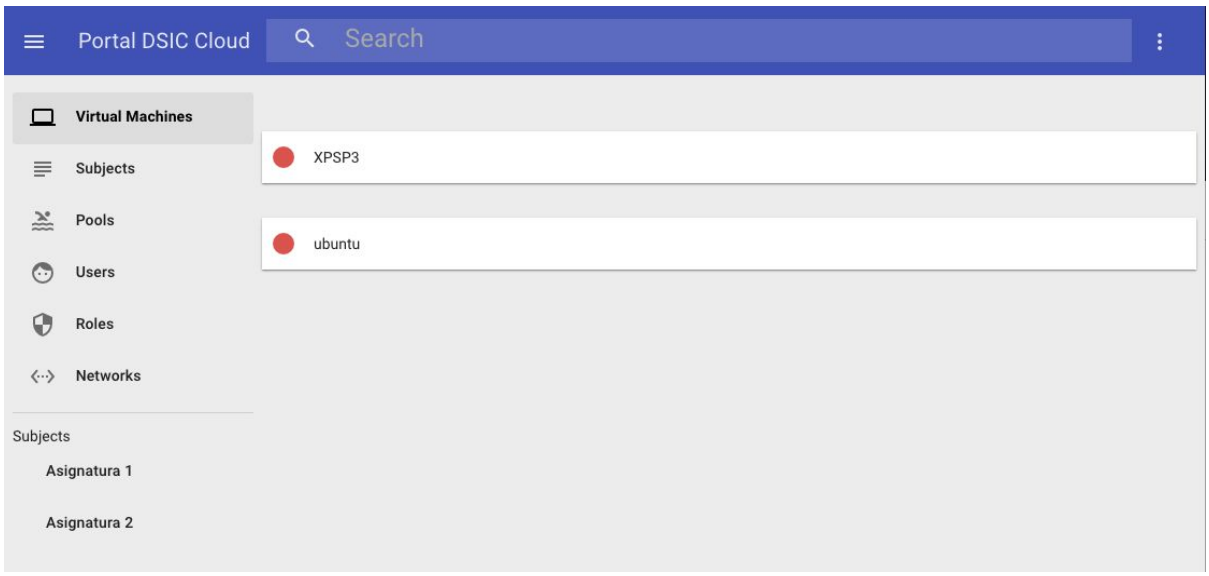


Figura 12. Listado de máquinas virtuales en modo escritorio.

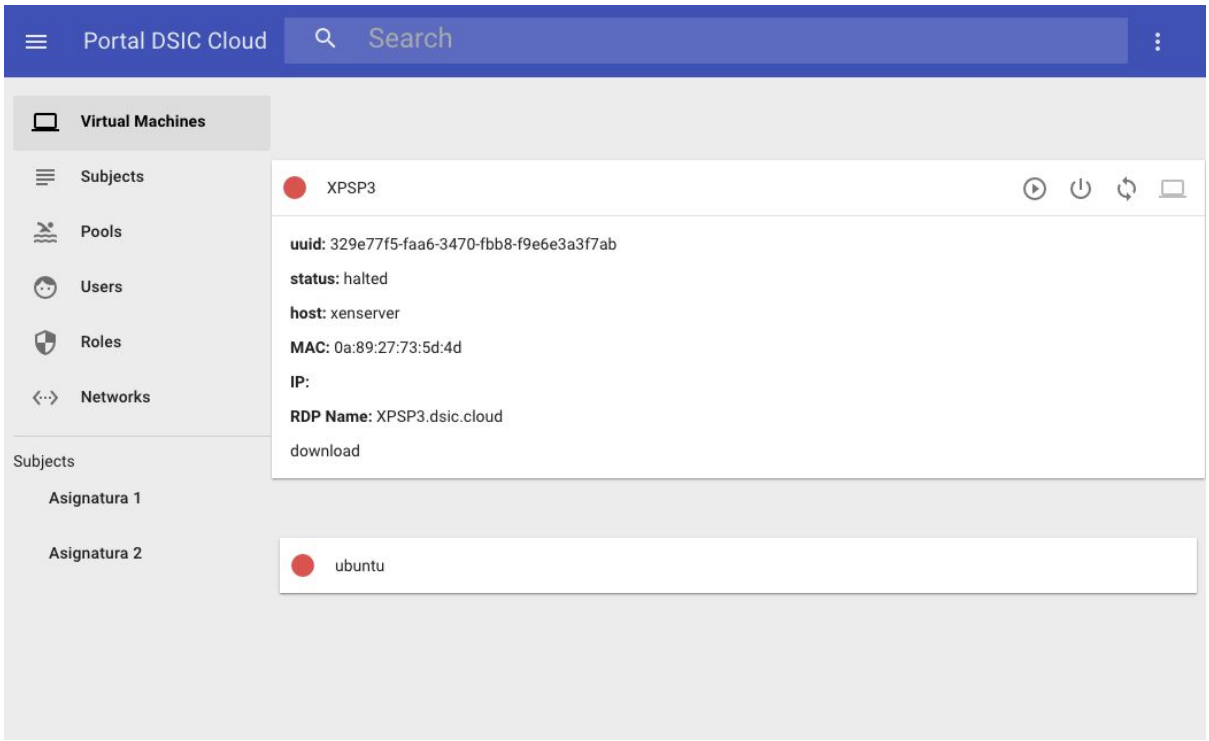


Figura 13. Detalle de máquina virtual en modo escritorio.

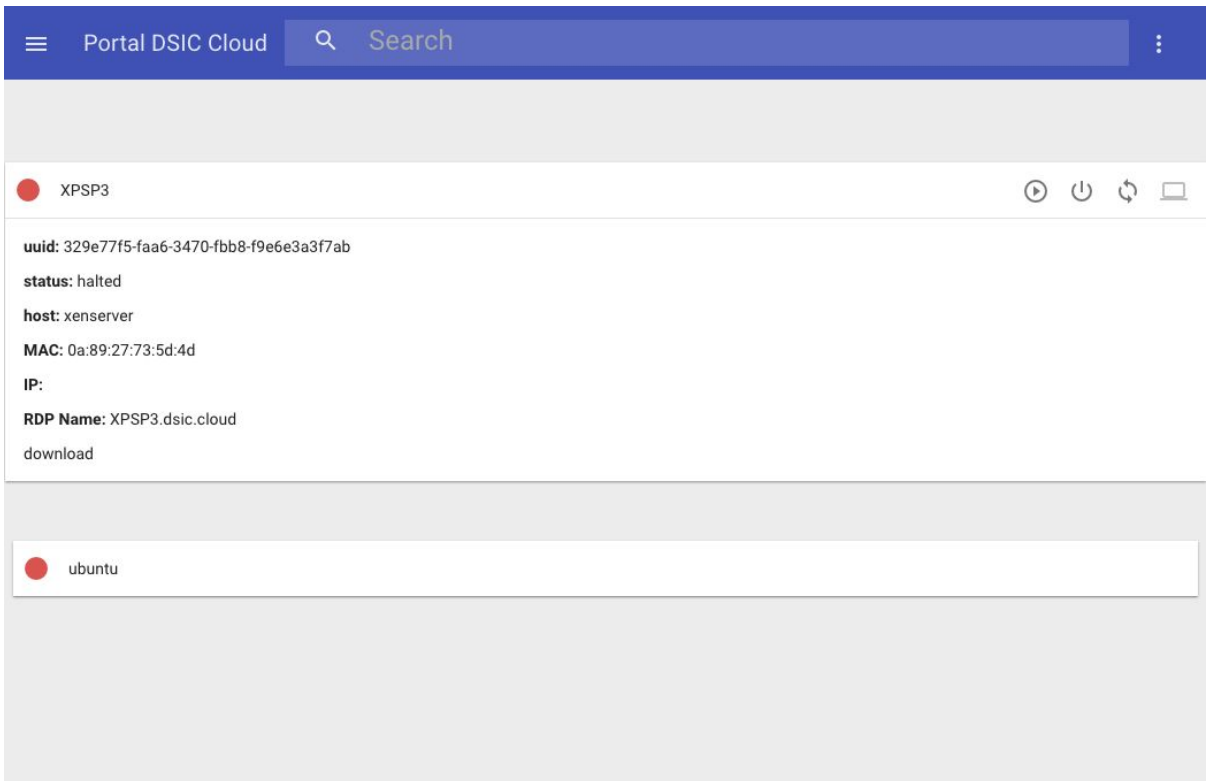


Figura 14. Detalle de máquina virtual en modo escritorio con el menú oculto.

5. Arquitectura de la aplicación

Para entender la estructura final de nuestra aplicación, vamos a describir, con dos ejemplos, cómo interaccionan los distintos componentes de nuestra arquitectura.

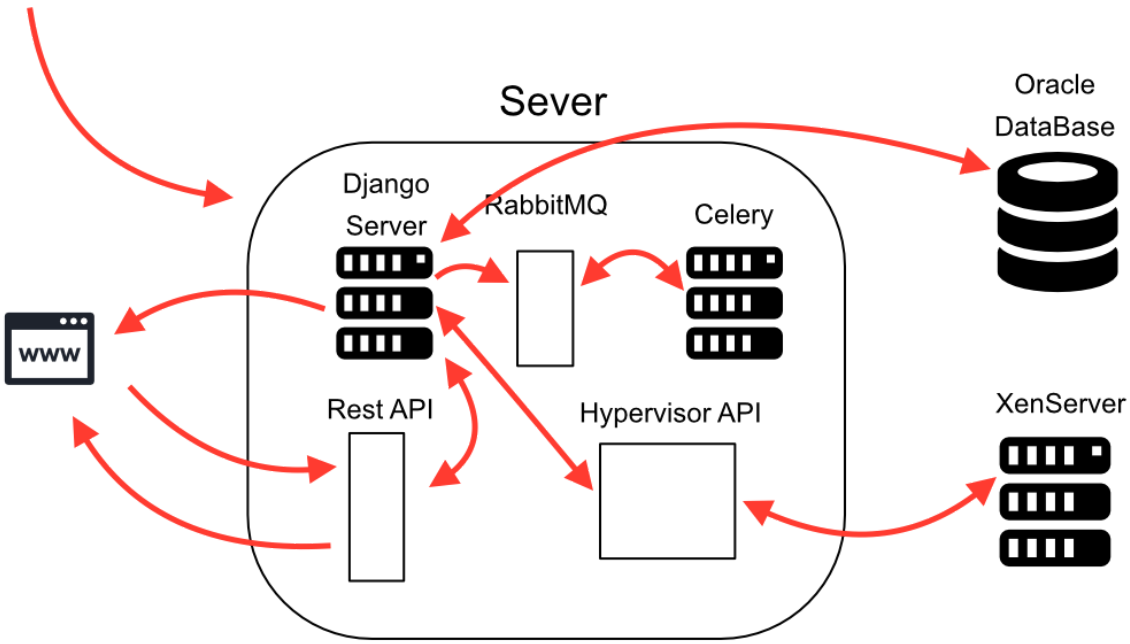


Figura 15. Despliegue actual de la aplicación [17] [18].

El proceso de una petición al servidor es el siguiente (obviado algunos pasos para simplificar, como la entrada por nginx/uwsgi o apache).

Supongamos que, el usuario accede a su listado de máquinas virtuales, estos son los pasos que seguirá la petición en sistema:

- ❏ Nuestro servidor Django recibirá la petición y comprobará que el usuario está logueado mediante la sesión. Si el usuario no está registrado, será redireccionado hacia la pantalla de login.
- ❏ Si el usuario se encuentra registrado, se le devolverá un html con los estáticos asociados.
- ❏ El navegador recibe el html, lo interpreta y ejecuta el javascript asociado.
- ❏ Angular realiza un GET a nuestra API Rest para obtener el listado de máquinas virtuales. En la petición, por ejemplo, indicamos que queremos la primera página de veinte máquinas virtuales.
- ❏ El servidor recibe la petición y comprueba que el usuario está registrado, en caso de no estar logueado devolverá un código html 403.

- ❏ Si está logueado consultaremos en la base de datos que máquinas virtuales tiene asociadas dependiendo de su rol. Por ejemplo, si es un alumno, recibirá las máquinas que tenga asociadas, en cambio si se trata de un técnico, recibirá toda las máquinas virtuales del sistema.
- ❏ Una vez obtenidas las máquinas virtuales, se filtran de la uno a la veinte y se serializa a JSON.
- ❏ Una vez serializados, se envían al frontend que los renderiza.

En el ejemplo anterior hemos visto el proceso correspondiente a solicitar un listado, que sería un proceso de lectura. Si quisiéramos realizar una acción como apagar una máquina virtual, el proceso sería ligeramente distinto.

- ❏ Al clicar sobre la acción, en este caso detener la máquina virtual, se realiza un PATH a nuestra API Rest, indicando la máquina virtual.
- ❏ Una vez superada la validación de la petición, se crea una tarea, se actualiza el estado de la máquina virtual y se devuelve un 200 como que la acción se ha realizado correctamente.
- ❏ El servidor envía la tarea a RabbitMQ que la añade a la cola y la envía a Celery.
- ❏ Una vez llega a Celery, se crea una tarea de apagado en XenServer, el cual le devolverá un identificador de tarea.
- ❏ Con el identificador de tarea, creamos otra tarea con la que comprobamos el estado de ésta periódicamente hasta que finalice.
- ❏ Una vez finalizada actualizamos el estado de la base de datos según el resultado.
- ❏ Mientras dura el proceso, el frontend preguntará de manera periódica al backend si el estado de la máquina virtual ha sido modificado.

6. Escalabilidad de la aplicación

En caso de que la aplicación necesitase escalar tenemos dos opciones:

- Escalar de forma vertical.
- Escalar Horizontalmente.

6.1 Escalar de forma vertical

En este caso actualizaremos el equipo en el que se encuentra el servidor, por ejemplo, añadiendo un procesador más potente, más ram o en caso de estar desplegado en una máquina virtual asignarle más cores o más memoria virtual.

6.2 Escalar de forma horizontal

Este caso es más complicado, ya que no todas las aplicaciones están preparadas para ello. Consiste en agregar más equipos. Un ejemplo sería añadir otro servidor.

Contamos con varias formas de escalar horizontalmente. Como la base de datos ya se encuentra en otra máquina, el primer paso a la hora de necesitar más recursos sería mover RabbitMQ y Celery a otra máquina, quedando el sistema de la siguiente manera:

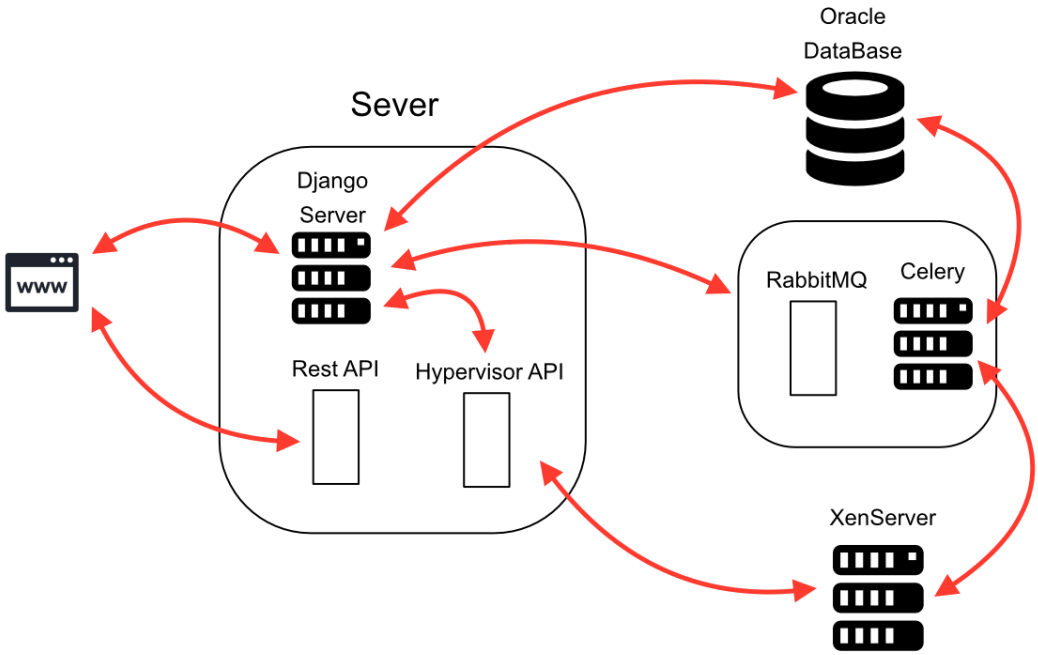


Figura 16. Despliegue con RabbitMQ y Celery separados.

En caso de necesitar más recursos tendríamos que duplicar el servidor django. Para ello tendríamos que crear un Balanceador de carga (Load Balancing). Esto es debido a que necesitamos un único punto de entrada a nuestra aplicación, si crearemos varios servidores sien el “paraguas” del balanceador, cada uno tendría su propia dirección y solo uno podría contestar a las peticiones. Añadiendo con el balanceador de carga el sistema quedaría de la siguiente manera:

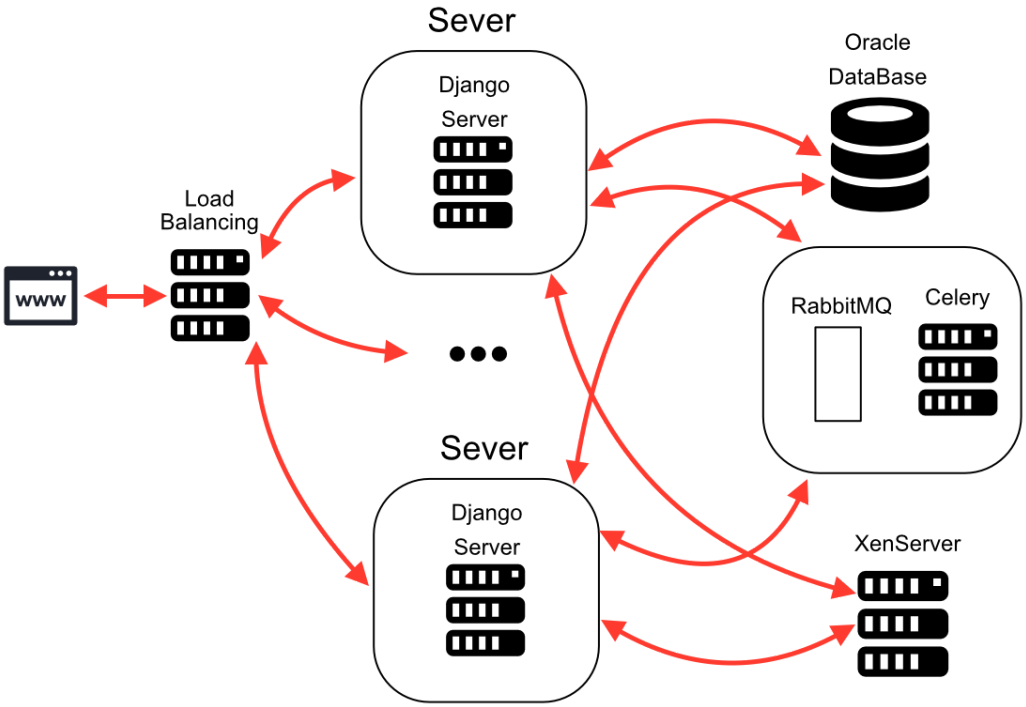


Figura 17. Despliegue con varios servidores.

En caso de necesitar escalar más la aplicación se podrían añadir más servidores.

7. Código fuente

El proyecto se encuentra disponible en un repositorio git privado dentro del DSIC:

https://mygit.dsic.upv.es/developers/portal

La idea barajada en un primer momento fue que el proyecto estuviese en un repositorio público y cualquier usuario pudiera acceder a él para adaptarlo a sus necesidades y aprovechar las ventajas del código libre. No obstante, conforme avanzaba el proyecto, la dependencia sobre la infraestructura del DSIC fue aumentado, desde los usuarios, los roles, a cómo se crean las máquinas virtuales. Esto supone que el proyecto sea menos genérico y por el contrario, más centrado en cubrir las necesidades concretas del DSIC. No obstante, esto no impide que en un futuro sea interesante realizar un fork del proyecto y eliminar todas estas dependencias para que la comunidad pueda enriquecerse de nuestro trabajo.

8. Trabajo Futuro

Como mencionamos al principio de esta memoria, esta aplicación no está terminada. Se trata solo de la base para lo que espero sean muchos años de desarrollo de funcionalidades, donde noveles y expertos se valgan del trabajo aquí depositado.

A continuación, enumeramos algunas de las tareas que se han quedado en el tintero y cuyo desarrollo consideramos sería interesante.

- XenServer proporciona un api para obtener estadísticas de las distintas máquinas virtuales, pues mostrar estas estadísticas en tiempo real en la página de cada máquina virtual sería algo que seguro muchos usuarios agradecerían.
- El sistema actualmente se encuentra en ingles, pero esta preparado para traducirse a cualquier idioma. Para ello, faltaría crear los textos en los distintos idiomas.
- Se puede gestionar el sistema de tareas desde el frontend, preguntando por la tarea en vez de por el estado de la máquina virtual. De esta forma podríamos obtener más información en caso de error.
- Considerar la actualización de la librería a Angular 2 cuando su desarrollo esté finalizado. Según los análisis de las pruebas realizadas sobre la versión alpha de la librería, el rendimiento mejora considerablemente.

9. Conclusiones

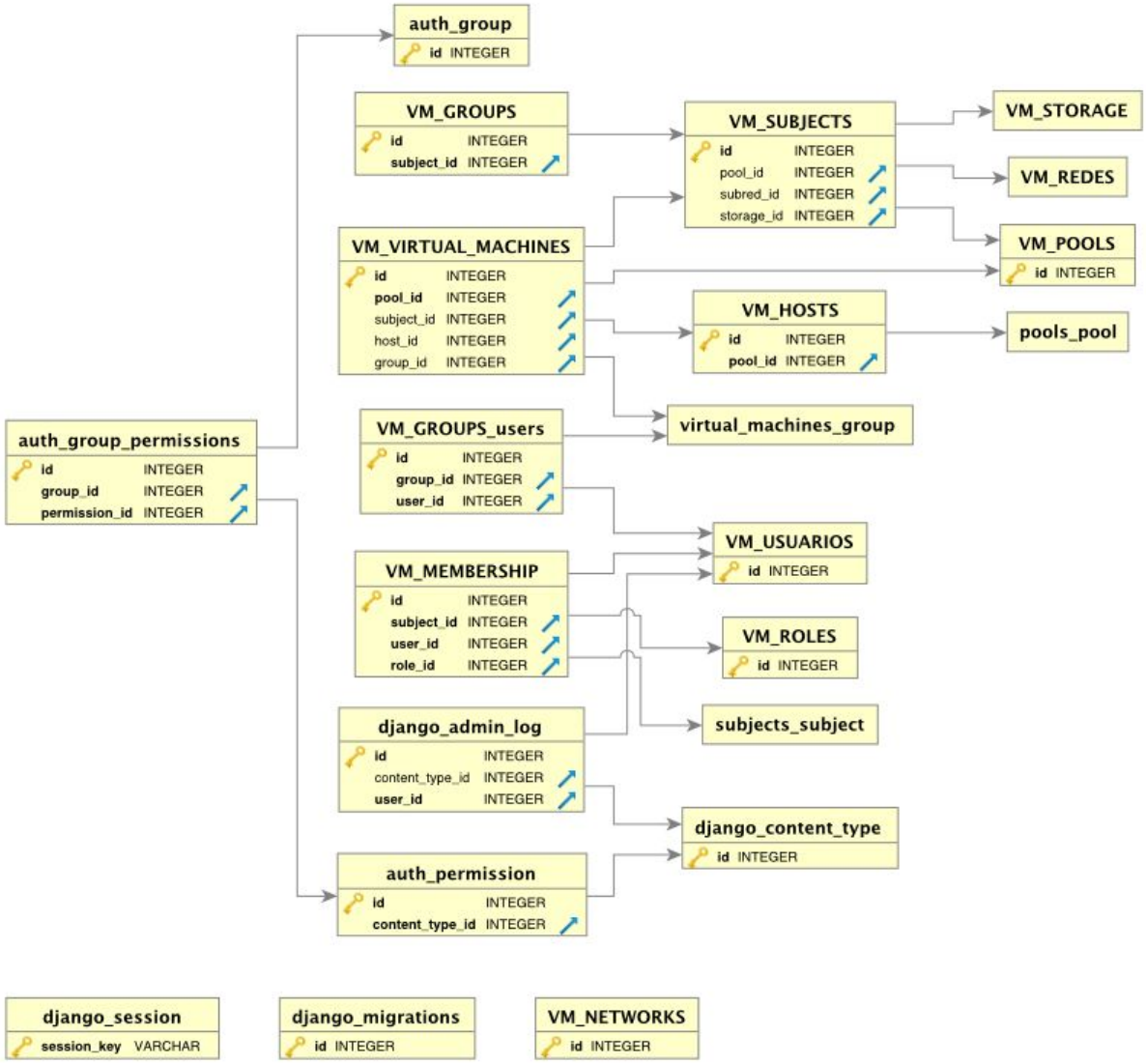
Durante el transcurso de este proyecto hemos podido observar las bondades de aplicar las metodologías ágiles. En un proyecto con una gran incertidumbre inicial, como era el caso, si hubiéramos cometido el error de emplear una metodología tradicional para su desarrollo, habríamos tenido que invertir un gran esfuerzo en la obtención y documentación de requisitos y probablemente, en el momento de la entrega de la aplicación resultaría no adecuarse a las expectativas del cliente.

Utilizando metodologías ágiles hemos podido adaptarnos a las nuevas necesidades del cliente, independientemente del momento del desarrollo en el que han ido surgiendo. De la misma forma, hemos podido solucionar los problemas a medida que éstos se han ido planteando. Al mantener un continuo contacto con el cliente, éste se ha podido sentir más identificado con el proyecto y ha podido ir priorizando las tareas que eran más importantes para él. Además, hemos podido introducir poco a poco a nuevos integrantes en el equipo de desarrollo, sin experiencia en la aplicación, ni en desarrollo web. Estos integrantes han ido realizando tareas más complicadas, iteración tras iteración, y esto ha permitido que a la entrega de este documento ya fueran capaces de administrar la plataforma y crear nuevas funcionalidades.

Aún queda mucho trabajo por realizar, pero nuestro objetivo no era finalizar el proyecto sino crear la base de una aplicación mantenible y escalable. Y ese objetivo, podemos decir con orgullo, que ha sido alcanzado. La aplicación cubre todas las funcionalidades del anterior portal desde un punto de vista moderno y más productivo. Además, añade nuevas funcionalidades que anteriormente debían hacerse manualmente por los técnicos, lo que supone un ahorro de tiempo y trabajo para los mismos y mayor comodidad y rapidez para los usuarios.

10. Apéndices

Apéndice A. Estructura de la Base de Datos.



Apéndice B. Manual de la aplicación

De manera general, en toda la aplicación, el campo de búsqueda de la parte superior, permite filtrar entre las diferentes listas del apartado en concreto en el que se esté. A la derecha del campo de búsqueda, existe un menú desplegable en el que se pueden ocultar los nombres de los grupos de máquinas para el apartado de “Virtual Machines” o desconectarse de la aplicación.

MANUAL ROL USUARIO

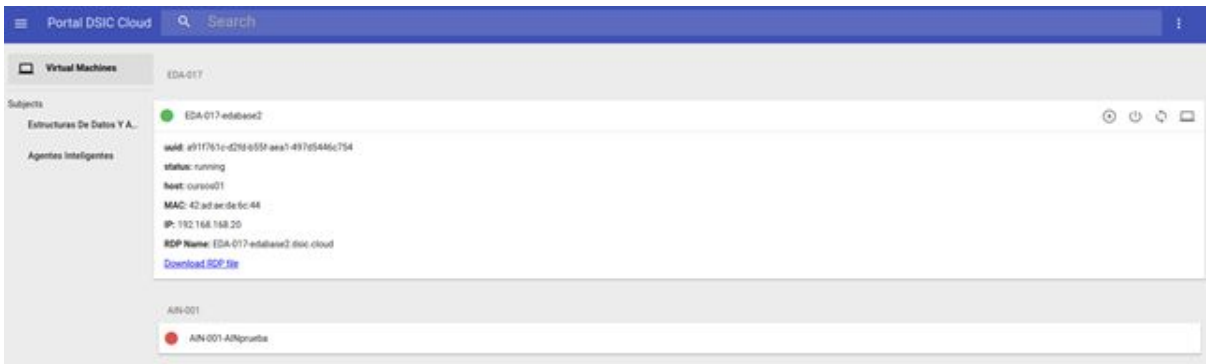
Tras hacer login, se mostrará en pantalla una lista de las máquinas virtuales de las que dispone.

Para cada máquina virtual, el usuario podrá encontrar información sobre la misma, como su nombre, IP, MAC, su estado (verde para encendido y rojo para apagado), etc. Además se tendrá acceso a descargar un fichero RDP para arrancar la máquina virtual desde una aplicación de escritorio remoto.

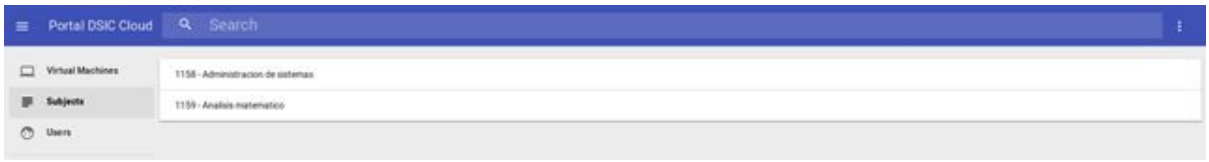
En cada máquina virtual se pueden llevar a cabo las siguientes acciones:

- Encender la máquina virtual
- Apagar la máquina virtual
- Sincronizar estado de la máquina virtual
- Acceso a la consola VNC para utilizar la máquina virtual

Además, en el menú de la izquierda disponemos de una lista de las asignaturas en las que tenemos máquinas virtuales asignadas. Haciendo clic en una asignatura filtrará las máquinas virtuales de dicha asignatura. Este menú se puede ocultar si pulsamos en el botón junto a 'Portal DSIC Cloud'.

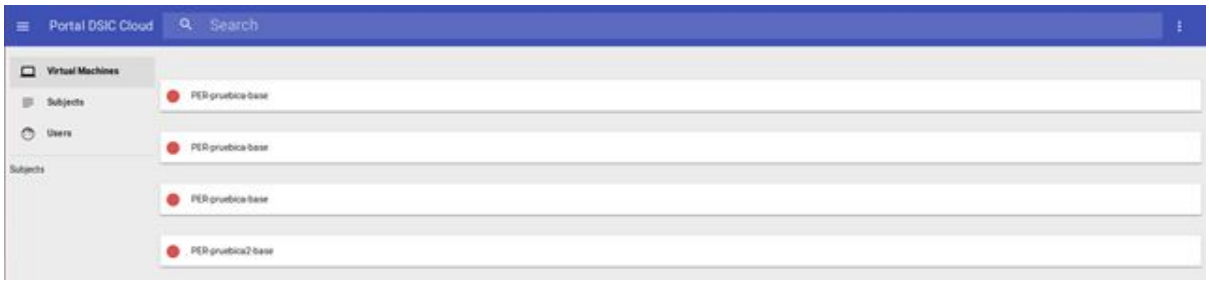


MANUAL ROL PROFESOR



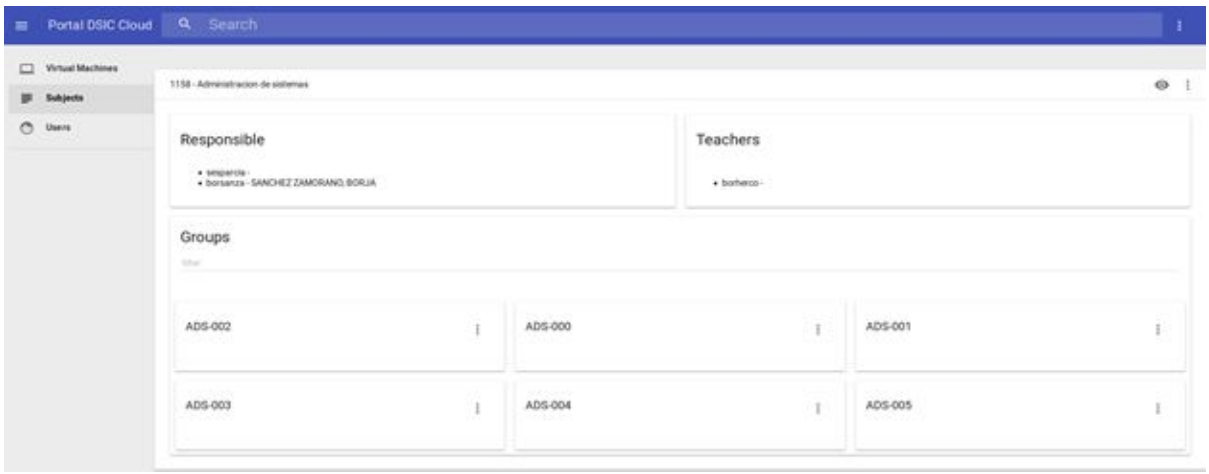
Menú “Virtual Machines”

- Un profesor puede realizar las mismas acciones que un “alumno” en este menú.



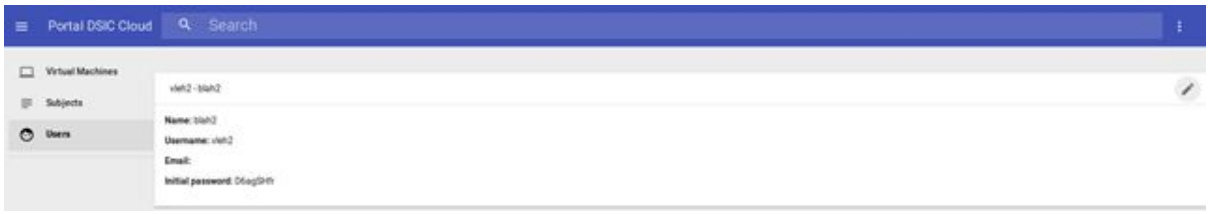
Menú “Subjects”

- Se muestra un listado de las asignaturas en las que el usuario es profesor.
- Sobre la misma barra de la asignatura se ofrecen 2 funcionalidades:
    1º Esconder las máquinas virtuales para que no aparezcan en la lista del menú anterior.
    2º Apagar todas las máquinas virtuales de la asignatura.
- Al clicar sobre una asignatura se muestra la siguiente información:
    1º Responsables de la asignatura
    2º Profesores de la asignatura
    3º Grupos de máquinas de la asignatura, el profesor puede editar los alumnos asignados a cada grupo.



Menú “Usuarios”

-En este apartado se puede consultar la información relativa a los usuarios de las asignaturas asignadas al profesor, también se muestran los alumnos “genéricos” (pertenecientes a cursos).

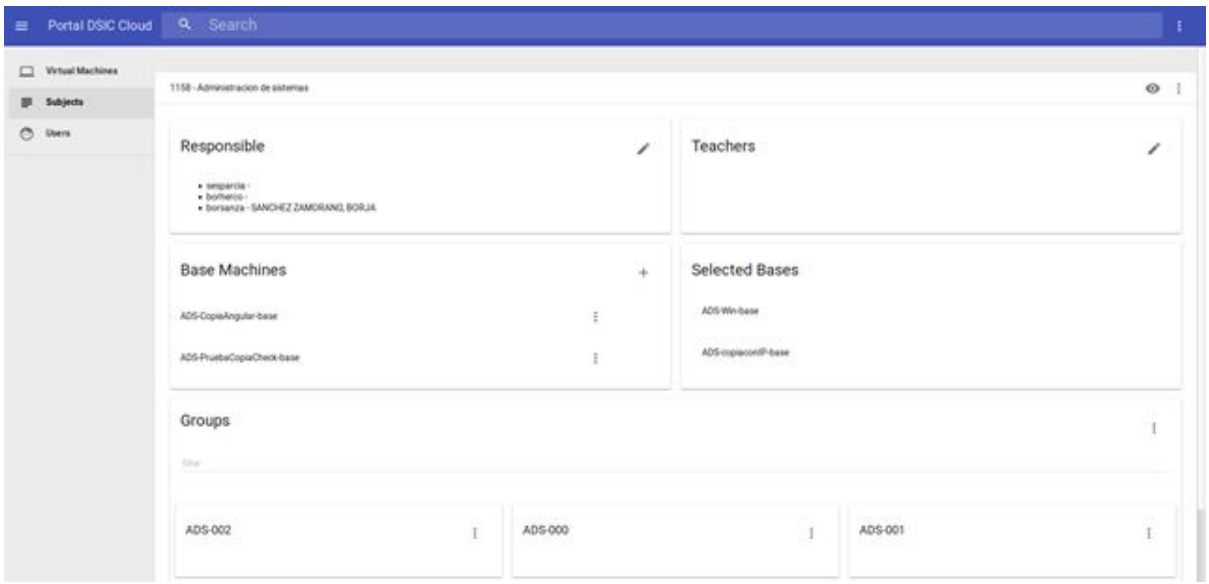


MANUAL ROL RESPONSABLE

Menú “Virtual Machines”

- Un responsable puede realizar las mismas acciones que un “alumno” o “profesor”en este menú.

Menú “Subjects”



- Un responsable puede realizar además de las mismas acciones que un “alumno” o “profesor” las siguientes:

-Puede editar tanto los responsables como los profesores de la asignatura.

-El responsable también puede crear “máquinas base”, a través de unas plantillas (“templates”) preestablecidas, o creando copias de máquinas base ya creadas.

-Asimismo puede cambiar el nombre de las “máquinas base” ya creadas o borrar las existentes.

-Una vez exista una máquina base y si no hay grupos ya creados, pueden añadirse al grupo de “bases seleccionadas”. Para ello se utiliza el botón de

añadir del apartado “Selected Bases”, se eligen los cambios deseados y se salva.

-Por último se pueden crear nuevos grupos de máquinas seleccionando “Add groups” en el menú contextual del apartado. Esta opción se puede ejecutar de dos maneras, o bien introduciendo el número de grupos a crear, o bien importando un fichero “xlsx”. Una vez añadidos los grupos, se bloquea la opción de seleccionar bases. Para volver a crear un “Grupo Base” se debería de borrar todos los grupos y volver a seleccionar las bases o avisar a un técnico para que realice la operación sin necesidad de borrar los grupos.

-En cuanto al apartado de “Groups” el responsable además de editar los usuarios de cada grupo, puede borrar los grupos de manera individual o elegir borrar todos los grupos de la asignatura en el menú de “Groups”.

Menú “Usuarios”

-En este apartado se puede consultar y editar la información relativa a los usuarios de las asignaturas de las cuales se es responsable, así como los alumnos de cursos impartidos por el responsable actual, se muestra la misma información que para el rol de “Profesor”.

MANUAL ROL TÉCNICO

Menú “Virtual Machines”

- Un técnico puede realizar las mismas acciones que un “responsable”en este menú.

Menú “Subjects”

- Un “Técnico” puede realizar las mismas acciones que un “Responsable”en este menú.
- Además un “Técnico” tiene la opción de “Apply changes”, en el menú contextual de “Groups” con el que puede aplicar los cambios de bases seleccionadas sobre los grupos de máquinas. Dicho cambio en el grupo de bases seleccionadas también es llevado a cabo por un usuario con el rol “Técnico”. Por seguridad estos cambios deben realizarse de uno en uno, es decir, cada vez que se realice un cambio en “Selected Bases” se deben aplicar los cambios.

Menú “Pools”

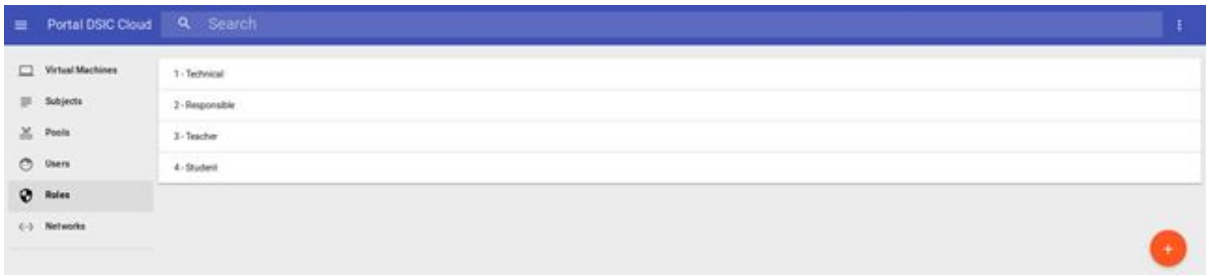
- En este menú se pueden ver y editar los datos de las “pools” y además sincronizarlas con las bases de datos para corregir disparidades.
- Con el botón de añadir se pueden crear nuevas “pools” o pasando puntero por encima se dará acceso al icono que permite la sincronización de todas las “pools”.

Menú “Usuarios”

- Un técnico puede realizar las mismas acciones que un “Responsible”en este menú.

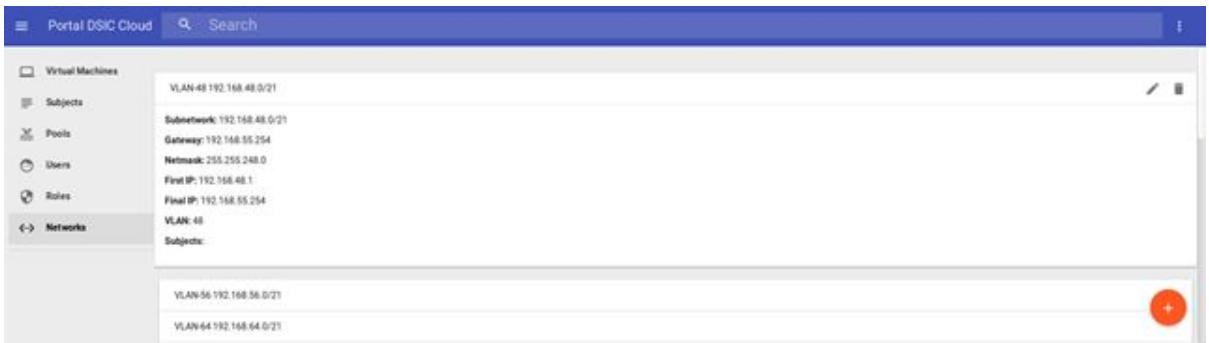
Menú “Roles”

- Desde este menú se pueden crear o editar los roles existentes para los usuarios.



Menú “Networks”

- Aquí se pueden visualizar las redes existentes, visualizar sus datos y añadir o borrar redes.



11. Referencias

[1] Desarrollo tradicional vs desarrollo ágil.
http://www.optimusinfo.com/traditional-vs-agile-software-development/

[2] Metodología de desarrollo de software.
https://es.wikipedia.org/wiki/Metodolog%C3%ADa_de_desarrollo_de_software

[3] Manifiesto ágil.
https://es.wikipedia.org/wiki/Manifiesto_%C3%A1gil
http://www.agilemanifesto.org/iso/es/

[4] Trello.
https://trello.com/

[5] Material design.
https://material.google.com/

[6] Material Design Lite.
https://getmdl.io/started/

[7] Angular Material.
https://material.angularjs.org/latest/

[8] Material Icons.
https://design.google.com/icons/

[9] XenServer.
http://xenserver.org/

[10] XenCenter.
http://xenserver.org/partners/developing-products-for-xenserver/21-xencenter-development/88-xc-dev-home.html

[11] Python XenAPI.
https://pypi.python.org/pypi/XenAPI

[12] XenAPI compatible con python 3.
https://github.com/saruba/xen-api

[13] Django Rest Framework.
http://www.django-rest-framework.org/

[14] Celery.
http://www.celeryproject.org/

[15] RabbitMQ.
https://www.rabbitmq.com/

[16] noVNC.
https://kanaka.github.io/noVNC/

[17] Imágen base de datos.
https://commons.wikimedia.org/wiki/File:Octicons-database.svg

[18] Imágen servidor.
https://commons.wikimedia.org/wiki/File:Octicons-server.svg