TRABAJO DE FÍN DE TÍTULO

Grado en Ingeniería Informática – Tecnologías de la Información

AUTOMATIZACIÓN DEL DESPLIEGUE DE MÁQUINAS VIRTUALES PARA LOS LABORATORIOS DE LA ESCUELA DE INGENIERÍA INFORMÁTICA

Autor

Alberto Sosa García

Tutores

Dr. Alexis Quesada Arencibia – Ciencias de la Computación e Inteligencia Artificial

Dr. Carmelo Rubén García Rodríguez – Ciencias de la Computación e Inteligencia Artificial

Diciembre 2018 – Las Palmas de Gran Canaria

Descripción del documento

Trabajo de Fin de Título en el Grado de Ingeniería Informática, con intensificación en Tecnologías de la Información, de la Universidad de Las Palmas de Gran Canaria presentado por el alumno:

Alberto Sosa García

Título del proyecto

Análisis, diseño e implementación de una herramienta software para el despliegue de máquinas virtuales en los laboratorios de la Escuela de Ingeniería Informática.

Tutores

Dr. Alexis Quesada Arencibia
Dr. Carmelo Rubén García Rodríguez

Agradecimientos

Agradezco el tiempo invertido a mis tutores Dr. Alexis Quesada y Dr. Rubén García, por sacar tiempo y dedicación para la resolución de dudas y apoyo en el desarrollo de este proyecto.

También agradecer la motivación recibida por parte de los compañeros del Dpto. de Computación Científica y Tecnológica del Instituto Tecnológico de Canarias. En especial la enseñanza y ayuda en el uso de nuevas tecnologías por parte de Rafael Nebot, Jefe de Sección de Programación y tutor de prácticas externas.

Y como no, el apoyo de mi familia para sacar el proyecto adelante y el de mi pareja, Gemma García, por ayudarme en los últimos pasos antes de finalizar este Trabajo de Fin de Título.

Resumen

Los equipos de los laboratorios de la Escuela de Ingeniería Informática y del Departamento de Informática y Sistemas cuentan con dos particiones base: una con un sistema Windows y otra con un sistema Linux. Cada asignatura tiene unos requerimientos a nivel de software, lo que conlleva que el personal del Centro de Cálculo lleve a cabo tareas tediosas en dichas particiones.

Este proyecto pretende llevar a cabo la virtualización de las infraestructuras de los laboratorios para dar soporte a una herramienta de gestión de máquinas virtuales que reemplazarán a los sistemas actuales. Así, esas tareas se resumirán en operaciones básicas con las máquinas virtuales personalizadas para cada asignatura.

Para ello se utilizará software libre para el despliegue de las infraestructuras de virtualización y el desarrollo de la herramienta de gestión. Además, se utilizarán tecnologías vistas en las prácticas externas universitarias.

Abstract

The laboratories systems of the Computer Engineering School and the Department of Computing and Systems have a minimum of two base partitions: one with Windows environment and the other one with Linux environment. Each subject demand software requirements, so the workers of the Data Center must to do tedious tasks on these partitions.

This project pretends to deploy a virtual infrastructure in each laboratory to support the management tool of full customized virtual machines that replaces the actual partitions to transform tedious tasks into basic virtual machines operations.

For this, it will be used open-source software for deployment of the virtual infrastructures and development of the management tool, in addition to the use of technologies seen in the university internships.

Índice de contenidos

Índice de tablas

Tabla de ilustraciones

Introducción

Para que sea posible la docencia en los laboratorios de la Escuela de Ingeniería Informática (EII en adelante) y del Departamento de Informática y Sistemas, los profesores de las diversas asignaturas del grado y másteres exigen, al comienzo de un nuevo curso académico y/o cuatrimestres, ciertos requisitos software, como por ejemplo la instalación de distintas versiones de un mismo software, entornos de desarrollo, particiones dedicadas a asignaturas de sistemas, etc. Esta forma de trabajo presenta dos inconvenientes importantes que tienen que ver con la sobrecarga e inconsistencia de los sistemas, ya que todos los equipos de los laboratorios, salvo excepciones, cuentan con dos particiones base: una con un entorno Windows y otra con un entorno Linux. Por otro lado, existe el problema de compartir un mismo laboratorio para distintas asignaturas, por lo que, si un alumno de forma accidental corrompe la partición actual o, en el peor de los casos, el equipo entero, afectaría al trabajo y desarrollo de las distintas asignaturas de otros compañeros. Además, siguiendo la metodología actual, cualquier solicitud de cambio en los requisitos software en una asignatura implica una nueva clonación de todos los discos duros para crear uniformidad en los equipos de los laboratorios. Como se puede observar, son tareas que tienen un alto costo en tiempo de gestión si se tiene en cuenta el elevado número de equipos con los que cuentan los distintos laboratorios de la escuela.

Para agilizar los procesos de gestión, este proyecto pretende llevar a cabo la virtualización de las infraestructuras de los laboratorios, de tal forma que esas particiones base sean sustituidas por máquinas virtuales. De esta manera, aquellas tareas tediosas que realiza el personal del CCDIS, pasarán a ser operaciones básicas con las máquinas virtuales. Así se consigue tener en los equipos un único sistema base de trabajo Linux con aquellos paquetes software imprescindibles para un entorno de escritorio, además de contar con las tecnologías de virtualización necesarias para ejecutar máquinas virtuales completamente personalizadas en base a las necesidades de las asignaturas.

Además, la infraestructura de virtualización en los equipos se verá respaldada por una herramienta de gestión a medida según requisitos del personal del CCDIS, que hará posible la gestión y administración de las máquinas virtuales: creación de máquinas virtuales, generación de plantillas, despliegue en los laboratorios, etc., de forma que la gestión de los equipos pase de ser distribuida a centralizada.

Estructura del documento

Este documento se encuentra estructurado en varios capítulos, en los que se detalla el desarrollo de los siguientes apartados:

- Capítulo 1: Estado actual y objetivos iniciales
  Una vez descrito el problema, se detalla el proceso actual que sigue el personal del CCDIS. Además, se realiza un análisis de las alternativas existentes, para presentar posteriormente las distintas propuestas de valor que ofrece el proyecto.

- Capítulo 2: Competencias cubiertas
  En este capítulo se enumeran las distintas competencias cubiertas con el desarrollo general del proyecto, en las que se incluyen las competencias específicas.

- Capítulo 3: Aportaciones
  A pesar de las ventajas técnicas que aporta el proyecto, también realiza aportaciones en distintos aspectos, como es en el ámbito profesional y personal.

- Capítulo 4: Normativa y legislación
  Todo proyecto se ve afectado y tiene que ser desarrollado en función de la normativa y aspectos legales vigentes. Además, se analiza el impacto que tienen las licencias software utilizadas para el desarrollo.

- Capítulo 5: Metodología de trabajo y planificación
  En este capítulo se comenta y justifica la metodología de trabajo escogida. Se detalla la planificación inicial del proyecto y su evolución durante el desarrollo.

- Capítulo 6: Tecnologías, herramientas e infraestructura hardware
  Para el desarrollo de este proyecto se ha utilizado una serie de tecnologías y herramientas escogidas bajo unos ciertos criterios. Además, se detalla la infraestructura hardware y entornos utilizados para las fases de desarrollo y preproducción de la solución.

- Capítulo 7: Análisis
  Como su título indica, se analiza el problema al que da solución el proyecto. Esto abarca desde la justificación de las tecnologías escogidas, hasta los actores y requisitos software que están presentes.

- Capítulo 8: Diseño

  Una vez identificados los actores y requisitos del sistema, se presenta la arquitectura del sistema: componentes de la infraestructura de virtualización y de la herramienta de gestión, estructura de datos, etc.

- Capítulo 9: Desarrollo

  En este capítulo se explica en detalle el desarrollo de la herramienta de gestión web. Se comentarán puntos como la estructura del código fuente, control de acceso, entre otros.

- Capítulo 10: Resultados, conclusiones y trabajo futuro

  Una vez desarrollada la solución, se evalúan sus resultados para sacar conclusiones de las propuestas de valor frente a las alternativas y así obtener aspectos a mejorar en un trabajo futuro.

- Bibliografía

  Se realiza un listado de los distintos documentos, sitios web, bibliografía, enlaces de interés, etc. consultados para el desarrollo del proyecto.

- Anexos

  En este apartado se incluye información complementaria del proyecto:
  - Anexo I: instalación y configuración de la infraestructura y aplicación.
  - Anexo II: guía básica del usuario.
  - Anexo III: recursos RESTful.

1. Estado actual y objetivos iniciales

1.1. Estado actual

En la actualidad, el personal del CCDIS cuenta con una serie de pasos bien detallados para la instalación y configuración de los dos sistemas base en los equipos.

Se comienza con la selección de un equipo de partida para la configuración de la imagen Windows que se carga a través de Clonezilla y se conecta a la red de datos para su activación. Luego, se configuran las cuentas genéricas y se le da nombre al equipo siguiendo un criterio nemotécnico. Por ejemplo, para un Dell Optiplex 790 del laboratorio 2-3, se le pone como nombre de equipo "OPT790-L23", para un Asus Essential del laboratorio 1-2, "ASUSES-L12", etc.

A partir de aquí, se comienza con las actualizaciones del sistema: Windows, Office y software básico. Además, se instala el software solicitado por el profesorado para las distintas asignaturas. Lo ideal es que esté disponible para todos los laboratorios, pero en algunos casos no es nada práctico debido a la complejidad en la instalación: necesidad de cierto hardware para instalar los drivers, conflicto de versiones de un mismo software, necesidad de permisos de administración para algunos procesos, etc.

Posteriormente, se cargan los ficheros para las tareas administrativas a través de la unidad de red "\serdisiniciolab". El fichero "InicioInicio.xml" consta de una serie de acciones llevadas a cabo tras el primer arranque del sistema después de la clonación: activa Windows y Office, pone el nombre real del equipo y reinicia. El fichero "Iniciosc-admin.xml" es una tarea que se ejecuta siempre que se inicia el sistema Windows y permite realizar tareas administrativas o de instalaciones desatendidas a posteriori. Por último, el fichero "Inicioapagar.xml" incorpora la tarea de apagado del equipo todas las noches a las 22:00 por si este se encontrase encendido. Una vez terminada la copia de las tareas, se apaga el equipo y se procura no iniciar el sistema Windows para que no se ejecuten las tareas asociadas al primer arranque del sistema ya que modificaría la imagen. Por último, se utiliza un servidor Clonezilla en modo *multicast* para realizar la clonación de los equipos de forma distribuida.

Para la instalación del sistema Linux, se inician los equipos vía red por PXE. Realmente, se trata de una instalación desatendida de la distribución CentOS 6.4 de 64-bits, por lo que, una vez finalizada la instalación, se apaga el equipo y se crea la imagen del disco con la herramienta Clonezilla, la cual genera una copia exacta del disco en un disco externo con el que luego se realiza la clonación en los equipos de los laboratorios.

1.2. Objetivos

En primer lugar, con este proyecto se pretende modificar la infraestructura actual, en la que se convierten los equipos de los laboratorios en host anfitriones capaces de ejecutar, como mínimo, las dos máquinas virtuales en sustitución de las particiones base actuales. Así, se logra cambiar la gestión actual de los equipos de los laboratorios, cambiando la infraestructura distribuida a una centralizada en la que el personal del CCDIS sea capaz de gestionar de forma eficiente cada equipo a través del despliegue simultáneo de las máquinas virtuales desde un servidor de virtualización.

Para realizar las labores de gestión, se facilitará una herramienta de gestión totalmente desarrollada a medida en función de los requisitos del personal. Esta consiste en una aplicación web accesible desde la red interna apropiada para dichas tareas.

Por otra parte, también se pretende familiarizar a los alumnos de los cursos primerizos con entornos de trabajo Linux y virtualización. Así se consigue que los estudiantes alcancen un cierto grado de madurez en este tipo de sistemas que usan asignaturas de cursos superiores, para superarlas sin grandes dificultades.

Así mismo, la realización del Trabajo de Fin de Grado supone una gran labor de las distintas fases que se pueden abordar en un proyecto real de gestión IT en el mundo laboral, de forma que más que una solución, se puede asimilar como una primera toma de conceptos ante un trabajo futuro.

1.3. Alternativas existentes

Por lo general, las grandes organizaciones tienen sus propios centros de datos que sustentan gran parte de los sistemas de información internos, e incluso externos en algunos casos. Con el paso del tiempo y la necesidad de afrontar nuevos proyectos, la cantidad de recursos y aplicaciones que hay que desarrollar aumenta de forma creciente. Además, estos hay que administrarlos y mantenerlos de forma eficiente con el menor impacto posible en la disponibilidad de los servicios, por lo que se hace inviable mantener tales sistemas sin que los DevOps [1] cuenten con herramientas software para agilizar y automatizar los procesos de configuración y administración de los recursos. [2]

Si trasladamos este escenario a la EII y al DIS, una posible alternativa para la gestión de los activos TI consistiría en aplicar políticas de gestión que usen herramientas tales como “Ansible” [3] o “Chef” [4], las cuales permiten realizar las tareas propias de gestión y administración de forma orquestada y centralizada. Esto se consigue a través de unos scripts en los que se definen las acciones a realizar, paquetes a instalar, configuración de red, administración de los usuarios, etc. A pesar de estar extendido su uso en entornos Linux, también tiene soporte para sistemas Windows. Aunque se aplicara el

uso de estas herramientas en la gestión de los equipos con la infraestructura actual, las particiones seguirán estando sobrecargadas en lo que a software respecta. Además, no da soporte a futuros avances de infraestructura o nuevos servicios.

De forma alternativa a los centros de datos tradicionales, surgen los centros de datos definidos por software (SDDC de sus siglas en inglés). Un SDDC tiene la infraestructura de computación totalmente virtualizada, de forma que, gracias a esa abstracción, se logra una gestión automatizada de los recursos que la componen para aportar elasticidad y escalabilidad a las Tecnologías de la Información [5]. Gracias a este nuevo paradigma, la empresa VMware Inc., ya ofrece un sistema centralizado para la distribución de máquinas virtuales y aplicaciones alojadas en los SDDC: VMware Horizon [6]. Uno de los grandes inconvenientes que presenta esta solución radica en la necesidad de utilizar una infraestructura soportada por productos VMware (ESXi, VSA, NSX, etc.) que requieren de unas licencias elevadas de precio.

Otra posible solución basada en virtualización consistiría en usar estos equipos de los laboratorios como clientes ligeros, dentro de una arquitectura de servidor centralizada en la que se ofrezcan las máquinas virtuales como servicio. Se estaría tratando de implantar el concepto "Infraestructura como Servicio" (*IaaS – Infrastructure as a Service*) dentro de la infraestructura actual, lo cual resulta inviable debido al equipamiento de redes y servidores con los que cuenta actualmente la EII y el DIS.

Sin embargo, debido a las condiciones que se han impuesto para el desarrollo de este proyecto, se utilizarán tecnologías de virtualización para la infraestructura y de desarrollo web para la herramienta de gestión utilizando Libvirt como nexo entre ambas. Este conjunto de utilidades de virtualización denominado Libvirt, que se detallará posteriormente, tiene soporte para una multitud de hipervisores *opensource*, entre los que destacan Xen, Bhyve y KVM/QEMU. Estos tres hipervisores son muy similares en su visión lógica, ya que "convierten" el *kernel* del sistema operativo en un hipervisor a través de la incorporación de una serie de módulos, para que los recursos virtuales tengan acceso directo al hardware. [7]

Xen es un hipervisor desarrollado por la Universidad de Cambridge que permite alcanzar virtualización de alto rendimiento sin requerir de hardware especializado, excepto las extensiones VT o Pacífica de los procesadores Intel o AMD, respectivamente. [8]

Por otra parte, Bhyve es un hipervisor que soporta la mayoría de los procesadores Intel y AMD y que se ejecuta en sistemas FreeBSD, por lo que para este proyecto queda totalmente descartado ya que se utiliza un sistema CentOS. [9]

Por último, KVM es una solución para implementar virtualización completa con Linux, ya que el componente para el núcleo está incluido en Linux desde la versión 2.6.20 y permite ejecutar máquinas virtuales con sistemas operativos sin modificar.

La decisión de escoger KVM/QEMU se debe principalmente a un cierto grado de familiarización con esta tecnología en los sistemas con CentOS, de tal forma que el desarrollo de la infraestructura se vea agilizado.

Para el desarrollo web, existe una extensa variedad de *frameworks*, tanto para la parte cliente como para la parte servidor. Se pueden utilizar distintos lenguajes dependiendo de las necesidades técnicas o simplemente por cuestiones de gustos y/o conocimientos. Java, PHP, JavaScript o Python son los lenguajes más utilizados en la actualizad. La elección del *framework* Flask con Python o Angular con TypeScript (compilado a JavaScript) se fundamenta en que ambas partes son fácilmente integrables, por lo que se facilita, en gran medida, el desarrollo de la herramienta a través de dos aplicaciones.

1.4. Conclusiones

En la EII y el DIS existe un alto número de equipos que gestionar y personalizar por cada laboratorio, por lo que, para crear uniformidad en ellos, hay que realizar un proceso de clonación de los discos de los equipos. Este proceso conlleva a una parada considerable en el servicio de los laboratorios debido al tiempo de mantenimiento que requiere esta tarea. Además, un mínimo cambio en las especificaciones de los sistemas incurre en realizar una clonación de los discos de los equipos. Con una infraestructura centralizada de máquinas virtuales, se reduce en gran medida el impacto en el servicio, ya que estas se desplegarían en segundo plano.

Por otra parte, con el cambio de infraestructura y la introducción de tecnologías de virtualización, se pretende abrir un abanico de futuros servicios. Por ejemplo, se podría implantar el concepto de la Infraestructura como Servicio, ya que de esta forma no sólo se cambia la infraestructura actual para mejorar la gestión de los activos TI, sino que también se mejora la productividad y se reducen los costes de los recursos con los que se cuenta. [10]

Actualmente hay estudiantes que demandan recursos tecnológicos ofrecidos por la biblioteca para llevar a cabo tareas formativas. De esta manera, en lugar de realizar préstamo físico, se les ofrecería una máquina virtual como servicio extra. Esto es muy frecuente en las organizaciones que están modernizando sus infraestructuras y los trabajadores demandan acceder a sus recursos desde un entorno ajeno a la empresa, ya que, en lugar de acceder a un escritorio remoto de un equipo físico a través de una VPN, se accede a un escritorio totalmente virtualizado.

Por último, el proyecto consta de una serie de objetivos y tecnologías impuestas para su realización, por lo que, a pesar de existir otras alternativas, estas no se ajustan a los objetivos del proyecto. La solución al problema consta de una serie de requisitos que no ofrece ninguna otra solución, ya que se trata de desarrollar e implementar una infraestructura y su aplicación de gestión a medida tal y como se describe en la memoria de propuesta de este Trabajo de Fin de Grado.

2. Competencias cubiertas

El estudio del Grado en Ingeniería Informática implica la adquisición de unas competencias generales según el RD 1393/2007, competencias de la Universidad de Las Palmas de Gran Canaria según lo establecido por esta Universidad para todas sus titulaciones y competencias del título siguiendo con en el Anexo II de la Resolución de 8 de julio de 2009 de la Secretaría General de Universidades (BOE de 4 de agosto de 2009), en las que se incluyen las competencias específicas de la intensificación. [11]

2.1. Comunes a la Ingeniería Informática

2.1.1. CII01

"*Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.*"

Para el diseño y el desarrollo, se han estudiado y escogido tecnologías y herramientas, bajo unas determinadas licencias, que cubren los requisitos de calidad, seguridad y fiabilidad. Por ello, la aplicación web sigue unos principios de diseño que aseguran que sea fiable, tolerante a fallos y compatible con cualquier navegador en cualquier sistema operativo; segura, ya que existe un control de acceso a los distintos recursos de la aplicación y se almacena la información justa y necesaria; y de calidad, ya que cumple una serie de requisitos utilizando técnicas y metodologías actualizadas, lo que favorece el mantenimiento futuro de la aplicación.

2.1.2. CII03

"*Capacidad para comprender la importancia de la negociación, los hábitos de trabajo efectivos, el liderazgo y las habilidades de comunicación en todos los entornos de desarrollo de software.*"

A través de entrevistas con el personal del CCDIS, durante las fases de análisis y diseño, se ha logrado recoger los requisitos para el desarrollo de la aplicación. Es crucial una buena comunicación en la que se adapte el lenguaje y la terminología al cliente, para cometer el menor número de errores en la negociación de los requisitos.

2.1.3. CII05

"*Conocimiento, administración y mantenimiento sistemas, servicios y aplicaciones informáticas.*"

Para la puesta en marcha de este proyecto, se ha tenido que configurar una serie de aplicaciones y servicios que dan soporte a la aplicación web, tanto para la infraestructura de virtualización como para el despliegue de la aplicación. Gracias a la documentación de Red Hat Enterprise Linux 7/ CentOS 7 (RHEL 7 en adelante), posibilita que otras personas sean capaces de continuar la administración y el mantenimiento de todos los componentes que forman este proyecto.

2.1.4. CII07

"*Conocimiento, diseño y utilización de forma eficiente los tipos y estructuras de datos más adecuados a la resolución de un problema.*"

El sistema de información de la aplicación web utiliza varios tipos y estructuras de datos, cada uno de ellos acorde a los estándares de comunicación cliente-servidor, infraestructura de virtualización, o ciertos requisitos del cliente.

2.1.5. CII08

"*Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados.*"

Una vez analizados los requisitos del personal del CCDIS, se escogieron las tecnologías y lenguajes más adecuados para la envergadura del proyecto. Por ello, la continua actualización y la gran comunidad existente para estas, fue uno de los criterios más importantes para la selección de las tecnologías detalladas en el capítulo 6.

2.1.6. CII10

"*Conocimiento de las características, funcionalidades y estructura de los Sistemas Operativos y diseñar e Implementar aplicaciones basadas en sus servicios.*"

El conocimiento de las características y servicios que ofrece CentOS 7 resulta de gran importancia, ya que es necesaria la configuración y uso de varios servicios, tales como PolicyKit, módulos de autenticación conectables (PAM), suite de virtualización, etc., para integrar varios la aplicación web en la infraestructura.

2.1.7. CII12

"*Conocimiento y aplicación de las características, funcionalidades y estructura de las bases de datos, que permitan su adecuado uso, y el diseño y el análisis e implementación de aplicaciones basadas en ellos.*"

El sistema de información de la aplicación web incorpora una base de datos para tratar cierta información de necesidad en forma de objetos para aprovechar al máximo las ventajas de la programación orientada a objetos, como por ejemplo el mapeador de objetos relacionales SQLAlchemy que ofrece Flask para Python 3.

2.1.8. CII13

"*Conocimiento y aplicación de las herramientas necesarias para el almacenamiento, procesamiento y acceso a los Sistemas de información, incluidos los basados en web.*"

A pesar de que el *microframework* Flask tiene soporte *fullstack*, se ha decidido separar *frontend* y *backend* utilizando para ello tecnologías distintas, por lo que se tuvo que diseñar y desarrollar una API RESTful para facilitar el almacenamiento, procesamiento y acceso a la información del sistema.

2.1.9. CII16

"*Conocimiento y aplicación de los principios, metodologías y ciclos de vida de la ingeniería de software.*"

Se ha seguido una serie de fases, principios y metodologías estudiadas en las asignaturas propias de ingeniería del software. Así, es posible desarrollar una aplicación con buenas prácticas para minimizar los costos de tiempo y mantenibilidad.

2.1.10. CII17

"*Capacidad para diseñar y evaluar interfaces persona computador que garanticen la accesibilidad y usabilidad a los sistemas, servicios y aplicaciones informáticas.*"

La interfaz de usuario que incorpora el cliente web tiene un diseño adaptable a cualquier dispositivo y en cualquier navegador, por lo que garantiza que un usuario pueda acceder y hacer uso del sistema.

2.1.11. CII18

"*Conocimiento de la normativa y la regulación de la informática en los ámbitos nacional, europeo e internacional.*"

La aplicación y despliegue de la infraestructura debe recoger los aspectos señalados en las normativas de ámbito internacional, europeo y nacional. Como en este proyecto no se realiza un tratamiento de datos personales, la normativa queda acotada al cumplimiento de las licencias software utilizadas que se encuentran detalladas en el capítulo 4.

2.2. Ingeniería del Software

2.2.1. IS1

"*Capacidad para desarrollar, mantener y evaluar servicios y sistemas software que satisfagan todos los requisitos del usuario y se comporten de forma fiable y eficiente, sean asequibles de desarrollar y mantener y cumplan normas de calidad, aplicando las teorías, principios, métodos y prácticas de la ingeniería del software.*"

Se ha establecido una planificación previa para el desarrollo del proyecto, de forma que se garantiza un producto fiable y robusto al aplicar la teoría, principios y métodos prácticos de ingeniería del software. Así se consigue cubrir todos los requisitos establecidos.

2.2.2. IS2

"*Capacidad para valorar las necesidades del cliente y especificar los requisitos software para satisfacer estas necesidades, reconciliando objetivos en conflicto mediante la búsqueda de compromisos aceptables dentro de las limitaciones derivadas del coste, del tiempo, de la existencia de sistemas ya desarrollados y de las propias organizaciones.*"

Debido a las limitaciones de tiempo, se han especificado los requisitos más importantes recogidos en las entrevistas con el personal del CCDIS. De esta forma, se priorizan aquellas funcionalidades que dan lugar a una aplicación que cumpla las necesidades mínimas.

2.2.3. IS3

"*Capacidad de dar solución a problemas de integración en función de las estrategias, estándares y tecnologías disponibles.*"

Una vez analizado el problema, en la fase de análisis se han estudiado y valorado distintas tecnologías que den lugar a una solución de calidad. Para ello, se ha optado por elegir tecnologías web, las cuales se adaptan totalmente a la solución que se buscaba.

2.2.4. IS4

"*Capacidad de identificar y analizar problemas y diseñar, desarrollar, implementar, verificar y documentar soluciones software sobre la base de un conocimiento adecuado de las teorías, modelos y técnicas actuales.*"

El desarrollo de este proyecto implica detallar las distintas fases que se han abarcado, desde el análisis del problema hasta la generación de documentos o manuales de usuario.

2.3. Tecnologías de la Información

2.3.1. TI01

"*Capacidad para comprender el entorno de una organización y sus necesidades en el ámbito de las tecnologías de la información y las comunicaciones.*"

Desarrollar una aplicación e implementar una infraestructura a medida implica tener que conocer la organización o el departamento para el que se va a trabajar, sus procedimientos, recursos de los que dispone y sus necesidades. En este caso, el CCDIS ha sido el departamento que ha facilitado los distintos requisitos.

2.3.2. TI02

"*Capacidad para seleccionar, diseñar, desplegar, integrar, evaluar, construir, gestionar, explotar y mantener las tecnologías de hardware, software y redes, dentro de los parámetros de coste y calidad adecuados.*"

Tanto la infraestructura como la aplicación se han diseñado y configurado en base a los recursos de los que dispone el CCDIS, garantizando así que la solución pueda ser ejecutada sin problemas, de forma robusta y eficiente.

2.3.3. TI04

"*Capacidad para seleccionar, diseñar, desplegar, integrar y gestionar redes e infraestructuras de comunicaciones en una organización.*"

La integración de este proyecto en el CPD del CCDIS supone una serie de procesos de análisis, diseño y desarrollo en función de los recursos de los que dispone este departamento. De esta manera, se ha conseguido desplegar e integrar los distintos componentes que conforman el sistema en el CCDIS y los laboratorios de la EII.

2.3.4. TI05

"*Capacidad para seleccionar, desplegar, integrar y gestionar sistemas de información que satisfagan las necesidades de la organización, con los criterios de coste y calidad identificados.*"

Una vez conocidos los recursos de los que dispone el CCDIS, se han llevado a cabo las distintas fases del proyecto. Así, se ha logrado satisfacer los requisitos identificados adaptando la solución a los distintos criterios de costo y calidad.

2.3.5. TI06

"*Capacidad de concebir sistemas, aplicaciones y servicios basados en tecnologías de red, incluyendo Internet, web, comercio electrónico, multimedia, servicios interactivos y computación móvil.*"

El desarrollo de este proyecto lleva consigo el diseño y desarrollo de una aplicación web y la instalación y configuración de una serie de elementos que conforman la infraestructura de virtualización. La aplicación web se integra totalmente con la capa de virtualización, de tal forma que desde cualquier equipo se puede administrar la misma.

2.4. Trabajo Fin de Grado

2.4.1. TFG01

"*Ejercicio original a realizar individualmente y presentar y defender ante un tribunal universitario, consistente en un proyecto en el ámbito de las tecnologías específicas de la Ingeniería en Informática de naturaleza profesional en el que se sinteticen e integren las competencias adquiridas en las enseñanzas.*"

El proyecto se ha desarrollado a partir de una idea propia de los tutores, de la cual se ha tenido que hacer un análisis previo para diseñar y desarrollar la infraestructura y aplicación. Además, todo el trabajo realizado ha partido desde cero, ya que no existe referencia alguna para el proyecto actual.

3. Aportaciones

3.1. Entorno tecnológico

El desarrollo de este proyecto implica un cambio en la infraestructura actual de los equipos de los laboratorios. De esta forma, se consigue uniformidad en todos los equipos que serán gestionados desde una herramienta software desplegada en uno de los servidores del CCDIS. Además, se introduce el uso de las tecnologías de virtualización que se utilizan en muchas de las grandes organizaciones, ya que hasta ahora los laboratorios de la EII y el DIS sólo cuentan con VirtualBox para la ejecución de máquinas virtuales muy puntuales.

3.2. Entorno profesional

Con este cambio en se consigue, en primer lugar, agilizar los procesos de gestión y administración de los equipos, ya que la configuración software para las asignaturas se realizará a través de máquinas virtuales, de forma que dichos procesos se resumen en operaciones básicas con las máquinas. Además, aquellos profesores que estén familiarizados con entornos de virtualización podrían crear sus propias máquinas virtuales para posteriormente ser desplegadas en los laboratorios especificados por el personal del CCDIS. También existe un alto impacto en el alumnado, ya que desde el primer curso se le fomenta el uso de sistemas Linux y herramientas de virtualización. Así se consigue una mayor familiarización con estos entornos de cara a asignaturas del segundo ciclo del Grado en Ingeniería Informática, como Administración de Sistemas Operativos e Infraestructuras Tecnológicas para los Sistemas de Información, entre otras.

3.3. Personal

A nivel personal este proyecto ha sido todo un reto, puesto que al haber solicitado una propuesta de trabajo que cubra necesidades reales del personal del CCDIS, se ha tenido que abarcar por completo las fases de análisis, diseño y desarrollo. Además, se han realizado tareas que van desde la gestión de la infraestructura, hasta el desarrollo de la herramienta de gestión web que da soporte a esta, por lo que la carga de trabajo en las distintas fases del proyecto ha sido alta.

Por otra parte, el uso de nuevas tecnologías para el desarrollo implica un proceso amplio de documentación para abordar un proyecto de tal calibre, puesto que se ha llevado a cabo el desarrollo o la gestión de la infraestructura con las últimas versiones de los *frameworks* y del sistema operativo, respectivamente. Estas se encuentran en continua evolución, ya que son tecnologías que están en el día a día de grandes desarrollos, por lo que necesitan de una gran comunidad que de soporte.

4. Normativa y legislación

4.1. Licencias Software

Las licencias software establecen los contratos entre desarrolladores y usuarios en los que se otorgan una serie de permisos para el uso, distribución o modificación del código fuente de un producto. También establecen la cesión de determinados derechos del propietario, límites de responsabilidad ante fallos, ámbito de validez geográfico, etc. [12, 13]

Existe una gran variedad de licencias clasificadas según los derechos que cada autor se reserva sobre su obra, o según el destinatario. Para el desarrollo del proyecto han intervenido las siguientes licencias software:

4.1.1. Licencia GNU GPL

La Licencia Pública General (GPL) de GNU es una de las licencias más utilizadas, ya que el desarrollador conserva los derechos de autor, pero permite su libre distribución, modificación y uso, siempre que el nuevo software que se desarrolle quede bajo la misma licencia. [14]

CentOS 7 posee esta licencia al tratarse de una bifurcación a nivel binario de la distribución Red Hat Enterprise Linux (RHEL). También se incluyen los repositorios EPEL e IUS para los paquetes adicionales instalados en el sistema y el conjunto de herramientas de Libvirt. [15, 16, 17, 18]

El uso de uWSGI también se encuentra regulada por esta licencia, además de incluir una excepción a la hora de enlazar esta biblioteca en combinación con otros programas, permitiendo así su distribución sin ninguna restricción. [19]

Existe una versión reducida de esta licencia, más conocida por su nombre en inglés *GNU Lesser General Public License*, o por su acrónimo GNU LGPL. La principal diferencia entre la GPL y la LGPL se encuentra en que esta última puede enlazarse a software no-GPL. [20]

Esta licencia regula el uso de PolicyKit, usado para la gestión de los privilegios de los usuarios en el sistema operativo. [21]

4.1.2. Licencia MIT

Creada por el Instituto Tecnológico de Massachusetts, la licencia MIT es una licencia software permisiva, de tal forma que ofrece una alta compatibilidad con otras licencias, como por ejemplo la GNU GPL, pero no al contrario. Esta licencia también destaca porque reutiliza software dentro del software propietario, del cual no existe una forma libre de acceso a su código fuente. [22]

Bajo esta licencia se encuentran: el *framework* de desarrollo web Angular 7, la librería de componentes de diseño Clarity Design System y las librerías PyJWT y Python-PAM para la autenticación de usuarios en el *backend*. [23, 24, 25, 26]

4.1.3. Licencia BSD

Se trata de una licencia principalmente originada para sistemas BSD. A efectos prácticos es muy similar a la licencia MIT, ya que se trata de una licencia software permisiva que, al contrario de la licencia GPL, permite el uso de código fuente en software no libre. Sin embargo, existen varias revisiones entre las que destaca la llamada "BSD simplificada", la cual varía de cuatro a dos cláusulas. Además, se agrega un aviso en el que las opiniones y los puntos de vista de los contribuyentes del proyecto no representan necesariamente la visión del proyecto FreeBSD. [27]

El *microframework* Flask utilizado para el *backend* se encuentra regulado por la licencia BSD, mientras que, a pesar de existir una licencia comercial para Nginx denominada Nginx Plus, se puede utilizar la versión de código abierto bajo la licencia BSD simplificada. [28, 29]

4.1.4. Licencia PSFL

La *Python Software Foundation Licence* es una licencia de software libre permisiva al estilo de la licencia BSD, de manera que cumple con los requisitos OSI y es compatible con la licencia GPL. [30]

El lenguaje Python, que se ha usado para el desarrollo del *backend*, se encuentra regulado por esta licencia. [31]

4.1.5. JetBrains

JetBrains ofrece licencias de sus productos a estudiantes y profesores de los centros inscritos a su programa. Con esto se obtiene una subscripción no exclusiva y no transferible para cada producto para propósitos educativos por un período de un año. Esta licencia se aplica a los IDEs PyCharm y WebStorm usados para el desarrollo de la herramienta de gestión. [32, 33, 34]

4.1.6. Postman

El entorno de desarrollo para APIs denominado Postman, se encuentra regulado por su propia licencia indicada en el EULA del producto. Este acuerdo de licencia de usuario final tiene un total de quince puntos que abarcan una breve introducción y aceptación del acuerdo, los términos y condiciones para su uso, derechos de propiedad y no exclusividad, números de instancias que se permiten, etc. [35]

4.1.7. Resumen de licencias

En la siguiente tabla se muestra la relación entre las licencias software con las tecnologías, librerías y herramientas utilizadas.

	GNU GPL	GNU LGPL	MIT	BSD	BSD Simpl.	PSFL	JetBrains	Postman EULA
CentOS	X							
EPEL/IUS	X							
Libvirt	X							
uWSGI	X							
PolicyKit		X						
Angular			X					
Clarity DS			X					
PyJWT			X					
Python-PAM			X					
Flask				X				
Nginx					X			
Python						X		
PyCharm IDE							X	
WebStorm IDE							X	
Postman								X

Tabla 1: Licencias Software

5. Metodología de trabajo y planificación

5.1. Metodología de trabajo

El desarrollo de este trabajo sigue una metodología incremental, la cual se ajusta a la planificación inicial y recursos humanos con los que se dispone, es decir, una única persona. Con el modelo incremental, se aplican secuencias o iteraciones a lo largo del tiempo del calendario del proyecto, de forma que, al final de cada incremento, se realizan entregas de pequeñas piezas de software usables llamadas "incrementos". Así, las soluciones y modelos elaborados se analizan, diseñan, desarrollan y evalúan a lo largo de varias etapas, introduciendo avances o cambios significativos en el desarrollo que permiten llegar al objetivo fijado, previa evaluación de los tutores.

Al utilizar este modelo, el primer incremento suele incluir los requisitos mínimos, dejando las características complementarias para otros incrementos. Por ejemplo, en este proyecto es fundamental que la herramienta de gestión pueda realizar conexiones con el hipervisor del sistema e interactuar con los modelos de la base de datos, a pesar de que no esté implementado el sistema de control de acceso o un buen diseño de la interfaz de usuario, entre otros requisitos.

Una vez se evalúa este primer incremento, se utilizan las conclusiones obtenidas como retroalimentación para los incrementos posteriores, por lo que esta metodología consta de las siguientes fases:

- Análisis: en esta fase, caracterizada en un comienzo por la incertidumbre, se realiza una revisión de documentación con el objetivo de conocer el estado del arte en el ámbito de estudio o de la resolución de problemas.

- Diseño: se formula una propuesta teórica a partir de bocetos, diagramas UML, diagramas entidad-interrelación para el diseño de la base de datos, recursos RESTful, etc. Además, se escogen las herramientas, tecnologías y patrones de diseño necesarios para llevar a la práctica el modelo teórico planteado.

- Desarrollo experimental: cuando se finaliza el diseño y se da por "acabado", se desarrolla el código y se le aplica una selección de datos.

- Evaluación: se analizan y evalúan los resultados obtenidos en la fase experimental.

- Retroalimentación: una vez realizada la evaluación, se da comienzo a un nuevo incremento en el que se mejora el modelo planteado en base a la evaluación.

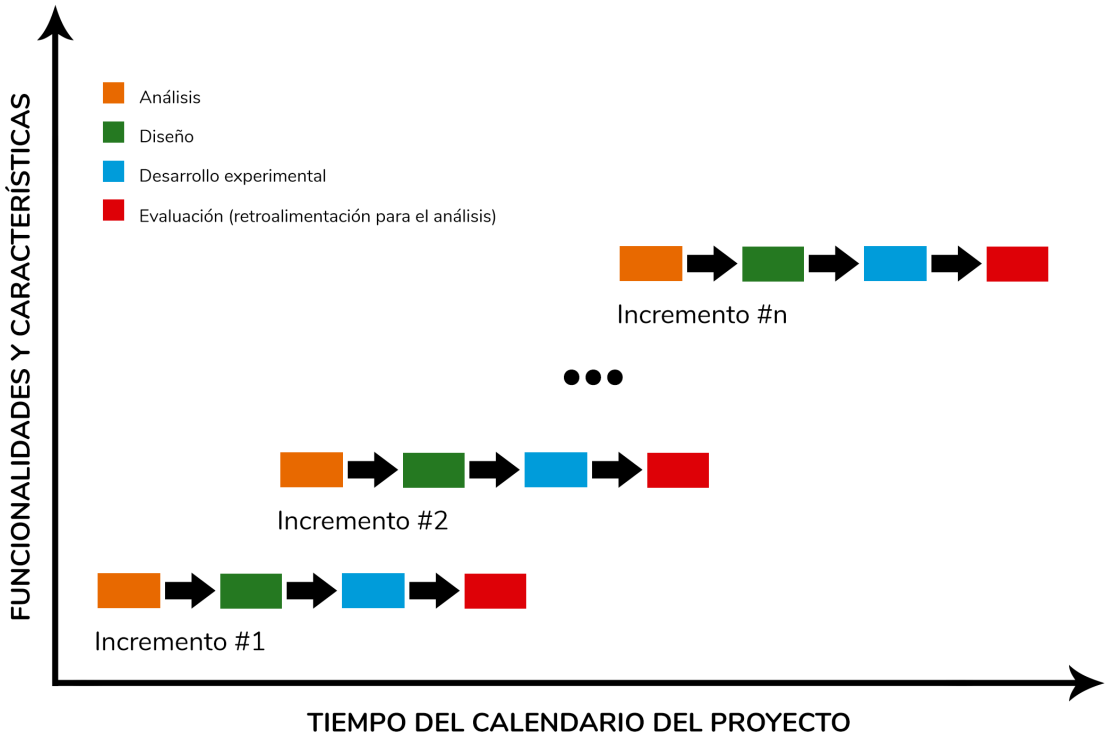

Ilustración 1: modelo incremental (elaboración propia)

Como se ha visto previamente, este trabajo pretende dar una solución única que no tiene alternativa alguna, por lo que el nivel de incertidumbre es alto. Al aplicar iteraciones en el proceso de desarrollo del software, se consigue sacar ventaja del conocimiento adquirido en el análisis y resolución de los problemas en iteraciones previas. Además, es una metodología que gestiona y se adapta a procesos que tienden a cambiar durante el ciclo de vida. [36]

5.2. Planificación inicial del proyecto

La planificación inicial acordada para el desarrollo del proyecto consta de cuatro fases. En la primera fase se realiza un análisis de requisitos basado en las necesidades del CPD del Departamento de Informática y Sistemas. Además, se realiza un estudio de las tecnologías, por lo que se establece una duración total de esta fase en 25 horas.

En la siguiente fase, de unas 185 horas aproximadas, se debe realizar el diseño de la plataforma, así como la instalación de los sistemas de las infraestructuras de red, almacenamiento y cómputos requeridos para dar soporte a los módulos software requeridos.

Posteriormente, se realiza la evaluación del proyecto en una duración estimada de 50 horas, en las que se debe desarrollar las plantillas de los sistemas a virtualizar, la generación de las máquinas virtuales basadas en estas plantillas y una prueba y análisis del rendimiento del sistema.

Para finalizar, se desarrolla la documentación de la memoria del trabajo de fin de título y los manuales de usuario, tareas que se estiman que tenga una duración de 40 horas.

El proyecto comienza con el análisis del estado del arte, en el que se realiza una captura de los requisitos basado en las necesidades del CPD del Departamento de Informática y Sistemas. Además, se recogen las funcionalidades básicas que debe tener el sistema a través de pequeñas entrevistas con personal del CCDIS.

Una vez identificados los requisitos, se continúa con el estudio de aquellas tecnologías que ofrezcan un buen producto acorde a lo solicitado, de forma que la solución final sea robusta y pueda evolucionar en trabajos futuros. Esta tarea resulta ser intensiva en documentación, ya que se seleccionan tecnologías que no se han utilizado o estudiado en profundidad para el desarrollo de las actividades en el grado.

Una vez finalizada la fase de análisis, se realiza una prueba de conceptos con un primer diseño del sistema. Tras comprobar que todas las piezas del sistema encajan y se comunican correctamente entre sí, se desarrolla una primera y pequeña pieza software en la que básicamente se realizan conexiones con el hipervisor en un sistema de pruebas y se interactúa con los modelos de la base de datos. También se realiza la instalación de los sistemas de las infraestructuras de red, almacenamiento y cómputo requeridos para el correcto funcionamiento de la aplicación.

Finalizada esta toma de contacto, da lugar a una serie de incrementos en los que se lleva a cabo el desarrollo de la herramienta y configuración de la infraestructura de virtualización a través de la metodología descrita previamente.

5.3. Ajustes de la planificación del proyecto

En la fase de análisis de las primeras iteraciones en el proceso de desarrollo, se detectó que el proyecto tomaría mayor envergadura de la esperada en un principio. Como consecuencia de ello, la estimación inicial de la duración de las distintas fases del proyecto fue inferior a la dedicación real requerida, tal y como se puede apreciar en la Tabla 2:

Fases	Duración estimada (horas)	Tareas	Duración real aproximada (horas)
Estudio previo / Análisis	25	Tarea 1.1. Análisis de requisitos basado en las necesidades del CPT del Dpto. de Informática y Sistemas.	40
		Tarea 1.2. Estudio de las tecnologías a utilizar.	
Diseño / Desarrollo / Implementación	185	Tarea 2.1. Diseño de la plataforma.	250
		Tarea 2.2. Instalación de los sistemas de las infraestructuras de red, almacenamiento y cómputo requeridos.	
		Tarea 2.3. Desarrollo de los módulos software requeridos.	
Evaluación / Validación / Prueba	50	Tarea 3.1. Desarrollo de las plantillas de sistemas a virtualizar.	60
		Tarea 3.2. Generación de las máquinas virtuales basadas en las plantillas.	
		Tarea 3.3. Prueba y análisis de rendimiento de un laboratorio virtual.	
Documentación / Presentación	40	Tarea 4.1. Desarrollo de la documentación de la memoria del Trabajo de Fin de Título.	50
		Tarea 4.2. Desarrollo de los manuales de usuario.	

Tabla 2: plan de trabajo

Como en el mercado actual no existe una solución que preste similitudes con los objetivos de este proyecto, el estudio de las tecnologías a usar ha de ser muy preciso y exhaustivo. Además, surgen contratiempos como la falta de características y funcionalidades básicas en las tecnologías impuestas para el desarrollo del proyecto, por lo que la solvencia de estos problemas requiere de horas de trabajo que no estaban contempladas en la planificación inicial y que alargan en gran medida la finalización del proyecto.

6. Tecnologías, herramientas e infraestructura hardware

6.1. Tecnologías

En la actualidad existe una gran variedad de tecnologías de código abierto tanto para el desarrollo de aplicaciones web como para la infraestructura de virtualización. No todas pueden ser usadas para todos los desarrollos, por lo que hay que realizar un estudio previo y una valoración de las alternativas para garantizar una buena solución y que pueda evolucionar en trabajos futuros.

6.1.1. La herramienta de gestión

6.1.1.1. *Aplicaciones web vs. Aplicaciones nativas*

En los últimos años, el desarrollo de aplicaciones web ha tomado gran protagonismo debido al incremento en el uso de dispositivos móviles conectados a internet.

Las aplicaciones web básicamente son herramientas accesibles desde una red bajo una arquitectura cliente-servidor, por lo que no necesitan ser instaladas en los dispositivos. Sin embrago, no están desarrolladas para un dispositivo en particular, por lo que en algunos casos no se aprovecha todo el potencial del hardware. Aún así, existen *frameworks* de desarrollo que ofrecen APIs para acceder a ciertas funciones nativas de los dispositivos, principalmente *smartphones*, como por ejemplo el acceso a la cámara, sensores de geolocalización, acelerómetro, interfaz de red, etc. Los dos más usados son Apache Cordova o Adobe PhoneGap. [37, 38]

En términos monetarios, las aplicaciones nativas son más caras de desarrollar, ya que por lo general son mejores en rendimiento y eficiencia. Esto se debe al uso de tecnologías y lenguajes de programación específicos para cada plataforma, como es el caso de iOS y Android, o multiplataforma que se ejecuten sobre una máquina virtual que sí está diseñada para cada plataforma, como es el caso de Java o Python.

Entre muchas de las ventajas de una aplicación web, destaca la experiencia del usuario: con diseños *responsive* se logra una mejor navegabilidad del usuario en distintos dispositivos, debido a que el contenido se adapta al tamaño y resolución de la pantalla en cuestión. Además, son muy flexibles en cuanto a acceso, ya que se puede acceder desde cualquier red permitida por la infraestructura, lo cual resulta de gran interés en entornos empresariales, como por ejemplo una empresa que disponga de un sistema de gestión interno (ERP, p.ej.) en una organización accesible desde cualquier puesto de trabajo y dispositivo.

Para aprovechar las ventajas de las aplicaciones web y las aplicaciones nativas, surgen las Aplicaciones Web Progresivas (PWA de sus siglas en inglés). Este tipo de aplicaciones web surgen gracias al uso de los *Service Workers* y otras tecnologías para que dichas aplicaciones puedan seguir ejecutándose en segundo plano, aunque se haya cerrado el navegador, o incluso si todavía no se ha abierto. Así, se consigue romper la barrera existente entre la web y las aplicaciones. [39, 40, 41]

El desarrollo de la herramienta de gestión se llevará a cabo a partir de dos aplicaciones, una para el cliente y otra para el servidor, que se integrarán en un único despliegue.

6.1.1.2. Desarrollo Web: el cliente

- **Angular**: se trata de uno de los *frameworks* de desarrollo web que más popularidad ha alcanzado en los últimos años entre las distintas tecnologías web y alternativas existentes. Destaca por el uso de Typescript sobre JavaScript puro, ya que potencia bastante el desarrollo cuando se pretende buscar la escalabilidad en las aplicaciones. Además, ofrece la posibilidad de utilizar plantillas declarativas para enlazar datos, aplicar inyección de dependencias y crear componentes reutilizables en la aplicación. También se trata de una herramienta orientada al desarrollo de aplicaciones SPA, modular y permite con gran facilidad la integración de otras tecnologías con Node.js. En su versión más reciente, soporta TypeScript Release 3.1, por lo que es capaz de usar sus características más relevantes, como por ejemplo el mapeo de tipos a tuplas y arrays, declaración de funciones como propiedades de una función y la inclusión del nuevo tipo de datos "*unknown*". Por otra parte, la documentación está muy bien detallada y mantenida, ya que el equipo de desarrollo está continuamente incorporando nuevas funcionalidades y mejorando el núcleo. [42, 43]

- **Clarity Design System**: bajo esta denominación, se encuentra la librería de componentes visuales que se ha utilizado para el diseño y desarrollo de la interfaz de usuario en la aplicación cliente. Es un proyecto *opensource* de la empresa VMware Inc. que no sólo ofrece el *framework* para HTML/CSS y componentes Angular (UI), sino que también incluye guías para mejorar la experiencia de usuario (UX) en base a estos, lo cual ayuda a realizar un buen diseño de las interfaces sin tener grandes conocimientos en este ámbito. A pesar de ser una librería prácticamente reciente, se ha decidido usarla porque ofrece componentes y "*layouts*" propios de interfaces de administración. Entre todas las librerías analizadas, Clarity Design System es la única que incorpora un componente "*wizard*" para realizar asistentes de instalación, ya que crear un componente similar utilizando otras librerías tendría cierta complejidad y no se lograría el mismo efecto con tales opciones de personalización [44].

6.1.1.3. *Desarrollo Web: el servidor*

- **Python 3**: es un lenguaje de programación orientado a objetos (entre otros paradigmas), interpretado, con tipado dinámico y multiplataforma. Además, tiene una sintaxis que favorece la legibilidad del código. Para el desarrollo de aplicaciones web tiene una gran ventaja, ya que no necesita de un ecosistema para la ejecución de las aplicaciones, al contrario que otros lenguajes como PHP o Java.

- **Flask**: es un *microframework* para Python que permite crear aplicaciones web de forma rápida y con un gran ahorro en líneas de código. Se basa en la especificación WSGI e incorpora el motor de plantillas Jinja2, aunque para este proyecto no se ha utilizado, ya que en su lugar se usa el compilado de la aplicación *frontend* para servir una interfaz de usuario elegante, dinámica y consistente.

- **SQLite**: es un sistema de gestión de base de datos relacional que cumple con los requisitos ACID. La biblioteca implementa la mayor parte del estándar SQL-92 y consultas complejas. No es necesario disponer de un servidor específico para establecer una comunicación, ya que desde la propia aplicación se puede acceder a la base de datos con el *driver* apropiado para ello.

- **sqlalchemy_utils**: es una librería de utilidades para SQLAlchemy. Incorpora tipos de datos tales como "IPAddressType" o "UUID", usados en los modelos de datos del sistema de información.

- **uWSGI**: es una aplicación que permite servir y desplegar aplicaciones web Python junto a servidores web como Nginx. Es la principal alternativa como servidor de producción, ya que no se recomienda en absoluto usar el servidor integrado de Flask para producción.

Además, para realizar un control de versiones de ambas aplicaciones y realizar el correspondiente despliegue en el sistema, se han usado las siguientes tecnologías:

- **Git**: es un sistema de control de versiones, pensado para la eficiencia, confiabilidad y mantenimiento de versiones de las aplicaciones cuando crecen en cantidad de ficheros. Aunque gestiona el desarrollo distribuido, en este proyecto se ha utilizado para generar backups de la aplicación antes de realizar cambios de importancia en los códigos fuentes.

- **Nginx**: es un servidor web/proxy inverso de código abierto, ligero y de alto rendimiento. En este proyecto cumple la función de proxy inverso, ya que la aplicación es servida a través de uWSGI.

6.1.2. Infraestructura de virtualización

Para poder implantar una herramienta de gestión es imprescindible contar con una infraestructura, debidamente configurada, que de soporte al sistema que se pretende diseñar. Esta infraestructura consta de múltiples equipos con un entorno Linux sobre el que se instalan ciertos componentes de virtualización.

6.1.2.1. ¿Qué es la virtualización?

Existen muchas definiciones sobre el concepto de "virtualización", pero todas parten de la misma base: la creación virtual de recursos tecnológicos a través de software. Estos recursos en su conjunto permiten crear máquinas capaces de ejecutar software, normalmente sistemas operativos, de forma concurrente y totalmente aisladas de otros programas en un mismo sistema. La capa software que permite la virtualización, controlando y gestionando los distintos recursos de un sistema físico, se llama "hipervisor". En este documento se hará gran uso de término "invitado" haciendo referencia al sistema operativo instalado en una máquina virtual. [45, 46]

Existen distintos métodos de virtualización:

- **Virtualización completa**: es el método de virtualización más rápido. La máquina virtual utiliza una versión no modificada del sistema operativo invitado, ya que hace uso de la CPU del host a través de un canal creado por el hipervisor.

- **Paravirtualización**: las máquinas virtuales utilizan una versión modificada del sistema operativo invitado. El hipervisor encamina las llamadas no modificadas del invitado a la CPU y otras interfaces, tanto físicas como virtuales. Debido a este encaminamiento, este método es más lento que la virtualización completa.

- **Virtualización por software**: para ejecutar sistemas operativos sin modificar se utiliza la traducción binaria u otras técnicas de emulación. El hipervisor realiza una traducción en las llamadas para que puedan ser reconocidas por el anfitrión o host. Al tener que realizarse dicha traducción, este método es más lento que los dos anteriores.

- **“Containerización”**: mientras que los métodos anteriores crean instancias separadas de núcleos de sistemas operativos, la “containerización” opera en lo alto de un núcleo existente, creando instancias aisladas del sistema operativo anfitrión o host. Esta tipología tiene la versatilidad de la virtualización completa o la paravirtualización, pero permite tener contenedores de poco peso y fáciles de manejar idóneos para tareas de desarrollo o incluso llevar a cabo entornos de producción que requieran de poca carga y mantenimiento.

6.1.2.2. *Componentes*

- **CentOS 7:** *Comunity Enterprise Operating System*, más conocido como CentOS, es distribución Linux estable derivada a nivel binario de la distribución Red Hat Enterprise Linux (RHEL). Se administra de forma similar a este, siendo la principal diferencia la eliminación de todas las referencias a las marcas y logos propiedad de Red Hat. Es muy popular entre los usuarios de sistemas Linux, hosting web y pequeñas empresas, ya que el principal objetivo de esta distribución es ofrecer a los usuarios un sistema operativo de “clase empresarial” gratuito. Este proyecto pretende expandir su objetivo de establecer CentOS como una plataforma líder para tecnologías de código abierto emergentes que provienen de otros proyectos como OpenStack. [47, 48]

- **Libvirt:** es un conjunto de utilidades y herramientas que facilitan la gestión de máquinas y recursos virtuales, como el almacenamiento o las interfaces de red. Este conjunto incluye una librería API, el demonio (libvirtd) y una utilidad de línea de comandos (virsh). Destaca dentro del software open-source de virtualización por la compatibilidad con los hipervisores más utilizados en el mercado actual, como KVM, Xen, VMware ESX, entre otros. La integración de la API libvirt en la herramienta de gestión se realiza a través de una traducción o ‘*bindings*’ del lenguaje C a Python. Así, una vez importado el paquete “libvirt-python” se tiene acceso a todos sus elementos desde la aplicación. [49, 50]

- **KVM:** *Kernel-based Virtual Machine* (KVM), es una solución completa de virtualización para sistemas Linux. Básicamente convierte un núcleo Linux en un hipervisor que permite que la ejecución de máquinas virtuales aisladas entre sí. Se trata de un hipervisor de tipo-1 o nativo, es decir, se ejecuta directamente en el hardware del equipo y ya forma parte del núcleo de Linux desde la versión 2.6.20 en adelante. Esto se consigue gracias a la incorporación de una serie de módulos al propio *kernel* del sistema operativo, por lo que no se ejecuta sobre este. [51, 52]

Dicha solución completa se instala en CentOS a través de cuatro grupos de paquetes que contienen todo el software necesario: “Virtualization Hypervisor”, “Virtualization Platform”, “Virtualization Client” y “Virtualization Tools”. En la tabla que se muestra a continuación se detalla la relación que hay entre los grupos y los paquetes:

Grupo de paquetes	Descripción	Paquetes obligatorios	Paquetes opcionales
Virtualization Hypervisor	Instalación de host de virtualización	libvirt, qemu-kvm, qemu-img	qemu-kvm-tools
Virtualization Client	Clientes para instalar y gestionar instancias de virtualización	gnome-boxes, virt-install, virt-manager, virt-viewer, qemu-img	virt-top, libguestfs-tools, libguestfs-tools-c
Virtualization Platform	Provee una interfaz para acceder y controlar máquinas virtuales y contenedores	libvirt, libvirt-client, virt-who, qemu-img	fence-virtd-libvirt, fence-virtd-multicast, fence-virtd-serial, libvirt-cim, libvirt-java, libvirt-snmp, perl-Sys-Virt
Virtualization Tools	Herramientas para gestión de imágenes virtuales fuera de línea	libguestfs, qemu-img	libguestfs-java, libguestfs-tools, libguestfs-tools-c

Tabla 3: grupos de paquetes de virtualización (fuente: Red Hat, "Virtualization Deployment and Administration Guide")

6.2. Herramientas

A lo largo de las fases del proyecto se han usado una serie de herramientas, tanto para el desarrollo de la aplicación web como para el desarrollo de la API RESTful:

- **WebStorm 2018.2**: es un IDE multiplataforma para el desarrollo web utilizando principalmente JavaScript y TypeScript. Facilita en gran medida el desarrollo, organización y mantenimiento de proyectos gracias a la multitud de plugins disponibles y la integración de los sistemas de control de versiones.

- **PyCharm 2018.2**: también bajo la propiedad de JetBrains, PyCharm es un IDE multiplataforma para el desarrollo de aplicaciones Python. Ha sido una herramienta crucial para el desarrollo del backend, ya que ofrece un depurador bastante intuitivo y potente para aplicaciones estructuradas en módulos, así como la posibilidad de conectar con las bases de datos que usa la aplicación desde la misma interfaz de usuario.

- **Firefox Quantum 60.0.2**: se trata del último navegador web diseñado por Mozilla. Cuenta con herramientas de desarrollo de gran utilidad para depurar las aplicaciones web.

- **Postman 6.1.3**: es un entorno de desarrollo de APIs (*ADE - API Development Environment*) que cubre todas las fases del ciclo de vida de una API: diseño y maquetado, depuración, testeo automático, documentación, monitorización y publicación. En este proyecto se ha usado para realizar llamadas de prueba a la API y examinar las respuestas obtenidas por el *backend*.

6.3. Infraestructura hardware

La puesta en preproducción de este proyecto se respaldada por la infraestructura hardware del CPD del Dpto. de Informática y Sistemas. El despliegue de la herramienta de gestión y la infraestructura de virtualización que le da soporte se encuentra en un servidor rack Dell PowerEdge 1950, con un procesador Intel Xeon y 8GB de memoria RAM DDR2 ECC, con el sistema operativo CentOS 7 en su instalación 'Minimal' sobre un sistema de almacenamiento RAID1.

Para el desarrollo de la aplicación, se ha utilizado un MacBook Pro con el sistema operativo macOS High Sierra. El equipo cuenta con un procesador Intel i5 3210M y 16GB de memoria RAM. En él se han instalado las herramientas de desarrollo, por lo que la validación de los requisitos de la aplicación se ha realizado en un entorno local conectando con el hipervisor remoto del servidor de preproducción. Así, se evita sobrecargar el sistema de desarrollo.

Como gran parte del desarrollo se realiza fuera de la red de la ULPGC, se ha preparado una réplica del entorno de preproducción sobre un servidor rack personal IBM x3250 M3 con un procesador Intel Xeon X3430 de 4 núcleos y 8GB de memoria RAM DDR3 ECC, junto a un disco duro de 500GB. Este equipo cubre las necesidades de una infraestructura de virtualización para soportar la ejecución simultánea de varias máquinas virtuales sin ningún problema.

Luego, cada laboratorio cuenta con una serie de equipos que se pretende que sean idénticos en especificaciones hardware. Estos equipos por lo general tienen buena potencia de cálculo, cantidad de memoria RAM y almacenamiento, por lo que tanto el despliegue como la ejecución de las máquinas virtuales, se puede llevar a cabo sin mayores problemas.

7. Análisis

7.1. Introducción

La fase de análisis es crucial en cualquier proyecto, ya que es la etapa en la que se centra gran parte de la atención en el problema en cuestión: proponer una alternativa a la metodología actual de la administración y gestión de los equipos de los laboratorios de la EII.

Con el objetivo principal de aplicar tecnologías de virtualización, no sólo se consigue unificar todos los equipos para facilitar su gestión desde una herramienta desarrollada a medida, sino que también se está incluyendo la infraestructura necesaria para dar soporte a futuros servicios, como la posibilidad de almacenar las imágenes de las máquinas virtuales en un almacenamiento remoto.

Por ello y para este análisis, el proyecto lo podemos dividir en dos grandes partes: la infraestructura y la herramienta de gestión.

7.2. La infraestructura

7.2.1. ¿Por qué usar virtualización?

La virtualización trae consigo muchas ventajas, de las que este proyecto se puede aprovechar: en primer lugar y objetivo principal, se consigue una administración centralizada y simplificada de las máquinas virtuales. Desde una o varias herramientas de gestión se pueden administrar los distintos recursos de la infraestructura, sin la necesidad de tener que planificar largos períodos de mantenimiento en los laboratorios. De esta forma, se realizan tareas sobre las máquinas virtuales, principalmente, sin que afecte al desarrollo normal de las clases.

La optimización de los recursos hardware es otro de los puntos fuertes de la virtualización que benefician a este trabajo. Gracias a esto, se consigue pasar de dos sistemas base Windows y Linux sobrecargados de software y entornos de desarrollo, a un sistema base Linux con máquinas virtuales especializadas, por lo que el costo en espacio de todos los sistemas y tiempo de puesta en marcha de los equipos se ve notablemente reducido.

A esto se le suma el aislamiento, de forma que, si en una máquina virtual ocurre un fallo general del sistema, el resto de las máquinas no se ven afectadas. Esto también permite crear máquinas virtuales para entornos de prueba en asignaturas que traten cuestiones

delicadas, como por ejemplo sucede en Administración de Sistemas Operativos o Fundamentos de Seguridad.

Además, la posibilidad de descontextualizar máquinas virtuales aporta valor en la gestión lógica de los laboratorios, ya que permite generar plantillas a partir de estas y posteriormente distribuirlas a lo largo de los laboratorios como si se tratase de una primera instalación. Estas dos operaciones básicas, la generación de plantillas y su consiguiente distribución, sustituirían a la metodología costosa de la clonación de discos duros para garantizar la homogeneidad de los equipos.

7.3. La herramienta de gestión

Una vez escogidos los componentes necesarios de la infraestructura para llevar a cabo la virtualización, se requiere de un entorno de gestión y administración de estos recursos. Para ello, se desarrolla una aplicación web que será utilizada por el personal del CCDIS, ya que son los encargados de la gestión y administración de los laboratorios.

Esta se desarrolla a partir de dos aplicaciones: el cliente y el servidor. Ambas partes integradas en una sola aplicación constituyen el entorno web de gestión.

7.3.1. Actores

En la aplicación web se identifica un único actor que utilizará el sistema: el usuario “dvls”. La existencia de un único actor se debe a que la herramienta sólo realiza labores de gestión y administración de la infraestructura de virtualización en los laboratorios. Será el personal del CCDIS el que tenga acceso a la plataforma para realizar las siguientes funciones principales:

- Estadísticas del sistema: desde una vista dedicada, el usuario podrá obtener información sobre el sistema, como es el consumo de memoria RAM o la cantidad de laboratorios o plantillas que se han definido.

- Gestión de los dominios: desde la creación hasta la eliminación y desde el encendido hasta el apagado, el usuario podrá gestionar los distintos dominios del sistema DVLS.

- Agrupación lógica de los equipos en laboratorios: es la principal función por la que destaca la aplicación desarrollada. El personal del CCDIS será capaz de agrupar de forma lógica los equipos de los laboratorios, de forma que se consigue centralizar y automatizar la distribución de las máquinas virtuales.

- Generación de plantillas: una vez se configure una máquina virtual, se clona dicha máquina a plantilla para posteriormente generar nuevas máquinas virtuales y personalizarlas acorde a nuevos requisitos hardware o software.

7.3.2. Requisitos

Para representar los requisitos funcionales de la aplicación se utiliza el Lenguaje Unificado de Modelado (UML). Este lenguaje sigue un estándar aprobado por la Organización Internacional de Normalización (ISO), por ello, cualquier persona que entienda dicho estándar podrá conocer los requisitos funcionales que presenta la aplicación. [53]

Los casos de uso que se presentan en el siguiente apartado se han recogido tanto de los objetivos del proyecto como de pequeñas entrevistas con personal del CCDIS para identificar sus necesidades.

7.3.2.1. Casos de uso

A continuación, se muestran los distintos casos de uso para el único usuario existente en el sistema. Para facilitar su lectura ha sido dividido en varios diagramas:

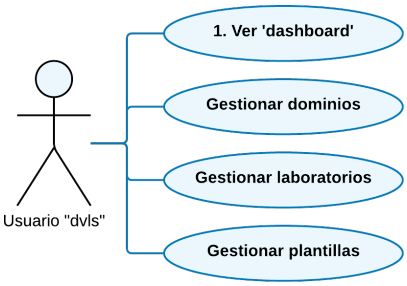

Ilustración 2: diagrama de casos de uso (resumen)

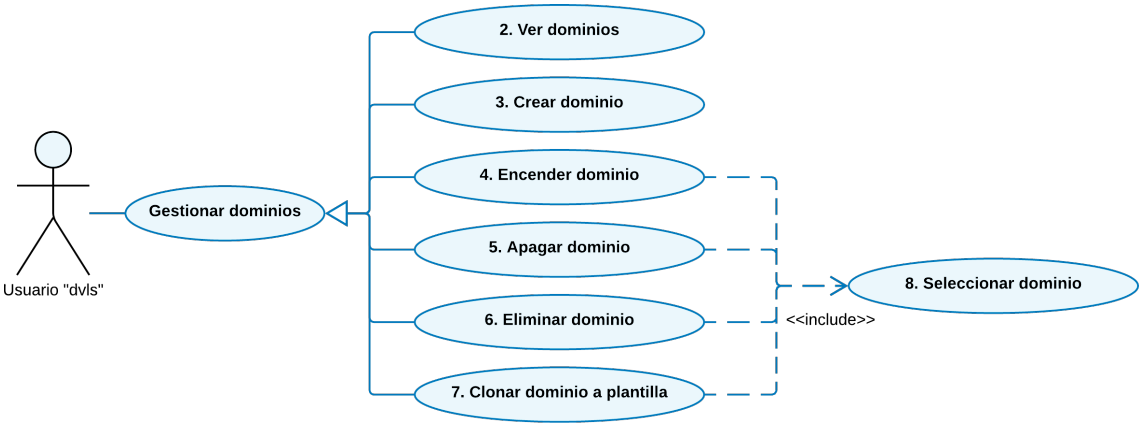

Ilustración 3: diagrama de casos de uso (gestionar dominios)

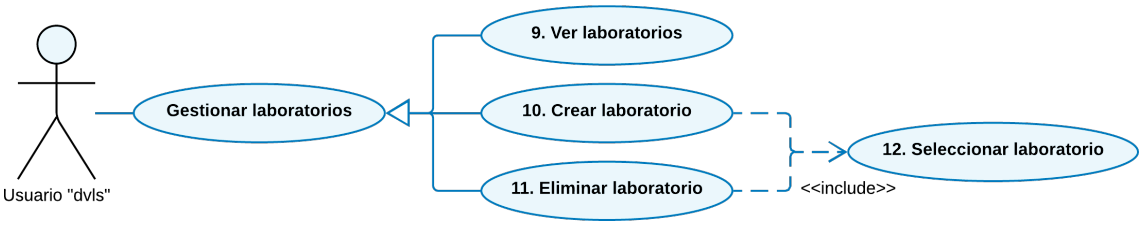

Ilustración 4: diagrama de casos de uso (gestionar laboratorios)

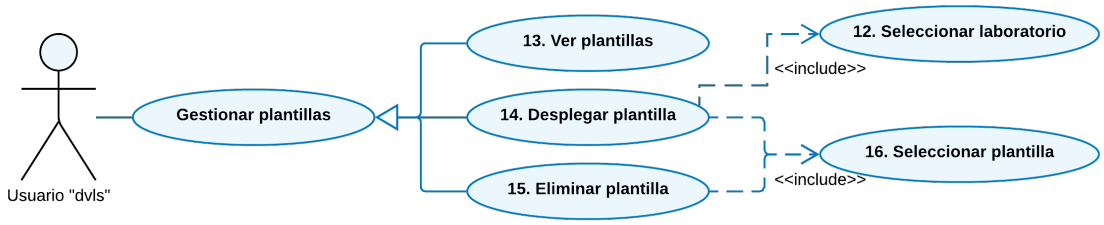

Ilustración 5: diagrama de casos de uso (gestionar plantillas)

7.3.2.2. *Especificación de los casos de uso*

En este punto se realiza una especificación más detallada de los casos de uso más importantes en la aplicación:

CASO DE USO: #3	**TÍTULO: CREAR DOMINIO**		
DESCRIPCIÓN	El usuario define las características básicas de un nuevo dominio a través de un asistente.		
ACTORES	Usuario “dvls”		
PRECONDICIONES	· El usuario “dvls” debe estar autenticado en el sistema. · Se debe haber establecido conexión previa con el hipervisor.		
FLUJO NORMAL	PASO	ACCIÓN	
	1	Se selecciona el método de instalación del dominio.	
	2	Se selecciona el tipo de sistema operativo invitado.	
	3	Se introduce el tamaño del disco de almacenamiento.	
	4	Se introduce la configuración básica del dominio.	
	5	Se revisa el resumen de las características del dominio.	
	6	Se envían los parámetros de configuración al servidor.	
POSTCONDICIONES	· Se define un nuevo dominio al finalizar la ejecución de la orden ‘virt-install’ en el servidor. · Se cierra el asistente en caso de no haber excepciones.		
VARIACIONES	PASO	ACCIÓN	
	-	-	
EXTENSIONES	PASO	CONDICIÓN	CASO DE USO
	-	-	-
EXCEPCIONES	PASO	CAUSA	
	6	El método de instalación no es correcto.	
	6	Ha ocurrido un error al ejecutar la orden ‘virt-install’ en el servidor.	
OBSERVACIONES	· Si el usuario escoge importar una imagen existente como método de instalación, se salta el paso #3.		

Tabla 4: especificación de casos de uso (crear nuevo dominio)

CASO DE USO: #7	TÍTULO: CLONAR DOMINIO A PLANTILLA		
DESCRIPCIÓN	El usuario introduce los parámetros necesarios para la clonación de un dominio a una plantilla.		
ACTORES	Usuario “dvls”		
PRECONDICIONES	· El usuario “dvls” debe estar autenticado en el sistema. · Se debe haber establecido conexión previa con el hipervisor. · El dominio debe tener un sistema operativo instalado. · El dominio debe estar apagado.		
FLUJO NORMAL	PASO	ACCIÓN	
	1	Se introduce el nombre de la plantilla.	
	2	Se introduce una breve descripción sobre el propósito/contenido de la plantilla.	
	3	Se envían los parámetros al servidor.	
POSTCONDICIONES	· Se define un nuevo dominio cuando finaliza la ejecución de la orden correspondiente en el servidor. · Se cierra el componente modal con el formulario.		
VARIACIONES	PASO	ACCIÓN	
	-	-	
EXTENSIONES	PASO	CONDICIÓN	CASO DE USO
	-	-	-
EXCEPCIONES	PASO	CAUSA	
	3	Se viola una restricción de unicidad en el sistema de información.	
OBSERVACIONES	· Debido a una mala configuración de los permisos en el despliegue de la aplicación, la base de datos puede no tener permisos de escritura.		

Tabla 5: especificación de casos de uso (clonar dominio a plantilla)

CASO DE USO: #10	TÍTULO: CREAR NUEVO LABORATORIO		
DESCRIPCIÓN	El usuario introduce los parámetros necesarios para la especificación de un nuevo laboratorio.		
ACTORES	Usuario “dvls”		
PRECONDICIONES	· El usuario “dvls” debe estar autenticado en el sistema.		
FLUJO NORMAL	PASO	ACCIÓN	
	1	Se introduce el nombre de la plantilla.	
	2	Se introduce una breve descripción sobre el propósito/contenido de la plantilla.	
	3	Se envían los parámetros al servidor.	
POSTCONDICIONES	· Se define un nuevo laboratorio cuando finaliza la operación en la base de datos. · Se definen los hosts que pertenecen al rango de direcciones IP especificado en los parámetros del laboratorio. · Se cierra el componente modal con el formulario.		
VARIACIONES	PASO	ACCIÓN	
	-	-	
EXTENSIONES	PASO	CONDICIÓN	CASO DE USO
	-	-	-
EXCEPCIONES	PASO	CAUSA	
	3	Se viola una restricción de unicidad en el sistema de información.	
OBSERVACIONES	· Debido a una mala configuración de los permisos en el despliegue de la aplicación, la base de datos puede no tener permisos de escritura.		

Tabla 6: especificación de casos de uso (crear nuevo laboratorio)

CASO DE USO: #14	TÍTULO: DEPLEGAR PLANTILLA		
DESCRIPCIÓN	El usuario introduce el nombre del dominio que se desplegará en los equipos del laboratorio que deberá seleccionar.		
ACTORES	Usuario “dvls”		
PRECONDICIONES	· El usuario “dvls” debe estar autenticado en el sistema.		
FLUJO NORMAL	PASO	ACCIÓN	
	1	Se introduce el nombre del nuevo dominio.	
	2	Se selecciona el laboratorio destino.	
	3	Se envían los parámetros al servidor.	
POSTCONDICIONES	· Se define un nuevo dominio en los equipos del laboratorio seleccionado. · Se cierra el componente modal con el formulario.		
VARIACIONES	PASO	ACCIÓN	
	-	-	
EXTENSIONES	PASO	CONDICIÓN	CASO DE USO
	-	-	-
EXCEPCIONES	PASO	CAUSA	
	3	No se pudo obtener la información de la plantilla.	
	3	No se pudo obtener la información del laboratorio.	
	3	El laboratorio no tiene ningún *host* relacionado.	
	3	No se puede establecer la conexión SSH: no se ha encontrado la clave pública del servidor.	
	3	Ha fallado la ejecución del comando enviado por SSH.	
OBSERVACIONES	· Debido a que el despliegue de la plantilla se realiza a través de la orden ‘virt-install’ enviada por SSH a cada uno de los equipos, no se obtiene ni la salida estándar, ni la salida de error de la ejecución de la orden en el equipo de destino.		

Tabla 7: especificación de casos de uso (desplegar plantilla)

8. Diseño

8.1. Arquitectura del sistema

El modelo cliente-servidor se considera el más apropiado para esta solución, ya que a través de una API RESTful se pueden gestionar y administrar los distintos recursos desde cualquier navegador dentro de la red apropiada para ello. Así, el desarrollo del sistema se consigue dividir en dos partes: el cliente, al que se le ha dado como nombre "DIS vLab Center (DVLC)", siguiendo el paradigma de diseño SPA; y el servidor o "DIS vLab Server (DVLS)", el cual se ejecuta como servicio en el equipo servidor e integra el código compilado del cliente. Además, la aplicación DVLS no sólo tiene una visión lógica de una API RESTful, sino que también se puede considerar como una aplicación que sirve de nexo entre el usuario final y Libvirt.

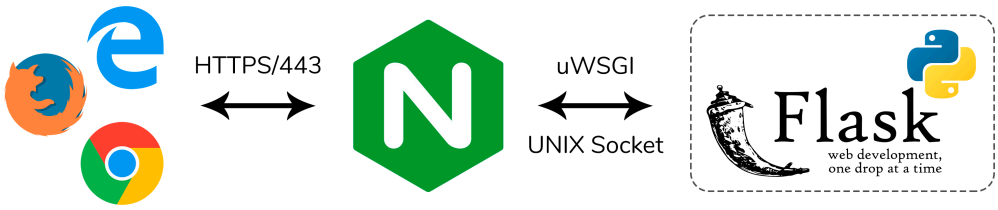

Ilustración 6: Arquitectura de la herramienta de gestión (elaboración propia)

Una de las grandes ventajas por las que se ha decidido diseñar una API REST (*Representational State Transfer* - Transferencia de Estado Representacional) es la flexibilidad que aporta. Los recursos que se envían o reciben no tienen una estructura fija, por lo que se pueden intercambiar distintas estructuras de datos: XML, JSON, YAML, etc. Además, la separación entre cliente y servidor hace que el producto escale fácilmente: cambios en la infraestructura, remodelación de la base de datos, cambios en las tecnologías/lenguajes, etc.

Al poner en producción una aplicación web Flask, es habitual que los usuarios no accedan directamente a la aplicación a través del servidor uWSGI, el cual provee los servicios de *hosting* para la aplicación. En su lugar, se hace uso del servidor web Nginx configurado como proxy inverso para que actúe como intermediario entre el cliente y el servidor uWSGI. Como el despliegue de la aplicación y el proxy inverso se encuentran en el mismo equipo, se utiliza el socket de UNIX para la comunicación entre Nginx y uWSGI por dos principales razones: seguridad y rendimiento.

Aunque el *microframework* de desarrollo web Flask proporciona el motor de plantillas Jinja2, se ha optado por el uso de Angular 7 para construir la aplicación cliente, ya que facilita en gran medida el diseño SPA. Por ello, cuando el usuario realiza una petición al servidor Nginx utilizando el protocolo HTTP/HTTPS al *endpoint* "/", se devuelve la aplicación cliente previamente compilada para un entorno de producción. A partir de este punto, el usuario dispone de una serie de acciones y recursos a los que puede acceder a través de la interfaz web.

8.2. Modelo de datos

La aplicación gestiona y almacena cierta información de forma interna en una base de datos SQLite3. Así, gracias al mapeador objeto-relacional SQLAlchemy, se trata la información de la base de datos como objetos en términos de programación, lo que ofrece una alta flexibilidad a la hora de operar con dichos datos. Esta base de datos consta de las siguientes tablas o entidades:

La entidad "Lab" representa la agrupación lógica de los equipos físicos de los laboratorios. Además de ofrecer información sobre el laboratorio en sí, como el rango de direcciones IP en el que se encuentra o el código de laboratorio, también ofrece información relevante sobre los equipos que lo componen de forma genérica: número máximo de vCPUs asignables a las máquinas virtuales, cantidad máxima de memoria RAM o la capacidad máxima de almacenamiento que disponen los equipos. Esta información es de gran utilidad para la configuración de las máquinas virtuales antes de ser desplegadas en los equipos.

Aparte de ser fundamental para generar el concepto de un laboratorio, la entidad "Host" sirve para almacenar la información necesaria para realizar la conexión con el hipervisor. De este modo, con la dirección IP, el usuario de la conexión y la clave pública SSH del servidor, previamente distribuida, se logra realizar la conexión.

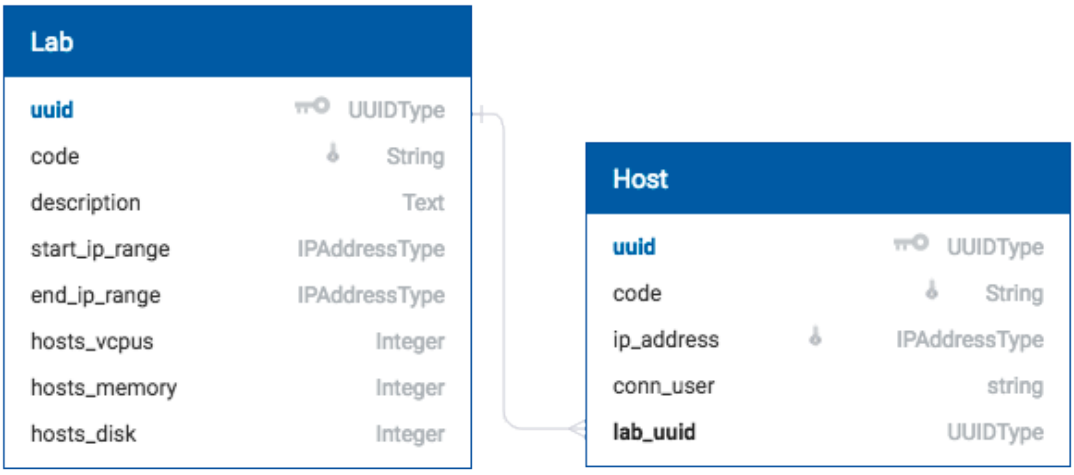

Ilustración 7: diagrama entidad-relación 'Lab' y 'Host'

Por otra parte, se tiene a la entidad “Template” que no tiene ninguna relación con ninguna otra entidad, ya que únicamente ofrece información sobre las rutas absolutas de los ficheros que conforman las plantillas en el sistema. Además, se incluye una breve descripción sobre el contenido o propósito de la plantilla, para posteriormente facilitar la creación de nuevas máquinas virtuales.

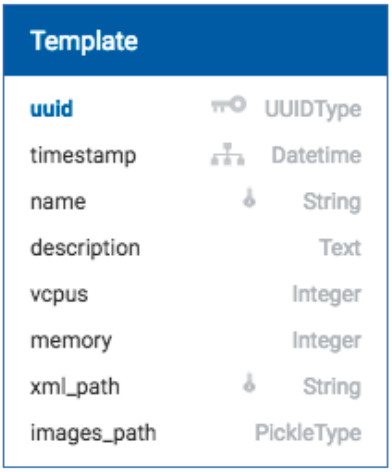

Ilustración 8: diagrama entidad-relación 'Template'

Por último, la entidad “Config” proporciona persistencia para ciertos parámetros internos de la aplicación, como por ejemplo el nombre de usuario con el que se realiza la autenticación, las rutas por defecto donde se encuentran las definiciones e imágenes de los dominios y la URI de conexión privilegiada con el *driver* local del hipervisor. Aunque en un principio no resulte necesario, se ha diseñado esta entidad para que, en un trabajo futuro, las configuraciones de la aplicación tengan cierto nivel de personalización.

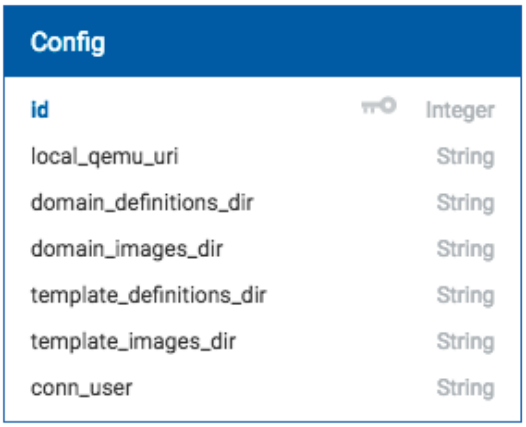

Ilustración 9: diagrama entidad-relacion 'Config'

8.3. Diseño arquitectónico: SPA

Una aplicación de una sola página (SPA del inglés, *Single Page Application*), como su nombre bien indica, es una aplicación web cuyo contenido se carga de forma dinámica y se muestra en una sola página, con el objetivo principal de prestar al usuario una mayor fluidez en la navegación. Además, se mejora la experiencia del usuario debido a los comportamientos *responsive* en los dispositivos y a la comunicación entre cliente y servidor, ya que los datos que se intercambian son más ligeros que en una aplicación o sitio web tradicional. [54, 55]

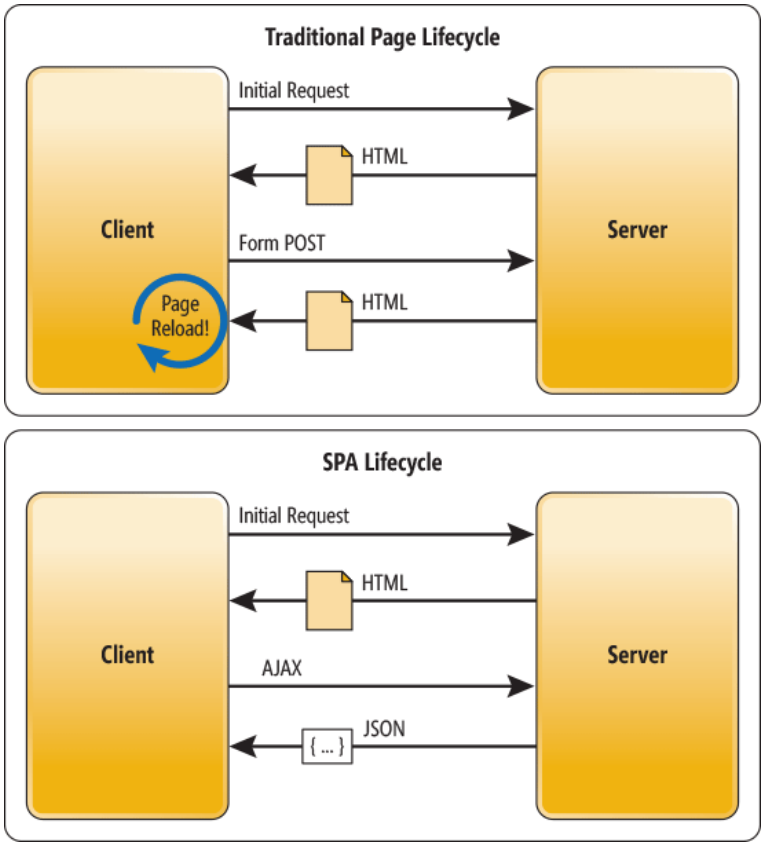

Ilustración 10: ciclo de vida tradicional vs. ciclo de vida SPA (fuente: MSDN Microsoft)

Para desarrollar Aplicaciones de una Sola Página, es conveniente hacer uso de *frameworks* cuya arquitectura esté orientada al desarrollo de aplicaciones SPA, como es el caso de Angular, ReactiveJS, Polymer, entre otros.

Seguir este paradigma es posible gracias al uso de Angular, ya que su propia arquitectura está orientada a la creación de componentes y módulos. De esta forma, las sensaciones con la que se encuentra el usuario son la de fluidez y comodidad en el uso.

8.4. Diseño arquitectónico: MVC

El uso de Angular como *framework* de desarrollo *frontend* y Flask para el desarrollo *backend*, facilita enormemente el diseño de aplicaciones con cierta organización y estructura, de forma que se puedan seguir patrones de diseño como es el caso del 'Modelo-Vista-Controlador'.

MVC sigue un ciclo que comienza con un usuario solicitando un recurso. Esta solicitud llega enrutador de la aplicación servidor DVLS, el cuál es capaz de detectar el controlador asociado a la URL de la petición. Una vez se llama al controlador, este realiza las operaciones pertinentes con los modelos que persisten en la base de datos. Lo más común es que los datos que ha solicitado el usuario sean devueltos en una respuesta HTTP con la vista, pero como se ha separado la aplicación en un desarrollo para el cliente y otro desarrollo para el servidor, los datos se devuelven al controlador de la aplicación cliente DVLC estructurados en JSON. Es entonces el controlador correspondiente al componente de la aplicación Angular el que muestra los datos en la vista.

De esta forma, el diseño MVC del sistema lo forman las dos aplicaciones integradas en una sola, tal y como se puede observar en la ilustración 11:

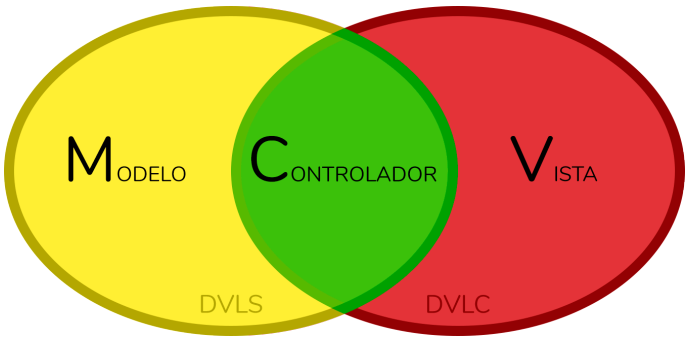

Ilustración 11: diseño arquitectónico - MVC

En Angular, la vista la componen los ficheros HTML y CSS que conforman las plantillas o "templates", dentro de la propia terminología. Por otra parte, se tiene el controlador que interactúa con las vistas y los modelos, e implementa la lógica de negocio en Typescript, aunque se compila en Javascript ES6.

9. Desarrollo

A continuación, se detalla el desarrollo de las dos aplicaciones que componen la herramienta de gestión. Sin embargo, el procedimiento para configurar la infraestructura de virtualización, el sistema operativo y el despliegue de la herramienta, se encuentra detallado en el Anexo I adjunto al final de este documento.

Para dar soporte a las tareas de gestión de los laboratorios y creación de máquinas virtuales, se ha desarrollado una herramienta de gestión a partir de dos aplicaciones: DIS vLab Client, el cliente; y DIS vLab Server, el servidor. A pesar de que dicha herramienta se encuentre compuesta por dos desarrollos, el compilado del *frontend* se incorpora a la aplicación DIS vLab Server, de forma que la herramienta se despliega en su totalidad a través de una sola aplicación.

9.1. El cliente: "DIS vLab Client"

La aplicación DIS vLab Client, o DVLC por sus iniciales, es la aplicación cliente que brinda una interfaz cómoda y amigable para realizar las labores de gestión y administración de los laboratorios, máquinas virtuales y despliegue de plantillas de forma simplificada. Está desarrollada en Angular 7 y en su esencia se trata de un cliente RESTful que se comunica con el *backend* DIS vLab Server.

9.1.1. Estructura del código fuente

A veces, encontrar la estructura de un proyecto en Angular no es trivial, ya que se tiene que seguir cierta estructura en los directorios para lograr modularidad y que la aplicación pueda crecer sin mayores problemas. Para facilitar esta labor a los desarrolladores, Angular ofrece unas guías de estilo y convenciones [56]. Por lo tanto, se ha considerado la siguiente estructura la más oportuna para el alcance de esta aplicación:

```
|-- app
    |-- app.component.html | .scss | .ts
    |-- app.module.ts
    |-- app-routing.module.ts
    |-- modules
        |-- [+] dashboard
        |-- [+] domains
        |-- [+] labs
        |-- [+] login
        |-- [+] settings
        |-- [+] templates
        |-- [+] ui
    |-- shared
        |-- [+] enums
        |-- [+] guards
        |-- [+] services
|-- assets
```

Bajo el directorio "*/app*" está el código fuente que compone la aplicación. En primer lugar, se encuentran los ficheros "*app.component.html | .scss | .ts*" que se corresponden con el componente principal de la aplicación, el cual se especifica en el *bootstrap* del módulo "*app*". Este componente básicamente actúa de "*placeholder*" para renderizar las páginas de los módulos que se detallan a continuación:

- **Dashboard**: este módulo contiene la página de inicio de la aplicación. Muestra información general acerca del equipo físico sobre el que está en ejecución la aplicación, información sobre el consumo de la memoria RAM y otros datos de interés sobre el sistema de información. En este directorio, se encuentra el fichero "*dashboard.module.ts*", correspondiente a la declaración del módulo, y los directorios "*components*" y "*pages*" que, tal y como su nombre indica, contienen los ficheros de los componentes y páginas que lo componen.

- **Domains**: de forma similar, este módulo contiene los componentes y las páginas que hacen posible la gestión de los dominios. Entre los distintos componentes que incluye, cabe destacar el "*wizard*" totalmente personalizado para el proceso de creación de un nuevo dominio. Con este asistente el usuario es capaz de crear un nuevo dominio a golpe de clic.

- **Labs**: para la gestión de los laboratorios también se ha generado un módulo. Del mismo modo, bajo el directorio raíz de este se encuentra el fichero de declaración del módulo y los directorios "*components*" y "*pages*".

- **Login**: el módulo de login implementa la página en la que se muestra el componente con el formulario para el inicio de sesión en el sistema. Sigue la misma estructura de directorios de los módulos anteriores.

- **Settings**: este módulo se incorpora para administrar ciertos parámetros de la aplicación que garantizan la persistencia de la configuración que afecta al funcionamiento de la aplicación servidor DIS vLab Server. Entre estos parámetros, destaca la URI de conexión con el hipervisor local, ubicación de las plantillas y usuario autorizado para iniciar sesión en la aplicación. Además, se incorpora una página para leer el registro de logs que emite el servidor que ayuda en creces a detectar problemas en el desarrollo normal de la actividad en la aplicación.

- **Templates**: la gestión de las plantillas y el despliegue en los laboratorios también se incluye en un módulo propio. Del mismo modo, la estructura del directorio raíz del módulo se compone de un directorio para los componentes y otro para las páginas, además del fichero de declaración de módulo.

- **Ui**: este módulo tiene gran importancia, ya que es el que compone toda la interfaz de usuario. En el fichero de declaración del módulo se importa la librería Clarity Design System y se exportan los distintos *layouts* que se han creado para las páginas de los módulos previamente descritos. Además, para que estos puedanr hacer uso de los *layouts*, deben hacer referencia a éste en la declaración de "*imports*".

Por otra parte, se tiene el directorio "*shared*" que contiene recursos compartidos para toda la aplicación. Estos recursos pueden ser servicios, como por ejemplo el servicio de datos para las llamadas con la API RESTful, el servicio de autenticación o el servicio para gestionar las tareas que se producen en la aplicación; enumerados, para definir un conjunto de constantes con un propósito específico; y los llamados "guards", que son clases que implementan una interfaz con las que el enrutador de la aplicación puede decidir si debe permitir o denegar la navegación a una ruta solicitada.

Por último, en el directorio "assets" se pueden encontrar recursos estáticos. En este caso, se almacenan las imágenes utilizadas para el logo de la cabecera o el favicon de la aplicación.

9.1.2. Control de Acceso

El control de acceso en la aplicación cliente se basa en la clase que contiene el fichero "*auth.guard.ts*", dentro del directorio "*shared/guards*". Esta clase implementa la interfaz "CanActivate" [57], que tal y como su nombre indica implementa ciertos métodos que deben retornar, en función de la lógica de negocio, si los componentes del enrutador deben ser accesibles o no. El método "*canActivate(...)*" es llamado desde el enrutador en las rutas padre, el cual va a implementar toda la lógica de negocio para la autenticación de las rutas, ya que las rutas hijas del enrutador llaman a éste método a través de "*canActivateChild(...)*".

```
canActivate(
  next: ActivatedRouteSnapshot,
  state: RouterStateSnapshot): Observable<boolean> | Promise<boolean> | boolean {
  console.log('AuthGuard # canActivate called');

  if (!this.authService.isTokenExpired()) { return true; }

  // Store the attempted URL for redirecting
  this.authService.redirectUrl = state.url;

  console.log('AuthGuard # canActivate @ Access denied! Navigating to login...');
  // Navigate to the login page
  this.router.navigate(['/login']);
  return false;
}
```

Ilustración 12: método 'canActivate()' del fichero 'auth.guard.ts'

La autenticación del usuario en la aplicación se realiza a través de JWT, que se verá en detalle en el apartado de la aplicación servidor, pero en el cliente el mecanismo es muy simple: comprobar si el token de sesión ha expirado o no. Esto se realiza a través de una función declarada en el servicio de autenticación, la cual decodifica el token devuelto por el servidor para extraer el tiempo de expiración (ilustración 13) y lo comprueba con la fecha actual (ilustración 14).

```
private static getTokenExpirationDate(token: string): Date {
  const decoded = jwt_decode(token);

  if (decoded.exp === undefined) { return null; }

  const date = new Date(0);
  date.setUTCSeconds(decoded.exp);
  return date;
}
```

Ilustración 13: función para obtener la fecha de expiración del token de sesión.

```
isTokenExpired(token?: string): boolean {
  if (!token) { token = this.getToken(); }
  if (!token) { return true; }

  const date = AuthService.getTokenExpirationDate(token);
  if (date === undefined) { return false; }
  return !(date.valueOf() > new Date().valueOf());
}
```

Ilustración 14: función para comprobar si el token de sesión ha expirado.

9.2. El servidor: “DIS vLab Server”

La aplicación DIS vLab Server (DVLS en adelante) se puede considerar como el nexo entre Libvirt y la interacción del usuario con esta a través de la aplicación cliente DVLC. En su esencia, DVLS se trata de una aplicación *backend* con una API RESTful, diseñada según los requisitos especificados y modelos de datos del sistema de información, y un conjunto de llamadas a herramientas del sistema que permiten la integración completa de la aplicación DVLC con el hipervisor en cuestión.

9.2.1. Estructura del código fuente

El código fuente se encuentra estructurado de tal forma que se tiene una buena organización y escalabilidad para un trabajo futuro:

```
|-- app
    |-- api
       |-- endpoints
           |-- dashboard.py
           |-- domains.py
           |-- frontend.py
           |-- hosts.py
           |-- labs.py
           |-- login.py
           |-- logs.py
           |-- templates.py
       |-- api.py
       |-- utils.py
    |-- core
        |-- __init__.py
        |-- config.py
    |-- models
        |-- [+] migrations
        |-- db.sqlite3
        |-- models.py
    |-- [+] static
    |-- main.py
|-- venv
|-- requirements.txt
|-- dvls.ini
|-- wsgi.py

(Se obvian algunos ficheros que no son significativos para la explicación de la estructura)
```

En primer lugar, observamos que el proyecto utiliza un entorno virtual para la gestión y uso de los paquetes, así como el servidor uWSGI. La configuración de este se encuentra especificada en el fichero dvls.ini, el cual configura el número de procesos que se lanzarán, el socket a utilizar, la elevación de los permisos de usuarios a través del *uid* y *gid*, y la ubicación del fichero de *logs*, entre otros parámetros posibles.

Bajo el directorio "*/app*" encontramos una serie de módulos que componen la aplicación: un módulo para la API, uno para el núcleo de la aplicación y otro para los modelos. También se encuentra el directorio que integra los ficheros estáticos del cliente, que son el resultado de haber compilado previamente la aplicación DVLC.

Dentro del módulo para la API, se tiene un directorio para los "*endpoints*" que contiene un fichero por cada tipo de recurso. Así, si en un futuro se desean añadir recursos, basta con añadir un nuevo fichero. También se tiene un fichero con utilidades compartidas entre los distintos *endpoints*, como por ejemplo la función que retorna una respuesta en formato *text/json* a una petición, o la función que realiza la comprobación de la existencia del *token* de sesión en la cabecera de la petición.

En el módulo "core", se encuentra el núcleo de la aplicación. Este consta del fichero __init__.py, que inicializa la aplicación, y el fichero config.py, el cual otorga una configuración mínima inicial a la aplicación.

La persistencia del sistema de información y los modelos se encuentran en un módulo propio, "models", el cual contiene el directorio de las migraciones que genera el módulo Flask-Migrate, la base de datos "db.sqlite3" y el fichero con las clases correspondientes a los modelos, "models.py".

Por último, se tiene el directorio para los ficheros estáticos de la aplicación, que en este caso se tratan de los ficheros que genera la salida del comando "ng build –prod" en la aplicación DVLC. Estos ficheros son los que componen toda la vista y la lógica de la interfaz de usuario, por lo que se implementa un *endpoint* propio para servir la aplicación cliente.

9.2.2. Extensiones Flask

DVLS está desarrollado con Flask, un *microframework* que permite crear de forma sencilla aplicaciones web. Una de las grandes ventajas que ofrece Flask es la posibilidad de añadirle extensiones para aumentar las funcionalidades del microframework. Incluso existen una serie de guías para que los desarrolladores desarrollen sus propias extensiones [58]. A continuación, se detallan los paquetes de las extensiones Flask que se han usado para el desarrollo de DVLS:

- **Flask-Cors:** facilita el manejo del intercambio de recursos de origen cruzado (Cross Origin Resource Sharing - CORS). De esta forma, se hacen posible las llamadas AJAX de origen cruzado. Su uso es muy sencillo, ya que por defecto habilita el soporte CORS en todas las rutas, para todos los orígenes y métodos. [59]

- **Flask-Migrate:** esta extensión sirve de soporte para el manejo de las migraciones de las bases de datos usando el mapeador objeto-relacional SQLAlchemy. Facilita una serie de herramientas a través de línea de comando para realizar las operaciones con la base de datos. [60]

- **Flask-SQLAlchemy:** añade el soporte de SQLAlchemy a la aplicación. SQLAlchemy es un mapeador objeto-relacional que, en la programación orientada a objetos, permite implementar las tareas de gestión de los datos por manipulación de objetos en lugar de utilizar SQL. [61]

9.2.3. Control de Acceso

Dado que la aplicación será usada por personal de gestión y administración, se ha decidido implementar el control de acceso de los usuarios utilizando los módulos intercambiables de autenticación del sistema CentOS 7: *Pluggable Authentication Modules*, *PAM*. [62]

De esta forma, las credenciales que el usuario envía al *backend* son comprobadas utilizando el módulo Python-PAM. Básicamente este módulo recoge las credenciales y realiza la comprobación en dos fases: primero, comprueba si el usuario existe en el fichero */etc/passwd*; y segundo, si este existe, comprueba la contraseña con la información almacenada en el fichero */etc/shadow*. Por seguridad, se requiere que el usuario, con el que se tenga acceso a la aplicación, disponga de una contraseña que cumpla la política de contraseñas y seguridad de la información del CCDIS. [63]

Si la comprobación de las credenciales finaliza con éxito, se utiliza la librería PyJWT, que permite codificar y decodificar los JSON Web Tokens (JWT), para generar un token de sesión y posteriormente enviárselo al cliente. A partir de este momento, todas las

llamadas que se realicen a los *endpoints* con el decorador "@token_required", necesitaran recibir el *token* de sesión para realizar las operaciones pertinentes con los recursos. Si la autenticación ha sido denegada, se le notifica al cliente que las credenciales enviadas son incorrectas.

9.2.4. Generación de plantillas

Una plantilla, en términos generales, es un clon de la máquina virtual origen. La principal diferencia se encuentra en que la imagen del dominio está descontextualizada.

La descontextualización de una imagen a través de la conocida utilidad "virt-sysprep" en Linux o "sysprep.exe" en Windows, generaliza el sistema operativo invitado para que, a la hora de generar nuevas máquinas a partir de la plantilla, se trate la configuración de ese sistema como una primera instalación. [64]

Esto implica eliminar cuentas de usuario, claves SSH generadas, identificadores MAC de las interfaces de red, entre otras muchas acciones.

9.2.5. Registro de logs

Para facilitar la detección de errores, en el desarrollo normal de la actividad en la aplicación, se ha implementado un sistema de *logs* que utiliza la librería *logging* de Python. De esta forma, se puede realizar un seguimiento de los eventos que se producen en la aplicación servidor, indicando distintos niveles de registro: DEBUG, INFO, WARNING, ERROR, CRITICAL. Así, dependiendo de la importancia del evento, se clasifican estos mensajes para que el técnico del CCDIS identifique el error fácilmente.

Esto resulta de gran utilidad para detectar si existen errores a la hora de desplegar una plantilla en un laboratorio, ya que, en caso de producirse un error, se registra el equipo en el que no se pudo realizar la tarea, además de indicar el mensaje de la excepción correspondiente.

10. Resultados, conclusiones y trabajo futuro

10.1. Resultados y conclusiones

Una vez concluidas las distintas iteraciones del proceso de desarrollo, se ha obtenido un sistema que cumple los objetivos principales de este trabajo. En primer lugar, se ha cambiado la infraestructura actual de los equipos, convirtiendo las dos particiones base actuales a máquinas virtuales. Y, por otro lado, se ha desarrollado una herramienta de gestión, que es capaz de generar nuevas máquinas virtuales y personalizarlas en función de los requisitos software de las asignaturas.

La implantación de una infraestructura de virtualización, en sustitución a la infraestructura actual de los equipos, ha sido exitosa, ya que las dos máquinas base que reemplazan a las particiones actuales se ejecutan con buen rendimiento, gracias al uso de KVM como solución completa de virtualización para los equipos.

Además, con la puesta en marcha de la herramienta de gestión se ha conseguido reducir la carga de trabajo de gestión de los laboratorios de la EII y del DIS. Así, los técnicos del CCDIS pueden llevar a cabo tareas que abarcan desde la creación de las máquinas virtuales hasta el despliegue en los laboratorios de nuevas máquinas basadas en plantillas, a través de un sistema de información que permite realizar los distintos procesos a golpe de clic. Con esta nueva forma de trabajo, no es necesario impedir el uso de los equipos de los laboratorios, ya que el despliegue de las nuevas máquinas virtuales se realiza en segundo plano.

Las tecnologías escogidas, tanto para la herramienta de gestión como para la infraestructura, han sido acertadas: el desarrollo se ha visto facilitado en gran medida, ya que las distintas tecnologías tienen una documentación y comunidad extensa. Así, ante cualquier contratiempo es posible obtener una solución casi inmediata. También son tecnologías estables y actualizadas, ya que cada vez se incentiva más el *opensourcing*.

Sin embargo, la API que utiliza Libvirt no cubre las expectativas personales que se tenían al principio. Aunque a nivel de sistema operativo haga uso de ficheros XML para definir los recursos, la API podría trabajar con otro tipo de estructuras de datos más cómodas y que se ajusten a los estándares RESTful, en este caso. Manejar ficheros XML en lugar de ficheros JSON es una labor muy poco práctica si se trabaja en una arquitectura con dicho tipo de interfaz de comunicación ya que requiere, o del envío de ficheros para pasarlos directamente a la API de libvirt, o la construcción de estos ficheros XML en el *backend*. Por ello, no se aprovecha todo el potencial que tiene Python o Typescript en el

manejo de diccionarios. Debido a esto se hacen llamadas, en algunos casos, a utilidades de virtualización, como “virt-install” o “virt-clone” con un sencillo envío de parámetros.

Esta primera toma de conceptos ante un cambio en la metodología de gestión actual ha concluido con éxito. Aún así, el proyecto es altamente escalable gracias a las tecnologías que se han utilizado, por lo que puede evolucionar en trabajos futuros, tanto por el personal del CCDIS como por futuros Trabajos de Fin de Título.

10.2. Trabajo futuro

Como se ha visto, el desarrollo de este trabajo consiste en una toma de conceptos para un cambio en la infraestructura actual de los equipos, ya que el proyecto tiene un punto de partida con un alto grado de incertidumbre como para desarrollar un sistema más complejo en el período de duración del Trabajo de Fin de Grado.

Sin embargo, se ha realizado un análisis sobre aspectos que se pueden mejorar o incluir en un trabajo futuro, como puede ser un próximo TFG o Trabajo de Fin de Máster.

10.2.1. Celery: *Distributed Task Queue*

Celery es un conjunto de herramientas que permite trabajar fácilmente con múltiples servicios a través de la gestión de colas de tareas o trabajos asíncronos. Se basa en el paso de mensajes de forma distribuida. A pesar de que esté enfocada para operaciones en tiempo real, se puede utilizar como planificador de tareas. Su arquitectura se fundamenta en tres elementos: los productores, la cola y los consumidores. En el contexto de las tareas asíncronas, los "productores" ponen tareas en cola y son los consumidores los que comprueban la cabeza de la pila de trabajos pendientes. Extrae el primero y lo ejecuta. [65]

Existen distintas formas de implementar un sistema de colas. Un sistema robusto y concurrente son las bases de datos SQL, aunque son muy lentas. Sin embargo, las bases de datos NoSQL son más rápidas, pero muchas veces no se confía en su fiabilidad. Así que, para construir las colas, se deben usar herramientas rápidas, fiables y que permitan concurrencia, como RabbitMQ, Redis y SQS. [66]

Como se observa, esta herramienta aumentará el potencial de la herramienta de gestión, ya que será capaz de gestionar una cola de trabajos que puedan ser ejecutados de forma asíncrona. Con esto se logrará desarrollar un nuevo sistema de notificaciones que permita al usuario ser informado del progreso en todo momento, o realizar el despliegue de las plantillas de forma asíncrona a cada uno de los equipos que conforman un laboratorio.

10.2.2. Integración de WebSockets

Dentro de la arquitectura cliente-servidor que siguen las aplicaciones DIS vLab Client y DIS vLab Server, la comunicación se realiza a través de peticiones asíncronas. Es por ello por lo que al interactuar con los modelos del sistema de información o al finalizar una tarea, se deba refrescar el componente en cuestión realizando llamadas al servidor.

Una mejora significativa en la aplicación consistiría en integrar la tecnología WebSocket, la cual proporciona un canal de comunicación bidireccional y *full-duplex* sobre un único socket TCP. De esta forma, el cliente permanece de forma continua a la escucha de eventos que emite el servidor cuando se realizan operaciones sobre los modelos, o termina un trabajo, por ejemplo.

Con la implementación de WebSockets, se tendría en el cliente DVLC el registro de las tareas en tiempo real, actualización de los recursos en los listados (cuando se cambia el estado de una máquina virtual, p.ej.). En el lado del servidor, se combinaría la tecnología de colas distribuidas de Celery con WebSockets, de tal manera que se lograría tener un estado de progreso en las tareas. Así, el usuario tendría información del progreso en tareas de larga duración, como puede ser el proceso de clonación de una máquina virtual con varios discos de gran tamaño. [67, 68]

10.2.3. Funcionalidades y características

Como se ha visto, la aplicación tiene implementadas las funcionalidades básicas y fundamentales para el despliegue de las máquinas virtuales (y las tareas de gestión asociadas) en los laboratorios. Una mejora futura para la aplicación consistiría en aumentar las posibilidades de administración de los recursos. Así, se podrían gestionar los almacenes y volúmenes de almacenamiento, tanto para el servidor DVLS como para los hosts de forma individual. Lo mismo sucede con las interfaces de red, ya que hasta ahora todas las máquinas usan la interfaz configurada por defecto al instalar los paquetes de virtualización. Esto resulta de gran utilidad para aquellas asignaturas que requieran de varias máquinas virtuales con varias interfaces de red, como ocurre en Infraestructuras Tecnológicas para los Sistemas de Información.

El diseño actual de las aplicaciones cliente y servidor permite escalar fácilmente el número de parámetros que reciben las ordenes para definir una nueva máquina virtual. Con una modificación en los formularios del cliente y en la interfaz RESTful, es suficiente para contemplar nuevas opciones en el asistente de la creación de una nueva máquina virtual. Por ejemplo, se podría añadir más de un disco de almacenamiento, varias interfaces de red, etc.

10.2.4. Cliente VNC

Por otra parte, la gestión actual de las máquinas virtuales se realiza a través de conexiones VNC, por lo que es imprescindible disponer de un cliente VNC. Existe un cliente web denominado “noVNC” que utiliza HTML5 con WebSockets y Canvas, y tiene soporte para el cifrado del protocolo WebSocket. Así, la gestión del contenido de la máquina virtual se integraría dentro del ecosistema del proyecto, sin necesidad de depender de software externo para el acceso remoto. [69]

Con esta mejora, junto a la mejora en el número de funcionalidades en la aplicación, la herramienta de gestión se acercaría más a un entorno de gestión propio de máquinas virtuales, como el cliente web que ofrece VMware en la instalación del hipervisor ESXi, o Proxmox VE en las distribuciones Debian.

10.2.5. “Containerización”

Debido a la gran cantidad de paquetes y software del que depende el proyecto, una buena práctica para el despliegue de la aplicación consistiría en empaquetar todo el entorno en un contenedor para Docker o Kubernetes. De este modo, se facilita la distribución del sistema y no se dedica todo un equipo servidor íntegramente a la factoría de máquinas virtuales.

10.2.6. Integración de pruebas: TDD

Una buena práctica en el desarrollo de aplicaciones que comienzan a tomar gran envergadura consiste en la integración de pruebas a través del *Test Driven Development (TDD)*. De este modo, las distintas pruebas que se realicen ayudarán a desarrollar algunas funcionalidades en la aplicación. Con el uso de TDD, primero se escriben las pruebas y se ejecutan. Estas fallarán puesto que no se ha escrito código para superarlas. Una vez escrito el código, se vuelven a ejecutar las pruebas y se va refinando el código hasta que se superen las especificaciones de cada uno de los requisitos. [70]

10.2.7. Almacenamiento centralizado

Para posibilitar que los alumnos no dependan de un puesto de trabajo fijo para el desarrollo de las asignaturas, las imágenes de las máquinas virtuales podrían residir en un almacenamiento centralizado. Así, se tendría un repositorio por cada laboratorio para las imágenes de las máquinas, por lo que, si se avería alguno de estos equipos, no se pierde el trabajo realizado hasta el momento.

El problema actual para implementar esta mejora reside en la infraestructura de red, la cual no es la adecuada para el tráfico de datos que supone tener las imágenes en una cabina de discos.

Bibliografía

[1] Atlassian, “DevOps”, (en línea. 26, noviembre 2018).
Disponible en: https://es.atlassian.com/devops

[2] Red Hat, “What’s IT automation”, (en línea. 26, noviembre 2018).
Disponible en: https://www.redhat.com/en/topics/automation/whats-it-automation

[3] Ansible, (en línea. 26, noviembre 2018).
Disponible en: https://www.ansible.com/

[4] Chef, (en línea. 26, noviembre 2018).
Disponible en: https://www.chef.io/chef/

[5] VMware, “Software Defined Data Centers”, (en línea. 26, noviembre 2018).
Disponible en: https://www.vmware.com/es/solutions/software-defined-datacenter.html

[6] VMware, “Horizon”, (en línea. 26, noviembre 2018).
Disponible en: https://www.vmware.com/products/horizon.html

[7] G2Crowd, “Best Server Virtualization Software”, (en línea. 26, noviembre 2018).
Disponible en: https://www.g2crowd.com/categories/server-virtualization

[8] Xen Project, (en línea. 26, noviembre 2018).
Disponible en: https://www.xenproject.org/

[9] Bhyve, (en línea. 26, noviembre 2018).
Disponible en: https://wiki.freebsd.org/bhyve

[10] Red Hat, “Cloud Computing”, (en línea. 26, noviembre 2018).
Disponible en: https://www.redhat.com/es/topics/cloud

[11] EII, “Objetivos y competencias del Grado en Ingeniería Informática”, (en línea. 26, noviembre 2018).
Disponible en: http://www.eii.ulpgc.es/tb_university_ex/?q=objetivos-y-competencias-del-gii

[12] Wikipedia, “Licencias software”, (en línea. 26, noviembre 2018).
Disponible en: https://es.wikipedia.org/wiki/Licencia_de_software

[13] BBVA OPEN4U, “Licencias software”, (en línea. 26, noviembre 2018).
Disponible en: https://bbvaopen4u.com/es/actualidad/las-5-licencias-de-software-libre-mas-importantes-que-todo-desarrollador-debe-conocer

[14] Wikipedia, “Licencia GNU GPL”, (en línea. 26, noviembre 2018).
Disponible en: https://es.wikipedia.org/wiki/GNU_General_Public_License

[15] Red Hat Enterprise Linux, (en línea. 26, noviembre 2018).
Disponible en: https://www.redhat.com/es

[16] Repositorios EPEL, (en línea. 26, noviembre 2018).
Disponible en: https://fedoraproject.org/wiki/EPEL/es

[17] Repositorios IUS, (en línea. 26, noviembre 2018).
Disponible en: https://ius.io/

[18] Libvirt, (en línea. 26, noviembre 2018).
Disponible en: https://libvirt.org/

[19] uWSGI, (en línea. 26, noviembre 2018).
Disponible en: https://uwsgi-docs.readthedocs.io/en/latest/

[20] Wikipedia, “Licencia LGPL”, (en línea. 26, noviembre 2018).
Disponible en: https://es.wikipedia.org/wiki/GNU_Lesser_General_Public_License

[21] Wikipedia, “PolicyKit”, (en línea. 26, noviembre 2018).
Disponible en: https://es.wikipedia.org/wiki/PolicyKit

[22] Wikipedia, “Licencia MIT”, (en línea. 26, noviembre 2018).
Disponible en: https://es.wikipedia.org/wiki/Licencia_MIT

[23] Angular, (en línea. 26, noviembre 2018).
Disponible en: https://angular.io/

[24] VMware, “Clarity Design System”, (en línea. 26, noviembre 2018).
Disponible en: https://vmware.github.io/clarity/

[25] PyJWT, (en línea. 26, noviembre 2018).
Disponible en: https://pyjwt.readthedocs.io/en/latest/

[26] Python-PAM, (en línea. 26, noviembre 2018).
Disponible en: https://pypi.org/project/python-pam/

[27] Wikipedia, “Licencia BSD”, (en línea. 26, noviembre 2018).
Disponible en: https://es.wikipedia.org/wiki/Licencia_BSD

[28] Flask, (en línea. 26, noviembre 2018).
Disponible en: http://flask.pocoo.org/

[29] Nginx, (en línea. 26, noviembre 2018).
Disponible en: https://www.nginx.com/

[30] Wikipedia, “Licencia PSFL”, (en línea. 26, noviembre 2018).
Disponible en: https://es.wikipedia.org/wiki/Python_Software_Foundation_License

[31] Python, (en línea. 26, noviembre 2018).
Disponible en: https://www.python.org/

[32] JetBrains, (en línea. 26, noviembre 2018).
Disponible en: https://www.jetbrains.com/

[33] JetBrains, “PyCharm”, (en línea. 26, noviembre 2018).
Disponible en: https://www.jetbrains.com/pycharm/

[34] JetBrains, “WebStorm”, (en línea. 26, noviembre 2018).
Disponible en: https://www.jetbrains.com/webstorm/

[35] Postman, “EULA”, (en línea. 26, noviembre 2018).
Disponible en: https://www.getpostman.com/licenses/postman_eula

[36] “Software Engineering, A Practitioner’s Approach, 5th Edition, Roger S. Pressman, Ph.D.”

[37] Apache Cordova, (en línea. 26, noviembre 2018).
Disponible en: https://cordova.apache.org/

[38] Phonegap, (en línea. 26, noviembre 2018).
Disponible en: https://phonegap.com/

[39] Google, “Progressive Web Apps”, (en línea. 26, noviembre 2018).
Disponible en: https://developers.google.com/web/progressive-web-apps/

[40] Xataka, “¿qué es una aplicación web progresiva?”, (en línea. 26, noviembre 2018).
Disponible en: https://www.xataka.com/basics/que-es-una-aplicacion-web-progresiva-o-pwa

[41] Arsys, “Service Workers”, (en línea. 26, noviembre 2018).
Disponible en: https://www.arsys.es/blog/programacion/service-worker/

[42] Campus MVP, “Las 5 principales ventajas de usar Angular para crear aplicaciones web”, (en línea. 26, noviembre 2018).
Disponible en: https://www.campusmvp.es/recursos/post/las-5-principales-ventajas-de-usar-angular-para-crear-aplicaciones-web.aspx

[43] Jaxenter, “Angular 7”, (en línea. 26, noviembre 2018).
Disponible en: https://jaxenter.com/angular-7-release-151024.html

[44] VMware, “Wizards con Clarity Design System”, (en línea. 26, noviembre 2018).
Disponible en: https://vmware.github.io/clarity/documentation/v0.13/wizards

[45] Red Hat, “What is virtualization”, (en línea. 26, noviembre 2018).
Disponible en: https://www.redhat.com/es/topics/virtualization/what-is-virtualization

[46] Red Hat, “Virtualization Getting Started”, (en línea. 26, noviembre 2018).
Disponible en: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/virtualization_getting_started_guide/chap-virtualization_getting_started-what_is_it

[47] CentOS, “About”, (en línea. 26, noviembre 2018).
Disponible en: https://www.centos.org/about/

[48] Wikipedia, “CentOS”, (en línea. 26, noviembre 2018).
Disponible en: https://es.wikipedia.org/wiki/CentOS

[49] Libvirt Wiki, “What is libvirt”, (en línea. 26, noviembre 2018).
Disponible en: https://wiki.libvirt.org/page/FAQ#What_is_libvirt.3F

[50] Libvirt, “Bindings”, (en línea. 26, noviembre 2018).
Disponible en: https://libvirt.org/bindings.html

[51] Red Hat, “Virtualization Deployment and Administration Guide”, (en línea. 26, noviembre 2018).
Disponible en: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/virtualization_deployment_and_administration_guide/sect-installing_the_virtualization_packages-installing_virtualization_packages_on_an_existing_red_hat_enterprise_linux_system

[52] Red Hat, “What is KVM”, (en línea. 26, noviembre 2018).
Disponible en: https://www.redhat.com/es/topics/virtualization/what-is-KVM

[53] Wikipedia, “Lenguaje Unificado de Modelado”, (en línea. 26, noviembre 2018).
Disponible en: https://es.wikipedia.org/wiki/Lenguaje_unificado_de_modelado

[54] Microsoft, “Single-Page Applications”, (en línea. 26, noviembre 2018).
Disponible en: https://msdn.microsoft.com/en-us/magazine/dn463786.aspx

[55] Angular University, “What are the benefits of SPA”, (en línea. 26, noviembre 2018).
Disponible en: https://blog.angular-university.io/why-a-single-page-application-what-are-the-benefits-what-is-a-spa/

[56] Angular, “Styleguide”, (en línea. 26, noviembre 2018).
Disponible en: https://angular.io/guide/styleguide

[57] Angular, “CanActivate”, (en línea. 26, noviembre 2018).
Disponible en: https://angular.io/api/router/CanActivate

[58] Flask, “Extensions”, (en línea. 26, noviembre 2018).
Disponible en: http://flask.pocoo.org/extensions/

[59] Flask, “Flask-Cors”, (en línea. 26, noviembre 2018).
Disponible en: https://flask-cors.readthedocs.io/en/latest/

[60] Flask, “Flask-Migrate”, (en línea. 26, noviembre 2018).
Disponible en: https://flask-migrate.readthedocs.io/en/latest/

[61] Flask, “Flask-SQLAlchemy”, (en línea. 26, noviembre 2018).
Disponible en: http://flask-sqlalchemy.pocoo.org/latest/

[62] Red Hat, “PAM Configuration Files”, (en línea. 26, noviembre 2018).
Disponible en: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system-level_authentication_guide/pam_configuration_files

[63] Red Hat, “Hardening your system with tools and services”, (en línea. 26, noviembre 2018).
Disponible en: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/security_guide/chap-hardening_your_system_with_tools_and_services#sec-Password_Security

[64] Libguestfs, “virt-sysprep”, (en línea. 26, noviembre 2018).
Disponible en: http://libguestfs.org/virt-sysprep.1.html

[65] Celery Project, (en línea. 26, noviembre 2018).
Disponible en: http://www.celeryproject.org/

[66] Vinta, “Celery overview architecture and how it works”, (en línea. 26, noviembre 2018).
Disponible en: https://www.vinta.com.br/blog/2017/celery-overview-archtecture-and-how-it-works/

[67] Wikipedia, “Websocket”, (en línea. 26, noviembre 2018).
Disponible en: https://es.wikipedia.org/wiki/WebSocket

[68] Socket.IO, (en línea. 26, noviembre 2018).
Disponible en: https://socket.io

[69] noVNC, (en línea. 26, noviembre 2018).
Disponible en: http://novnc.com

[70] Scotch, “Build a RESTful API with Flask. The TDD way”, (en línea. 28, noviembre 2018).
Disponible en: https://scotch.io/tutorials/build-a-restful-api-with-flask-the-tdd-way#toc-running-our-tests

Anexos

Anexo I: instalación y configuración de la infraestructura y aplicación.

DIS vLab Server (DVLS) es una aplicación web *fullstack* para gestionar los laboratorios virtuales de la Escuela de Ingeniería Informática y del Departamento de Informática y Sistemas de la Universidad de Las Palmas de Gran Canaria. Realmente, un laboratorio virtual es un concepto lógico creado a partir del conjunto de ordenadores físicos con una instalación de CentOS 7.x GNOME Desktop, y los paquetes de virtualización necesarios, en una sala del edificio. Así mismo, DVLS se ejecuta como servicio sobre una instalación de CentOS 7.x Minimal en un equipo o servidor de altas prestaciones. Esta estación de trabajo gestiona el hipervisor local para personalizar y generar plantillas que posteriormente serán desplegadas en los laboratorios virtuales, a través de una conexión SSH con los hipervisores de los equipos remotos.

Prerrequisitos

Para desplegar la aplicación DVLS es necesario contar con una instalación CentOS 7.x Minimal actualizada, con los repositorios EPEL e IUS y los siguientes grupos y paquetes:

- “Virtualization Platform” (grupo)
- “Virtualization Hypervisor” (grupo)
- “Virtualization Tools” (grupo)
- “Virtualization Client” (grupo)
- “Development” (grupo)
- “libvirt-devel.x86_64”
- “libguestfs-tools”
- “python36u”
- “python36u-devel”
- “python36u-pip”
- “nginx”
- “openssl”

Repositorio y dependencias

El código fuente será distribuido a través de un repositorio actualizado y accesible, o en un fichero comprimido junto a este documento. Para comenzar con la instalación y preparación del equipo servidor, clone el repositorio remoto o, en su lugar, descomprima el código fuente del fichero comprimido y sitúelo en el directorio recomendado */usr/lib/dvls*. Dicho repositorio no incluye las dependencias y librerías que se necesitan

para su ejecución, por lo que se necesita del gestor de entornos virtuales '*virtualenv*' de Python para descargar las dependencias. Para ello, utilice el gestor de paquetes Python '*pip*' y posteriormente cree un nuevo entorno virtual en el directorio raíz del código fuente de la aplicación DVLS. Active el entorno virtual e instale con la herramienta '*pip*' las dependencias registradas en el fichero *requirements*.txt.

```
# pip3.6 install virtualenv
# cd /usr/lib/dvls
# virtualenv venv
# source venv/bin/actívate
(venv) # pip install -r requirements.txt
```

Configuración del sistema

Usuarios y grupos

La aplicación DVLS se ejecutará como servicio del sistema a través de un usuario no privilegiado llamado '*dvls*', por cuestiones de seguridad. Si no lo creó en el proceso de instalación de CentOS 7.x, añádalo al sistema actual. Las credenciales de este nuevo usuario serán las de acceso a la aplicación web, por lo que la contraseña debe ser bastante segura. Además, el usuario '*dvls*' debe formar parte de los grupos del sistema y '*libvirt*' para garantizar ciertos permisos que permitan la ejecución de la aplicación sin ningún error.

```
# useradd dvls
# passwd dvls
# usermod -a -G libvirt dvls
```

Firewall

Por defecto, CentOS 7.x trae activado el firewall, por lo que debe añadir una regla para el tráfico entrante HTTP o HTTPS, en función de si elige utilizar SSL en la configuración del proxy inverso que se verá posteriormente.

```
# firewall-cmd --add-service=http --permanent
# firewall-cmd —-add-service=https --permanent
# firewall-cmd --reload
```

PolicyKit

Todos los usuarios sin privilegios de administrador que formen parte del grupo '*libvirt*' tienen asignada una política de acceso a los recursos de la API *libvirt* por defecto. El fichero que especifica dicha regla se puede encontrar en la ruta */usr/share/polkit-1/rules.d/50-libvirt.rules*.

Sin embargo, todos los recursos que sean gestionados, a nivel del sistema de ficheros, a través de una conexión privilegiada con *libvirt*, tendrán como propietario al usuario y grupo '*root*'. Por ello, todas las órdenes que ejecute el usuario '*dvls*' y haga uso de estos recursos terminarán con un código de error.

En el proceso de clonación de un dominio a plantilla se ejecuta el binario */usr/bin/virt-sysprep* para descontextualizar los discos de un dominio. Como se explicó en el punto anterior, estos recursos del dominio se encuentran bajo el dominio del usuario '*root*', por lo que hay que añadir una acción y regla al sistema de PolicyKit para controlar la ejecución de dicho binario como usuario no privilegiado a través de pkexec. Esta herramienta se puede considerar análoga a "sudo", ya que si no se especifica un usuario el binario es ejecutado como usuario administrativo.

En primer lugar, hay que definir una nueva acción dentro del contexto de PolicyKit. Para ello, se crea el fichero */usr/share/polkit-1/actions/es.ulpgc.dis.dvls.virt-sysprep.policy* y se añade el siguiente contenido:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE policyconfig PUBLIC
 "-//freedesktop//DTD PolicyKit Policy Configuration 1.0//EN"
 "http://www.freedesktop.org/standards/PolicyKit/1.0/policyconfig.dtd">
<policyconfig>
  <vendor>DVLS</vendor>
  <vendor_url>https://www.dis.ulpgc.es/</vendor_url>

  <action id="es.ulpgc.dis.dvls.virt-sysprep">
    <description>Run 'virt-sysprep'</description>
    <message>Authentication is required to descontextualize domain disks as
non-root user</message>
    <defaults>
      <allow_inactive>no</allow_inactive>
      <allow_active>no</allow_active>
    </defaults>
    <annotate key="org.freedesktop.policykit.exec.path">/usr/bin/virt-
sysprep</annotate>
  </action>
</policyconfig>
```

Una vez añadida la acción, se crea una regla para permitir la ejecución de dicha acción a cualquier usuario que forme parte del grupo '*libvirt*'. Para ello, se crea el fichero */etc/polkit-1/rules.d/60-virt-sysprep.rules* con el siguiente contenido:

```
polkit.addRule(function(action, subject){
    if (action.id == "es.ulpgc.dis.dvls.virt-sysprep" &&
subject.isInGroup("libvirt")) {
        return polkit.Result.YES;
    }
});
```

Pluggable Authentication Modules

La aplicación utiliza el mecanismo de autenticación centralizado llamado Módulos de Autenticación Conectables (en inglés *Pluggable Authentication Modules - PAM*) para autenticar al usuario '*dvls*' en el sistema. Esto permite que la aplicación se desvincule de la lógica de negocio relacionada con el control de acceso y únicamente tenga que tratar la validación en la aplicación por *JSON Web Tokens*. Por defecto, la aplicación hace uso de un servicio personalizado para la aplicación, ya que únicamente hace falta autenticar al usuario contra el fichero */etc/shadow*. Para crear dicho servicio, ejecute la siguiente orden:

```
# echo "auth required pam_unix.so" > /etc/pam.d/dvls
```

Recursos NFS

Para que los equipos clientes puedan generar dominios nuevos a partir de las imágenes y definiciones de las plantillas, es imprescindible que tengan acceso remoto. Por defecto, DVLS está configurado para que estos ficheros se generen en el directorio /var/lib/dvls/ por lo que se debe indicar el directorio en el fichero */etc/exports* y añadir las reglas pertinentes al firewall del sistema.

Servicio DVLS

La aplicación web se ejecutará como un servicio del sistema. Para ello, cree un fichero en el directorio */etc/systemd/system/* cuyo nombre será el del servicio, p.ej. *dvls.service*. Abra el fichero con su editor de texto preferido y copie el siguiente contenido:

```
[Unit]
Description=uWSGI instance for DIS vLab Server
After=network.target

[Service]
WorkingDirectory=/usr/lib/dvls
Environment="PATH=/usr/lib/dvls/venv/bin:/usr/bin"
ExecStart=/usr/lib/dvls/venv/bin/uwsgi --ini dvls.ini

[Install]
WantedBy=multi-user.target
```

Antes de habilitar e iniciar el servicio, cambie el usuario y grupo propietario del directorio de la aplicación para poder ejecutarla:

```
# chown -R dvls:dvls /usr/lib/dvls
```

Configuración de Nginx

Secure Sockets Layer

Es una buena práctica utilizar el protocolo HTTPS en lugar de HTTP en aplicaciones web dentro de entornos corporativos, cuyo tráfico de red sea susceptible de ser monitoreado, redirigido o manipulado. Por esta razón, es recomendable generar un certificado auto-firmado usando OpenSSL. Si ya dispone de un certificado firmado por una autoridad certificadora, obvie este paso y vaya a la configuración del proxy inverso.

Asegúrese de que exista el directorio */etc/nginx/ssl*, donde se almacenará la clave y el certificado del servidor. Para generar la clave y el certificado use la siguiente orden:

```
# openssl req -x509 -nodes -days 3650 -newkey rsa:2048 -keyout
/etc/nginx/ssl/nginx.key -out /etc/nginx/ssl/nginx.crt
```

Configuración del proxy inverso

En el proyecto se incluye la dependencia de uWSGI, que es una aplicación para servir aplicaciones Python junto a servidores web como Nginx en este caso, que actuará de proxy inverso con soporte directo para el protocolo nativo de uWSGI. De esta forma, Nginx simplemente actuará como punto de escucha para peticiones HTTP o HTTPS que se redireccionarán al servidor uWSGI.

Para configurarlo como tal, cree un nuevo fichero de configuración bajo el directorio de configuraciones Nginx, p.ej. */etc/nginx/config.d/dvls.conf* e incluya la siguiente declaración de servidor:

```
server {
    listen 443 ssl;
    server_name dvls.dis.ulpgc.es;
    ssl_certificate /etc/nginx/ssl/nginx.crt;
    ssl_certificate_key /etc/nginx/ssl/nginx.key;

    location / {
        include uwsgi_params;
        uwsgi_pass unix:/usr/lib/dvls/dvls.sock;
        uwsgi_read_timeout 999999;
    }
}
```

Acceso a la interfaz web

Con la configuración anterior debería ser capaz de acceder a la interfaz web vía HTTPS con las credenciales del usuario '*dvls*' del sistema.

Solución de problemas

Si obtiene un error 502 proveniente de Nginx al acceder a la interfaz web y está usando SELinux en modo '*enforcing*', se debe a una política incorrecta de SELinux sobre el socket de la aplicación DVLS. Para corregirlo, necesita permitir el acceso a este recurso a través de una política generada con la herramienta *audit2allow*. Con la instalación de CentOS 7.x Minimal necesita instalar el paquete *policycoreutils-python* para utilizar *audit2allow*.

Una vez haya accedido a la interfaz web y obtenido el error 502 de Nginx, abra un terminal y ejecute las siguientes órdenes:

```
# grep nginx /var/log/audit/audit.log | audit2allow -M nginx
# semodule -i nginx.pp
```

De esta forma, se está añadiendo una política local de acceso generada a partir del mensaje de error que provoca que Nginx no se pueda comunicar con el socket de DVLS.

Anexo II: guía básica del usuario

La herramienta de gestión, denominada DIS vLab Server, consta de una interfaz web cómoda y simple para la gestión de los recursos. Esta interfaz, DIS vLab Client, ofrece un flujo de trabajo que comienza con la creación y configuración de un dominio y finaliza con el despliegue de la plantilla en un laboratorio. Para ello se tiene una serie de pasos y funcionalidades que se describen a continuación:

Inicio de sesión

Para acceder a la herramienta de gestión hay que introducir las credenciales del usuario 'dvls' que se configuró previamente en la instalación de la infraestructura y aplicación. Se trata de una autenticación en el sistema a través de PAM, por lo que no se tienen cuentas personales, ya que se trata de una herramienta de gestión para un procedimiento común entre todo el personal (ilustración 15).

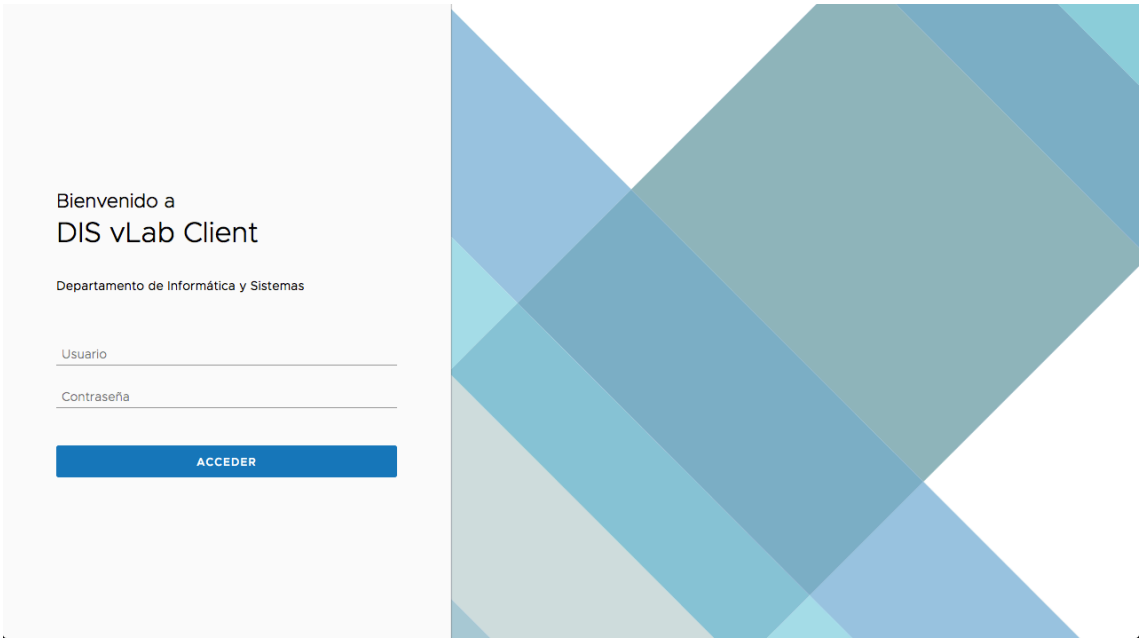

Ilustración 15: DVLC - Página de acceso a la aplicación

Una vez se tiene acceso, lo primero que se muestra es un "*dashboard*" con información sobre el sistema de información: breve resumen del equipo servidor, carga de la memoria RAM, dominios activos frente a los definidos en el hipervisor, laboratorios y plantillas creadas. Además, en la parte inferior de la disposición, se encuentra una tabla con los registros de los eventos que van sucediendo en el sistema (ilustración 16).

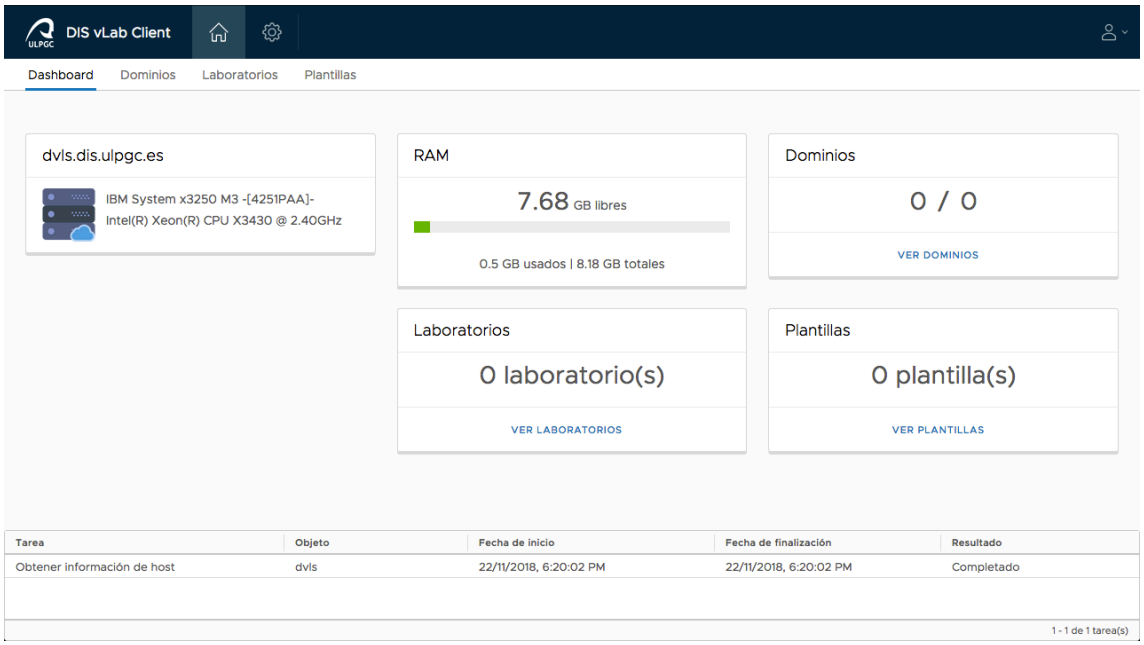

Ilustración 16: DVLC - Página principal "dashboard"

Creación de una máquina virtual

Desde la pestaña “Dominios” se accede al módulo de gestión de los dominios en el hipervisor local. Así, se tiene un listado con los dominios definidos con una serie de características. La aplicación permite gestionar la energía (encender, apagar y reiniciar) de los dominios, así como la posibilidad eliminarlos de forma permanente (incluida la imagen) o clonarlos a plantilla.

La instalación de un nuevo dominio se realiza a través de un asistente para personalizar una configuración mínima de una máquina virtual. Para ello, se hace clic en el botón “Nuevo dominio”. En primer lugar, hay que seleccionar el método de instalación de la máquina virtual. En el caso de esta guía, se seguirá el método por instalación local a través de una imagen ISO. Cabe destacar que los pasos del asistente cambiarán en función del método escogido.

Para el método de instalación local se debe introducir la ruta del fichero con la imagen ISO del sistema operativo o, en su caso, la ruta del volumen con el CDROM (ilustración 17).

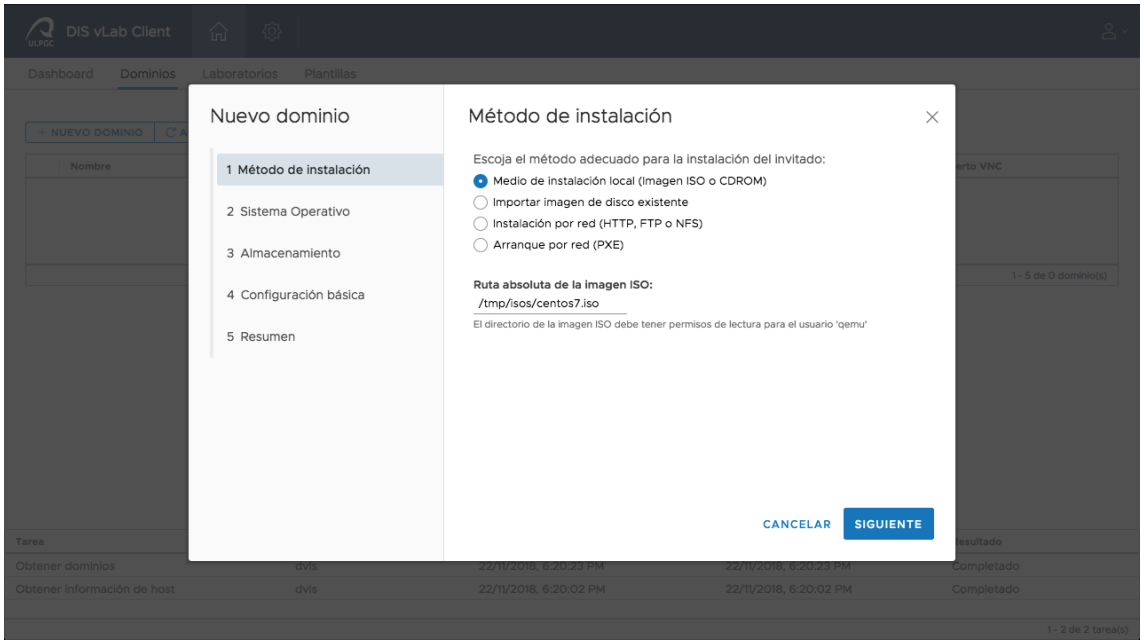

Ilustración 17: DVLC – Selección del método de instalación del sistema operativo invitado.

El siguiente paso consiste en seleccionar de una lista el tipo de sistema operativo invitado que contiene la imagen seleccionada. Se trata de una funcionalidad para que, en futuras versiones de la aplicación, se personalicen las opciones de configuración de forma automática en función del sistema operativo seleccionado. En este caso, la imagen contiene el instalador de CentOS 7 (ilustración 18).

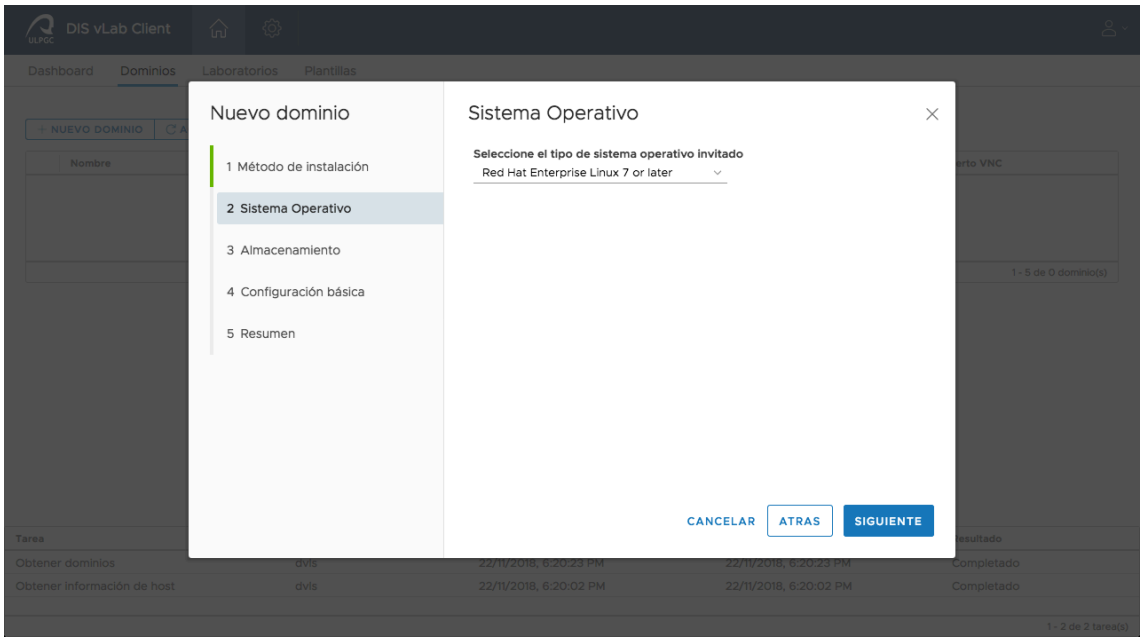

Ilustración 18: DVLC - Selección del sistema operativo invitado.

Este asistente configura un único volumen de almacenamiento local sobre el que se va a instalar el sistema operativo. El fichero se guardará con el mismo nombre de la máquina virtual, por lo que lo único que hay que especificar es su tamaño (ilustración 19).

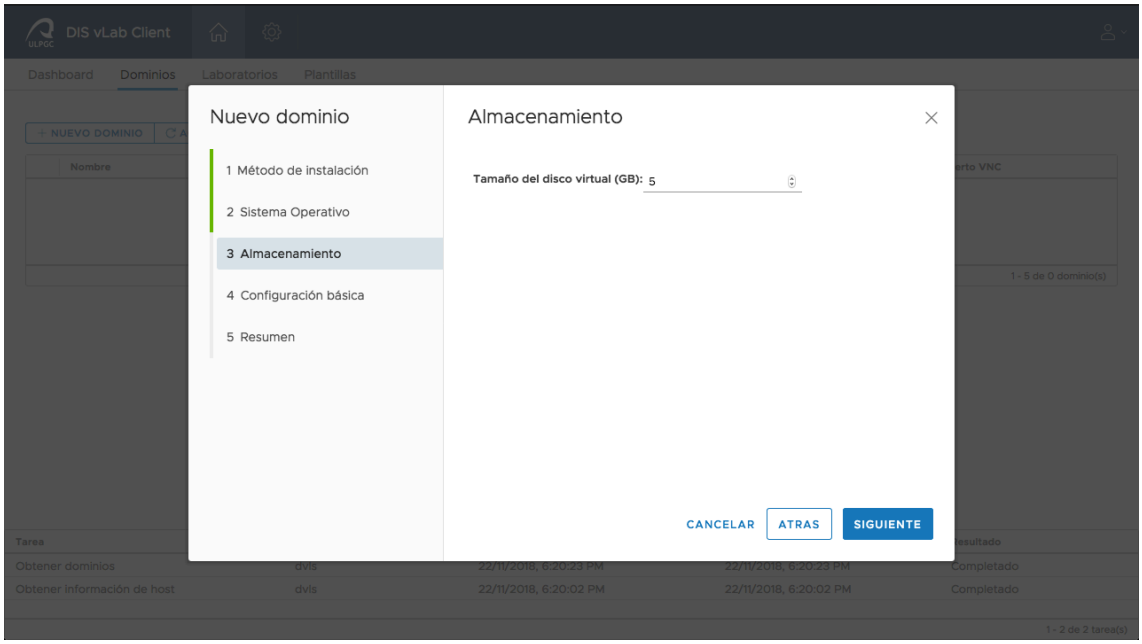

Ilustración 19: DVLC - Especificación del tamaño del disco virtual del dominio.

Por último, se especifican los parámetros correspondientes a la configuración básica del hardware de la máquina, así como su nombre y los gráficos VNC para la gestión remota (ilustración 20).

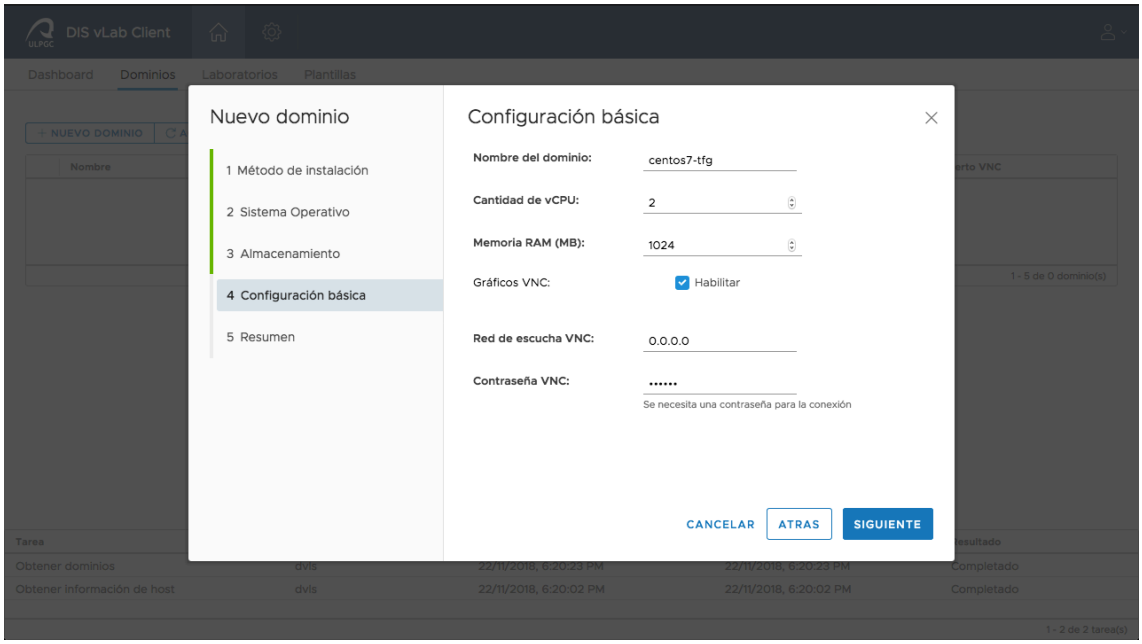

Ilustración 20: DVLC - Parámetros de la configuración básica del dominio.

Antes de finalizar el asistente, se muestra un resumen de todos los campos rellenados (ilustración 21).

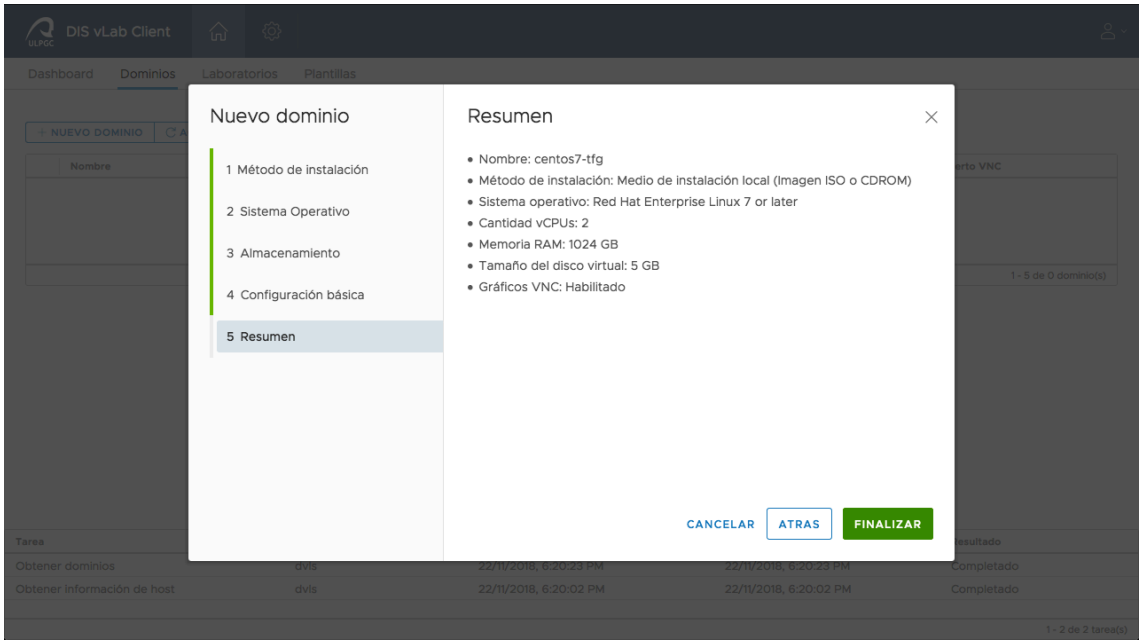

Ilustración 21: DVLC - Resumen de la configuración.

Al hacer clic sobre el botón “Finalizar”, se ejecuta una tarea en segundo plano que lanza la orden virt-install. Si esta ejecución finaliza con éxito, se mostrará el dominio en el listado de la página. Por defecto, el dominio se enciende una vez es definido, por lo que se le asigna un puerto VNC para la gestión remota que se explica a continuación (ilustración 22).

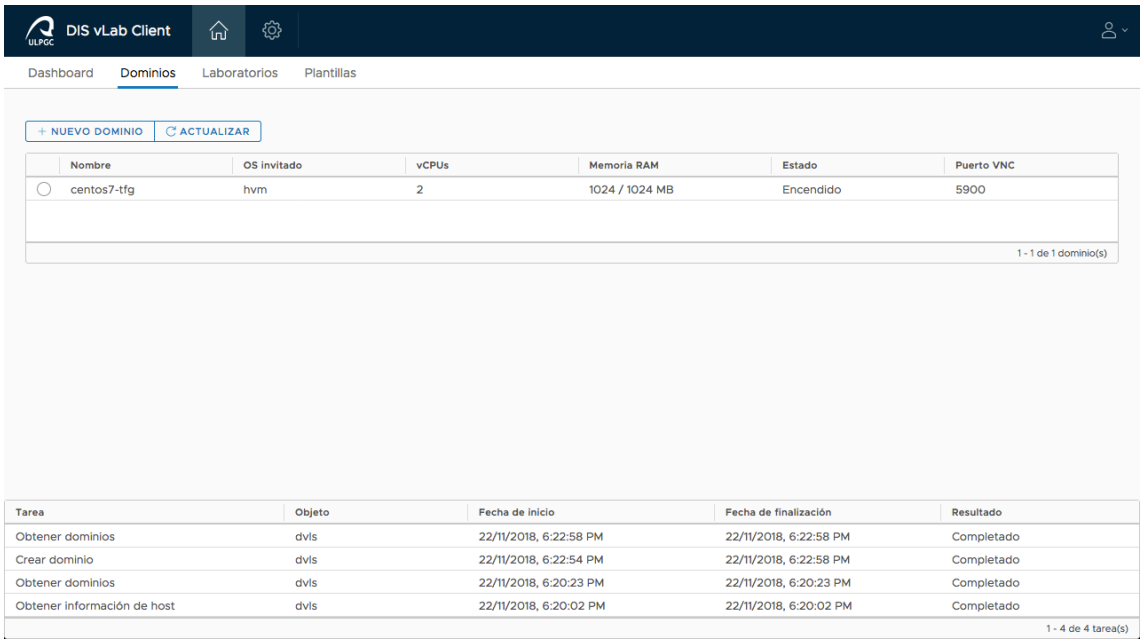

Ilustración 22: DVLC - Listado de los dominios definidos en el sistema.

Gestión de los dominios

Para llevar a cabo las tareas de instalación del sistema operativo o de gestión del propio sistema, se realiza una conexión a través de VNC. Para ello, en el listado se informa sobre el puerto que se ha asignado a la máquina en cuestión.

Desde un cliente VNC, se introduce el nombre del servidor DVLS acompañado del puerto VNC. Actualmente, la conexión no está configurada para que sea cifrada, por lo que es necesario establecer una contraseña en el asistente de la creación del dominio (ilustración 23 y 24).

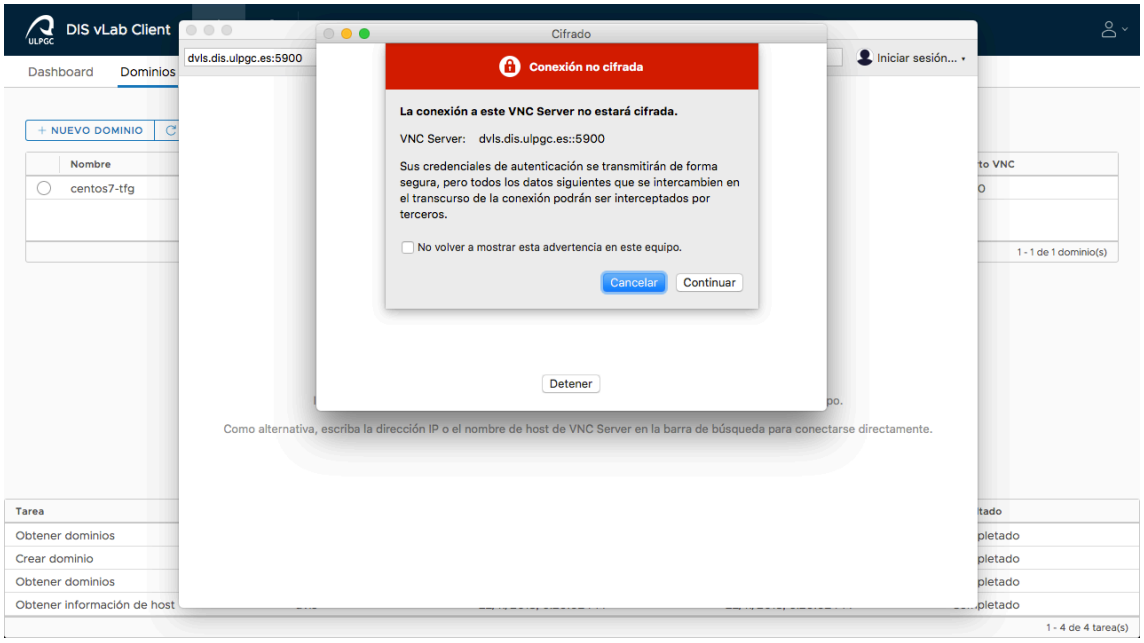

Ilustración 23: DVLC - Se alerta de que la conexión VNC no está cifrada.

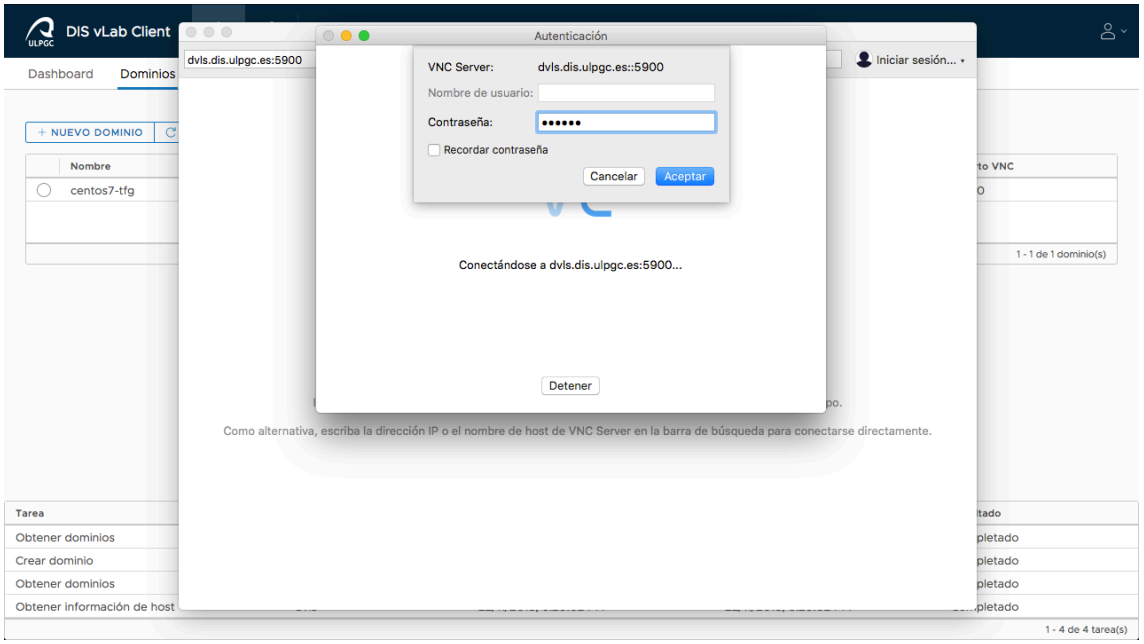

Ilustración 24: DVLC - Conexión VNC, credenciales para la conexión.

Si la conexión se realiza con éxito, se verá el contenido de la consola gráfica en una ventana del cliente VNC (ilustración 25). A partir de este punto, la configuración y gestión del escritorio virtualizado queda bajo la responsabilidad del personal del CCDIS.

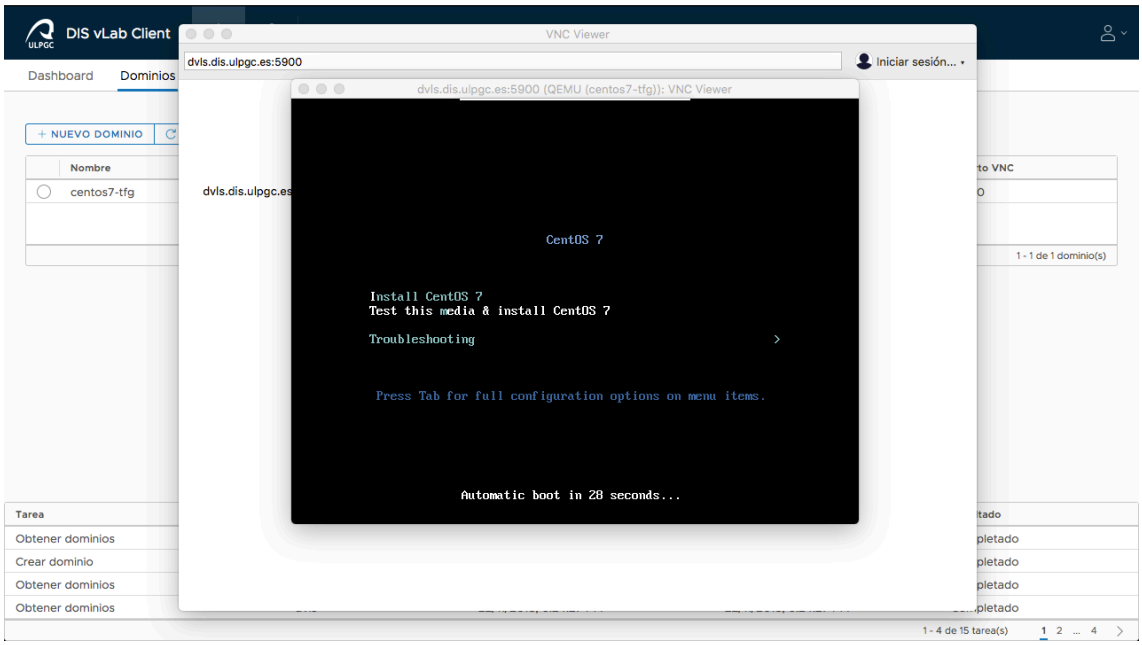

Ilustración 25: DVLC - Consola gráfica VNC del dominio.

Generar una plantilla a partir de un dominio existente

Una vez se ha realizado la configuración del sistema del dominio, se procede a generar una plantilla. Además, se da la opción para sistemas Linux de descontextualizar la imagen del sistema operativo, ya que si se trabaja con un sistema Windows se debe utilizar su utilidad *sysprep.exe*.

Para generar una plantilla, se selecciona el dominio en cuestión y en el menú desplegable se hace clic sobre “Clonar a plantilla”. Se abre una ventana modal en la que se debe introducir el nombre y una descripción significativa de la plantilla para facilitar la identificación de su contenido de cara al despliegue (ilustración 26).

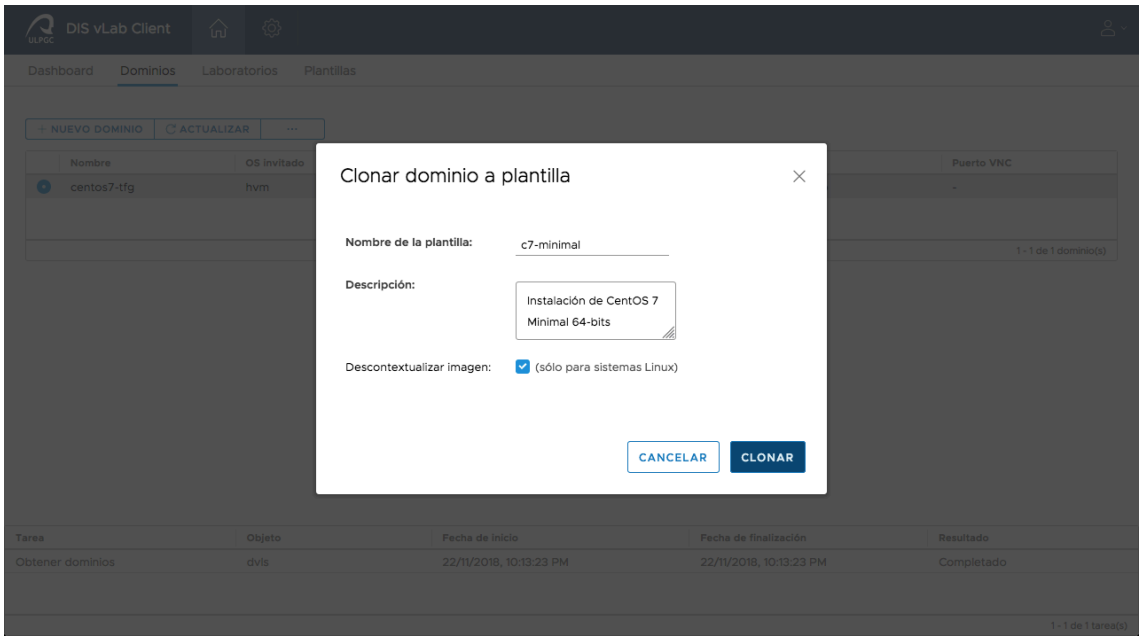

Ilustración 26: DVLC - Formulario para la generación de la plantilla.

El tiempo del proceso dependerá de la complejidad de la máquina virtual, principalmente por la imagen del disco de almacenamiento. Por ello se muestra una barra de proceso mientras se ejecuta la orden (ilustración 27).

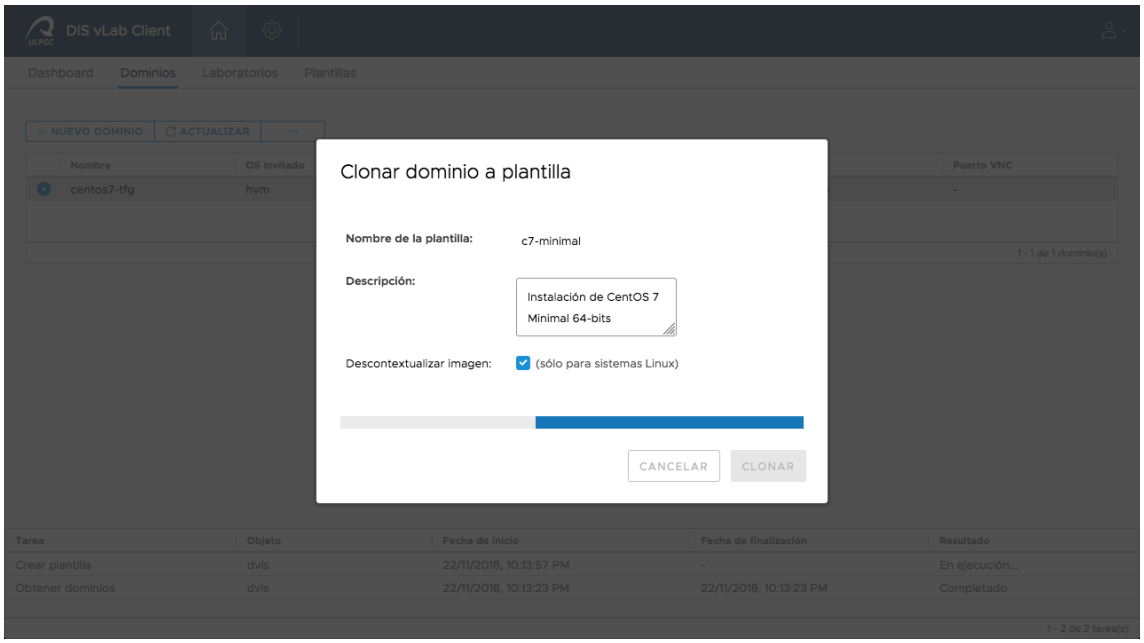

Ilustración 27: DVLC - Barra de progreso durante la ejecución del proceso de clonación.

Si la ejecución acaba con éxito, en el módulo de “Plantillas” en la aplicación se muestra la plantilla generada lista para ser desplegada (ilustración 28).

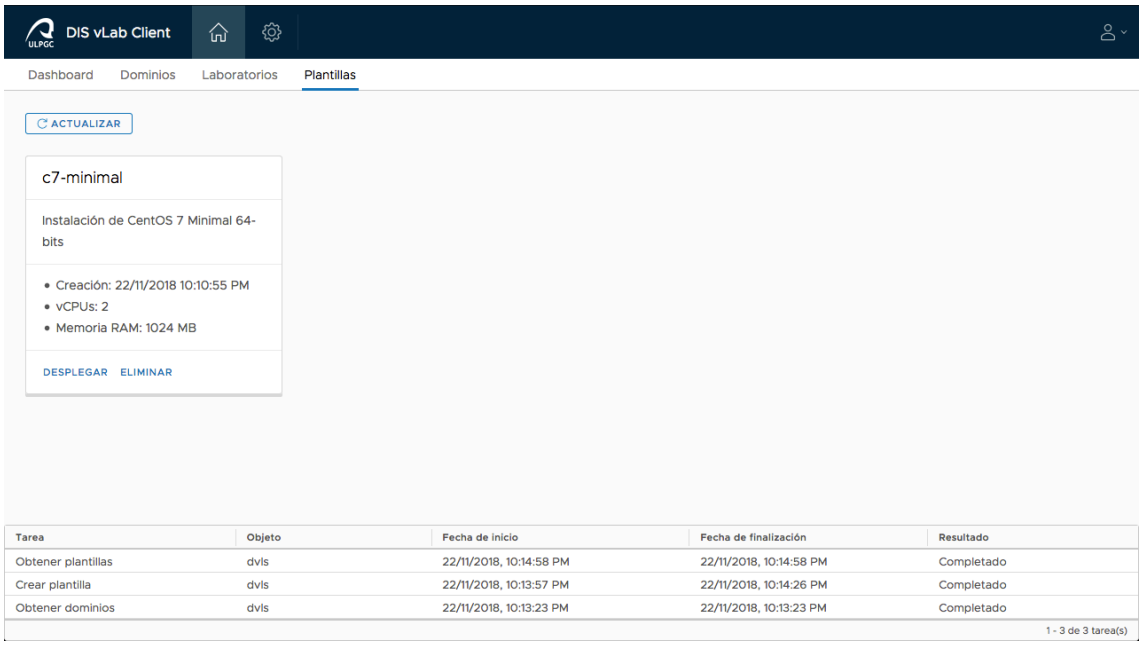

Ilustración 28: DVLC - Listado de las plantillas en el sistema.

Crear un laboratorio

Una plantilla no puede ser desplegada sin la existencia de entidades “laboratorio” en la base de datos, por lo que es necesario definir un laboratorio en el sistema. Esta definición consta de un código de laboratorio, así como una breve descripción, el rango de direcciones IP para poder identificar los equipos que lo conforman y las características hardware de los equipos para un carácter orientativo, ya que no se garantiza que todo el laboratorio cuente con el mismo tipo de equipos.

Cabe destacar que el rango de direcciones IP cobra gran importancia en la definición del laboratorio, ya que se registra un equipo por cada IP dentro de dicho rango para poder realizar una conexión remota con el hipervisor del equipo y así generar la máquina a partir de la plantilla seleccionada.

Para ello, desde la vista del módulo “Laboratorios” se hace clic sobre el botón “Nuevo laboratorio”. Aparecerá una ventana modal con el formulario para crear un laboratorio (ilustración 29).

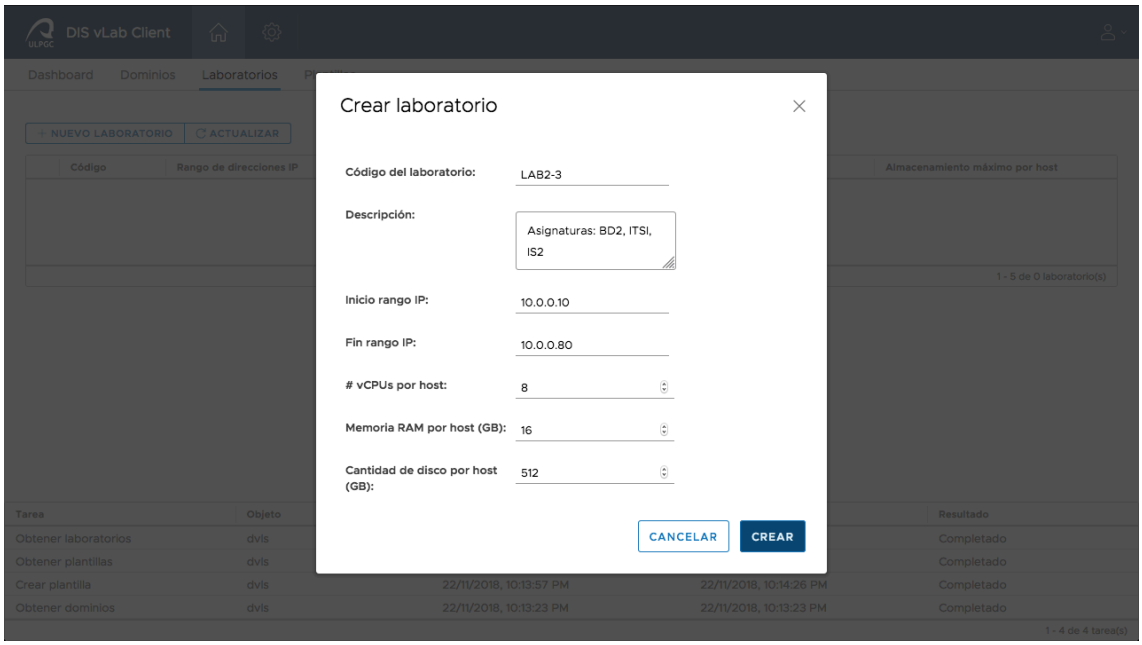

Ilustración 29: DVLC – Formulario para la creación de un nuevo laboratorio.

Despliegue de una plantilla

El último paso consiste en desplegar una de las plantillas registradas en el sistema. Para ello, simplemente se hace clic en el botón “Desplegar” de la plantilla en cuestión y se introduce en el formulario que se muestra, el nombre que tendrán las máquinas en los equipos de destino y el laboratorio objetivo (ilustración 30).

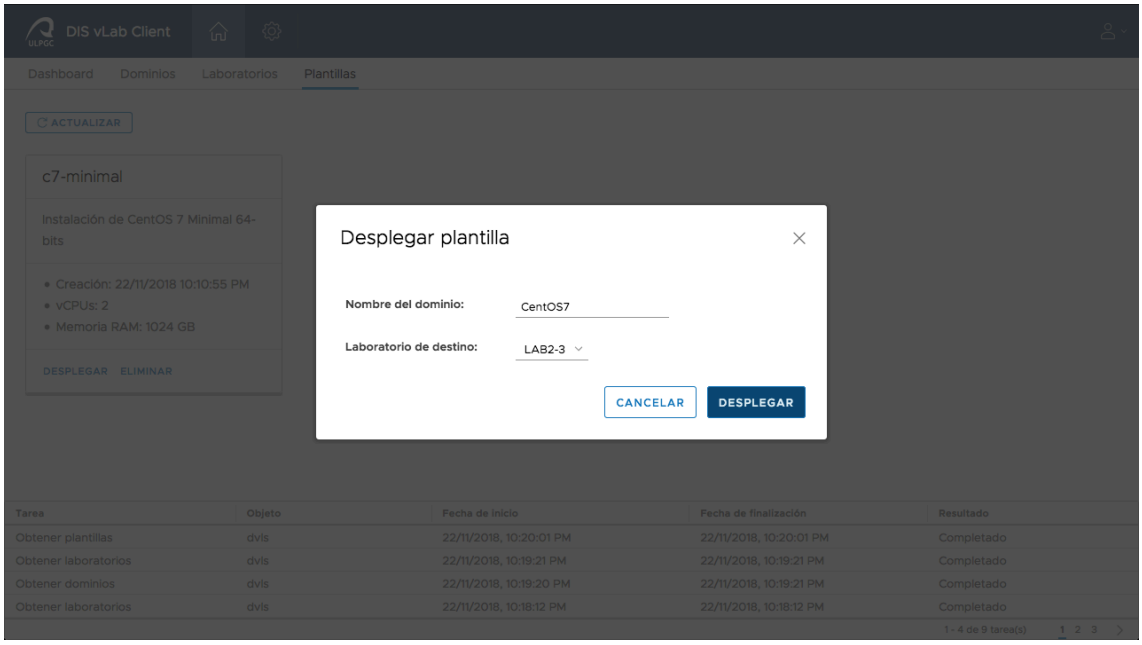

Ilustración 30: DVLC - Formulario para el despliegue de la plantilla.

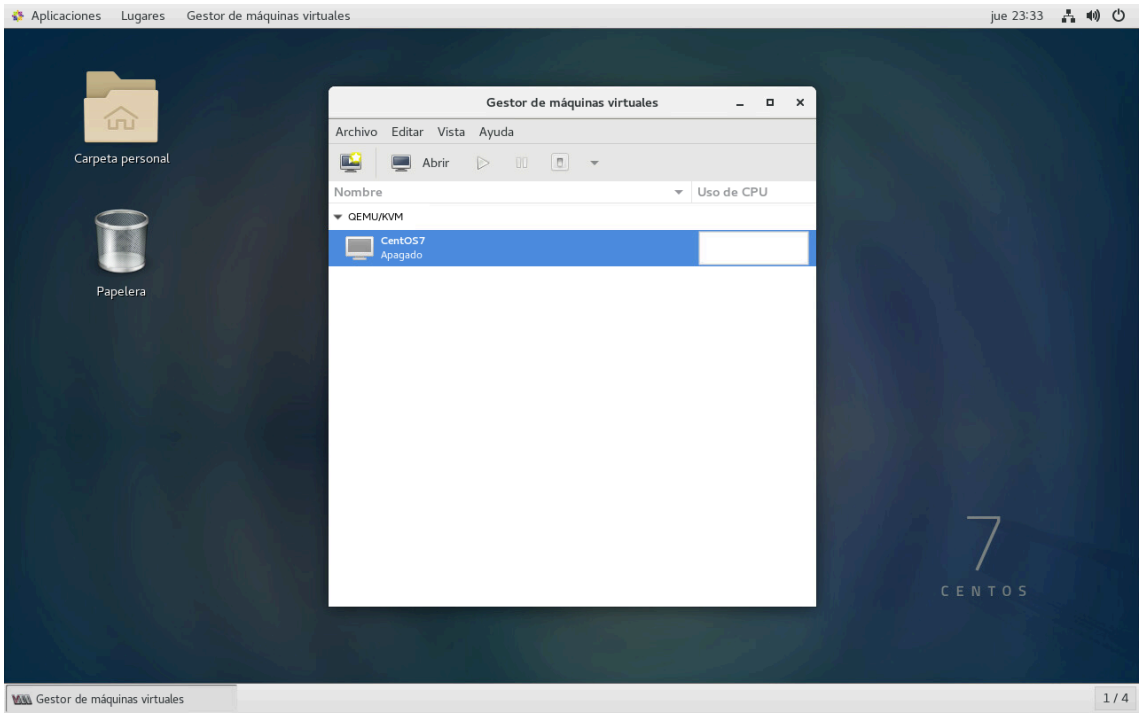

Ilustración 31: máquina virtual generada en un equipo de laboratorio.

Anexo III: recursos RESTful

En este anexo se presenta la estructura de datos que se intercambia en cada llamada cliente-servidor con la API RESTful, por si en un trabajo futuro se desea usar o modificar esta interfaz.

Dashboard

A continuación, se muestra la estructura de datos que se obtiene en primera instancia al autenticarse en la aplicación.

GET /api/dashboard/ → (Obtiene información básica sobre el sistema: información hardware, número de dominios definidos y cantidad de plantillas registradas)

```
{
    "system": {
        "hostname": "dvls.dis.ulpgc.es",
        "manufacturer": "IBM",
        "model": "x3250 M3"
    },
    "domains": {
        "total": 2,
        "active": 1
     },
    "cpu": {
        "arch": "x86_64",
        "model": "Intel ® Xeon x3430 2.40GHz",
        "cpus": 2,
        "freq": 2493, // MHz
        "numa_nodes": 1,
        "sockets_per_node": 2,
        "cores_per_socket": 1,
        "threads_per_core": 1
    },
    "memory": {
        "total": 4, // GB
        "used": 1.86, // GB
        "free": 2.14, // GB
        "load": 46 // Porcentaje
    },
    "templates": 3
}
```

Dominios

En este aparado se muestran los recursos asociados al modelo de los dominios.

POST /api/domains/ → (Creación de un nuevo dominio, incluye las cuatro tipologías de instalación)

```
{
    "installation_type": "iso|image|network|pxe”,
    "name": "CentOS_7",
    "memory": 2048,
    "vcpus": 2,
    "disk": {
        "path": "/path/to/imported/disk.qcow2",
        "size": 8
    },
    "cdrom": "/path/to/rhel7.iso",
    "import": true,
    "location": "http://example.com/path/to/os",
    "network": "bridge:br0",
    "os_variant": "rhel7",
    "graphics": {
        "vnc": true,
        "listen": "0.0.0.0",
        "password": "sysadmin.vnc"
    }
}
```

GET /api/domains/ → (Obtiene una lista de los dominios definidos)

```
[
    {
        "uuid": "",
        "name": "CentOS_7",
        "is_active": true,
        "os_type": "hvm",
        "memory": {
            "total": 4096,
            "used": 1024
        },
        "vcpus": 2,
        "state": 5,
        "vnc_port": 5900
    },
    …
]
```

DELETE /api/domains/<domain_name> → (Elimina el dominio <domain_name>)

```
{ }
```

PUT /api/domains/<domain_name>/start → (Enciende el dominio <domain_name>)

```
{ }
```

PUT /api/domains/<domain_name>/reboot → (Reinicia el dominio <domain_name>)

```
{ }
```

PUT /api/domains/<domain_name>/shutdown → (Apaga el dominio <domain_name>)

```
{ }
```

Laboratorios

Con estos recursos se realizan las operaciones CRUD sobre el modelo de datos que define el concepto de 'laboratorio' en el sistema de información.

POST /api/labs/ → (Crea un laboratorio)

```
{
    "code": "LAB2-3",
    "description": "Laboratorio de DAW2",
    "start_ip_range": "10.0.0.10",
    "end_ip_range": "10.0.0.20",
    "hosts": {
        "vcpus": 4,
        "memory": 16, // GiB
        "disk": 250 // GiB
    }
}
```

GET /api/labs/ → (Obtiene una lista con los laboratorios)

```
[
    {
        "uuid": "6ac3a7ce-6306-451e-9c38-77a22d80640e",
        "code": "LAB2-3",
        "description": "Laboratorio de DAW2",
        "start_ip_range": "10.0.0.10",
        "end_ip_range": "10.0.0.20",
        "hosts": {
            "total": 43,
            "vcpus": 4,
            "memory": 16, // GiB
            "disk": 250 // GiB
        }
    }, …
]
```

DELETE /api/labs/<lab_uuid> → (Elimina el laboratorio con UUID <lab_uuid>)

```
{ }
```

Plantillas

Del mismo modo, se muestra la estructura de datos de los recursos RESTful asociados al modelo 'plantillas'.

POST /api/templates/

(Crea una nueva plantilla)

```
{
    "name": "centos7_daw2",
    "description": "Sistema Linux con entorno de desarrollo para DAW2",
    "domain_uuid": "CentOS_7"
}
```

GET /api/templates/

(Obtiene una lista de las plantillas registradas)

```
[
    {
        "uuid": "6ac3a7ce-6306-451e-9c38-77a22d80640e",
        "timestamp": "20190916-20:19:50",
        "name": "centos7_daw2",
        "description": "Sistema Linux con entorno de desarrollo para DAW2",
        "xml_path": "/etc/libvirt/qemu/templates/centos7_template.xml",
        "images_path": [
            "/var/lib/libvirt/images/templates/centos7_template-disk0.qcow2",
            …
        ]
    },
    …
]
```

DELETE /api/templates/<template_uuid>

(Elimina la plantilla con UUID <template_uuid>)

```
{ }
```

POST /api/templates/<template_uuid>

(Genera un dominio a partir de la plantilla con UUID <template_uuid>)

```
{
    "domain_name": "nuevo_centos7",
    "lab_deployment": true,
    "lab_uuid": "6ac3a7ce-6306-451e-9c38-77a22d80640e "
}
```

Hosts

Por último, se muestran los recursos asociados al modelo fundamental para realizar los despliegues: el host.

POST /api/hosts/ → (Crea un nuevo host)

```
{
    "code": "LAB2-3_01",
    "ip_address": "10.0.0.14",
    "conn_user": "root",
    "lab_uuid": "6ac3a7ce-6306-451e-9c38-77a22d80640e "
}
```

GET /api/hosts/ → (Obtiene una lista con todos los hosts)

```
[
    {
        "uuid": "6ac3a7ce-6306-451e-9c38-77a22d80640e",
        "code": "LAB2-3_01",
        "ip_address": "10.0.0.15",
        "conn_user": "root",
        "lab_uuid": "6ac3a7ce-6306-451e-9c38-77a22d80640e "
    },
    …
]
```

DELETE /api/hosts/<host_uuid> → (Elimina el host con UUID <host_uuid>)

```
{ }
```