**ANTEPROYECTO DEL**

**TRABAJO DE FIN DE GRADO**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **INFORMACIÓN GENERAL** | | | | | |
| **Alumno/a** | Claudio María Martínez Velasco | | | | |
| **Titulación:** | Ingeniería de Computadores | | | | |
| **Tutor/es:** | Guillermo Pérez Trabado, Departamento de Arquitectura de Computadores | | | | |
| **Título** | Desarrollo de un orquestador de máquinas virtuales para entornos docentes y de investigación | | | | |
| **Subtítulo** *(solo si en grupo)* |  | | | | |
| **Título en inglés** | Development of a Virtual Machine Orchestrator for Educational and Research Environments | | | | |
| **Subtítulo en inglés** *(solo si en grupo)* |  | | | | |
| **Trabajo en grupo:** | **Sí** |  | **No** | X |  |
| **Otros integrantes del grupo:** |  | | | | |

|  |
| --- |
| **INTRODUCCIÓN** |
| La virtualización constituye uno de los pilares fundamentales de la infraestructura tecnológica actual, tanto en el ámbito académico como en el industrial. Su capacidad para permitir la coexistencia de múltiples entornos independientes sobre un mismo hardware ha abierto la puerta a soluciones de docencia más flexibles, entornos de pruebas seguros y sistemas de investigación escalables.  En este contexto surge la necesidad de un orquestador de máquinas virtuales (VMs) que proporcione una capa de abstracción por encima de los hipervisores subyacentes (como VMware, Proxmox u Oracle VirtualBox), con el fin de simplificar la administración, mejorar la modularidad y ofrecer un marco universal para diferentes aplicaciones. |
| *Contextualización del problema a resolver. Describir claramente de dónde surge la necesidad de este TFG y el dominio de aplicación. En caso de que el TFG se base en trabajos previos, debe aclararse cuáles son las aportaciones del TFG.* |
| Actualmente, existen soluciones comerciales y de código abierto para la gestión de máquinas virtuales, pero suelen estar ligadas a un hipervisor específico o carecen de la flexibilidad necesaria para adaptarse a escenarios de docencia universitaria. En dichos escenarios, es común la necesidad de desplegar **cientos** de máquinas virtuales organizadas por asignatura, alumno y práctica, lo que introduce una elevada complejidad si se gestiona directamente desde el nivel del hipervisor.  El precio de la solución de orquestación de WMWare se ha incrementado de forma importante. Una solución open-source podría ser más económica, aun contemplando costes de instalación, formación y mantenimiento.  La propuesta de este TFG consiste en el desarrollo de un orquestador modular de máquinas virtuales, implementado como un servicio Linux con interfaz REST, que proporcione un modelo de administración unificado basado en descriptores. Este orquestador permitirá definir, organizar y gestionar VMs de manera independiente de la tecnología de virtualización subyacente, incorporando además mecanismos de herencia de configuraciones, control de usuarios, y despliegue automático de infraestructuras educativas.  El trabajo se basa en librerías Python creadas anteriormente por Guillermo Pérez. Estas librerías hacen de interfaz con hipervisores VMWare, y también realizan un despliegue de una infraestructura virtual descrita en un fichero YAML. Asimismo, utiliza un IPAM externo para la gestión de direcciones IP en el despliegue de la arquitectura virtual.  En el presente TFG, se amplia esto creando un orquestador que maneja muchas máquinas virtuales, manteniendo la información persistente y coordinada con los hipervisores, así como lo mencionado anteriormente. Se adaptará el módulo de despliegue de infrasturcturas al orquestador. Será ampliable a otros hipervisores aparte de VMWare. Por la forma de organizar las máquinas virtuales, será de utilidad para otras aplicaciones, aparte de la docencia. |

|  |
| --- |
| **OBJETIVOS** |
| *Descripción detallada de en qué consistirá el TFG. En caso de que el objeto principal del TFG sea el desarrollo de software, además de los objetivos generales deben describirse sus funcionalidades a alto nivel.* |
| El **objetivo general** del TFG es diseñar e implementar un **orquestador de máquinas virtuales multiplataforma**, con una arquitectura modular y extensible, que pueda ser utilizado tanto en entornos docentes como de investigación. Objetivos específicos:  1. **Diseñar un modelo de descriptores de VMs** que represente de forma unificada tanto la definición como el ciclo de vida de una máquina virtual, incluyendo estados como Provisioned, Created, Running, Paused, Stopped y Failed. 2. **Implementar un servicio Linux** que mantenga un registro centralizado de todos los descriptores, actuando como capa de abstracción respecto a los hipervisores reales. 3. **Definir un sistema jerárquico en árbol** para organizar las VMs, permitiendo la herencia en cascada de configuraciones comunes y la aplicación de permisos y políticas de acceso en distintos niveles. 4. **Desarrollar conectores de comunicación para distintos hipervisores**, empezando con un hipervisor mock de pruebas y posteriormente extendiendo la compatibilidad a VMware. Y como desarrollo fututo, otros como Proxmox. 5. **Incorporar un sistema de control de usuarios y roles** (superusuario, manager y usuario simple) con permisos heredables en la jerarquía de carpetas. 6. **Diseñar un módulo especializado para despliegues docentes**, capaz de crear infraestructuras completas a partir de listados de alumnos y definiciones de asignaturas, integrándose con un sistema de gestión de direcciones IP (IPAM). 7. **Implementar una interfaz REST y una interfaz web** (UI) que permitan el acceso remoto y simplificado a las funcionalidades del orquestador. La interfaz web, en este TFG será un primer boceto sin pretensión de ser completa. 8. **Garantizar la coherencia de datos** entre las definiciones de VMs, los descriptores y las máquinas reales, estableciendo políticas claras de actualización y sincronización.   **IMPORTANTE: Dada la envergadura de un posible orquestador completo, en el presente TFG se desarrollarán funcionalidades básicas únicamente, dejando para un desarrollo futuro, la ampliación a funciones más complejas.** |

|  |
| --- |
| **ENTREGABLES** |
| *Listado de resultados que generará el TFG (aplicaciones, estudios, manuales, etc.)* |
| * Documentación describiendo la arquitectura del orquestador, el modelo de descriptores y las interacciones entre módulos. * Implementación de un módulo orquestador como servicio Linux con API REST (basado en FastAPI). * Implementación de un driver de hipervisor mock y, posteriormente, drivers para VMware. * Módulo de despliegue automático de asignaturas a partir de definiciones YAML. * Interfaz de usuario web (UI) desarrollada con algún framework de Python. * Sistema de gestión de roles y permisos basado en jerarquía de carpetas. * Repositorio GIT público con el código fuente. |

|  |
| --- |
| **MÉTODOS Y FASES DE TRABAJO** |
| **METODOLOGÍA:** |
| *Descripción de la metodología empleada en el desarrollo del TFG. Especificar cómo se va a desarrollar. Concretar si se trata de alguna metodología existente y, en caso contrario, describir y justificar adecuadamente los métodos que se aplicarán.* |
| El desarrollo se llevará a cabo siguiendo una metodología ágil basada en **SCRUM**, con iteraciones cortas (sprints) de una o dos semanas. Cada sprint incluirá planificación, desarrollo, revisión y documentación de resultados. |
| **FASES DE TRABAJO:** |
| *Enumeración y breve descripción de las fases de trabajo en las que consistirá el TFG.* |
| 1. Análisis y diseño inicial: Definición del modelo de descriptores, jerarquía en árbol y estados de ciclo de vida. 2. Implementación del hipervisor mock para pruebas independientes de hardware. 3. Desarrollo inicial del módulo orquestador: gestión básica de descriptores, almacenamiento en árbol y operaciones fundamentales. 4. Extensión del orquestador: herencia de configuraciones, control de usuarios y permisos. 5. Exposición del orquestador como servicio Linux: API REST con FastAPI y cliente CLI. 6. Integración de drivers para VMware. 7. Implementación del módulo de despliegue docente, con integración con un IPAM. 8. Diseño de la interfaz web (NiceGUI) para interacción remota. 9. Pruebas, validación y documentación final. |

|  |
| --- |
| **TEMPORIZACIÓN:** |
| *La siguiente tabla deberá contener una fila por cada una de las fases enumeradas en la sección anterior. En caso de tratarse de un trabajo en grupo, se añadirá una columna HORAS por cada miembro del equipo. Debe especificarse claramente el número de horas dedicado por cada alumno/a y la suma de horas individual deberá ser también de 296.* |
| |  |  | | --- | --- | | **FASE** | **HORAS** | | *Nombre Apellidos* | | 1. Análisis y diseño inicial | **30** | | 1. Implementación hipervisor mock | **20** | | 1. Desarrollo básico del orquestador | **40** | | 1. Extensión con herencia y permisos | **50** | | 1. API REST y servicio Linux | **40** | | 1. Drivers para VMware y Proxmox | **40** | | 1. Módulo de despliegue docente | **40** | | 1. Interfaz web | **20** | | 1. Pruebas y validación | **16** | | 1. Documentación y memoria |  | |  | **296** |   (\*) La documentación se irá realizando en cada fase. |

|  |
| --- |
| **ENTORNO TECNOLÓGICO** |
| **TECNOLOGÍAS EMPLEADAS:** |
| *Enumeración de las tecnologías utilizadas (lenguajes de programación, frameworks, sistemas gestores de bases de datos, etc.) en el desarrollo del TFG.* |
| Sistema Operativo Linux, y Linux Subsystem for Windows |
| Lenguajes de programación: Python 3, YAML/JSON. |
| Frameworks y librerías: FastAPI (REST), Pydantic, Box, NiceGUI (UI), UML (Visual Paradigm o similar). |
| Herramientas de virtualización: VMware ESXi, Proxmox VE, hipervisor mock. |
| Bases de datos y almacenamiento: SQLite o sistema de ficheros jerárquico. |
| Gestión de configuración y despliegue: Ansible, IPAM. |
| **RECURSOS SOFTWARE Y HARDWARE:** |
| Servidores Linux con VMware ESXi y/o Proxmox. |
| Máquinas virtuales para pruebas en entorno de desarrollo. |
| Sistema IPAM para gestión de direcciones IP. |
| Repositorios GitHub/GitLab. |
| Herramientas de documentación: LaTeX/LibreOffice, Visio o PlantUML. |
| *Listado de dispositivos (placas de desarrollo, microcontroladores, procesadores, sensores, robots, etc.) o software (IDE, editores, etc.) empleados en el desarrollo del TFG.* |
| Visual Studio CODE / Python |
| Máquinas virtuales con Linux ligeros. |
| VMWare Workstation y VMWare ESX para ejecutar las máquinas virtuales. |
| GitLab o GitHub para gestionar el repositorio GIT del proyecto. |
| Editor para UML (posiblemente MS Visio). |
| Herramientas de Inteligencia artificial: MS Copilot, Google Gemini, ChatGPT |

|  |
| --- |
| **REFERENCIAS** |
| *Listado de referencias (libros, páginas web, etc.)* |
| VMware. “vSphere Virtual Machine Administration Guide.” <https://docs.vmware.com>  Proxmox Virtual Environment Documentation. <https://pve.proxmox.com/pve-docs>  FastAPI Documentation. <https://fastapi.tiangolo.com>  HTTP client <https://www.python-httpx.org/>  NiceGUI Documentation. <https://nicegui.io>  YAML parse, <https://yaml.dev/doc/ruamel.yaml/>  Authentication <https://docs.authlib.org/en/latest/client/api.html>  IPAM <https://phpipam.net/documents/>  Python <https://www.python.org/doc/>  Rich API <https://rich.readthedocs.io/en/latest/> |

Málaga, 1 de Agosto de 2025

|  |  |  |
| --- | --- | --- |
| Firma tutor/tutora: | Firma cotutor/a: | Firma tutor/a coordinador/a: |