ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

DESPLIEGUE DE UN CMS DE LICENCIA LIBRE POR MEDIO DE UN CONTENEDOR

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO SUPERIOR EN REDES Y TELECOMUNICACIONES

CRISTHIAN LEONARDO CARRIÓN PARRA

DIRECTOR: ITALO ALEXANDER CARREÑO MENDOZA

DMQ, marzo 2023

CERTIFICACIONES

Yo, CRISTHIAN LEONARDO CARRIÓN PARRA declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

CRISTHIAN CARRIÓN

cristhian.carrion@epn.edu.ec

cris_cristhianc@hotmail.es

Certifico que el presente trabajo de integración curricular fue desarrollado por CRISTHIAN CARRIÓN, bajo mi supervisión.

ITALO CARREÑO

DIRECTOR

italo.carreno@epn.edu.ec

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el producto resultante del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

CRISTHIAN LEONARDO CARRIÓN PARRA

DEDICATORIA

Deseo dedicar el presente trabajo de integración curricular, al esfuerzo, trabajo y dedicación de mis padres, que se han convertido en luz y guía, a lo largo de toda mi vida. Enseñándome que, con esfuerzo, trabajo duro y dedicación, se alcanza cualquier meta impuesta.

De igual manera, a mi familia y hermanos, por el apoyo que me han regalado, durante la culminación de esta etapa en mi vida.

Cristhian Leonardo Carrión Parra

AGRADECIMIENTO

En primer lugar, quiero agradecer a Dios, que ha hecho posible la obtención de un nuevo escalón académico.

A mis padres, Nancy Parra y Sergio Carrión que, con su apoyo, consejo y ejemplo, han hecho posible obtener un grado más en mi formación académica.

A la Escuela Politécnica Nacional, que abrió sus prestigiosas puertas, ofreciéndome la oportunidad de convertirme en un profesional. Así como, a todos los profesores a cargo de instruirme en mi formación técnica y desempeño moral sobre todos los campos pertenecientes al correcto desempeño en mi campo.

Finalmente, al Ing. Italo Carreño, por su guía y apoyo en la culminación del presente escrito.

Cristhian Leonardo Carrión Parra

ÍNDICE DE CONTENIDOS

RESUMEN

Un contenedor, se presenta como una solución actual a las necesidades de almacenamiento, registro, desarrollo y optimización de una gran variedad de servicios desarrollados a treves de aplicaciones que, se ensamblan, partiendo de componentes requeridos por él, o los usuarios que exigen mayor capacidad en el sistema, De esta manera se presenta, uno de los sistemas con mayor soporte y desarrollo, *Docker*, un familiar ágil y lejano de las máquinas virtuales convencionales que propone una portabilidad y aislamiento de servicios y aplicaciones de terceros, utilizando un empaquetado que, posteriormente, logrará permitir al usuario, ejecutar el, o los servicios, de manera ágil y fácil, minimizando las complicaciones más comunes, y optimizando los recursos.

Es el propósito del presente escrito, es el de evidenciar el proceso completo, sobre el levantamiento de un *CMS* (Sistema de Gestión de Contenidos), por medio de un contenedor que, trabajando en conjunto con un sistema de gestión de bases de datos, *MariaDB,* almacenados localmente, que, proporcionarán el correcto levantamiento de la plataforma *Moodle*, corriendo en el sistema operativo de código libre *Ubuntu,* obedeciendo las características previamente estudiadas sobre los recursos de *hardware* y *software* que soportará y mantendrá la correcta operación del prototipo.

Finalmente, mostrará de manera amigable y funcional, la interfaz con los aspectos más importantes de la plataforma virtual de aprendizaje, por ejemplo: seguridad para el usuario, como el manejo de contraseñas, perfiles de acceso enfocada a una clase predeterminada, entregables para su posterior calificación y manejo cuantitativo de los resultados, interacción entre usuarios, etc.

Se mantiene un enfoque hacia perfiles de él, o los administradores, profesores y estudiantes que mantengan total acceso a la plataforma y a la red, en un ambiente *LAN*.

PALABRAS CLAVE: *CMS*, *Contenedor*, *Moodle*, Sistema de gestión de base de datos, *Docker*, *MariaDB*.

ABSTRACT

A container, is presented as a current solution to the needs of storage, registration, development and optimization of a wide variety of services developed for three applications that are assembled, starting from components required by it, or by users who demand greater capacity. In the system, in this way, one of the systems with the greatest support and development is presented, Docker, an agile and distant relative of conventional virtual machines that propose portability and isolation of third-party services and applications, using a packaging that, Subsequently, it will allow the user to execute the service or services in an agile and easy way, minimizing the most common complications and optimizing resources.

It is the purpose of this writing, is to demonstrate the complete process, on the lifting of a CMS (Content Management System), by means of a container that, working together with a database management system, MariaDB , stored locally, which will maintain the correct lifting of the Moodle platform, running on the open source operating system Ubuntu, obeying the characteristics previously studied on the hardware and software resources that will support and maintain the correct operation of the prototype.

Finally, show in a friendly and functional way, the interface with the most important aspects of the virtual learning platform, for example: security for the user, such as password management, access profiles focused on a predetermined class, deliverables for later qualification and quantitative management of the results, interaction between users, etc.

A focus is maintained towards profiles of him, or the administrators, teachers and students who maintain full access to the platform and the network, in a Local Area Network.

KEYWORDS: CMS, Container, Moodle, Database management system, MariaDB.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

Este proyecto tendrá como finalidad el despliegue funcional de una plataforma de aprendizaje *CMS* (Course Management System), implementado por medio de un contenedor, más precisamente, por el software gratuito *Docker Hub,* levantado en un sistema operativo de código libre, *Ubuntu,* elegido por su óptimo funcionamiento, sobre prácticas previamente realizadas, y sobre todo, le enorme cantidad de soporte que existe en las diferentes documentaciones elaboradas por parte de los propios desarrolladores orientados a esta clase de servicios en particular.

Este servicio tendrá como sus principales parámetros: establecer una satisfactoria disponibilidad, así como una óptima conexión y acceso para todas las terminales inteligentes que cumplan con las credenciales que autorizan el uso de la red de área local *LAN,* donde permanezca levantado el servicio, además, la dirección *IP* del terminal *host* de administración del *CMS*, donde, a su vez, se desplegará el servicio web *Moodle,* que actuará como plataforma de educación, donde se podrá visualizar e interactuar con los cursos generados por el administrados del servicio, hacia los diferentes perfiles (profesores y estudiantes), donde integrará, de manera dinámica y amigable interfaz de un determinado número de cursos disponibles, operado con diferentes permisos establecidos por el perfil de cada usuario.

El presente escrito abordará de manera puntual, el proceso de inicio a fin de cada uno de los aspectos técnicos y de mayor relevancia desarrollados por el autor, para un posterior análisis respecto al funcionamiento del mismo, las implicaciones que conlleva el manejo de este tipo de servicio, y el alcance de la misma operando en una determinada red de área local.

1.1 Objetivo general

Desplegar un *CMS* de licencia libre por medio de un contenedor.

1.2 Objetivos específicos

- Identificar los requerimientos para el despliegue del *CMS.*
- Seleccionar el *software* y el *hardware* acorde a los requerimientos encontrados.
- Despliegue del *CMS*.
- Realizar pruebas de funcionamiento del prototipo.

1.3 Alcance

Por medio del presente proyecto se busca desplegar un *CMS* de licencia libre en base a un contenedor. Por tal motivo, se realizarán las siguientes actividades:

- Configuración e instalación de todas las herramientas o paquetes necesarios.
- Configuración de los parámetros necesarios para el despliegue del *CMS*.
- Despliegue del *CMS*.
- Configuración de un curso de prueba en la plataforma de aprendizaje.
- Realización de las pruebas de funcionamiento.

1.4 Marco Teórico

CMS

Por sus siglas, *Content Management System.* Es *un* sistema de gestión de contenidos que, al tener un enfoque de gestión de contenido en una plataforma de educación, que se ejecuta a través de un contenedor y ejecutado en un ambiente *LAN,* donde organiza, estructura, categoriza y recopila la información que se entrega a ese sitio en particular [1], pero no todo son buenas noticias en el despliegue de este tipo de servicios por lo que, a continuación se muestra, en la **Tabla *1.1*** , una comparativa sobre las ventajas y desventajas que representa el levantamiento de este servicio [1] [2].

Tabla 1.1 Ventajas y desventajas del despliegue de un *CMS* [1] [2]

Sistema de Gestión de contenidos	
Ventajas	**Desventajas**
Permite una rápida incorporación de realimentación para los usuarios del sistema.	Un *CMS* siempre se considera en etapa beta.
Ambientes modulares.	Este sistema siempre está en constante actualización, lo que significa una mayor dificultad para mantenerse al día.
Utilización de micro formatos libres, previamente diseñados.	Es necesario contar con un desarrollador o grupo de desarrolladores web.
No se espera actualización para levantar el servicio.	Mayor flexibilidad que puede caer en diferentes direcciones o inconsistencias con respecto al diseño y, este factor

	aumenta proporcionalmente el número de desarrolladores para solventar este problema.
Acceso inmediato a los repositorios o base de datos basados en la nube, o localmente.	Algunos *CMS,* presentan dificultades o incompatibilidades con la variedad de navegadores que existen, y que deben solucionarse.
Automatización en diversos procesos educativos como, por ejemplo, de evaluación.	
Presenta una curva de aprendizaje hacia los usuarios más no, un modelo de pendiente.	

Moodle

Es una plataforma gratuita con un enfoque en el aprendizaje a manera de gestión web. Comprende una amplia gama de módulos integrados de manera única, segura y robusta, con una orientación hacia administradores, estudiantes y educadores que está aprobado y es preferido a nivel mundial por pequeñas y grandes instituciones de educación, contando con más de 200 millones de usuarios, haciendo uso de esta plataforma [3]. Presentado como una herramienta de personalización de plataformas virtuales de educación, hacia cualquier nivel institucional, obedeciendo los parámetros descritos por el o los desarrolladores, donde, el contenido del mismo, se almacena en su propio servidor web. Cuenta con más de 80 entidades capacitadas para dar soporte, conocidos como *Socios Moodle* [4]*,* siendo estos, una red de fundamentos, instrucción, albergue de varios asentadores que liderarán el desarrollo sobre la plataforma *Moodle* en línea.

Introducción a los Contenedores

Es una tecnología que se ha venido desarrollando por más de 40 años, debido a una constante migración, con respecto a la ejecución de aplicaciones tradicionales, a sistemas que optimicen recursos, ya que, ahora el enfoque avanza hacia el empaquetamiento de las dependencias, código y bibliotecas de estas aplicaciones. Este comportamiento se ha visto desarrollado como norma, cuando se habla a un nivel empresarial ya que, el servicio de un contenedor es capaz de prestar una amplia mejora hacia el uso compartido de los recursos. Los contenedores [5], logran alcanzar una

óptima creación de ambientes predeterminados, donde, las aplicaciones que se ejecutan de manera virtualizada, centran el encapsulamiento de las mismas o, de manera más global, logran establecer servicios o entornos informáticos predeterminados para que, posteriormente, el empaquetado de todas estas herramientas, tengan la posibilidad de ser ejecutadas en cualquier ambiente, ya sea en el escritorio, almacenados en la nube o en un sistema tradicional de TI [5] [6].

Sistema de gestión de base de datos *(SGBD)*

La principal función y alcance que debe cumplir este sistema en particular, se resume en el acrónimo *ACID,* por sus siglas en inglés, donde, especifican los cuatro principales parámetros de su funcionamiento [7]:

- Atomicidad *(Atomicity)*: Se resume con el término “todo o nada”, donde cada consulta debe tener validez, obedeciendo a un número determinado de pasos o secuencias, donde, si se presentara algún error o fallo en su ejecución, éste no puede finalizar a medias, es decir, todas estas consultas ocurren o ninguno de ellas.
- Consistencia *(Consistency)*: También conocido como coherencia, se produce cuando la base de datos muestra el estado “estable”, al finalizar la base de datos, exigiendo un monitoreo constante en cada una de las transacciones.
- Aislamiento *(Isolation)*: En este proceso se especifica el “no colapso” de las transacciones, es decir, establece los parámetros de cómo y cuándo se generan los cambios realizados por una base de datos por una operación. utilizando por ejemplo una función de bloqueo.
- Durabilidad *(Durability)*: Hace referencia al almacenamiento perpetuo de los datos tras finalizar con la transacción en cuestión [7] [8].

Red LAN

Por sus siglas, *Local Area Network*, o Red de Área Local, se resume en una interconexión de periféricos y/o terminales de computación, donde su extensión se encuentra limitada a un dominio de aproximadamente 200 (m). Se puede relacionar con el área de un edificio o a su vez, se puede relacionar con el siguiente diagrama, donde se busca idear un entorno común de red *LAN*.

En la **Figura *1.1*** se muestra el ejemplo de un esquema básico de red de área local, donde, de manera central, se muestra un dispositivo enrutador que interconecta a los hosts o terminales finales que, además, comparten un enlace desde y hacia internet.

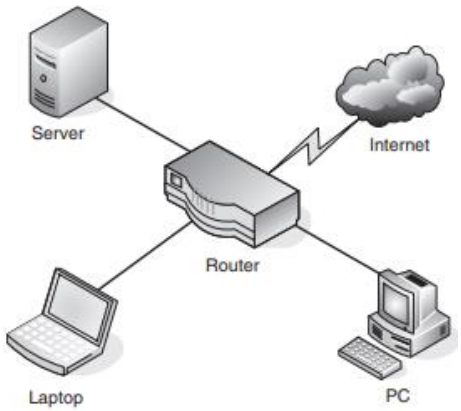

Figura 1.1 Ejemplo de Red de Área Local [9].

Cualquier elemento, que se encuentre en la red, excluyendo la computadora personal, la computadora portátil y el servidor que se muestran en la **Figura *1.1***, se consideran fuera de la *LAN* [9].

Bitnami

Es un repositorio de paquetes de *software* o instaladores con un enfoque hacia servidores y aplicaciones web con una estructura establecida en la **Figura *1.2***. *Bitnami* mantiene un enfoque hacia la virtualización, contando con soporte para los sistemas operativos: *Windows, Mac OS X, Solaris, Linux* y sus distribuciones, etc. Ofreciendo más de 130 tipos de aplicaciones para servidores *web*, listos para usar a un solo clic.

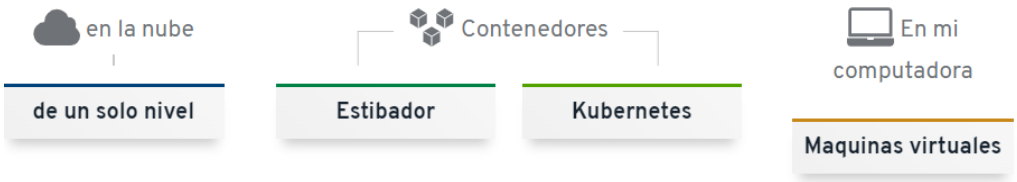

Figura 1.2 Alcance de *Bitnami*

Se puede ejecutar tanto de manera local como remota. Además, opera con la licencia libre *Apache,* en su versión actual 2.0, lo que permite la modificación y distribución sin tener en cuenta retribuciones por regalías [10].

2 METODOLOGÍA

Debido al particular enfoque en el que se desarrolló el levantamiento del presente servicio, la metodología experimental, es la base en la que se ejecutó el proceso para cumplir con el alcance propuesto, por lo que, se detalla, utilizando el mismo método, los

elementos para la obtención y cumplimiento de cada uno de los parámetros descritos, donde, de manera adecuada, se principia un previo estudio y análisis sobre los requerimientos necesarios y utilizados para el despliegue del *CMS*, que se complementarán con el requerido *hardware* y *software* que, a título del autor, mejor se adapten a los requerimientos descritos como por ejemplo, las herramientas *Docker* y *Docker Compose.*

Este servicio se ejecutará dentro del sistema operativo de licencia libre *Ubuntu*, que ha sido elegido por el soporte y la interfaz que se ofrecen hacia el usuario, logrando la óptima funcionalidad de todas las características sobre el motor de *Docker* que, con el fin de cumplir con el objetivo general, se trabajará en conjunto con el sistema de gestión de base de datos *MariaDB*, levantando la correcta ejecución del servicio, como por ejemplo, el almacenamiento instantáneo en la base de datos en cada cambio realizado por los usuarios sobre los perfiles de: administradores, profesores y estudiantes, que operarán únicamente con los permisos correspondientes que, a su vez, atados a las características de su perfil. Utilizando la mayoría de los recursos obtenidos de la plataforma *Moodle.*

Finalmente, se realizarán pruebas del funcionamiento de la plataforma de aprendizaje *Moodle*, así como la personalización del contenido que se proyectan en un entorno amigable para los diferentes perfiles de los usuarios, evidenciando así, el perfecto funcionamiento.

3 RESULTADOS

En este apartado, se detallan los diferentes requerimientos exigidos para el levantamiento y correcto uso del prototipo en cuestión, así como las especificaciones de las herramientas de *hardware* y *software,* utilizadas para cumplir con todas las funcionalidades dispuestas en el prototipo. Finalizando, con las pruebas correspondientes al funcionamiento de operación de la plataforma de aprendizaje *Moodle*.

3.1 Identificación de los recursos necesarios, para el despliegue del CMS, por medio de un contenedor

A continuación, se detalla los requerimientos necesarios tanto de *software* y *hardware* de cada uno de los elementos/complementos utilizados para el óptimo despliegue del *CMS* por medio de un contenedor, funcionando en un sistema operativo libre.

Docker

Es una plataforma libre que, de manera sin precedentes, ofrece gratuitamente automatización en cuanto a aplicaciones se refiere, integrando todas las dependencias en gran variedad de paquetes, de manera aislada.

Este *software,* ha logrado innovar con impactantes capacidades, que logran diferenciarse de otros desarrolladores, como, por ejemplo, la reducción de peso sobre embalado del *software* que se almacena en el contenedor, incluyendo todo aspecto técnico, como las librerías y complementos, obteniendo una optimización en la ejecución del mismo [11].

Docker Engine

Es una tecnología de código abierto cliente-servidor, que crea y contiene aplicaciones en relación con:

- *CLI* o un cliente a través de una interfaz de línea por comandos.
- Un servidor, que se ejecuta de manera desatendida o en segundo plano, de prolongada ejecución.
- *API* de *Docker o* interfaz de programación de aplicaciones, que precisan interfaces, a los que, los desarrolladores pueden derivar a segundo plano.

La *API* de *Docker* entra en la categoría de *API RESTful* (Interfaz de terminales computacionales que se ejecutan para intercambiar, de manera segura, información a través de ellas) [12], donde, se accede por medio de un *host HTTP,* como por ejemplo *curl* o *wget*.

Docker Engine exige requisitos generales para un correcto funcionamiento en ambientes *Linux*, por lo que el *host,* debe cumplir con los siguientes requerimientos [13]:

- *CPU* de 64 [bits] para virtualización y soporte en *Kernel*.
- Versión el *Linux* 2.6.20 o superior, ya que integra *KVM* (virtualización en el núcleo de *Linux*).

- Versión 5.2 o superior en *QUEMU*, que es, un emulador de código abierto para *VM,* que permite la ejecución en cualquier variedad de arquitectura o microprocesador basado en (32 y 64 bits, *MIPS, PowerPC, SPARC, e*tc.) [14].
- Por lo menos 4(Gb) en *RAM*.
- Cualquiera de los entornos de escritorio, que se describen a continuación:
    a. *Gnome:* Interfaz de escritorio gráfica, que genera un grupo central de aplicaciones, incorporándose con el escritorio. Convirtiéndola, en la interfaz de distribución de entorno más utilizada.
    b. *MATE:* Derivado y continuación de *Gnome2,* ofreciendo cambios con respecto al: administrador de archivos, editor de texto, gestor de ventanas, emulación de terminal, visor de imágenes y documentos. Mostrados con una interfaz de manera más clásica.
    c. *KDE:* La interfaz de escritorio que, se diferencia en su *software,* dado que, se agrupa en tres componentes: *Frameworks*, *Plasma* y *Aplications*, que son un conjunto de bibliotecas, que se aplican como base tecnológica para una interfaz más futurista.

Docker en Distribuciones *Linux*

Docker ha demostrado una compatibilidad y constante actualización hacia la gran diversidad de sistemas operativos, por lo que, a continuación, se detallan los principales recursos recomendados para el uso de *Docker Engine*, en la **Tabla *3.1*** debido a el enfoque hacia sistemas operativos de código libre [13]:

Tabla 3.1 Requisitos para *Docker Engine,* en la distribución de *Linux [13]*.

Distribución	Versiones	Recomendaciones
CentOS	7, 8, 9.	Los repositorios *CentOS extra* deben estar habilitados, usar *overlay2* como controlador de almacenamiento.
Debian	*Bullseye* y *Buster 10*, *Bullseye* y *Buster11*.	
Fedora	36, 37.	
Ubuntu	*Cinético* 22.10, *Jammy* 22.04, *Focal* 20.04, *Bionic* 18.04.	

Red Hat Enterprise Linux, (RHEL)	7, 8, 9. Paquetes disponibles solo en arquitecturas s390x *(IBM Z).*	Instalar "*overlay2"* como control de almacenamiento.
SUCE, Linux Enterprise Server (SLES)	15-SP3, 15 SP4. Paquetes disponibles solo en arquitecturas s390x *(IBM Z)*.	Los repositorios *OpenSUCE* y *SSC SUCE* deben estar habilitados.

Docker Compose

Dentro del desarrollo de *Docker,* existe un gran número de herramientas que complementan las funcionalidades que, optimizan y agilizan el almacenamiento local. *Docker Compose* hace posible estas características, integrando en sus propiedades un archivo con el formato de serialización *YAML,* para la ejecución instantánea de una o varias aplicaciones, con la instrucción para que, el servicio se encuentre ejecutado [15].

Características de *Docker Compose*

La principal característica de esta herramienta es la capacidad de almacenar multiplicidad de entornos, que se almacenan en un único *host,* conservando la información al que pertenece al volumen, partiendo de la creación de los diversos contenedores. Se utiliza una etiqueta o nombre que sean originarios del proyecto o servicio, logrando un aislamiento de diferentes entornos, otorgando la posibilidad de ejecutar un sin número de copias para el desarrollo en otros terminales de manera local. Teniendo en cuenta en que, si el servicio no ha presentado ningún cambio el almacenamiento es su cache, este se reinicia.

Docker Compose, toma automáticamente la atribución de reutilizar el o los contenedores existentes, lo que acelera el sistema al presentarse cambios realizados en su entorno. Además, este complemento logra una personalización en su composición, por la posibilidad de una contribución de un sin número de usuarios o entornos debido a la acogida de variables [16].

Funciones Principales de *Docker Compose*

Con tan solo un único comando, se podrá ejecutar una gran variedad de entornos, desarrollo, reseteo, producciones y flujos de trabajo *CI* (Integración continua) [17]. Aunque, se debe destacar la principal función, que es la de una gestión de tiempo o ciclo de vida de cada servicio, complementándose con cuatro diferentes funciones [18]:

- Empezar, concluir y restaurar el servicio.
- Visualizar el estado en el que se encuentra la ejecución del servicio.

- Enviar el estado de *output* del servicio en el que se encuentra ejecutando el servicio.
- Ejecutar un único comando que permitirá la ejecución global del servicio.

VirtualBox

Es una potente herramienta de virtualización que se ejecuta tanto en ambientes empresariales como domésticos. Idealmente desarrollados para procesadores de 32 y 64 bits en *AMD* o *Intel* [19]. También se destaca del resto de sistemas de virtualización, por ser una solución gratuita (código abierto) que opera bajo la licencia *GNU (General Public License) (GPL)* publicada por *Free Software Fundation* en su versión 3 [20].

En la actualidad, este *software* se ha adaptado a la globalidad de sistemas operativos más utilizados como: *Linux* en sus versiones (2.4, 2.6, 3.x, 4.x), macOS, *Solaris* y su distribución *OpenSolaris*, *Windows* en sus versiones (*NT* 4.0, 2000, *XP*, *Server* 2003, 7, 8, 10).

Es vital, señalar que *VirtualBox* se encuentra en constante desarrollo, ya que, al momento de exponer este escrito, cuenta con la versión 7.0, respaldado por la comunidad y dedicación de *Oracle*, cumpliendo los estándares que lo convierte en el favorito, si se trata de virtualización [20].

Ubuntu

Fundado por *Mak Shuttleworth* que, con la visión de trabajar con un sistema operativo *Linux* de mayor facilidad en su manejo, se desarrolló, utilizado como código fuente para el sistema operativo *Debian* que, a pesar de que ya se tenía una variedad considerable de distribuciones como *Xandros, MEPIS, Perogeny, Libranet. Ubuntu* se destaca por la adopción de código abierto que, asegura la gratuidad y soporte por la comunidad hacia el usuario [21].

En su primera versión (*Ubuntu* 4.0), publicada el 26 de octubre de 2004, se ha venido desarrollando regularmente una nueva versión, aproximadamente, cada 6 meses, por lo que, en momento de escrito el presente documento, se ha alcanzado la versión 22.04.2 para su versión de escritorio, donde se desarrolla el presente prototipo. Además, abarca en su desarrollo, versiones de tipo: *Desktop*, *Server* y *Mobil* [22]. En sus últimas versiones, los recursos recomendados para la instalación y ejecución de *Ubuntu* en la **Tabla *3.2*** [24] [25]:

Tabla 3.2 Recursos recomendados para la instalación y ejecución de *Ubuntu* 22.04 *[23] [24] [25]*

Clase	Procesador	Memoria del sistema	Espacio libre en memoria	Gráficos	Característica recomendada (opcional)
Desktop	2 (GHz) *Intel* o *AMD Dual Core* o superior.	8(Gb) de memoria *RAM* o superior.	25 (Gb) de memoria en disco duro o superior.	Tarjeta gráfica *VGA*, monitor con resolución de 800x600 pixeles.	Unidad de *DVD* o puerto *USB* para trasmisión de medios de instalación Conexión estable hacia internet.
Server	1 (Ghz) Intel o AMD.	2 (Gb) en *RAM* o superior.	5 (Gb) de memoria en disco duro, o superior.	Tarjeta gráfica *VGA*, monitor con resolución 800x600 pixeles.	Unidad de *DVD* o puerto *USB* para transmisión de medios de instalación. Conexión estable a internet.
Mobil	1(GHz) en *ARM Cortex-AS.*	4 a 8 (Gb) en *RAM* o superior.	512(Mb)-1(Gb).	320x480 Pixeles, pantalla multitouch.	Almacenamiento externo *SD*.

MariaDB

Es un sistema de base de datos de código abierto, desarrollada por la comunidad *MySQL.* Originalmente creada con un enfoque de discos giratorios, debido a su arquitectura, pero posteriormente se implementó la tecnología de dispositivos de memoria no volátil o *NVR,* que disparó un nuevo método de compresión al añadirla a su nueva arquitectura, lo que permitió el manejo de datos de transacciones con una alta tasa de disponibilidad.

MariaDB, evidentemente no ha tenido el mismo recorrido que *MySQL,* pero se ha adaptado bastante bien a los entornos de *Ubuntu,* tanto así, que se recomienda la

instalación de esta base de datos antes que sus competidores, haciendo especial énfasis en las versiones: 16.04, 18.04 y 20.04. Además, contar con los recursos recomendados: 1 [Gb] en *RAM*, de 500 [Mb] a 1 [Gb] en disco duro y *CPU* de 2 núcleos.

Agrega una de las principales funcionalidades, como servidor integrado, es el análisis de datos y un enorme repertorio de aplicaciones, que son capaces de soportar un nuevo y novedoso servicio *MariaDB Server* [26].

3.2 Elección de *hardware* y *software* que más se acopla y esté a disposición en base a los requerimientos encontrados

Dado a los recursos identificados en el apartado 3.1, a continuación, se detalla el sistema de *Hardware* y *Software* que se utilizó para el despliegue y administración del servicio *CMS*, dado que se adapta y supera en algunos aspectos recomendados, que logran cumplir con los objetivos ya expuestos en el presente escrito, al ejecutar, en su totalidad con el manejo y desarrollo del *CMS*.

Hardware

En la **Tabla *3.3***, se identifica los principales elementos a nivel de *software* que se utilizó para el levantamiento del servicio.

Tabla 3.3 Resumen de *software* utilizado *[27]*

Elemento	Descripción
Modelo del Equipo	Notebook *HP Pavilion* 17-f217ng.
Procesador	*Intel® Core™* i7-5500U *CPU@*, 2.40 [Ghz].
RAM	8[Gb], DDR3-1600 monocanal.
Sistema Operativo	64 [bits], procesador x64.
Pantalla	17.3’’m 16:9, 1600x900 pixeles.
Disco Duro	500[Gb] *SSD* (modificado).
Tarjeta Gráfica	*NVIDIA® GeForce* 830M.
Tarjeta de Red	*Realtek PCIe FE Family Controller*

Comparativamente con lo descrito en el apartado 3.1 iguala, en algunos aspectos, aunque en su mayoría, supera los requisitos para la ejecución de la totalidad de los elementos para el despliegue del *CMS,* en cualquiera de los ambientes *Linux.* Excediendo las recomendaciones, parar ejecutar el servicio con una mayor fluidez.

Software

Debido a la extensión de las especificaciones de *software* utilizados, para el despliegue del servicio, se ejemplificará en dos diferentes tablas, que se muestran a continuación, en la **Tabla *3.4***

Tabla 3.4 *Software* utilizado

Sistema	Versión
Sistema Operativo	*Windows* 10 pro© 2019 *Microsoft Corporation*
VirtualBox	7.0.2 r154219
Ubuntu	*Desktop* 22.04
Docker	20.10.21
Docker Compose	1.20.2 – 1
MariaDB	10.6
Moodle	3.7

Donde se puede comparar las versiones correspondientes a cada programa y complemento utilizado para el levantamiento del servicio con respecto propios del equipo de ejecución.

En la **Tabla *3.5***, se muestra las especificaciones con las que el sistema operativo *Ubuntu* se ejecuta en la aplicación para máquinas virtuales *VirtualBox* [19].

Tabla 3.5 *Software* asignado para la virtualización

Elementos para virtualización	Capacidad
Memoria base	5435 [Mb]
Número de procesadores	3
Memoria de video	103 [Mb]
Controlador gráfico	*VMSVGA*
Adaptador de red	*Intel* PRO/1000 *MT Desktop* (Adaptador puente *<<Realtek PCIe FE Family Controller>>*)

3.3 Levantamiento del *CMS*

Empezando con un análisis de todos requisitos previamente descritos en el anterior apartado, se procede con la ejecución de la virtualización en el *software*

correspondiente, *VirtualBox,* donde progresiva e intuitivamente se concede los permisos necesarios para la instalación y ejecución del sistema operativo libre.

Una vez conocida la interfaz del virtualizador, se procede con la ejecución de *Ubuntu,* obtenida por el desarrollador, a manera de extensión *".iso".* El sistema operativo basado en *Linux,* trabaja en conjunto con la herramienta *VirtualBox,* para iniciar y ejecutar su sistema libre, de manera correcta, concediendo (por parte del usuario), todos los permisos necesarios y configuración correspondiente. Obteniendo, finalmente, la interfaz visualizada en la **Figura *3.1***.

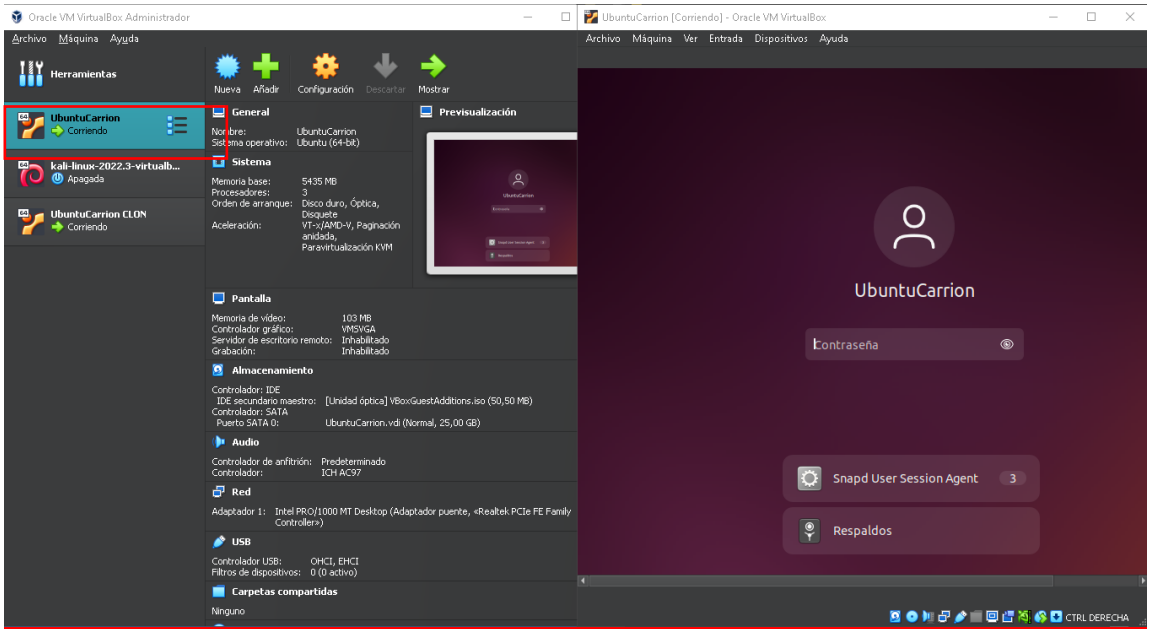

Figura 3.1 Interfaz de la virtualización de *Ubuntu.*

Donde, se muestra la primera interfaz virtualizada de *Ubuntu.* Además, se puede evidenciar los recursos asignados para su ejecución del sistema operativo libre.

En la ventana de comandos de nuestro sistema virtualizado, se ejecuta las dos líneas de comando, *update* y *upgrade,* identificado en la **Figura *3.2***.

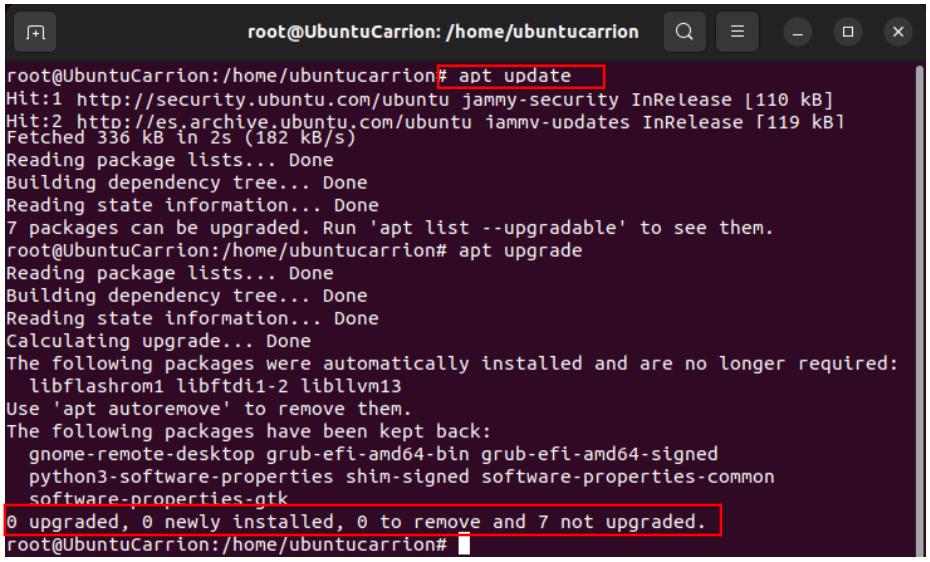

Figura 3.2 Actualización del *SO Ubuntu*

Nos permite actualizar las listas de los paquetes que se encuentran disponibles, a sus versiones más recientes y su posterior instalación, respectivamente. Luego se procede con la instalación de *Docker,* sus complementos y los directorios donde se almacenarán todas las variables que permitirán el correcto levantamiento del servicio.

Por recomendación del desarrollador *Docker,* en su repositorio oficial [28], propone deshacerse de versiones y/o contenedores que puedan estar previamente instalados en el sistema operativo, por lo que, en la **Figura *3.3***, donde, se ejecuta una línea de comando que se encarga de eliminar *Docker, Docker Engine y Docker.io.*

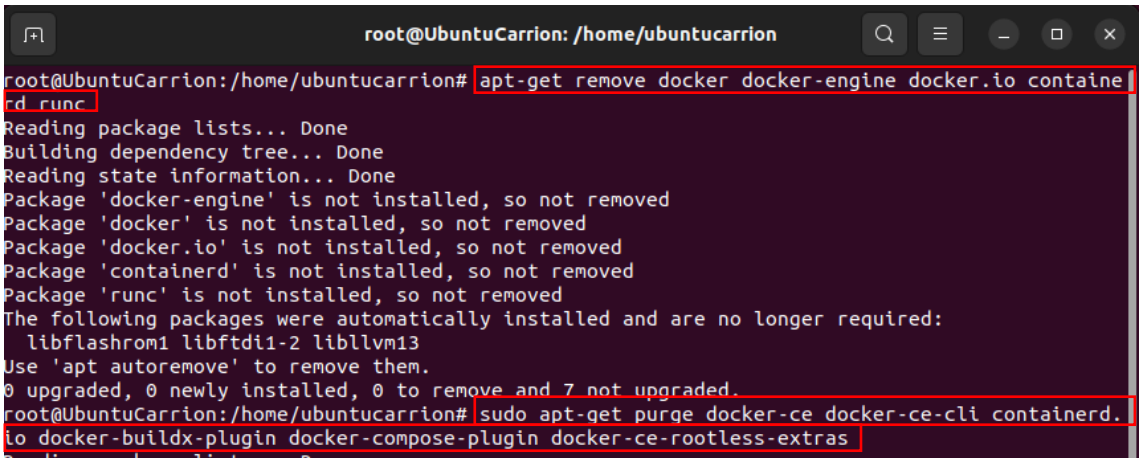

Figura 3.3 Desinstalación de posibles contenedores generados anteriormente

De manera más puntal, sin tener en cuenta que el resultado de la primera línea de comando que se describe como *"is not installed, so nor removed",* que significa, la

ausencia de contenido con esas etiquetas, en la mayoría de sus procesos. Se ejecutó la segunda línea de comando, que elimina los paquetes que complementos de la herramienta, asegurando así una completa desinstalación.

De manera resumida, se muestra los tres bloques de configuración para el repositorio de *Docker* en la **Figura *3.4***, donde:

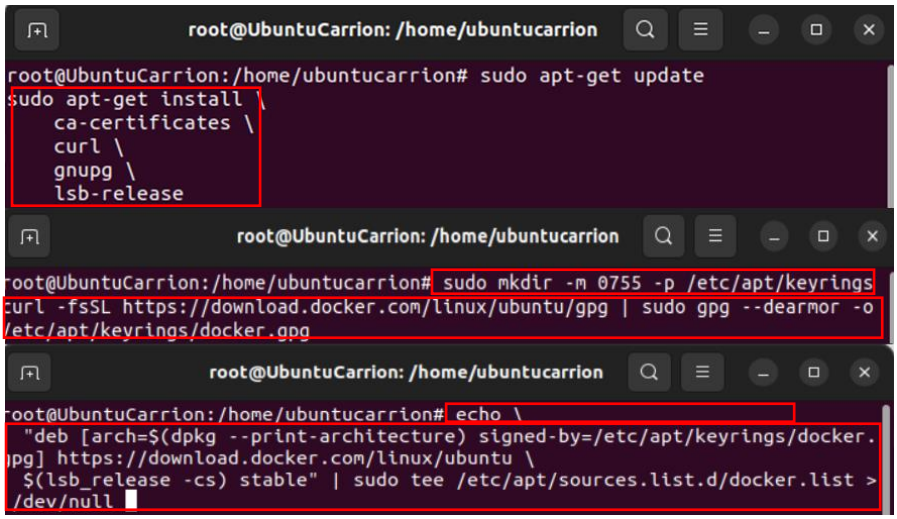

Figura 3.4 Configuración del repositorio de *Docker*

En el primer bloque se muestra la línea de comando que permite que el "*apt" de Linux* utilice los repositorios mediante *HTTP* [28], obteniendo la clave *GPG,* el *software* oficial de *Docker,* que sirve para el cifrado de manera híbrida ya que, combina la criptografía convencional con claves simétricas debido a la mayor velocidad que presentan con las claves públicas, para así, agilizar la compartición de dichas *keys* [29].

Además, convierte la codificación estándar *OpenGPG* y almacenándose en el directorio de *"apt",* estableciendo así los premisos necesarios para que dicho directorio lo descubra de manera confiable. En el segundo bloque, se ejecuta la clave *GPG*.

En el tercer y último bloque se configura el repositorio, ejecutando en un *"script"* de texto, que permite enlazar las características con la plataforma del desarrollador, detectando de manera automática la arquitectura del sistema para proceder con la descarga adecuada de los paquetes, procediendo con la verificación de la clave *GPG*.

Finalmente, se suma este repositorio a la lista de paquetes en el directorio *"sources.list.d"* [28].

Haciendo uso de la actualización de los paquetes *"apt",* se despliegan los paquetes recién añadidos en la **Figura *3.5***.

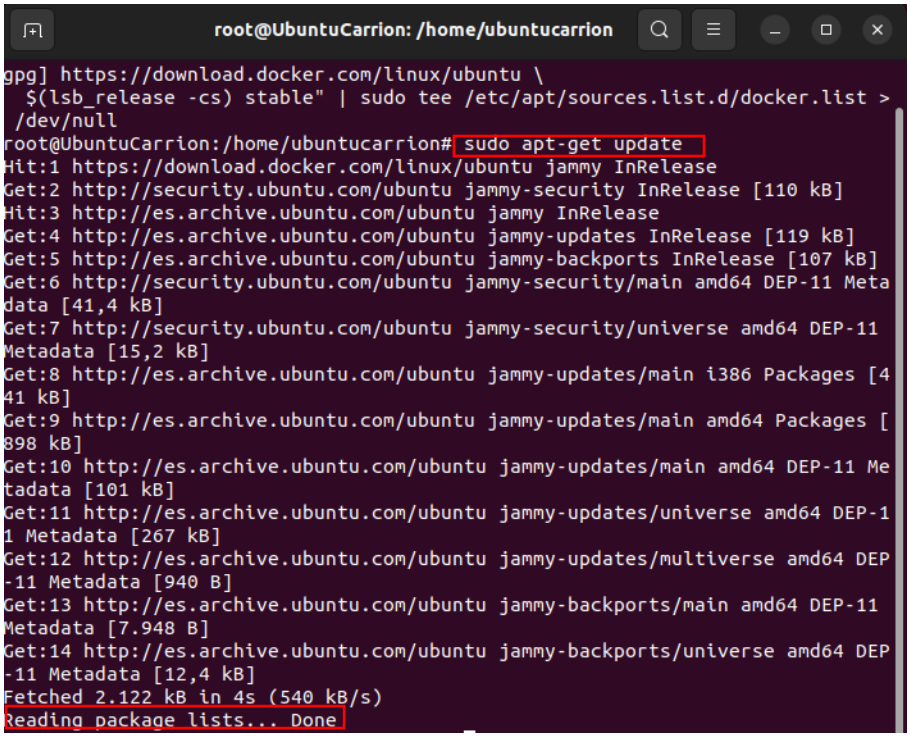

Figura 3.5 Evidencia de los paquetes *Docker*

Por lo que, se puede deducir que se encuentran alojados localmente de manera correcta [28]. Finalmente, se ejecutan las líneas de comandos descritos en la **Figura *3.6*** donde, además de instalar *Docker Engine,* se pueden sumar los componentes de la herramienta en el sistema como *"docker-ce",* que se ejecuta como la aplicación en segundo plano del motor de *Docker, "Docker-ce-cli"*, como la interfaz de interacción con el usuario del *CLI y "containerd.io",* que se encarga de la inicialización y finalización de los contenedores, es decir, la estabilidad durante su tiempo de vida.

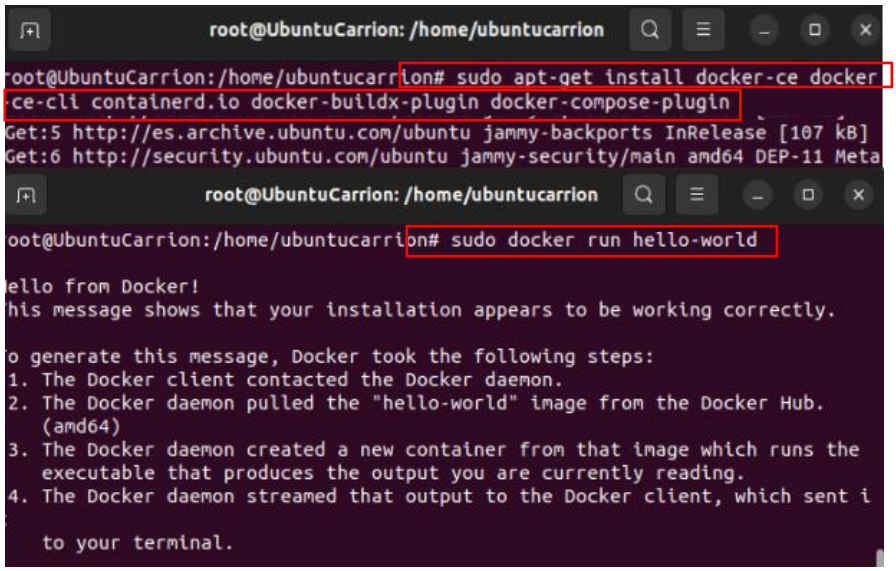

Figura 3.6 Instalación y comprobación de *Docker Engine*

Para evidenciar el correcto funcionamiento del contenedor, se genera un *script,* en el segundo bloque de la imagen, con la etiqueta *"hello-world"* en el contenedor de *Docker* [28], Cumpliendo con la creación de un contenedor a modo de prueba.

Para mantener un orden sobre el manejo de la serialización de datos, se propone el almacenamiento en dos diferentes directorios dentro del sistema operativo, tal como se muestra en la **Figura *3.7***.

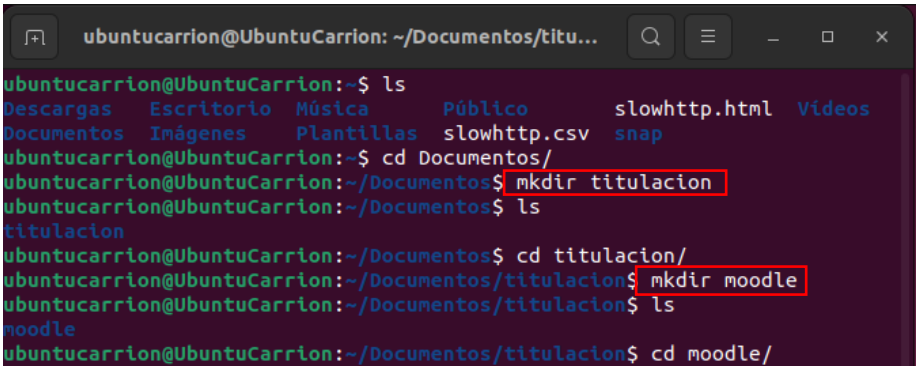

Figura 3.7 Creación de directorios

Una vez finalizada la creación de los directorios, se migra el manejo de código de *Ubuntu* 22.04.2 hacia *Visual Studio Code* [30], debido a que, entrega mejores prestaciones, tales como: uso de código abierto en su software, interfaz más amigable con respecto al manejo de código, ayuda a identificar las líneas de comando con extensión "*.YML",* así como los procesos de ejecución del mismo y el soporte que experimenta el usuario al uso de repositorios *Git.* Procediendo con la ejecución de las herramientas complementarias del contenedor.

Posteriormente, se accede desde un nuevo terminal de comando, para arrancar desde el directorio correspondiente, para establecer las tres líneas de comando: donde, la primera línea, se encarga de la instalación *Compose V2,* para el usuario dinámico en *$HOME*, el directorio correspondiente. La segunda línea se encarga de crear el directorio donde se alacena el *CLI* de los *plugins* de la configuración de *Docker.* Finalmente, en la tercera línea se indica la biblioteca *GitHub* [31] de soporte para la descarga de *Docker Compose*, ejecutando instantáneamente la descarga con los parámetros evidenciados en la **Figura *3.8***.

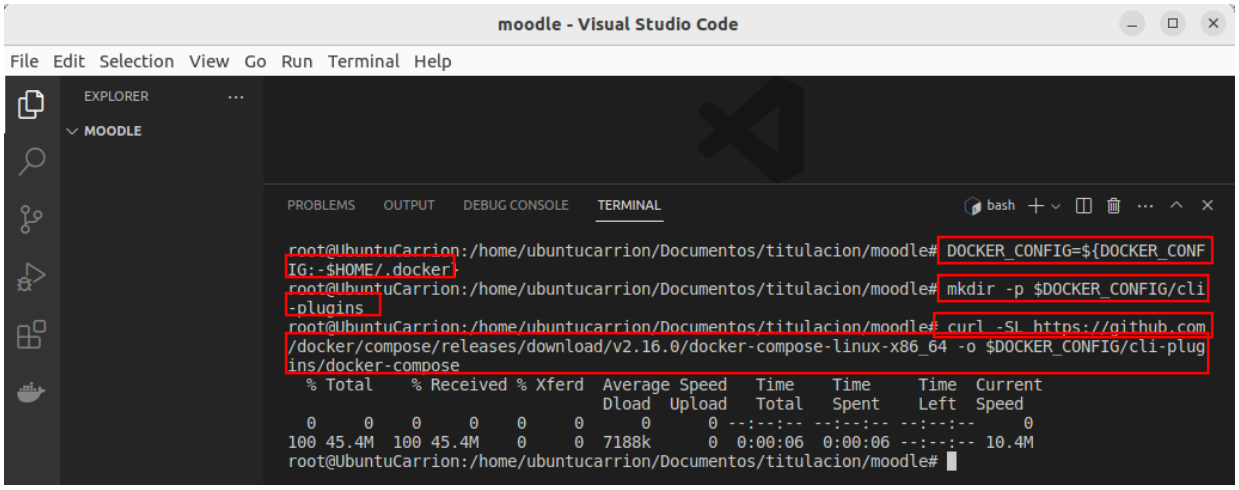

Figura 3.8 Instalación de *Docker-Compose*

En la **Figura *3.9***, se muestra en correcto funcionamiento del complemento *Docker Compose,* se establece el repositorio para la descarga del archivo *".yml"* [32], que almacena la imagen del servicio en lenguaje de programación "*YAML"***Figura *3.9***, el contenedor con el servicio de *Moodle y* las funcionalidades de la base de datos *MariaDB.*

Una vez ejecutado el comando, se evidencia en la parte superior izquierda de la figura la descarga del archivo de programación *"Docker-compose.yml".* Ambos procesos de levantamiento del *CMS* se detallarán posteriormente.

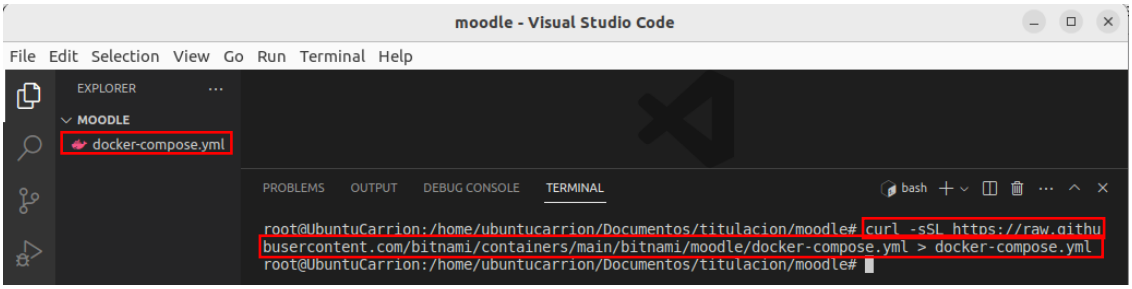

Figura 3.9 Levantamiento de *Bitnami* impulsado por *Moodle*

A continuación, se detallará el funcionamiento de las líneas de código en lenguaje *YAML* mostradas en la **Figura *3.10***.

Figura 3.10 Contenido de *docker-compose.yml*

- En la línea 1, se especifica la versión de la imagen del contenedor.
- De la línea 2 a la 30, se establece los parámetros para el desarrollo de las imágenes de la base de datos y el *CMS Moodle, donde:*
    a. En línea 4, se especifica el directorio donde se almacena localmente la imagen de la base de datos *MariaDB,* así como la versión utilizada, siendo esta la 10.6, con las etiquetas propias del origen de la imagen descargada.
    b. De la línea 7 a la línea 11 del código, se detallan las credenciales utilizadas para la administración del *CMS* en texto plano, establecida por defecto, para el primer acceso de la plataforma donde, posteriormente,

se editará dichas credenciales para trabajar con un mejor nivel de seguridad.

c. En la 13 línea de código, se especifica la ruta del almacenamiento de nuestra base de datos *MariaDB*.
d. De manera similar que el anterior apartado, las líneas de código antes descritas en la línea 15, se detalla la ubicación de la imagen, pero ahora de nuestro *CMS, Moodle.*
e. En la línea 17, se establece el puerto 80:8080 que, define el protocolo de transferencia por hipertexto *(HTTP)*, ejecutado en cada transacción *WWW,* definiendo así la semántica que se va a implementar en las diferentes terminales, (servidores, *host* y *proxies*) para su comunicación. Para evitar un probable conflicto con el puerto 80, se recurre a una alternativa para servidores web con el puerto 8080 [33].
f. En la línea 18, se establece los puertos para la implementación de *HTTPS,* ofreciendo un nivel mayor de seguridad [34].
g. De la línea 20 a la línea 25, se establece las credenciales para el acceso a la plataforma web, generadas de manera por defecto.
h. La línea 29 y 30, enlaza las dependencias de *Moodle* con la base de datos *MariaDB.*

- Finalmente, en las líneas de código 31 a 37, indica que el contenedor montará el volumen de los servicios de *Moodle* y la base de datos *MariaDB* en los directorios determinados.

Una vez, determinada la funcionalidad de la imagen de *Docker Compose,* se procedió con el levantamiento del servicio.

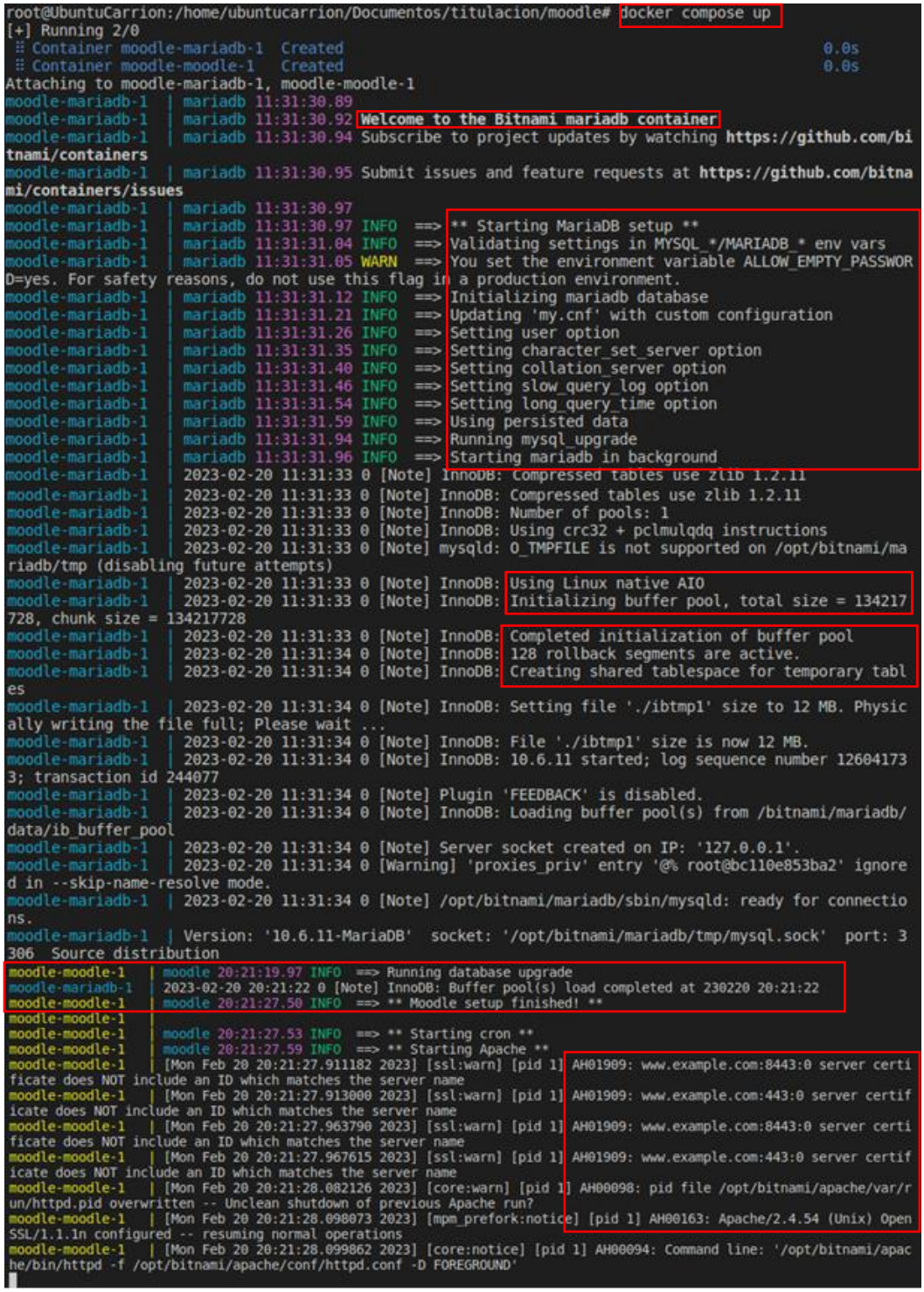

Figura 3.11 Ejecución de *Docker Compose*

Entre los más importantes se logran identificar:

- La inicialización del contenedor de la base de datos *MariaDB,* con *Bitnami LMS* con tecnología de servidor *Moodle(TM),* así como la descripción de los repositorios oficiales de soporte *GitHub*.

- Posteriormente, se muestra la interfaz de inicialización de la base de datos *MariaDB,* ejecutado la validación de la los elementos almacenados, así como el seteo de las variables de entorno.

  Se actualiza las opciones de arranque de la base de datos en el archivo *"my.cnf".* Se arranca: la configuración de las opciones del usuario, las características iniciales del servidor, parámetros del servidor que se ejecutan para la codificación de las reglas de operación del uso de los caracteres de idioma apropiados en la base de datos (intercalación del servidor), registros de consultas lentas de la base de datos ayudando de manera efectiva, a identificar las causas probables de rendimiento [35], el tiempo de vida de los datos que persisten al momento en el que se haya finalizado la aplicación o servicio. Finalmente, se ejecuta la base de datos *MariaDB* en segundo plano.
- Se muestra el subsistema de lectura/escritura nativa *(AIO nativ),* ejecutado en Linux, para realizar las solicitudes anticipadas de lectura/escritura en los archivos de los datos de las páginas.
- El siguiente bloque de mayor importancia, indica la inicialización y capacidad del *"buffer pool"* que, se encarga de mantener almacenado en la memoria todas las páginas que, previamente han sido leídas, para que, posteriormente, el tiempo de respuesta de un segundo acceso con la misma dirección, se reduzca notablemente, eliminando la necesidad de acudir nuevamente al disco duro.
- Se indica la finalización de la respectiva actualización de la base de datos, así como el establecimiento del servidor *web*.
- Finalmente, se establece la relación del servidor con los puertos correspondientes, asegurando así, el uso de la plataforma *CMS*, que se encuentra operando en conjunto con la base de datos establecida y, a su vez, se encuentra en constante comunicación con el servidor, obedeciendo el valor de las credenciales usadas, así como la dirección de red a la que se encuentra atada, similar al valor del dominio en una página *web* [36].

Mediante el comando evidenciado en la parte superior de la **Figura *3.12***, se despliega la lista de los contenedores generados, con las propiedades que describen el comportamiento de *Docker*, tales como: el identificador del contenedor, la etiqueta de la imagen del contenedor, la fecha de creación y el estado en el que se encuentra en relación a su acceso.

En la imagen, también se logra identificar el servicio de *Moodle* y la base de datos *MariaDB.*

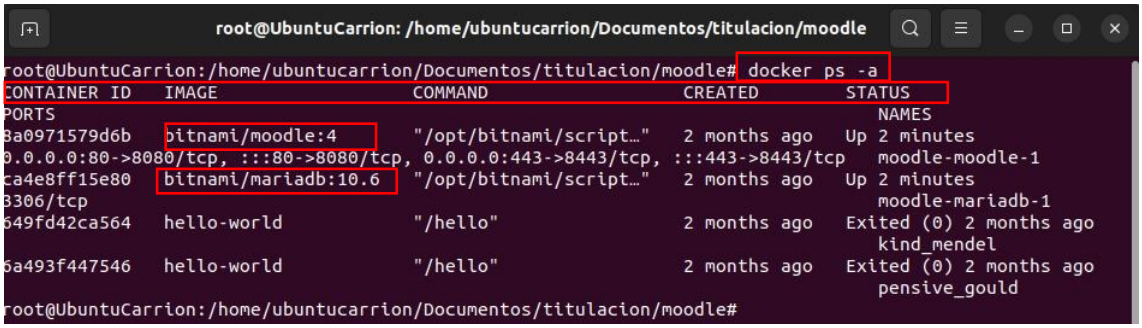

Figura 3.12 Listado de contenedores en ejecución

Dentro del terminal del Sistema Operativo *Ubuntu,* se logra identificar la dirección de red en IPv4 *localhost* 192.168.100.48, como se muestra en la **Figura *3.13***, vital para el alcance al *CMS*, a través de un aplicativo de navegador *web (Mozilla Firefox),* y así evidenciar el correcto funcionamiento del servicio.

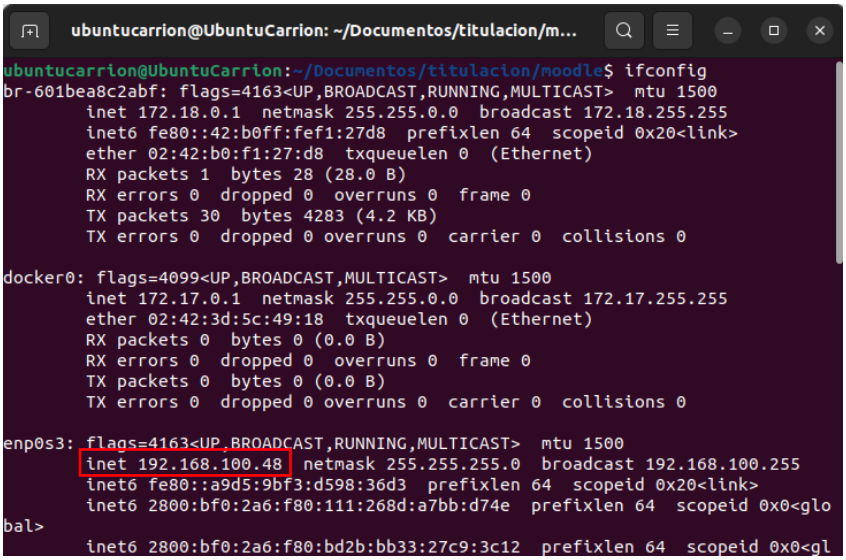

Figura 3.13 Dirección *IPv4* local del Sistema Operativo

En la **Figura *3.14*** se muestra la interfaz del primer acceso en el navegador *web*, utilizando la dirección descrita en la **Figura *3.13***.

En la parte superior derecha se muestra el botón que permite el despliegue de la ventana de autenticación para el acceso a la plataforma.

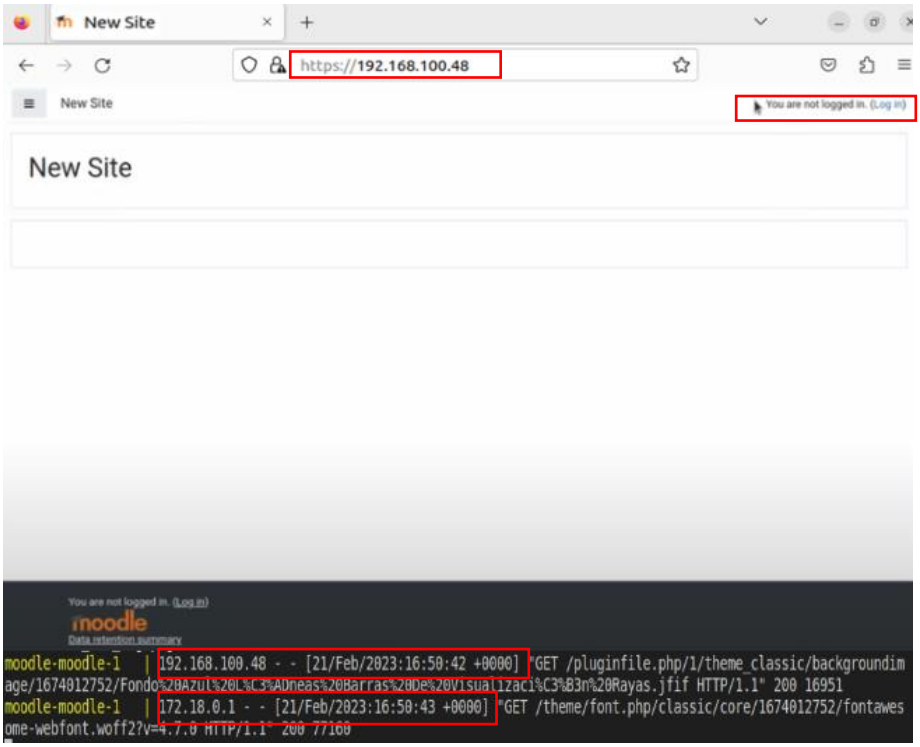

Figura 3.14 Primera interfaz del *CMS*

Se compara, en tiempo real, la ventana del navegador, con el entorno de desarrollo de código, donde se evidencia el registro automático del acceso al dominio con el *"localhost"* y la dirección *IPv4* de borde en el sistema operativo. Almacenando la dirección correspondiente, así como la fecha de la acción. Este proceso se verá reflejado en la ventana del terminal de *Docker Compose*, por lo que, se va a obviar el proceso de registro en la base de datos, y se va a evidenciar su almacenamiento, a manera de reinicio de servicio.

En la **Figura *3.15*** , se muestra las credenciales utilizadas para el ingreso a la plataforma *Moodle* en modo administrador, siendo estas *"user"* para el *ID* de usuario y *"bitnami"* como contraseña del mismo.

Estas credenciales son obtenidas por el desarrollador, en la descripción de la imagen en el contenedor [37].

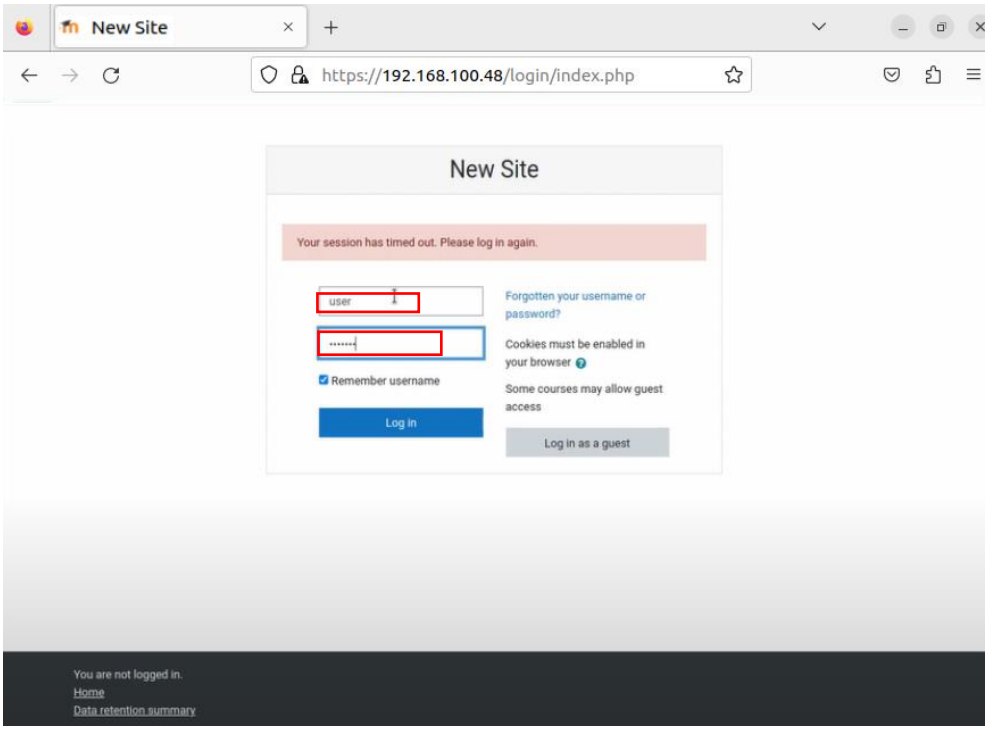

Figura 3.15 Ingreso a la plataforma con las credenciales generadas por defecto

Finalmente, se evidencia la primera interfaz que se genera de manera estándar por *Moodle,* mostrada en la **Figura *3.16***. Evidenciando así, el correcto levantamiento del sistema de gestión de contenidos para su posterior administración, con respecto a la creación de perfiles, credenciales, repositorios e interfaz visual hacia los usuarios del servicio.

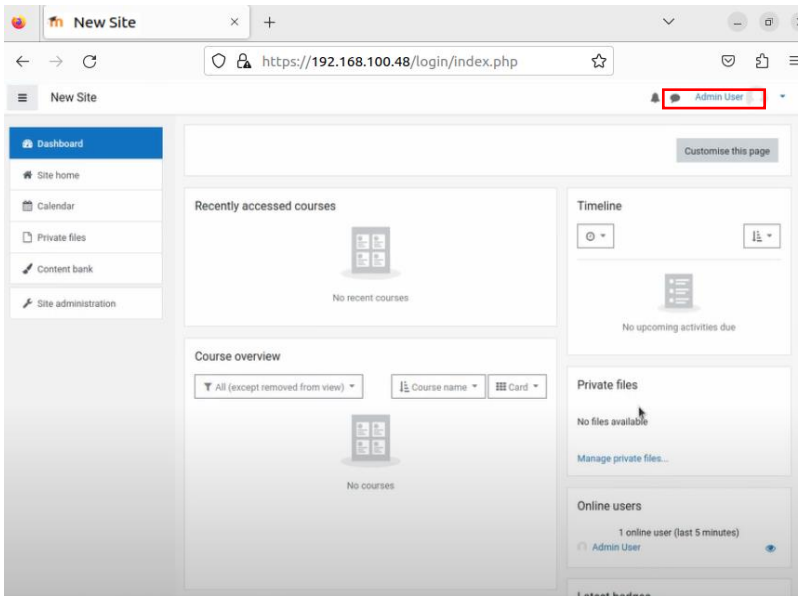

Figura 3.16 Primera interfaz de administración de *Moodle*

3.4 Pruebas sobre el funcionamiento de la *CMS*

Empezando por la personalización de la plataforma, se levanta el servicio, obedeciendo el proceso previamente descrito. Se accede de forma remota, utilizando la dirección del *"localhost"* del contenedor. Lo que limitará el consumo de recursos en el sistema operativo anfitrión del servicio.

A continuación, se muestran de manera resumida, los principales parámetros de edición, sobre la personalización de la interfaz del *CMS*, para lograr mayor familiaridad para los usuarios [38].

En la parte superior izquierda de la **Figura *3.17***, se identifica la dirección en el *CMS* que permite la personalización de los logotipos que se despliegan en el servicio, luego se personaliza la página de inicio para el *logueo* de los usuarios en la misma dirección establecida, finalizando con el almacenamiento de los cambios realizados a lo que se procederá el almacenamiento en la base de datos.

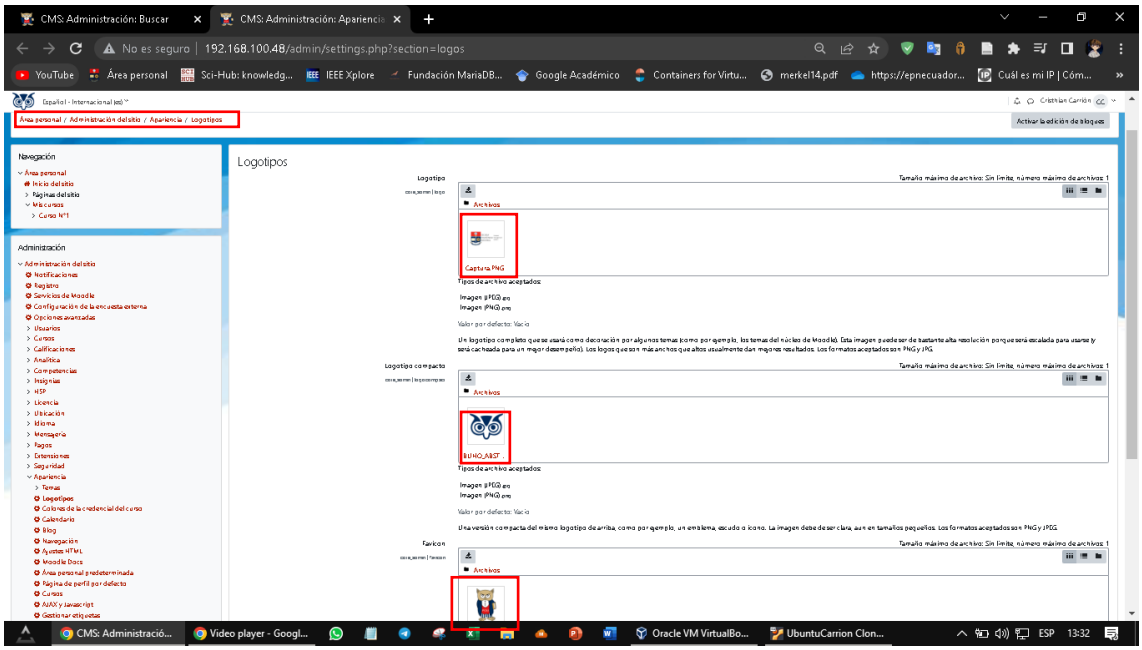

Figura 3.17 Personalización de logotipos del *CMS*

En la **Figura *3.18***, se muestra la dirección de los dentro del servicio que permite la creación de un nuevo curso, en la parte superior derecha, donde, de manera intuitiva se generan los campos mostrados.

Se evidencia en la parte izquierda de la figura, el hipervínculo que se genera, haciendo posible la configuración de la interfaz de curso en cuestión, con la etiqueta de *"*Curso N°1*"*.

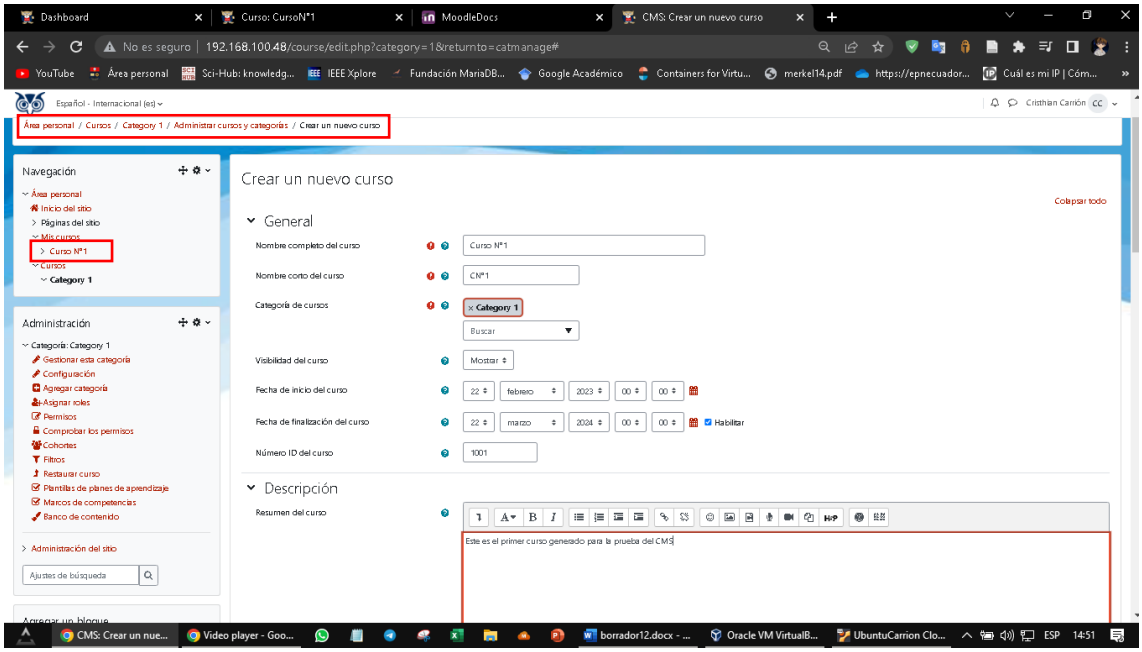

Figura 3.18 Creación de curso

En la **Figura *3.19***, se muestra la dirección, interfaz y los campos llenados para generar una cuenta destinada al uso de un profesor, asignando las credenciales correspondientes para el correspondiente acceso a la plataforma.

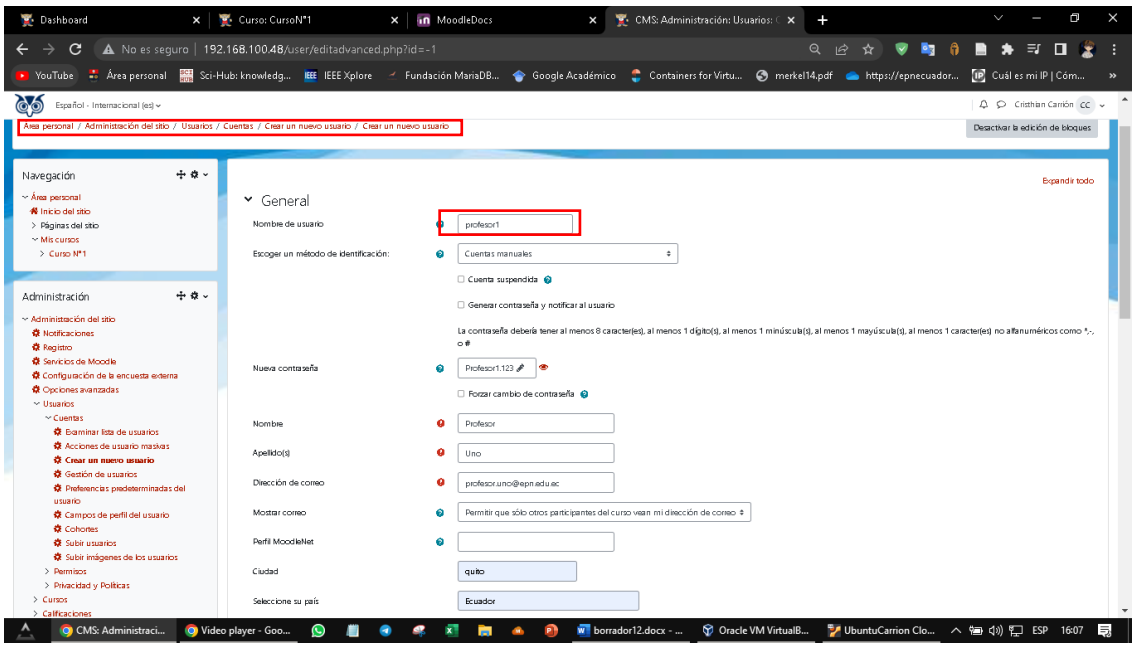

Figura 3.19 Creación de usuario "Profesor"

De manera similar a la **Figura *3.19***, se genera las cuentas de los usuarios destinadas para los estudiantes. Obteniendo un número total de usuarios de: 2 profesores, 10

estudiantes y un administrador, con todos los campos correspondientes, evidenciado en la **Figura *3.20***.

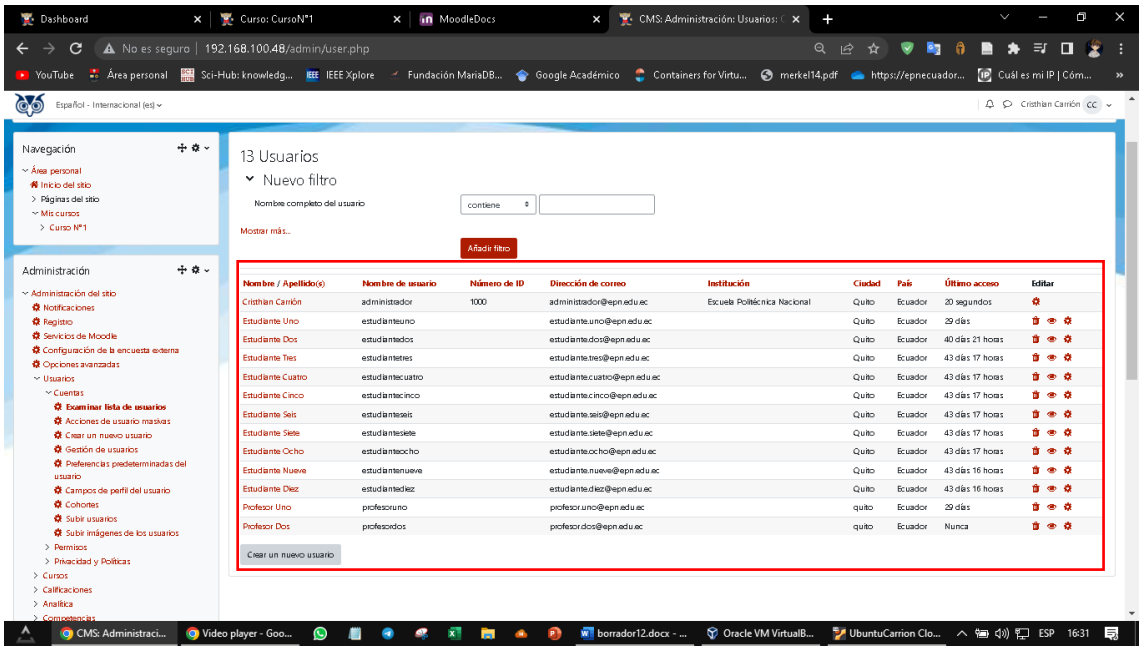

Figura 3.20 Lista de usuarios

Para asegurar la correcta concesión de permisos hacia los perfiles ya definidos, se establecen los roles hacia los tres diferentes grupos de usuarios: Administrador, Profesor y Estudiante, mostrado en la **Figura *3.21***, donde, se especifican las funciones de edición que pueden o no realizar en la plataforma.

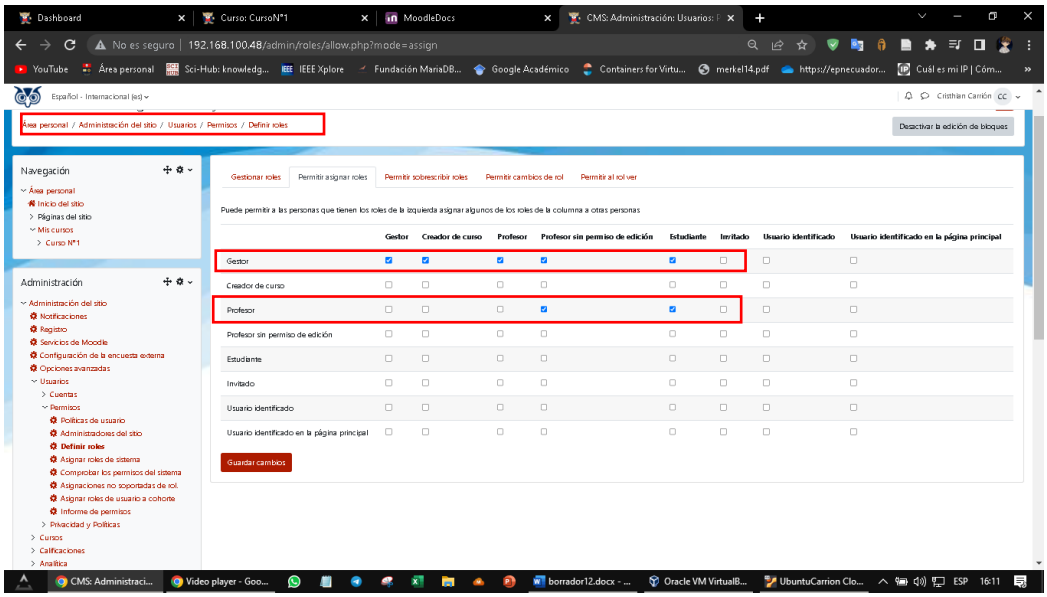

Figura 3.21 Definición de roles

Una vez generado el curso a modo de prueba, se procede a inscribir a los usuarios en el aula virtual, proceso que se identifica en la **Figura *3.22***, donde, se selecciona el usuario y se asigna el rol que desempeñará en el “CursoN°1”.

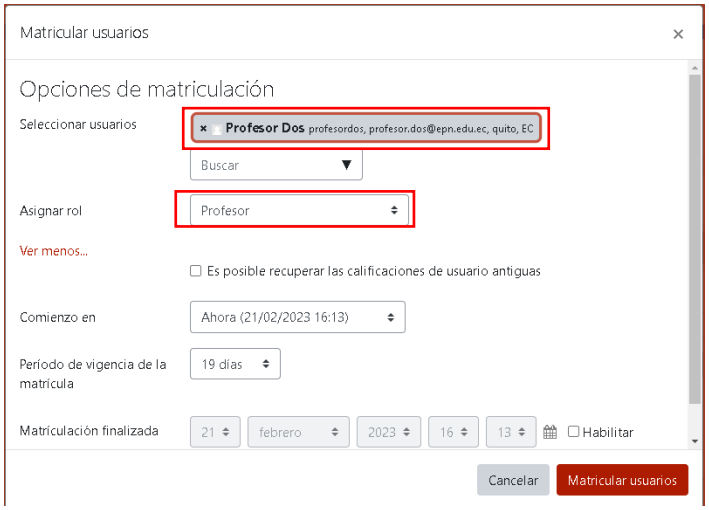

Figura 3.22 Matriculación en “Curso N°1”

En la **Figura *3.23***, se puede visualizar la lista completa de perfiles considerados en el aula virtual, así como la asignación previa de las credenciales que permitirán el acceso al aula virtual, contando con un usuario y contraseña única, generadas de manera administrativa.

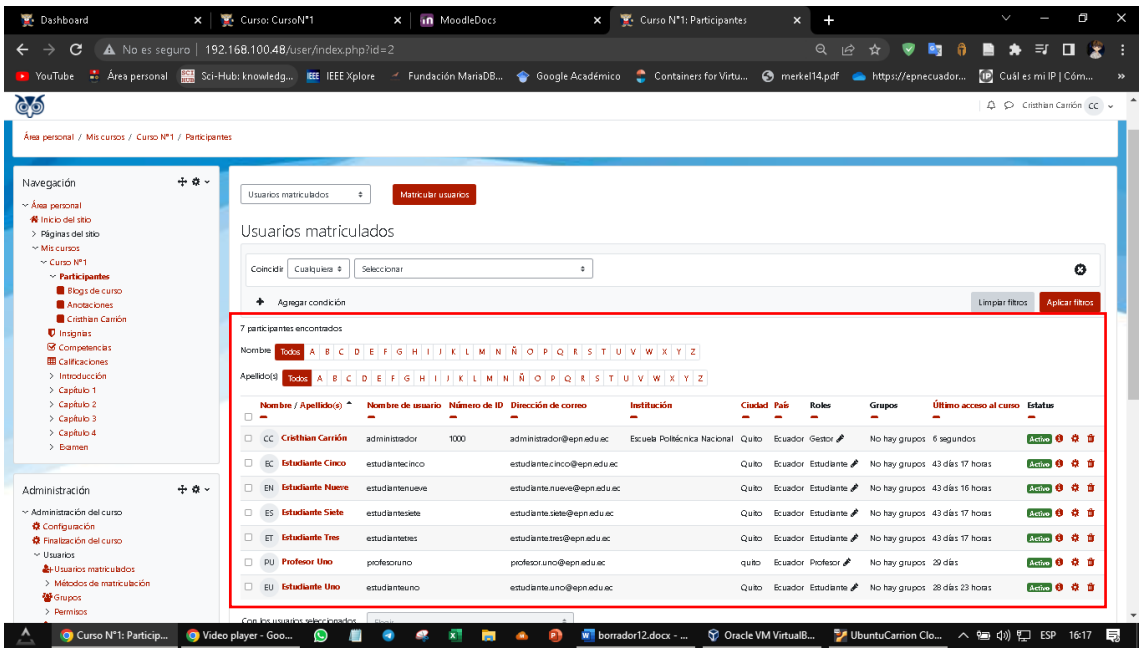

Figura 3.23 Listado de usuarios en “Curso N°1”

Para evidenciar el alcance de operación del *CMS,* se accederá a los perfiles previamente creados, desde otro terminal, conectado a la misma red *LAN.*

En la **Figura *3.24***, se muestra el enlace establecido hacia la red desde el equipo que ejecuta el contenedor con todos los servicios derechos que, se muestra en la parte izquierda de la imagen. En la parte derecha, se muestra la conexión hacia internet obtenida a través de la misma red, pero, usando un diferente equipo *(host)*.

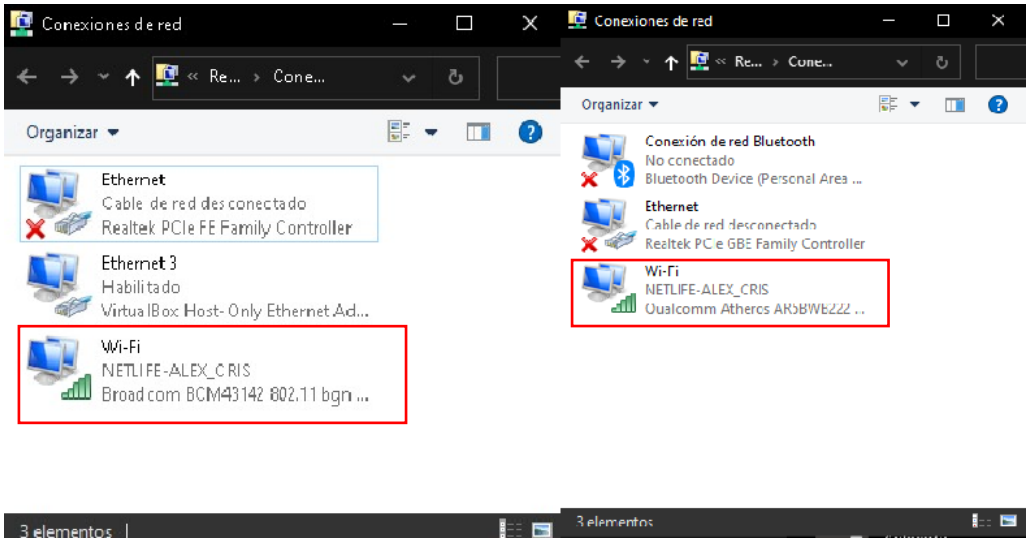

Figura 3.24 Conexión de red, en ambos terminales

Haciendo uso de la dirección de red obtenida de la **Figura *3.25*** se accede al *CMS* desde un *Host* diferente, al equipo que contiene el sistema de gestión de contenidos, obteniendo así, la certeza de la disponibilidad de la plataforma virtual, hacia todos los elementos finales con capacidad de establecer conexión a la red *LAN*.

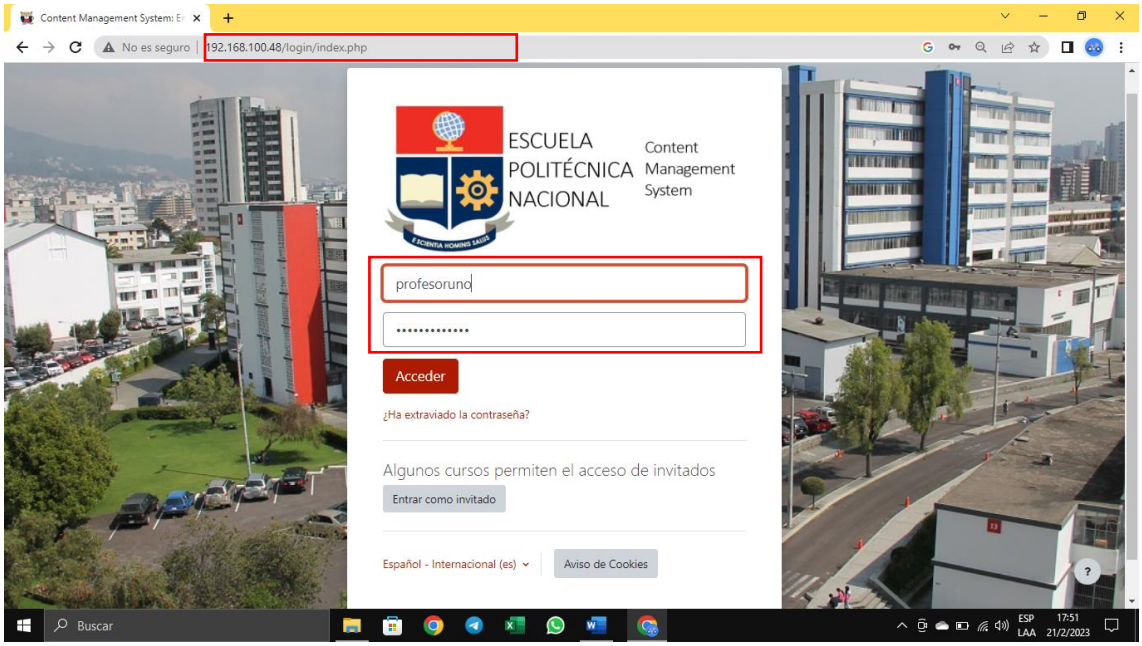

Figura 3.25 Acceso a perfil de profesor desde diferente *host*

A continuación, a manera de prueba, se va a generar contenido con el perfil de profesor, y diferentes aplicaciones en cuatro diferentes módulos o capítulos.

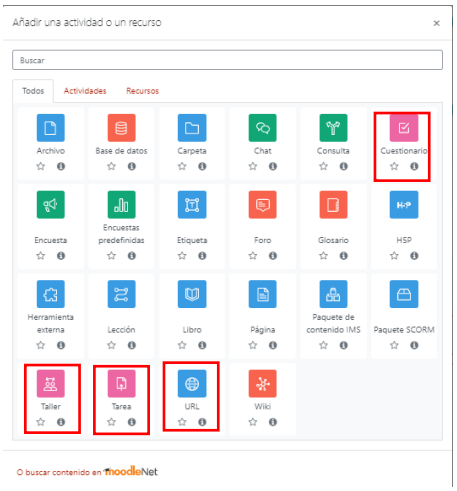

Figura 3.26 Recursos para generar actividades

Una vez ingresado al perfil de profesor dentro de la plataforma *Moodle,* el *CMS* ofrece una gran variedad de elementos prefabricados, de los cuales, 4 se ejecutarán a manera de prueba, indicados en la **Figura *3.26***, en cuatro diferentes módulos o capítulos, como se indica a continuación.

A manera de prueba, se genera 4 diferentes elementos de entrega y posterior evaluación: tarea, taller, cuestionario y examen, evidenciados en la **Figura *3.27***, designado a un elemento por cada capítulo.

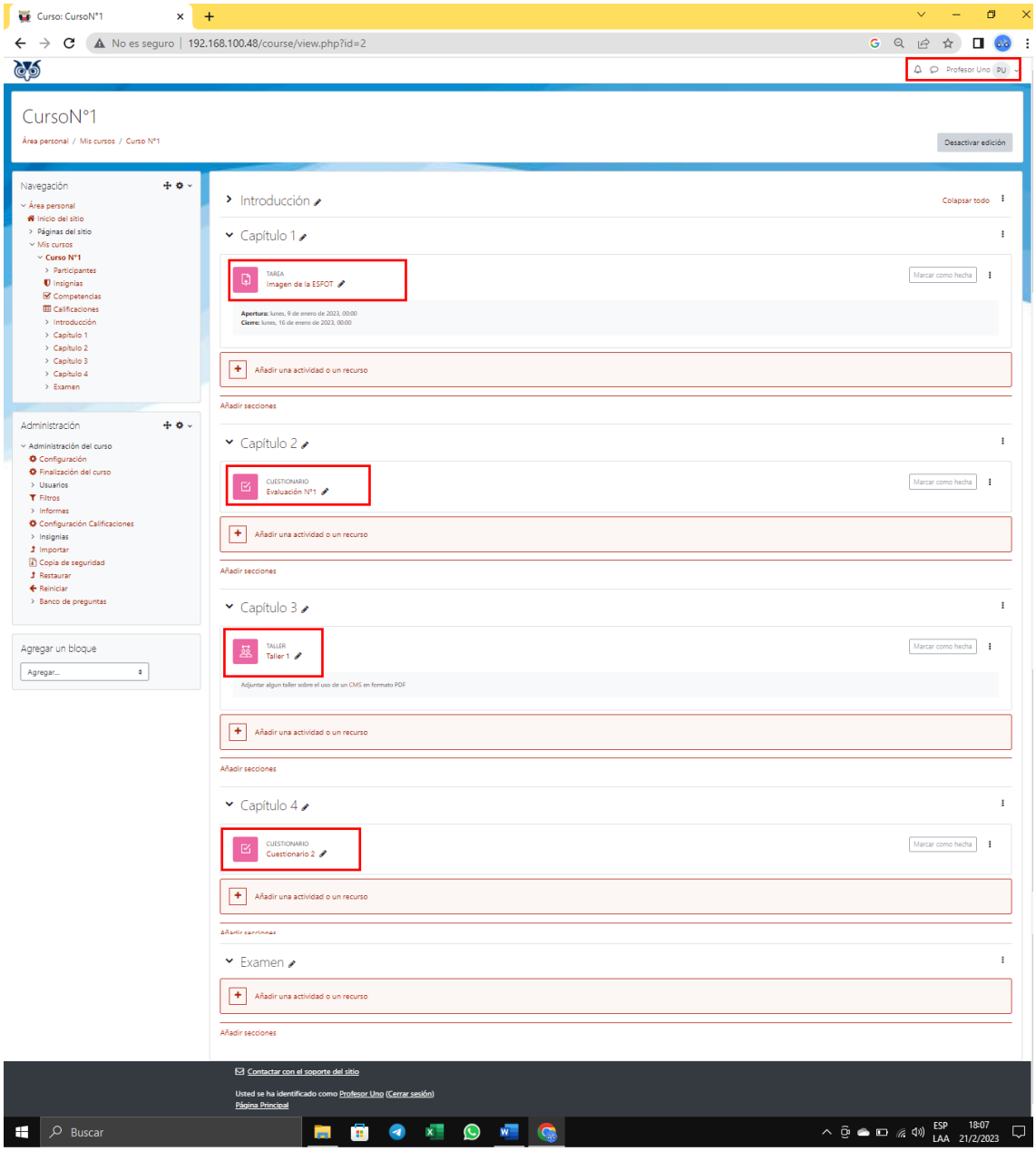

Figura 3.27 Contenido generado en “Curso N°1”

Cada elemento de evaluación contiene sus propios parámetros de entrega. A continuación, solo se va a ejemplificar el contenido de cuestionario, debido a la relación más puntual con el perfil de estudiante.

Posteriormente, se evidencia el contenido y la interfaz de “Evaluación N°1”, como se muestra en la **Figura *3.28*** , los elementos implementados, sobre el cuestionario propuesto en “Curso N°1” done, se evidencia la dirección para configurar los módulos con las preguntas correspondientes, así como las respuestas asignadas a cada una de esas preguntas.

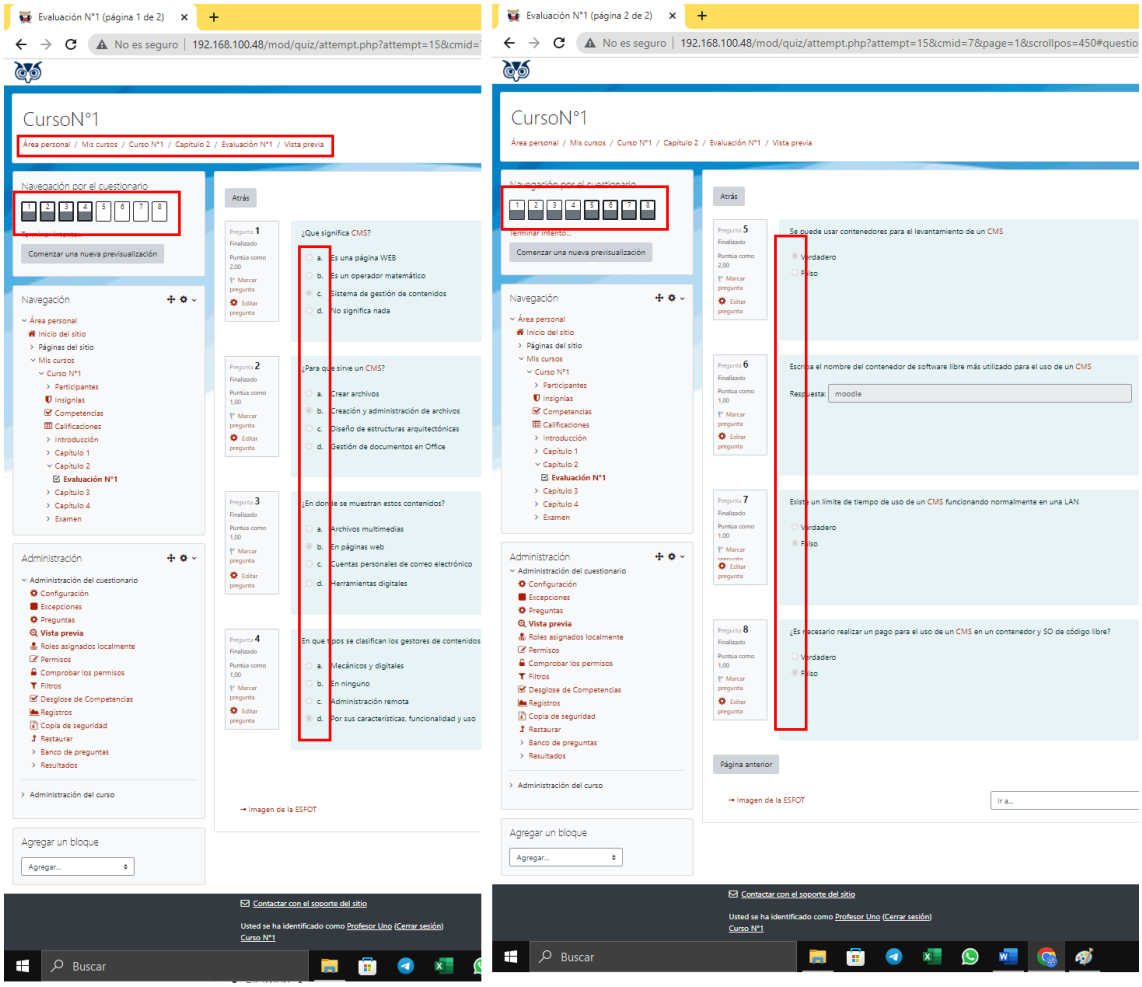

Figura 3.28 Cuestionario generado con el perfil “profesoruno”

A manera de prueba, se procedió a acceder desde un tercer terminal *(host)*, para el testeo de la cuenta de estudiante.

Desde un tercer terminal (dispositivo móvil), se accede a la dirección *IPv4* propia del *CMS*, que se logra identificar en la parte superior de la **Figura *3.29***.

En la parte inferior se muestra el perfil de acceso, mostrado por las siglas *“EU”*, es decir, el perfil “estudianteuno”.

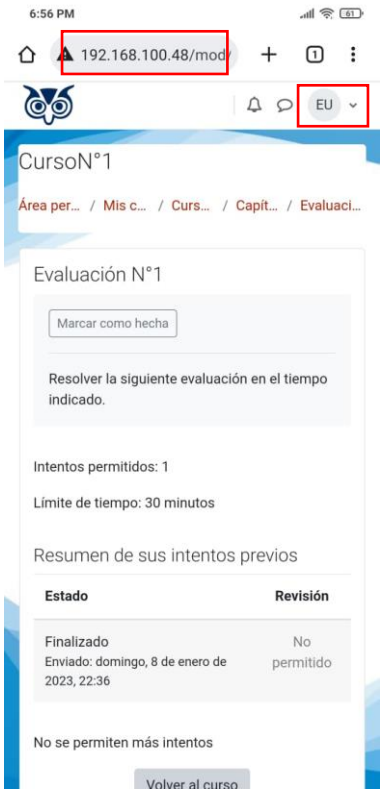

Figura 3.29 Evaluación desde dispositivo móvil, realizada en perfil de estudiante

Con respecto al resto del contenido, se muestra la evidencia de haber concluido con la evaluación previamente diseñada.

Finalmente, en la **Figura *3.30***, se evidencia el funcionamiento del *CMS* al comprobar los resultados obtenidos del evento de evaluación, realizado por el segundo terminal.

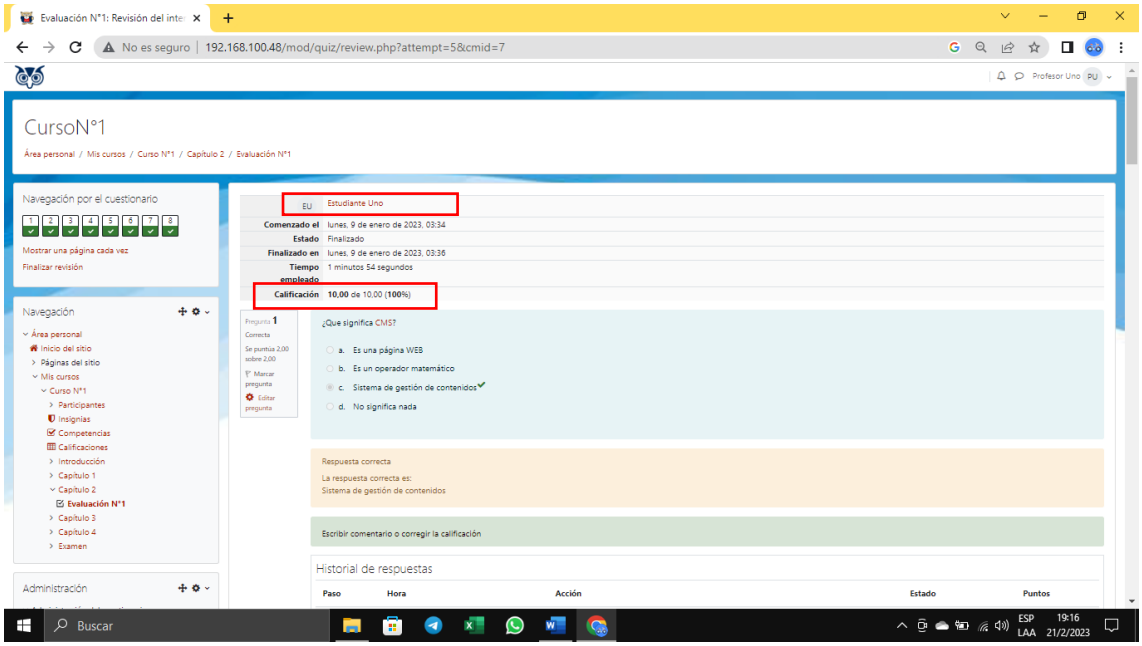

Figura 3.30 Revisión de evento de evaluación desde perfil del profesor

En la plataforma, existe un gran número de interfaces creadas a manera de prueba, donde, una parte de estas interfaces se resumen en la **Figura *3.31***, ya que, tabula cada evento realizado por los 5 perfiles de estudiantes, de manera automática.

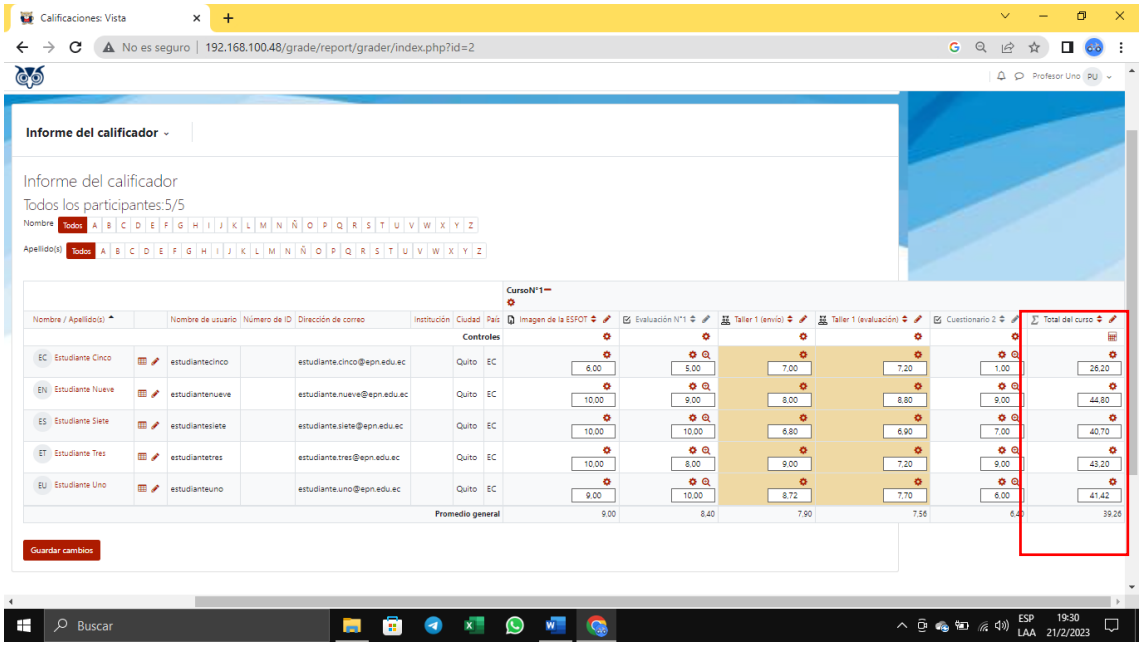

Figura 3.31 Gestión de las calificaciones de los estudiantes

4 CONCLUSIONES

- Es enorme el impacto que se obtiene con la implementación de *Docker,* ha demostrado una revolución sobre los términos de movilidad, tasa de compresión, eficacia y eficiencia, cuando se habla de aplicaciones. Donde, el principal elemento que lo potencia es el soporte que se encuentra en constante actualización, creado por grupos de desarrolladores que, de manera gratuita, entregan a la comunidad, nuevas y novedosas técnicas para el despliegue de contenedores además de *Moodle* como: *MySQL, Tomcat, Apache, Saleor, SSL, Drupal, WordPress, Nginx,* etc.
- La imágenes o archivos *file,* que se ejecutan dentro del contenedor se prestan de manera similar que un *script,* obedeciendo a todo su sistema de ficheros previamente establecidos, en los que basan el servicio, a manera de punto de partida, siempre y cuando se encuentre en ejecución *Docker* y/o sus complementos. También es correcto afirmar que una imagen en *Docker* se comporta tal como lo haría una instantánea o plantilla en un sistema de máquinas virtuales.
- El uso de la base de datos *MariaDB,* ofrece mejores prestaciones, ya que, otorga un mejor rendimiento, más ligero y de mayor rapidez, con respecto a la base de datos *MySQL,* además, en la actualidad utiliza módulos de código cerrado, mientras que *MariaDB,* opera libremente en su totalidad. Utilizando más de 12 motores nuevos para el almacenamiento, y más de 200 mil conexiones.
- Mediante el establecimiento de los recursos necesarios de *hardware* y *software,* así como todas las herramientas necesarias, se cumplió, sin complicaciones la creación y manejo del prototipo.
- Se concluye que, las propiedades generadas por defecto con respecto a la interfaz de *Moodle,* son extensas, cubriendo casi en su totalidad a todos los requerimientos del usuario, por lo que, para el levantamiento del prototipo se realizaron las principales pruebas: la personalización de la interfaz de *Login,* instalación y preferencia de idioma español (latino), eliminación de marcas *Moodle*, disposición del calendario, actividades próximas, cursos accedidos recientemente, realización de entregables elaborados por 5 diferentes perfiles de estudiantes en 4 diferentes entregables y su respectiva calificación, contacto por mensajería desde y hacia los perfiles de estudiantes y profesores hacia el soporte de la plataforma (administrador), entre otros.

- Una función importante de *Docker Compose*, es el *"buffer pool"* que se encarga de almacenar en memoria, los procesos de las páginas que previamente han sido leídas en disco duro, de manera que, al momento de ejecutar una segunda lectura, el tiempo de respuesta se reduce, ya que ahora no se recurre al accedo del disco duro.
- Un contenedor desarrollado a través de un sistema operativo libre *Linux*, ofrece mayor facilidad con respecto a la gestión de varios procesos debido al uso del sistema *init,* ofreciendo la facilidad de ejecutar varias aplicaciones, como si se tratara de una sola.

5 RECOMENDACIONES

- Es posible que se genere alguna clase de contratiempo con el arranque de la base de datos, que puede tener relación con el archivo *"my.cnf"* que se encarga de almacenar las opciones de arranque de la base de datos, las cuales se deben ejecutar en un orden específico. Por lo que, es probable que el administrador experimente una falta de actualización en los cambios realizados en la plataforma, por ejemplo, en *Moodle*, Elementos que ya no son parte de la personalización de la plataforma del CMS. Para solventar estos escenarios, se recomienda la comprobación manual del archivo, que, por lo general se encuentra en las siguientes direcciones: *"/etc/my.cnf", "$MYSQL_HOME/mi.cnf", "/etc/mysql/my.cnf",* donde, si se evidencia más de un elemento, el contenedor cargará cada uno estos, lo que se derivará en la anulación de estos elementos entre sí. Se recomienda el reinicio del servidor una vez editado el archivo y eliminar cualquier elemento extra en *"my.conf"* como, por ejemplo, *"—defaults-file"*.
- Si se provee de mayor capacidad gráfica en el terminal, se recomienda asignar la mayor cantidad de procesador gráfico hacia la virtualización, superando la recomendada por el desarrollador, ya que, en la práctica se experimentan interrupciones en su uso.
- Con respecto al alcance del servicio, se recomienda tener en cuenta el manejo de la red *LAN*, donde, a manera de ejemplo, se propone dos escenarios experimentados a lo largo de la implementación del servicio. El primero, una red *LAN home* (domestica), y una segunda, una red *LAN* a nivel institucional *(EPN)*. Al iniciar el contenedor y todos los parámetros de la plataforma correspondientes, se alcanzó inmediatamente la disponibilidad hacia todos los

host contenidos en la red, si embargo, al ejecutar el mismo servicio a nivel institucional solo se logró obtener el servicio en el equipo que se encuentra ejecutando el contenedor. Se recomienda el cambio de la dirección *MAC* del equipo, el tipo de adaptador a “adaptador tipo puente” y denegar el modo promiscuo a través de la máquina virtual. Debido a que, por lo general, a nivel institucional se trabajan con protocolos que agilizan la conexión en todos los puntos de red, generando una única dirección *localhost* en todos los terminales virtualizados, creando conflicto al momento de levantar un servicio específico.

- Si no se dispone de recursos suficientes para la ejecución de *Ubuntu* en una máquina virtual, se recomienda la utilización de Ubuntu Server, ya que, al ejecutarse sin interfaz gráfica, garantiza un mayor rendimiento en un equipo de gama media/baja y no reduce las funcionalidades finales que nos ofrecen los contenedores, ya que, como se evidenció en el presente escrito, el manejo e implementación de base de datos, imágenes de *Docker* y sus complementos, no exigen mayores recursos.
- A pesar de que puede asignar una traducción de dirección IP hacia un dominio en texto del *localhost* del contenedor, logrando así alcanzar una mayor familiaridad, no se recomienda utilizar este proceso, debido a que este entorno se puede transportar hacia otras redes de área local, donde es muy probable que, la dirección asignada *IPv4,* sea diferente a la primera.
- Se recomienda el uso de un terminal diferente al incorporado por defecto en el sistema operativo *Linux,* ya que, por ejemplo, al trabajar con *Visual Studio Code,* mantenemos un entorno más amigable con respecto a la identificación de los elementos contemplados dentro de la imagen de los contenedores, para entender de mejor manera su comportamiento.

6 REFERENCIAS BIBLIOGRÁFICAS

[1] E. Ramalingam, «Research Paper on Content Management Systems (CMS): Problems in the Traditional Model and Advantages of CMS in Managing Corporate Websites,» 2016.

[2] R. Vacek, «Putting the Library Website in Their Hands: The Advantages and Challenges of a Homegrown Content Management System,» 2008.

[3] Moodle, «Acerca de Moodle,» 26 12 2022. [En línea]. Available: https://docs.moodle.org/all/es/Acerca_de_Moodle#:~:text=Moodle%20es%20una%20plataforma%20de,crear%20ambientes%20de%20aprendizaje%20personalizados.. [Último acceso: 23 01 2023].

[4] Moodle, «Find your Moodle partner,» [En línea]. Available: https://moodle.com/solutions/certified-service-providers/. [Último acceso: 23 01 2023].

[5] V. G. a. K. M. a. A. G. da Silva, «Containers for virtualization: An overview,» *Applied Computer Systems,* vol. 23, nº 1, pp. 21-27, 2018.

[6] D. a. o. Merkel, «Docker: lightweight linux containers for consistent development and deployment,» *Linux j,* vol. 239, nº 2, p. 2, 2014.

[7] J. a. D. D. a. M. T. a. A. D. a. T. N. Lindstrom, «Non-Volatile Memory System and Applications Symposium (NVMSA),» *NVM aware MariaDB database system,* pp. 1-6, 2015.

[8] S. Watts, «ACID Explained: Atomic, Consistent, Isolated & Durable,» bmc, 24 04 2020. [En línea]. Available: https://www.bmc.com/blogs/acid-atomic-consistent-isolated-durable/. [Último acceso: 12 01 2023].

[9] B. Gambrel, «Networking Fundamentals,» [En línea]. Available: https://fti.uajy.ac.id/wp-content/uploads/2018/02/Networking-Fundamentals.pdf. [Último acceso: 23 01 2023].

[10] E. F. MEDINA, «PLATAFORMA BITNAMI - MOODLE,» 2018. [En línea]. Available:

http://repositorio.unsa.edu.pe/bitstream/handle/UNSA/8500/EDMfamee.pdf?sequence=1&isAllowed=y. [Último acceso: 17 02 2023].

[11] T. Bui, «Analysis of docker security,» *arXiv preprint arXiv:1501.02967.,* 2015.

[12] Amazon, «¿Qué es la API RESTful?,» 2022. [En línea]. Available: https://aws.amazon.com/es/what-is/restful-api/. [Último acceso: 15 02 2023].

[13] D. Docs, «Docker Engine overview,» 2023. [En línea]. Available: https://docs.docker.com/engine/. [Último acceso: 15 02 2023].

[14] Debian, «Package: qemu (1:5.2+dfsg-11+deb11u2),» [En línea]. Available: https://packages.debian.org/stable/qemu. [Último acceso: 15 02 2023].

[15] Docker, «Descripción general de Docker Compose,» [En línea]. Available: https://docs.docker.com/compose/. [Último acceso: 20 01 2023].

[16] D. y. P. B. y. C. F. F. y. D. J. P. y. A. A. Reis, «Developing Docker and Docker-Compose Specifications: A Developers' Survey,» *IEEE Access,* vol. 10, pp. 2318-2329, 2022.

[17] Microsoft, «Creación de flujos de trabajo de integración continua (CI) mediante Acciones de GitHub,» [En línea]. Available: https://learn.microsoft.com/es-es/training/modules/github-actions-ci/. [Último acceso: 21 01 2023].

[18] IONOS, «Digita guide IONOS,» IONOS, 03 09 2019. [En línea]. Available: https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/bases-de-datos/. [Último acceso: 15 01 2023].

[19] Oracle, «VirtualBox,» Oracle, [En línea]. Available: https://www.virtualbox.org/. [Último acceso: 12 02 2023].

[20] F. Mehnert, «Genode 14.02 supports VirtualBox on Nova microhypervisor,» 28 02 2014. [En línea]. Available: https://web.archive.org/web/20141010165126/https://www.virtualbox.org/pipermail/vbox-dev/2014-February/012126.html. [Último acceso: 12 02 2023].

[21] B. M. a. H. J. B. Al Housani, «The Linux review - Ubuntu desktop edition - version 8.10,» *2009 International Conference on the Current Trends in Information Technology (CTIT),* pp. 1-6, 2009.

[22] K. M. Mujahid Tabassum, «Software Evolution Analysis of Linux (Ubuntu) OS,» de *2014 International Conference on Computational Science and Technology (ICCST)*, 2014.

[23] Ubuntu, «Download Ubuntu Desktop,» [En línea]. Available: https://ubuntu.com/download/desktop. [Último acceso: 12 02 2023].

[24] Ubuntu, «Get Ubuntu Server,» [En línea]. Available: https://ubuntu.com/download/server. [Último acceso: 12 02 2023].

[25] UbPorts, «Ubuntu Touch,» [En línea]. Available: https://ubuntu-touch.io/es/. [Último acceso: 12 02 2023].

[26] MariaDB, «About MariaDB,» [En línea]. Available: https://mariadb.org/es/. [Último acceso: 13 01 2023].

[27] H.-P. D. Co, «HP Customer Support - Knowledge Base,» [En línea]. Available: https://support.hp.com/sg-en/document/c04588758. [Último acceso: 15 02 2023].

[28] D. Docs, «Instalar Docker Engine en Ubuntu,» 2022. [En línea]. Available: https://docs.docker.com/engine/install/ubuntu/. [Último acceso: 16 02 2023].

[29] M. Ashley, «Guía de ``Gnu Privacy Guard",» [En línea]. Available: https://www.gnupg.org/gph/es/manual.html. [Último acceso: 16 02 2023].

[30] Microsoft, «Edición de código, Visual Studio Code,» 2023. [En línea]. Available: https://code.visualstudio.com/. [Último acceso: 17 02 2023].

[31] GitHub, «Docker Compose V2,» [En línea]. Available: https://github.com/docker/compose. [Último acceso: 17 02 2023].

[32] Bitnami, «Bitnami LMS impulsado por Moodle™ LMS,» 17 02 2023. [En línea]. Available: https://bitnami.com/stack/moodle. [Último acceso: 18 02 2023].

[33] C. Mateu, «Desarrollo de Aplicaciones Web,» Eureca Media, 03 2004. [En línea]. Available: https://libros.metabiblioteca.org/bitstream/001/591/1/004%20Desarrollo%20de%20aplicaciones%20web.pdf. [Último acceso: 18 02 2023].

[34] V. Docs, «Configuración de la puerta de enlace 8443,» VMware Horizon Cloud Service, 28 10 2021. [En línea]. Available: https://docs.vmware.com/es/VMware-

Horizon-Cloud-Service/services/hzncloudmsazure.admin15/GUID-5AB10EF7-0CC1-4E90-8C48-AD093021F959.html. [Último acceso: 18 02 2023].

[35] Mysql, «El registro de consultas lentas,» [En línea]. Available: https://dev.mysql.com/doc/refman/5.7/en/slow-query-log.html. [Último acceso: 19 02 2023].

[36] d. Mysql, «MySQL Documentation,» [En línea]. Available: https://dev.mysql.com/doc/. [Último acceso: 19 02 2023].

[37] D. Hub, «Bitnami LMS impulsado por Moodle™ LMS,» [En línea]. Available: https://hub.docker.com/r/bitnami/moodle. [Último acceso: 19 02 2023].

[38] Moodle, «Documentation,» 28 11 2022. [En línea]. Available: https://docs.moodle.org/all/es/P%C3%A1gina_Principal. [Último acceso: 19 02 2023].

7 ANEXOS

La lista de los Anexos se muestra a continuación:

ANEXO I. Certificado de originalidad

ANEXO II. Enlaces

ANEXO I: Certificado de Originalidad

CERTIFICADO DE ORIGINALIDAD

Quito, D.M. 2 de marzo de 2022

De mi consideración:

Yo, ITALO ALEXANDER CARREÑO MENDOZA, en calidad de Director del Trabajo de Integración Curricular titulado DESPLIEGUE DE UN CMS DE LICENCIA LIBRE POR MEDIO DE UN CONTENEDOR elaborado por el estudiante CRISTHIAN LEONARDO CARRIÓN PARRA de la carrera en TECNÓLOGÍA SUPERIOR EN REDES Y TELECOMUNICACIONES, certifico que he empleado la herramienta Turnitin para la revisión de originalidad del documento escrito completo, producto del Trabajo de Integración Curricular indicado.

El documento escrito tiene un índice de similitud del 15%

Es todo cuanto puedo certificar en honor a la verdad, pudiendo el interesado hacer uso del presente documento para los trámites de titulación.

NOTA: Se adjunta el link del informe generado por la herramienta Turnitin.

https://epnecuador-my.sharepoint.com/:b:/g/personal/italo_carreno_epn_edu_ec/EafXADcWHT9PnUByTtNrVSEBgfLDJAqEM9hXF_Ut40UWLA?e=cb87d4

Atentamente,

ITALO ALEXANDER CARREÑO MENDOZA

Docente

Escuela de Formación de Tecnólogos

ANEXO II: Enlaces

Anexo II.I Código QR de la implementación y pruebas de funcionamiento