



INSTITUTO TECNOLÓGICO DE PABELLÓN DE ARTEAGA

TECNM CAMPUS PABELLÓN DE ARTEAGA

**“SCRIPTS QUE AUTOMATIZAN LA CREACIÓN
DEL ENTORNO LAMP PARA LA INSTALACIÓN DE REDES DE
SENSORES EN GRANJAS VERTICALES (VERTICAL FARMS)”**

Que como requisito para obtener el grado de
LICENCIADO EN INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN Y
COMUNICACIONES

Presenta:

Ricardo Guadalupe Gomez Martínez

Director:

M.S.C. Eduardo Flores Gallegos

Pabellón de Arteaga, Aguascalientes, Junio del 2022

DEDICATORIA

A mis padres Elvira y Leonardo, por ser un ejemplo de esfuerzo y dedicación, gracias por sus consejos que hicieron de mí una mejor persona y por su apoyo incondicional a lo largo de todo el proceso, soy afortunado por tenerlos como padres.

Mi hermano Diego por la convivencia, el cariño y el apoyo a lo largo del proceso académico y a lo largo de la vida.

Mi abuela Martina quien fue pieza clave en todo esto, gracias por sus palabras de aliento, el cariño y todo el apoyo brindado.

A toda mi familia porque de una u otra manera me acompañan siempre en mis metas y sueños, siempre los llevo en mi mente y mi corazón, a todos ustedes gracias por tanto.

ÍNDICE GENERAL

1. Generalidades del proyecto	15
1.1. Introducción	15
1.2. Problemas a resolver priorizandolos	17
1.3. Justificación	18
1.4. Objetivo general	19
1.5. Objetivos específicos	19
2. Marco teórico	20
2.1. Industria 4.0	20
2.1.1. Internet de las cosas (IoT)	21
2.2. Ambiente de virtualización para aplicaciones	21
2.2.1. Virtualización	21
2.2.2. Importancia de la virtualización	22
2.2.3. Tipos de virtualización	22
2.3. Virtualización de contenedores	23

2.3.1.	Sistemas basados en contenedores virtuales	23
2.3.2.	Contenedor virtual	23
2.3.3.	Imagen de contenedor virtual	23
2.4.	Herramientas en la creación de contenedores	24
2.4.1.	Podman	24
2.4.2.	Docker	24
2.4.3.	Contenedor Vs máquina virtual tradicional	26
2.4.4.	Orquestador	27
2.5.	Creación del entorno con software y servicios en Linux	28
2.5.1.	Shell	28
2.5.2.	Bash	29
2.5.3.	Debian	29
2.5.4.	APT	29
2.6.	Aplicación, uso de Docker y scripts	30
2.7.	Herramientas de usabilidad para el sistema y usuario	31
2.7.1.	Git	32
2.7.2.	Http	32
2.7.3.	Servidor web Apache	32
2.7.4.	MYSQL	33
2.7.5.	Html	33

ÍNDICE GENERAL	5
2.7.6. PHP	34
2.7.7. BIOS (Basic Input Output System)	34
2.7.8. JavaScript	34
2.7.9. CSS	35
2.8. TDD(Desarrollo Dirigido por Tests)	35
2.9. Sistemas cerrados de producción de plantas	35
2.9.1. Granjas verticales	35
2.9.2. Agricultura de precisión	36
2.9.3. Redes de sensores	36
3. Desarrollo de la propuesta	37
3.1. Requerimientos del sistema	37
3.1.1. Requerimientos de los scripts	37
3.1.2. Requerimientos de la página	38
3.2. Diseño general del sistema	39
3.2.1. Diseño de la interfaz para la página web	41
3.3. Implementación TDD	42
3.4. Construcción de los scripts	43
3.4.1. Procedimiento para la instalación de Docker	43
3.4.2. Implementación y despliegue de la imagen de Docker con los aditamentos necesarios	44

3.4.3.	Estructura base para el script principal	45
3.4.4.	Retoques a la estructura base del script principal	48
3.4.5.	Estructura de los scripts secundarios	50
3.5.	Construcción de la página web	52
3.5.1.	Estructura del menú desplegable	52
3.6.	Generador de scrips	53
3.7.	Evaluación	55
3.7.1.	Script principal	55
3.7.2.	Scripts secundarios	55
3.7.3.	Página web	56
4.	Resultados	57
5.	Discusión	59
6.	Conclusiones	60
A.	Anexos	61
A.1.	Anexo 1: Manual de instalación y configuración de Linux Mint	61
A.2.	Descarga de los archivos necesarios para la instalación del sistema operativo de trabajo	61
A.2.1.	Creación del medio de arranque para la instalación	62
A.2.2.	Instalación de Linux Mint	65

A.3. Instalación y configuración de Linux Mint	70
A.3.1. Configuración de Linux Mint para el desarrollo de los scripts .	70
A.4. Anexo 2	74

ÍNDICE DE FIGURAS

2.1. Arquitectura Docker	25
3.1. Casos de uso para los scripts	38
3.2. Casos de Uso Página Web	39
3.3. Script principal mediante la página	40
3.4. Scripts secundarios mediante la página	40
3.5. Diseño de la interfaz de usuario	41
3.6. Diseño de la interfaz de usuario final	41
3.7. Ciclo TDD	42
3.8. Estructura base del script	45
3.9. Asignación de permisos (Fuente: Linux Mint)	46
3.10. Ejecución de script (Fuente: Linux Mint)	47
3.11. Ejecución de script con éxito (Fuente: Linux Mint)	47
3.12. Localhost mostrando el contenido del HTML (Fuente: Linux Mint)	48
3.13. Script principal con los retoques aplicados	49
3.14. Script secundario con los retoques aplicados	50

3.15. Estructura HTML	52
3.16. Estructura JavaScript	52
3.17. Formulario para el generador de scripts en HTML básico	53
3.18. Algoritmo PHP para generar scripts	54
A.1. Interfaz página oficial (Fuente: página oficial Linux Mint)	61
A.2. Servidores espejo (Fuente: página oficial Linux Mint)	62
A.3. Opciones de descarga (Fuente: página oficial Rufus)	63
A.4. Interfaz del programa (Fuente: Rufus)	64
A.5. Interfaz del Boot menú	65
A.6. Pantalla de inicio al arrancar (Fuente: Linux Mint)	66
A.7. Escritorio de Linux Mint (Fuente: Linux Mint)	66
A.8. Selección de idioma (Fuente: Linux Mint)	67
A.9. Distribución de teclado (Fuente: Linux Mint)	67
A.10. Códecs multimedia (Fuente: Linux Mint)	68
A.11. Tipo de instalación (Fuente: Linux Mint)	68
A.12. Asignación de usuario y contraseña (Fuente: Linux Mint)	69
A.13. Proceso de instalación (Fuente: Linux Mint)	69
A.14. Actualización de repositorios (Fuente: Linux Mint)	70
A.15. Actualización de de los paquetes (Fuente: Linux Mint)	71
A.16. Clave GPG (Fuente: Linux Mint)	72

A.17.Integración del nuevo repositorio (Fuente: Linux Mint)	73
A.18.Instalación Sublime Text (Fuente: Linux Mint)	73

ÍNDICE DE TABLAS

2.1. Ventajas y desventajas de Docker	27
3.1. Requerimientos para los scripts	37
3.2. Requerimientos para la página	38
3.3. Evaluación de objetivos del script principal	55
3.4. Evaluación de objetivos del los scripts secundarios	55
3.5. Evaluación de objetivos de la página	56

AGRADECIMIENTOS

A mi tutor M.S.C. Eduardo Flores Gallegos, por compartir conmigo sus conocimientos técnicos, por introducirme en el mundo Linux y en el Laboratorio de Iluminación Artificial. Gracias por su paciencia y sus palabras de aliento, por facilitarme el material para trabajar en muchas ocasiones. Usted formó parte importante en este proyecto con sus aportes profesionales.

A la Dra. Nivia Iracemi Escalante Garcia por sus sabias palabras, el apoyo moral a lo largo de todo el proceso y sus consejos. A usted mi profesora la llevaré siempre conmigo, usted fue pieza clave en mi preparación profesional, le debo gran parte de mis conocimientos. Gracias por su paciencia, dedicación y constancia, por compartir conmigo sus conocimientos, ¡la estimo mucho!

A Tristán Nathaniel Huerta Valdivia por el apoyo brindado con el desarrollo de la plataforma web, fue un honor trabajar contigo y gracias por los conocimientos compartidos.

Y mi profundo agradecimiento a todas las autoridades y al personal que conforman el Laboratorio de Iluminación Artificial, por confiar en mí, abrirme las puertas, permitirme realizar todo el proceso de investigación en sus instalaciones y proporcionarme los materiales necesarios para trabajar. ¡Gracias por todo!



Pabellón de Arteaga, Ags., **21/julio/2022**

ING. RICARDO LARA COLÓN
JEFE DEL DEPARTAMENTO DE CIENCIAS BÁSICAS

PRESENTE

Por medio del presente doy el visto bueno a la Tesis de Licenciatura titulada **“SCRIPTS QUE AUTOMATIZAN LA CREACIÓN DEL ENTORNO LAMP PARA LA INSTALACIÓN DE REDES DE SENSORES EN GRANJAS VERTICALES (VERTICAL FARMS)”** del estudiante **RICARDO GUADALUPE GOMEZ MARTÍNEZ** con número de control 171050183 de la carrera de Ingeniería en Tecnologías de la Información y Comunicaciones. Dicho trabajo ya fue revisado por cada uno de los miembros del comité tutorial y el estudiante ya realizó los cambios sugeridos, por lo que autorizamos su impresión.

Sin otro particular, aprovecho la ocasión para enviarle un cordial saludo, quedo de Usted.

ATENTAMENTE

Excelencia en Educación Tecnológica®
Tierra Siempre fértil®

M.S.C. EDUARDO FLORES GALLEGOS
MIEMBRO DE COMITÉ TUTORIAL

ING. ANA CRISTINA TELLO VAZQUEZ
MIEMBRO DE COMITÉ TUTORIAL

DRA. NIVIA IRACEMI ESCALANTE GARCIA
MIEMBRO DE COMITÉ TUTORIAL



Pabellón de Arteaga, Ags.,

21/julio/2022

RICARDO GUADALUPE GOMEZ MARTÍNEZ
ESTUDIANTE DE LA CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN Y
COMUNICACIONES
NO. DE CONTROL 171050183

PRESENTE

Por medio de este conducto me permito comunicar a Usted que habiendo recibido los votos aprobatorios de los(las) revisores(as) de su trabajo de Tesis titulado: **"SCRIPTS QUE AUTOMATIZAN LA CREACIÓN DEL ENTORNO LAMP PARA LA INSTALACIÓN DE REDES DE SENSORES EN GRANJAS VERTICALES (VERTICAL FARMS)"**, hago de su conocimiento que puede imprimir dicho documento y continuar con los trámites para la presentación de su acto protocolario de titulación integral por Tesis, así mismo fungiendo como **Presidente Eduardo Flores Gallegos, Secretaria Nivia Iracemi Escalante García y Vocal Ana Cristina Tello Vázquez.**

Sin otro particular, aprovecho la ocasión para enviarle un cordial saludo, quedo de Usted.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
Tierra Siempre fértil®

ING. RICARDO LARA COLÓN
JEFE DEL DEPARTAMENTO DE CIENCIAS BÁSICAS

RESUMEN

El objetivo del presente proyecto es diseñar y construir scripts que automaticen la creación de un entorno de desarrollo web LAMP, para la implementación de redes de sensores dentro de las granjas verticales del LIA en el TECNM campus Pabellón de Arteaga, que sirven para concentrar y organizar la adquisición de información además, que permita la toma de decisiones dentro de LIA.

El desarrollo de este proyecto beneficiará a los sectores o áreas de cultivo del Laboratorio de Iluminación Artificial dentro y fuera del Instituto Tecnológico de Pabellón de Arteaga con la generación de scripts que automatizan el entorno LAMP para la instalación de redes de sensores en las granjas verticales permitirá llevar una estricta organización en el control y monitoreo de la información permitiendo así tomar decisiones sobre el desarrollo y crecimiento de los diferentes cultivos. Además, de que la información puede servir de referencia para futuras experimentaciones sobre la predicción de la producción de cultivos.

La automatización del entorno LAMP se realizo haciendo uso de Bash y Docker, el primero utilizado para la construcción de los scripts, permite escribir ordenes mediante linea de comandos, las cuales el sistema operativo Linux ejecutara de manera secuencial permitiendo automatizar el proceso, el segundo facilita el desligue del entorno LAMP haciendo uso de un contenedor con los paquetes necesarios, posibilita el despliegue y la configuración del entorno mediante linea de comandos.

Como resultado se obtienen 5 scripts y una página web la cual proporciona la interfaz para la descarga de los scripts y posibilita crear scripts prellenados para la base de

datos, de los 5 scripts, el primero de ellos despliega y configura el entorno LAMP, los otros cuatro están encargados de configurar la base de datos.

CAPÍTULO 1

GENERALIDADES DEL PROYECTO

1.1 INTRODUCCIÓN

Actualmente la agricultura, además de la producción a campo abierto, se practica en una amplia variedad de ambientes modificados, entre los que destacan los invernaderos con o sin control ambiental con cultivos en sistemas hidropónicos, sustratos inertes o en suelo, mismos que representan un ejemplo de ecosistemas artificiales para desarrollar la agricultura intensiva. En México, existen muchas regiones con condiciones naturales idóneas para el establecimiento de granjas verticales debido a ello la agricultura protegida se ha desarrollado en forma acelerada, dado que permite obtener productos de calidad tanto para mercado nacional como de exportación. De esta forma, el empleo de granjas verticales y la agricultura protegida están contribuyendo ampliamente en la producción de alimentos y en el desarrollo de varias zonas agrícolas de México. En años recientes, los cultivos hortícolas han presentado tendencia hacia la obtención de producción anticipada o fuera de estación, en condiciones diferentes a aquellas en las que tradicionalmente se cultivaban a campo abierto. Esta tendencia ha creado la necesidad de usar diversos elementos, herramientas, materiales y estructuras en la protección de cultivos con la finalidad de obtener productos de mejor calidad. A esta actividad se le conoce como agricultura protegida (también llamada horticultura protegida) y en gran medida ha sido propiciada por el desarro-

llo de materiales plásticos agrícolas.

Las granjas verticales (vertical farms) son una alternativa al problema de la falta de espacio para cultivar alimentos para consumo humano, que cumpla con las necesidades de las superpoblaciones actuales [1]. Los espacios dedicados al cultivo controlado requieren el uso de redes de sensores, estos con el fin de monitorear los cultivos, abriendo la posibilidad de llevar un registro y un control de los datos obtenidos, como lo son la temperatura, humedad, CO², entre otros. Para la implementación de redes de sensores, es necesario contar con un servidor que almacene los datos, la instalación y configuración del servidor puede tornarse complicada.

La inclusión en contenedores es un enfoque de desarrollo de software en el que una aplicación o un servicio, sus dependencias y su configuración (extraídos como archivos de manifiesto de implementación) se empaquetan como una imagen de contenedor. La aplicación en contenedor puede probarse como una unidad e implementarse como una instancia de imagen de contenedor en el sistema operativo (SO) host.

Una alternativa para llevar adecuadamente la administración de servicios o procesos es la utilización de los contenedores de software que actúan como una unidad estándar de implementación de software que puede contener diferentes dependencias y código. De esta manera, la inclusión del software en contenedor permite implementarse en entornos con pocas modificaciones o ninguna en absoluto.

Los contenedores también aíslan las aplicaciones entre sí en un sistema operativo compartido. Las aplicaciones en contenedor se ejecutan sobre un host de contenedor que a su vez se ejecuta en el sistema operativo (Linux o Windows). Por lo tanto, los contenedores tienen una superficie significativamente menor que las imágenes de máquina virtual (VM). Cada contenedor puede ejecutar una aplicación web o un servicio al completo.

En este documento se plantea el uso de script y el uso de contenedores en conjunto, para facilitar el despliegue y configuración de un servidor LAMP (Linux, Apache, MYSQL, PHP), lo cual facilitar en general la implementación de una red de sensores

[2]. Lo anterior con el objetivo del presente proyecto, diseñar y construir scripts que automaticen la creación de un entorno de desarrollo web LAMP, para la implementación de redes de sensores dentro de las granjas verticales del LIA en el TECNM.

1.2 PROBLEMAS A RESOLVER PRIORIZANDOLOS

Las granjas verticales (vertical farms) son una alternativa al problema de la falta de espacio para cultivar alimentos para consumo humano, que cumpla con las necesidades de las superpoblaciones actuales. Debido a lo anterior, la agricultura de precisión tiene como principio el manejo de la variabilidad espacial y temporal de los suelos que ha evolucionado e integrando tecnologías más amplias en las diversas etapas de los cultivos, donde es fundamental el control de la información de todo tipo para la administración de los procesos de producción [1].

En el Laboratorio de Iluminación Artificial (LIA) se realizan cultivos de hortalizas para consumo humano donde variables como temperatura, humedad, CO² y cantidad y calidad de luz deben ser monitoreadas constantemente para evaluar el desarrollo y rendimiento del cultivo. Uno de los problemas que se presenta en LIA es que no se lleva un registro ni control de la información obtenida dentro de cada área de producción, por lo cual surge la necesidad de implementar redes de sensores. De forma general, la implementación de redes de sensores ha funcionado adecuadamente en un espacio, pero al extenderse su implementación el control de cada red en los diferentes espacios se complica, lo que lleva a que no se tomen adecuadas decisiones sobre los cultivos generados. Adicionalmente, impide identificar de que espacio controlado es procedente la información. Lo que conlleva nuevamente en pensar en otro tipo de tecnología que nos brinde el control y administración de los datos. Además, de la identificación del espacio de donde proviene la información. También, la información recabada por los sensores, la implementación de este servidor, puede tornarse complicado para alguien que no tienen muchos conocimientos sobre el tema.

1.3 JUSTIFICACIÓN

Las granjas verticales fueron diseñadas para facilitar la producción agrícola dentro de estructuras verticales como edificios. Dentro de las versiones a gran escala y pueden integrar granjas verticales son edificios existentes, bloques de oficinas, dentro de almacenes reutilizados, contenedores, invernaderos, entre otros. Lo anterior con el objetivo de seleccionar espacios reducidos en zonas determinadas y contribuir al desarrollo sostenible del medio ambiente mediante las técnicas de iluminación artificial, captación de agua de lluvias para sistemas de riego y demás factores ambientales tomando ventaja en la reducción de consumo y costo de transporte y/o logística. Siendo un factor importante para la innovación y el crecimiento de la región debido a que la FAO, la Organización de las Naciones Unidas para la Agricultura y la Alimentación, está alertando: en 2050 la población mundial alcanzará los 9.6 mil millones y por tanto harían falta dos planetas para satisfacer sus necesidades alimentarias. Según la organización, para alimentar a todas estas personas, la agricultura mundial tendrá que producir un 70 % más de alimentos que en la actualidad, utilizando solo un 5 % más de recursos naturales [3].

El desarrollo de este proyecto beneficiará a los sectores o áreas de cultivo del Laboratorio de Iluminación Artificial dentro y fuera del Instituto Tecnológico de Pabellón de Arteaga con la generación de scripts que automatizan el entorno LAMP para la instalación de redes de sensores en las granjas verticales permitirá llevar una estricta organización en el control y monitoreo de la información permitiendo así tomar decisiones sobre el desarrollo y crecimiento de los diferentes cultivos. Además, de que la información puede servir de referencia para futuras experimentaciones sobre la predicción de la producción de cultivos.

1.4 OBJETIVO GENERAL

Diseñar y construir scripts que automaticen la creación de un entorno de desarrollo web LAMP, para la implementación de redes de sensores dentro de las granjas verticales del LIA en el TECNM campus Pabellón de Arteaga.

1.5 OBJETIVOS ESPECÍFICOS

- Analizar el proceso existente para el almacenamiento de la información.
- Recopilar los requerimientos del sistema con base a las necesidades dentro de LIA.
- Diseñar la estructura base del sistema con base en los requerimientos.
- Automatizar la creación de contenedor Docker con el entorno LAMP a través Bash.
- Construir página web que aloje los scripts y proporcione la interfaz para la descarga de los mismos.
- Evaluar el funcionamiento de la página web y los scripts.
- Generar repositorio en Github con la documentación necesaria para que el usuario pueda hacer uso de los scripts y la página.

CAPÍTULO 2

MARCO TEÓRICO

En este apartado se describen los antecedentes y la motivación que dieron origen al desarrollo de este proyecto. Además, definimos los conceptos teóricos y tecnologías involucradas con la creación del entorno LAMP para la instalación de Redes de Sensores en Granjas Verticales (Vertical Farms).

2.1 INDUSTRIA 4.0

La revolución de Internet ha impactado radicalmente en la economía y en la sociedad del siglo XXI. El término de industria 4.0 tiene la misión de promover la manufactura computarizada impulsada por el incremento de las capacidades computacionales y de conectividad. El surgimiento de nuevas formas de interacción hombre-máquina como pantallas táctiles y sistemas de realidad aumentada, además de las habilidades notables para la transferencia de instrucciones digitales al mundo físico como la robótica y la impresión 3D [4].

2.1.1 INTERNET DE LAS COSAS (IoT)

IoT se compone de un sistema de dispositivos de computación interrelacionados como autos, casas, animales, máquinas, robots, personas, por mencionar algunos. Contienen identificadores únicos y permiten recopilar, analizar e intercambiar información sin intervención humana [4, 5].

Adicionalmente, el internet de las cosas ofrece oportunidades para el mercado como los fabricantes de dispositivos electrónicos, objetos inteligentes, sensores, actuadores y Gateways; para proveedores de servicios de internet y computación en la nube. Es una tecnología transversal para aplicación como ciudades inteligentes, hogares, edificios, medicina y salud, industria, transporte, logística, agricultura de precisión [6, 7].

2.2 AMBIENTE DE VIRTUALIZACIÓN PARA APLICACIONES

Uno de los modelos de servicios es la virtualización dado que tiene gran impacto en el mundo académico como en la industria. Además, su interés por el ahorro en costos y operacionales debido a que la gestión de recursos de manera centralizada. Además, facilita la identificación de los servicios que se ponen en marcha teniendo una sola sintaxis de manejo en el sistema, siendo más sencillo de manejar que cuando el hardware se encuentra de manera individual [8].

2.2.1 VIRTUALIZACIÓN

Por lo tanto, la virtualización es un método que permite integrar y compartir recursos. Se crea una capa de abstracción entre el hardware de la máquina física (host)

y el sistema operativo de la máquina virtual (virtual machine, guest). Donde se establece un medio para generar una versión virtual de un módulo o recurso como un servidor, un sistema operativo una red o incluso un dispositivo de almacenamiento en el cual al recurso se lo divide en uno o más entornos de ejecución. Es decir, es la combinación de hardware y software que posibilita a un recurso físico funcionar como múltiples recursos lógicos [9].

2.2.2 IMPORTANCIA DE LA VIRTUALIZACIÓN

La importancia de la virtualización se ve reflejada en la reducción de los costos de hardware y una mejora del uso de los servidores. Dado que, con el creciente costo de energía, cada vez más organismos e instituciones sienten la necesidad de pasar a un centro de datos ecológico (cantidad reducida espacio, energía para almacenar menores servidores físicos) a través del uso de tecnologías de virtualización dentro del centro de datos a nivel de servidores, estaciones de trabajo y aplicaciones [10].

2.2.3 TIPOS DE VIRTUALIZACIÓN

Desde el punto de vista del OS invitado (en inglés guest) tiene completo control del hardware sobre el que se está ejecutando. Virtualizar es básicamente abstraer al guest de los recursos físicos. Según como se realiza esta abstracción encontramos: Virtualización completa (full virtualization), Paravirtualización (paravirtualization) y Contenedores (OS virtualization).

Aunque existen diferentes tipos para ejecutar la virtualización, para el desarrollo de este proyecto se utilizara la virtualización a través de contenedores. Los contenedores tienen algunas similitudes con las máquinas virtuales, pero son diferentes en el enfoque arquitectónico, lo que hace a los contenedores más portables y eficientes [11].

2.3 VIRTUALIZACIÓN DE CONTENEDORES

La virtualización basada en contenedores es una aproximación a la virtualización donde se ejecuta la capa de virtualización como una aplicación aislada dentro del sistema operativo del equipo anfitrión. Concede la creación de nuevos entornos de ejecución (comparable con máquinas virtuales) y que reciben nombre de contenedores [12].

2.3.1 SISTEMAS BASADOS EN CONTENEDORES VIRTUALES

Se refieren a las soluciones de software (aplicaciones) que tienen como base de ejecución los contenedores virtuales. De esta manera aprovechan los beneficios de los contenedores permitiendo que se puedan ejecutar varias aplicaciones de manera paralela y aislada unas con otras [13].

2.3.2 CONTENEDOR VIRTUAL

Los contenedores virtuales (container) son un conjunto de elementos entre los que se encuentran un sistema de archivos, configuraciones (recursos, entorno, seguridad), un conjunto de perfiles, propiedades (arquitectura de contenedor, efímera o persistente y el nombre) y estado de ejecución [13].

2.3.3 IMAGEN DE CONTENEDOR VIRTUAL

Para una correcta utilización de los contenedores se requiere de una imagen que es definida como un archivo estático e inmutable que incluye código ejecutable para que pueda ejecutar un proceso aislado en la infraestructura de tecnología de la informa-

ción (TI). La imagen se integra bibliotecas, herramientas y otras configuraciones de plataformas que un programa de software necesita para ejecutarse en una plataforma de contenedorización como Docker o CoreOS Rkt, Podman, entre otros. La imagen comparte el kernel del sistema operativo de su anfitrión máquina [14].

2.4 HERRAMIENTAS EN LA CREACIÓN DE CONTENEDORES

2.4.1 PODMAN

Podman (Pod Manager) es una herramienta nativa de Linux y la última palabra de moda en tecnología de contenedorización. Fue desarrollada por RedHat como otro motor de contenedores, siendo una solución de código abierto y sin demonios; diseñado para desarrollar, administrar y ejecutar aplicaciones utilizando contenedores e imágenes en un entorno Linux [15].

2.4.2 DOCKER

Docker es un motor de contenedores de código libre que utiliza una virtualización ligera permitiendo automatizar el despliegue de aplicaciones mediante contenedores virtuales. Esto significa que puede encapsular las aplicaciones dentro de un contenedor junto con sus dependencias y ejecutarlos sobre cualquier equipo que tenga instalada la plataforma Docker, usualmente servidores Linux, nubes públicas y privadas [16].

2.4.2.1 ARQUITECTURA DOCKER

De acuerdo a la definición anterior, la función Docker hace posible la virtualización de un servidor en diversos fragmentos de forma independiente, permitiendo compartir recursos en una máquina anfitriona. Adicionalmente, se pueden encapsular entornos y aplicaciones que posteriormente puedan ser accedidas o utilizadas por otros espacios de trabajo, facilitando el uso y configuración de diferentes herramientas.

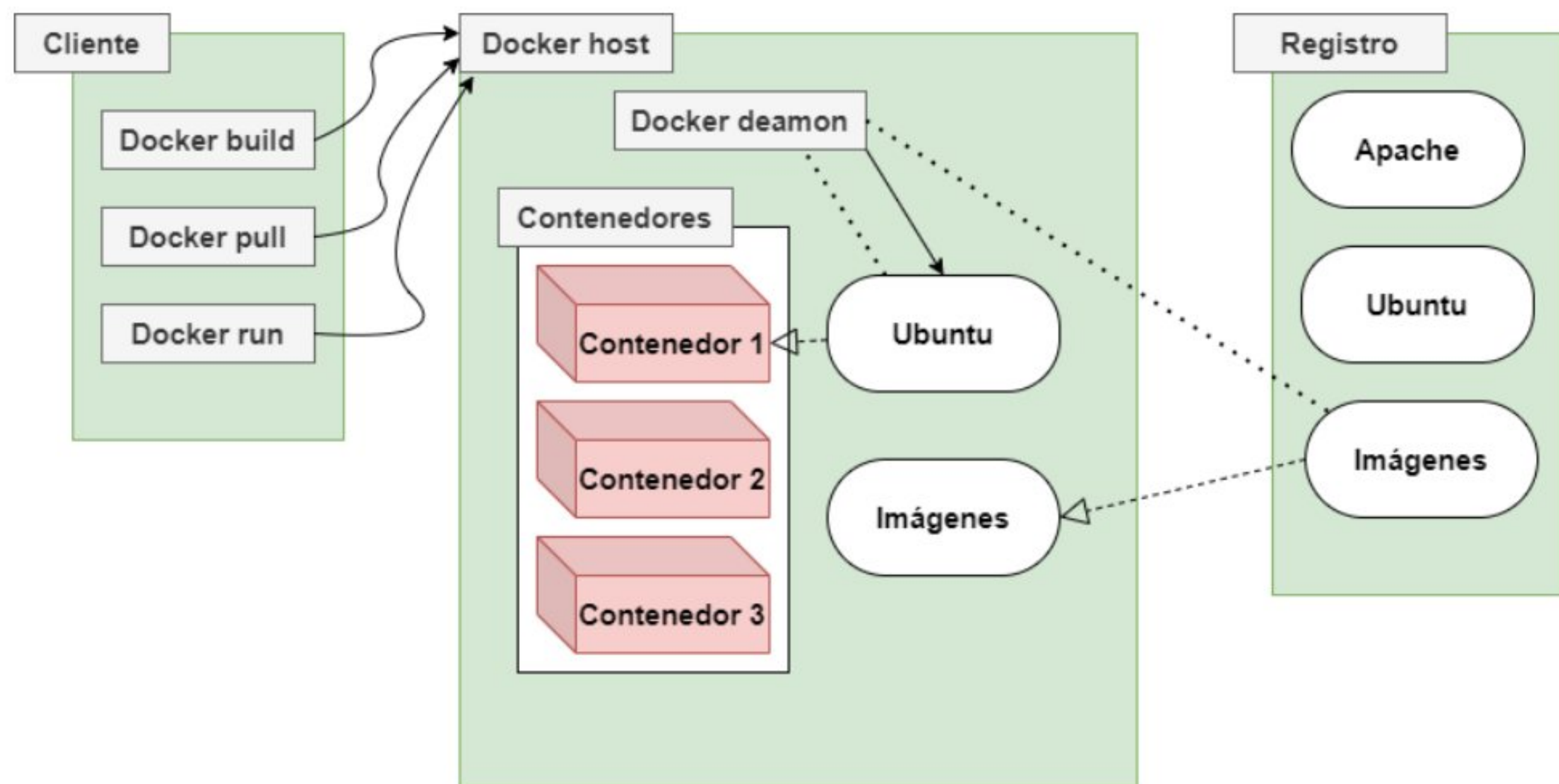


Figura 2.1: Arquitectura Docker

La Figura 2.1 representa el diseño de la arquitectura utilizada por Docker, consta de 3 capas independientes, a continuación, se describe cada una de ellas [17]:

- **Cliente:** Ejecuta instrucciones directamente sobre el anfitrión de Docker para descargar y desplegar imágenes y contenedores.
- **Docker Host:** Esta capa contiene un Docker Daemon, se encarga de controlar las imágenes y los contenedores que un cliente puede generar.
- **Docker registry:** Esta capa está encargada de organizar y publicar las imágenes que un usuario puede llegar a utilizar.

2.4.3 CONTENEDOR VS MÁQUINA VIRTUAL TRADICIONAL

Los contenedores Docker pueden ser vistos como herramientas más flexibles para el empaquetamiento, entrega y despliegue de software y de sus aplicaciones. Adicionalmente, mientras las máquinas virtuales requieren virtualización completa creando una computadora virtual, los contenedores utilizan y comparten las características estándar del kernel. Por lo que los contenedores proveen un mejor desempeño al costo de cierta rigidez en cuanto a compatibilidad [16].

2.4.3.1 VENTAJAS Y DESVENTAJAS DE DOCKER SOBRE LA VIRTUALIZACIÓN TRADICIONAL

[18] En la Tabla 2.1 se muestran algunas de las ventajas y desventajas de utilizar Docker.

Tabla 2.1: Ventajas y desventajas de Docker

Ventajas	Desventajas
Los contenedores se replican fácilmente y se inician en segundos.	Se necesita acceso a internet para la descarga de las imágenes.
Consume menos recursos de hardware, y los que consume van directamente a la aplicación.	En Windows solo soporta 64 bits con sistema Operativo Windows 8 en adelante.
Las imágenes e instancias suelen ocupar menos espacio que las máquinas virtuales.	En Linux solo soporta arquitecturas de 64 bits, Ubuntu 16.04 en adelante.
Existen miles de imágenes que pueden ser descargadas y modificadas libremente.	En algunas versiones de Docker podría dar error debido al constante desarrollo.
La instalación de un servicio solo consiste en ejecutar una línea de comando.	
Tiene habilitado un control de versiones, parecido a “Git”.	

Como podemos observar existe un mayor número de ventajas que de desventajas, es importante validar estas cuestiones antes de implementar Docker en determinado proyecto, en este caso se utilizó debido a que agiliza el proceso de automatización mediante los scripts.

2.4.4 ORQUESTADOR

Para tener una gestión y monitorización de los contenedores en esta nueva infraestructura, se desarrolla un orquestador. Este programa implementa las siguientes

funcionalidades [16]:

- Listado de contenedores activos por nodo.
- Arranque y parada de contenedores bajo demanda.
- Migración bajo demanda y automática en caso de posible caída del nodo o saturación del mismo.
- Monitorización, gráficos e información detallada de cada contenedor.
- Monitorización y gráficos de los nodos.

2.5 CREACIÓN DEL ENTORNO CON SOFTWARE Y SERVICIOS EN LINUX

El uso de herramientas como Linux permite generar y unir los comandos del sistema con estructuras de control de flujo y disponer de un entorno de prototipo rápido de herramientas para la automatización de tareas. A continuación, se describen cada una de los comandos que permitieron la fundamentación de esta investigación [16].

2.5.1 SHELL

Los shell son necesarios para ejecutar la gran mayoría de programas instalados en la computadora. En informática, el shell o intérprete de línea de comandos es el programa informático que proporciona al usuario una interfaz para acceder a los servicios del sistema operativo [25].

2.5.2 BASH

Bash es un intérprete de comandos que ejecuta, una por una y en orden secuencial, las instrucciones introducidas por el usuario o contenidas en un script y devuelve los resultados. Actúa como interfaz entre el kernel Linux y los usuarios o programas en modo texto, incorpora numerosas utilidades de programación y mejoras sobre sh, su shell predecesora, debido a que es una herramienta desarrollada por GNU, suele ser utilizada por defecto en las distribuciones GNU/Linux actuales [25].

2.5.3 DEBIAN

Es un sistema operativo libre que incorpora las librerías de GNU y hace uso del Kernel de Linux, este último es una pieza de software, que permite la comunicación entre el software y el hardware del PC y fue creada en un principio por Linus Torvalds y desarrollado por miles de programadores a lo largo del mundo. Debian está disponible para un gran número de arquitecturas de computadoras por lo cual puede funcionar en casi todos los ordenadores personales, incluyendo la mayoría de los modelos más antiguos [26].

2.5.4 APT

APT son las siglas de «herramienta avanzada de paquetes» («Advanced Package Tool»). Lo que hace este programa «avanzado» es su enfoque sobre paquetes. No sólo los evalúa individualmente sino que los considera como un todo y produce la mejor combinación posible de paquetes dependiendo de lo que esté disponible y sea compatible (según dependencias) [27].

2.6 APLICACIÓN, USO DE DOCKER Y SCRIPTS

Desde su creación, las tecnologías de virtualización han impulsado la escalabilidad y flexibilidad que demandan los servicios ofrecidos por la gestión de archivos en la nube, permitiendo crear múltiples servidores virtuales (instancias) sobre un mismo equipo de hardware. También, la tecnología de virtualización ha permitido el desarrollo de sistemas de software debido a que las necesidades de interacción con los usuarios se encuentran en constante evolución. Adicionalmente, los contenedores virtuales están siendo parte de una solución cada vez más adoptada por diferentes instituciones y organizaciones para resolver problemas que se presentan durante la integración de múltiples piezas de software requeridas para sistemas y procesos [19].

Dentro de los desarrollos propuestos, López [17] diseñó una arquitectura soportada en Docker para el despliegue automático de contenedores en procesos de desarrollo de software orientados a la web. Todo lo anterior, con el objetivo de proporcionar herramientas que permitan al estudiante capacitarse en la construcción de sistemas.

Los autores demostraron la importancia de aplicar Docker para la gestión de contenedores. Además, organizaron la aplicación de múltiples virtualizaciones en una máquina o crear contenedores con acceso entre ellos. Proponen el uso de Kubernetes, enfatizando en el nivel de conocimiento que debe poseer un usuario para determinar configuraciones precisas a la hora de establecer ambiente de pruebas o de producción [5].

En Pacheco [18] Se presentó una comparativa de una arquitectura de microservicios de contenedores Dockers con una arquitectura tradicional (monolítica) a través de la comprobación aplicativa para verificar el rendimiento y los recursos utilizados.

Castro [20] creó una estrategia de obtener una base de datos distribuida heterogénea con replicación en un ambiente CentOS con la codificación de un script bash con software libre. Obtuvo un ODBC que configuró dos gestores sin gastar dinero con

optimización de tiempo. Haciendo que el proceso sea realmente sencillo.

Los scripts bash pueden llegar a tener un sin fin de utilidades, en Luquin [21], se han presentado para la automatización en el proceso de instalación, control, creación y restauración de copias de seguridad en una red local a través de scripts y SSH.

Adicionalmente, Rodríguez [22] automatizó distintos aspectos de la instalación de servicios en GNU/Linux en sistemas y bajo costo con la ayuda de scripts que automatizan tareas como la instalación del sistema operativo y el despliegue de servicios. Los scripts generan una imagen de disco preconfigurada de Arch Linux ARM.

La otra tarea importante es el despliegue de servicios, por lo que se han implementado scripts capaces de desplegar algunos servicios y su configuración.

Ortega [23], nos propone Una herramienta para disponer de un entorno de ejecución con el uso de Docker contiene todos los programas de software para realizar prácticas de laboratorio facilitando la generación de entornos de aprendizaje. Conlleva a evitar problemas de incompatibilidad durante el proceso de instalación.

Mendez [24] avaló el uso de Contenedores Docker como una estrategia de virtualización presentando fundamentos, ventajas y desventajas respecto a las limitaciones de seguridad con relación a las máquinas virtuales.

2.7 HERRAMIENTAS DE USABILIDAD PARA EL SISTEMA Y USUARIO

La fundamentación de los siguientes conceptos conceden una adecuada administración del software y servicios.

2.7.1 GIT

Es un sistema donde se registran el control de versiones sobre un conjunto de archivos de diferentes tamaños, de modo que en cualquier momento se puedan recuperar versiones anteriores o alguna en específico, haciendo el trabajo más eficiente. Cada repositorio guarda un historial de los cambios realizados, esta acción permite revertir el código un estado anterior y permite visualizar quien realizó la última modificación en el proyecto [18].

2.7.2 HTTP

El Protocolo de transferencia de hipertexto (HTTP) es una capa de aplicación Protocolo para transmitir documentos hipermedia, como HTML, fue diseñado para la comunicación entre navegadores web y servidores web, pero también se puede utilizar para otros fines. HTTP es un protocolo sin estado, lo que significa que el servidor no guarda ningún dato (estado) entre dos solicitudes [28].

2.7.3 SERVIDOR WEB APACHE

El Proyecto del servidor HTTP Apache es un esfuerzo para desarrollar y mantener un servidor HTTP de código abierto para sistemas operativos actuales, incluidos UNIX y Windows. El objetivo de este proyecto es brindar un servidor seguro, eficiente y extensible que proporcione servicios HTTP en sincronía con los estándares HTTP actuales. El servidor HTTP Apache ("httpd") se lanzó en 1995 y ha sido el servidor web más popular en Internet desde abril de 1996. Ha celebrado su 25 aniversario como proyecto en febrero de 2020 [29].

2.7.4 MYSQL

Es un sistema gestor de bases de datos (SGBD, DBMS por sus siglas en inglés) muy conocido y ampliamente usado por su simplicidad y notable rendimiento. Aunque carece de algunas características avanzadas disponibles en otros SGBD del mercado, es una opción atractiva tanto para aplicaciones comerciales, como de entretenimiento precisamente por su facilidad de uso y tiempo reducido de puesta en marcha. Esto y su libre distribución en Internet bajo licencia GPL le otorgan como beneficios adicionales (no menos importantes) contar con un alto grado de estabilidad y un rápido desarrollo.

MYSQL está disponible para múltiples plataformas, la seleccionada para los ejemplos de este libro es GNU/Linux. Sin embargo, las diferencias con cualquier otra plataforma son prácticamente nulas, ya que la herramienta utilizada en este caso es el cliente *mysql-client*, que permite interactuar con un servidor MYSQL (local o remoto) en modo texto. De este modo es posible realizar todos los ejercicios sobre un servidor instalado localmente o, a través de Internet, sobre un servidor remoto [30].

2.7.5 HTML

HTML (HyperText Markup Language) es un lenguaje muy sencillo que permite describir hipertexto, es decir, texto presentado de forma estructurada y agradable, con vínculos o enlaces (hyperlinks) que conducen a otros documentos o fuentes de información relacionadas y con inserciones multimedia (gráficos, sonido, entre otros.) [31].

2.7.6 PHP

El lenguaje PHP (cuyo nombre es acrónimo de PHP: Hypertext Preprocessor) es un lenguaje interpretado con una sintaxis similar a la de C++ o JAVA. Aunque el lenguaje se puede usar para realizar cualquier tipo de programa, es en la generación dinámica de páginas web donde ha alcanzado su máxima popularidad. En concreto, suele incluirse incrustado en páginas HTML (o XHTML), siendo el servidor web el encargado de ejecutarlo [32].

2.7.7 BIOS (BASIC INPUT OUTPUT SYSTEM)

Es un chip programado que está insertado en la placa base. La BIOS da instrucciones a la memoria, monitor, teclado, impresora y a otros periféricos, permite que el ordenador los reconozca en el arranque y puedan funcionar. Podemos decir que el BIOS tiene instalado un sistema operativo básico que se encarga de algunos componentes electrónicos del ordenador. De todos modos, la BIOS no se encarga de la gestión del ordenador, eso lo hace el sistema operativo [33].

2.7.8 JAVASCRIPT

Es un lenguaje de scripts dinámico que admite construcción de objetos basada en prototipos. Intencionalmente, la sintaxis básica es similar a Java y C++ para reducir la cantidad de conceptos nuevos necesarios para aprender el lenguaje [34].

2.7.9 CSS

CSS (en inglés Cascading Style Sheets) es lo que se denomina lenguaje de hojas de estilo en cascada y se usa para estilizar elementos escritos en un lenguaje de marcado como HTML. CSS separa el contenido de la representación visual del sitio [35].

2.8 TDD(DESARROLLO DIRIGIDO POR TESTS)

Técnica de diseño e implementación de software para diseñar software que se centra en tres pilares fundamentales [36]:

- La implementación de las funciones justas que el cliente necesita y no más.
- La minimización del número de defectos que llegan al software en fase de producción.
- La producción de software modular, altamente reutilizable y preparado para el cambio.

2.9 SISTEMAS CERRADOS DE PRODUCCIÓN DE PLANTAS

2.9.1 GRANJAS VERTICALES

Son aquellas granjas que se desarrollan en edificios para llevar a cabo la producción, ya sea como agricultura, o ganadería en el interior de estos [1].

2.9.2 AGRICULTURA DE PRECISIÓN

La denominada “*Agricultura de Precisión*” contribuye específicamente a optimizar el uso de los recursos e insumos al utilizarlos de forma adecuada en el lugar y momento preciso, evitando una dosis excesiva o en un lugar innecesario.

Al día de hoy un campo productivo ya no se concibe sin la tecnología digital, no es cuestión de lujo, sino de una necesidad en el uso de tecnologías avanzadas para lograr maximizar el nivel de producción, enfrentando el desafío por falta de agua, suelos erosionados, nuevas plagas, malezas mas agresivas y enfermedades mas resistentes [37].

2.9.3 REDES DE SENSORES

Una red de sensores es un sistema que consiste de dispositivos autónomos usando sensores distribuidos espacialmente, para monitorear cooperativamente condiciones físicas o ambientales, tales como temperatura, humedad relativa, radiación solar, entre otros [2].

CAPÍTULO 3

DESARROLLO DE LA PROPUESTA

3.1 REQUERIMIENTOS DEL SISTEMA

3.1.1 REQUERIMIENTOS DE LOS SCRIPTS

La Tabla 3.1 muestra las funcionalidades requeridas para los scripts.

Tabla 3.1: Requerimientos para los scripts

ID	Descripción
R1	Instalar paquetes necesarios en caso de ser requerido.
R2	Configurar entorno de desarrollo LAMP.
R3	Pedir información para crear la base de datos.
R4	Crear base de datos con la información proporcionada por el usuario.

La Figura 3.1 representa los casos de uso de los scripts.

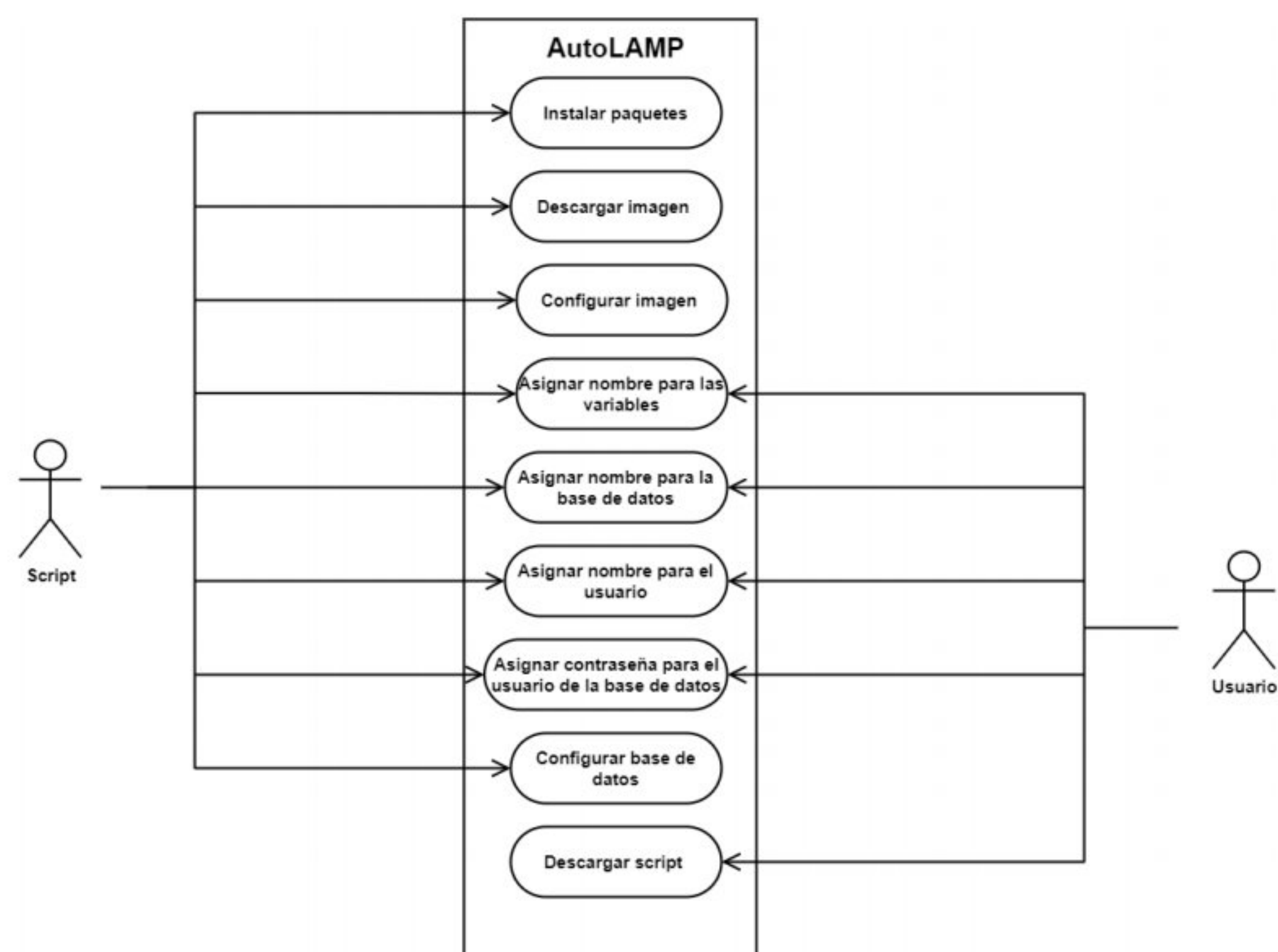


Figura 3.1: Casos de uso para los scripts

3.1.2 REQUERIMIENTOS DE LA PÁGINA

Para la página web las funcionalidades requeridas son desglosadas en la Tabla 3.2.

Tabla 3.2: Requerimientos para la página	
ID	Descripción
RP1	Menú desplegable que proporcione la interfaz para la descarga de los scripts.

La Figura 3.2 muestra los casos de uso de la página web.

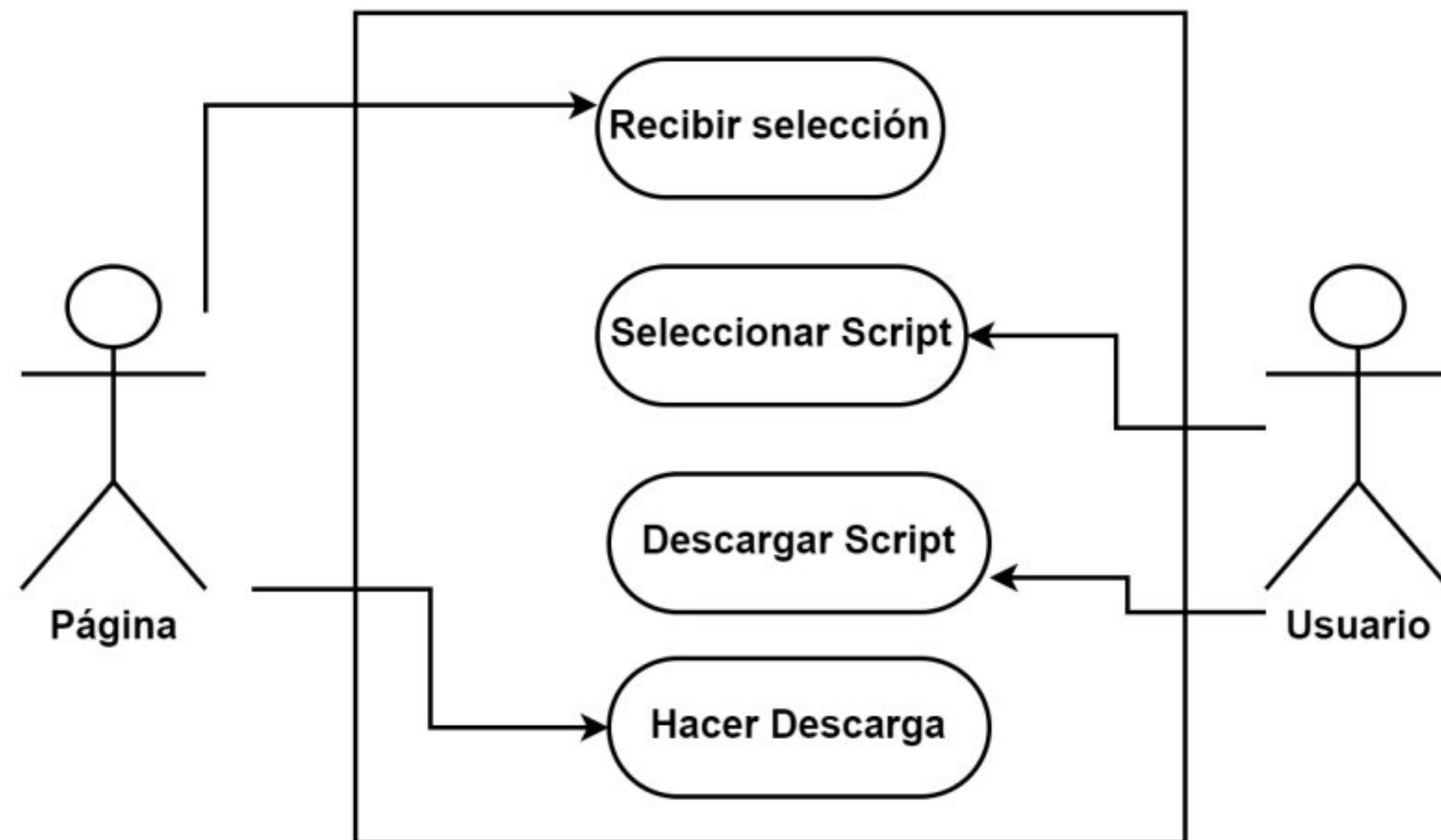


Figura 3.2: Casos de Uso Página Web

3.2 DISEÑO GENERAL DEL SISTEMA

La Figura 3.3 se presenta el diseño inicial del sistema mediante un diagrama de flujo, se presentan los pasos a realizar para hacer uso del sistema y se emplea una validación dentro del script, para terminar si es necesario instalar los paquetes requeridos.

A grandes rasgos así es como funcionará el sistema para desplegar el servidor, una vez este se encuentre en ejecución, los scripts secundarios serian implementados como se muestra en la Figura 3.4.

La página proporciona la interfaz para la descarga de los scripts, por el momento es su única función, se trata de mantener el sistema simple y funcional, sin tantas opciones que puedan confundir al usuario.

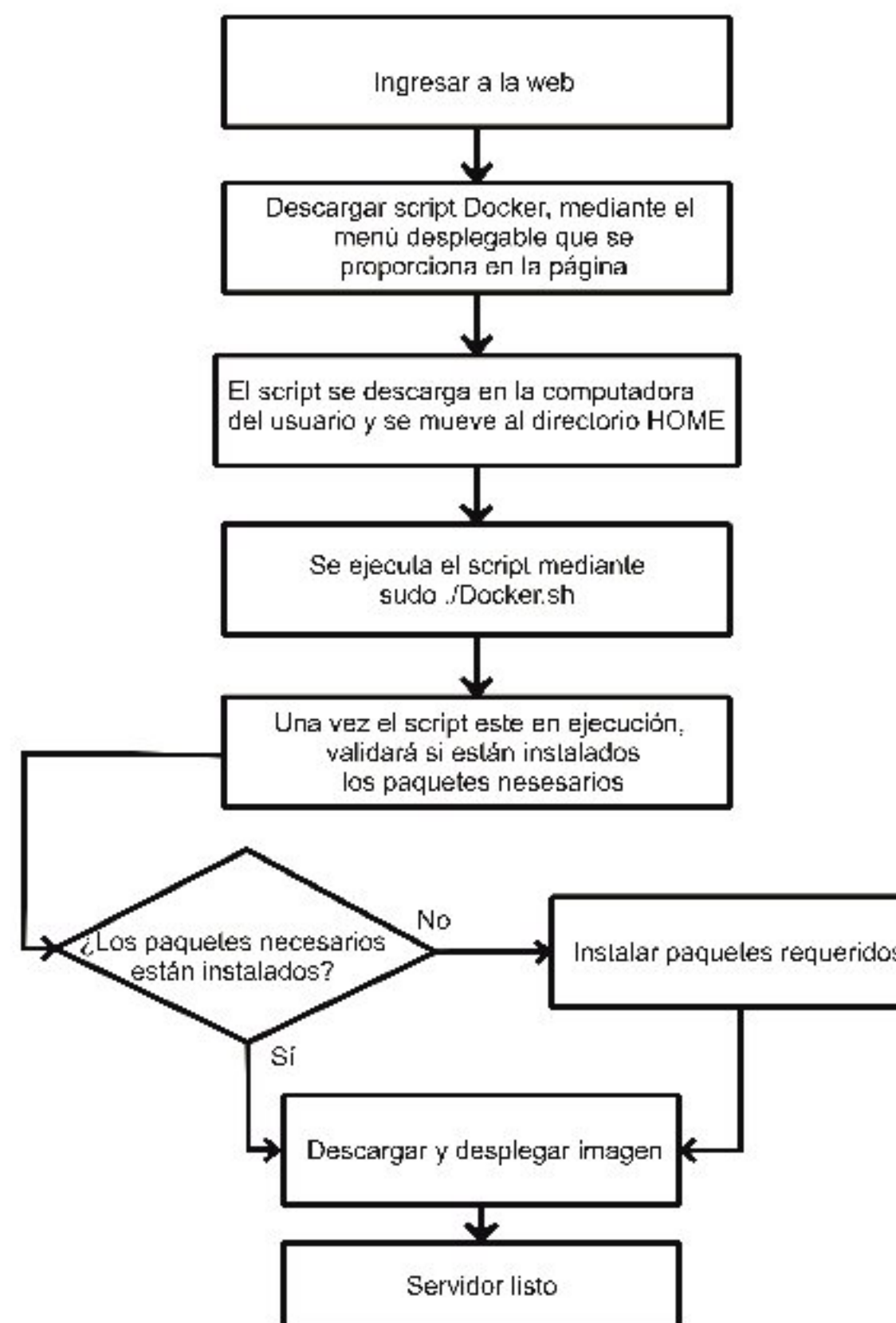


Figura 3.3: Script principal mediante la página

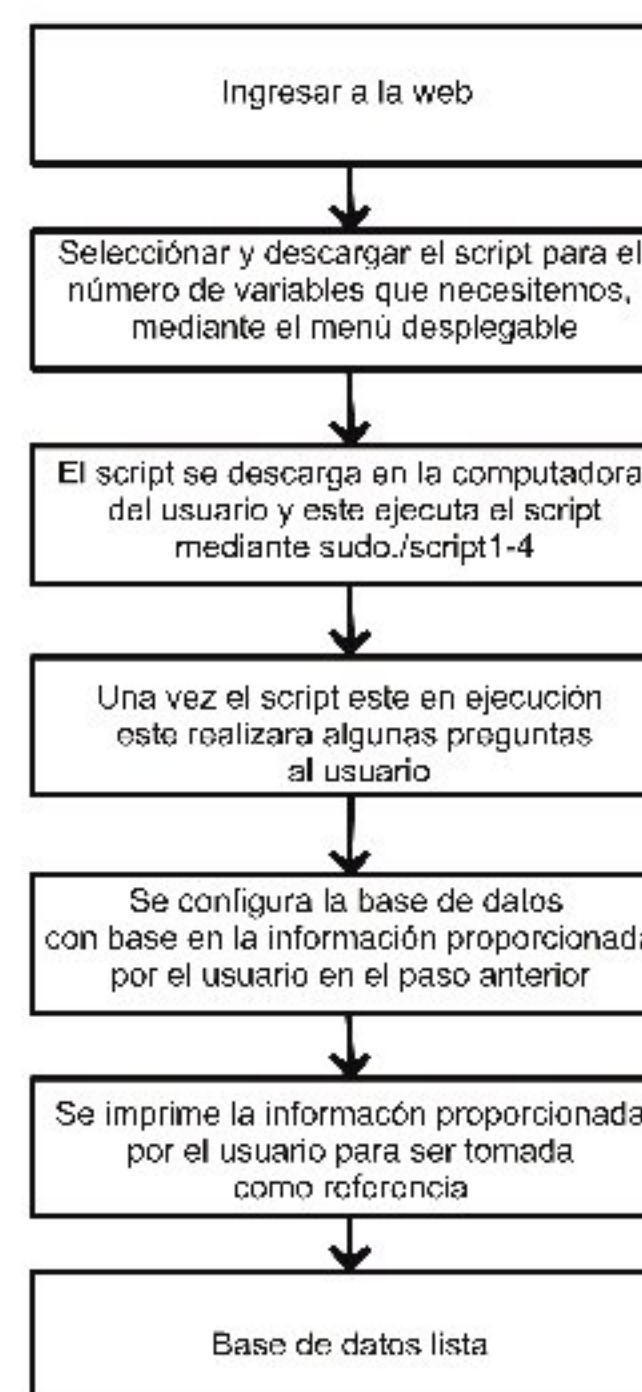


Figura 3.4: Scripts secundarios mediante la página

3.2.1 DISEÑO DE LA INTERFAZ PARA LA PÁGINA WEB

Al principio se pensó en algo como lo que se muestra en la Figura 3.5, un diseño minimalista que incluya solo un menú desplegable para proporcionar la descarga de los scripts al usuario.

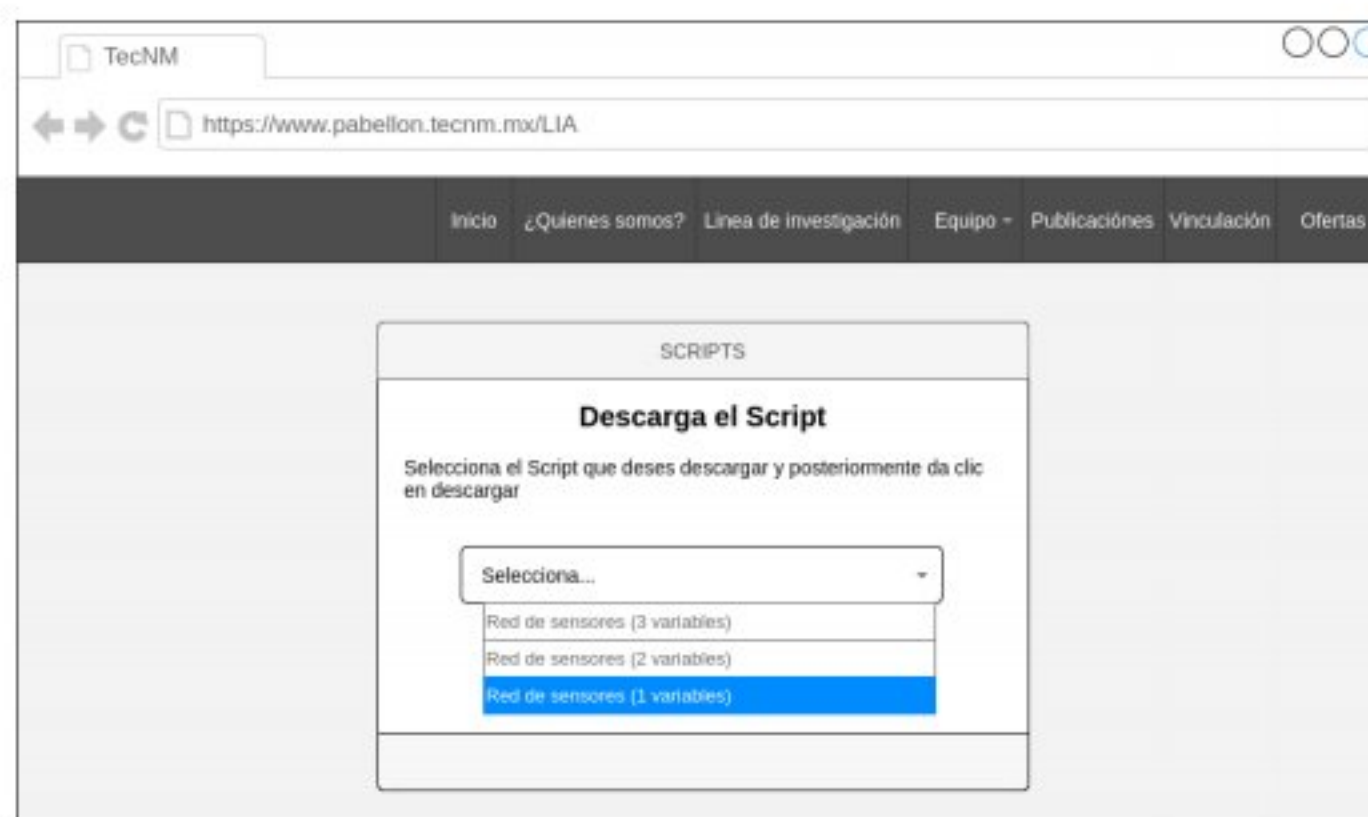


Figura 3.5: Diseño de la interfaz de usuario

Cada opción en el menú desplegable estará ligada a un script, cuando el usuario seleccione alguna de ellas, este se descargará automáticamente y podrá ser ejecutado. Al final se recicló el diseño de la página del LIA quedando como se muestra en la Figura 3.6.



Figura 3.6: Diseño de la interfaz de usuario final

3.3 IMPLEMENTACIÓN TDD

Para que el desarrollo fuera realizado de una manera mas organizada, y en general los scripts resultantes tuvieran el menor numero de Bugs posibles, se implemento TDD, en cual consiste en aplicar casos de prueba para especificar y validar lo que hará el código, se realizan pequeñas pruebas de funcionamiento y en caso de que la prueba fallé, se reescribe el código, este proceso se repite hasta que el código pase la prueba con éxito, con el fin de hacer el código más simple y sin errores [36]. El proceso se muestra de una manera gráfica en la Figura 3.7.

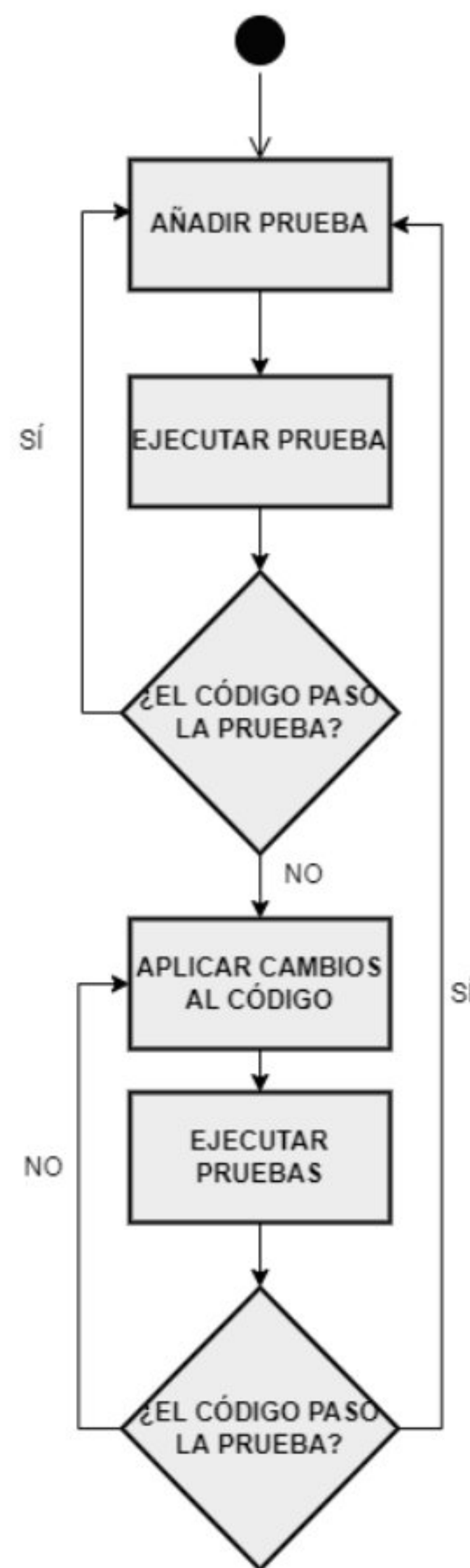


Figura 3.7: Ciclo TDD

3.4 CONSTRUCCIÓN DE LOS SCRIPTS

3.4.1 PROCEDIMIENTO PARA LA INSTALACIÓN DE DOCKER

Antes de automatizar un proceso mediante el uso de scripts primero hay que conocer el proceso en sí, por lo cual, en esta subsección se realizó una búsqueda sobre el ¿como instalar Docker?, después de revisar varias paginas donde decían como hacerlo, al final se optó por tomar el proceso de la *pagina oficial*.

Existen varias maneras de realizar la instalación, en esta ocasión se hará mediante el uso de un repositorio, debido a que su uso facilita las tareas de instalación y actualización, el proceso para instalar Docker es el siguiente:

Actualizamos nuestros repositorios

```
sudo apt-get update
```

Instalamos los siguientes paquetes, que permiten hacer uso de repositorios a través de HTTPS.

```
sudo apt-get install ca-certificates curl gnupg lsb-release
```

Agregamos la clave GPG oficial de Docker.

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --  
dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

Agregamos el repositorio.

```
echo \  
“deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]  
https://download.docker.com/linux/ubuntu \  
focal stable” | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Actualizamos nuestros repositorios nuevamente.

```
sudo apt-get update
```

Instalamos Docker Engine.

```
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

Los comandos anteriores nos servirán como referencia para poder crear los script, se tomó como base el repositorio para Ubuntu por cuestiones prácticas, no debería haber problemas si se ejecuta en Debian o en alguna otra distribución basada en él. Los comandos son meramente ilustrativos, en caso de querer replicarlos, se recomienda ir a la página oficial y copiarlos desde ahí, debido a que los caracteres especiales se escriben un poco diferente en LaTeX, y esto puede provocar pequeñas variaciones en el comando.

3.4.2 IMPLEMENTACIÓN Y DESPLIEGUE DE LA IMAGEN DE DOCKER CON LOS ADITAMENTOS NECESARIOS

Al igual que en la subsección anterior, se realizó una búsqueda con el fin de encontrar una imagen adecuada que nos permita implementar un servidor LAMP mediante Docker.

Al final de la búsqueda se optó por utilizar la imagen llamada “***DockerLAMP***”, creada por el usuario ***Matt Rayner*** de la comunidad de DockerHub, la imagen nos proporciona Apache, MYSQL y PHP, todo en una sola imagen, se proporciona también un manual de uso con los comandos a utilizar, para poder desplegar un contenedor de la imagen y poder hacer uso de él. Para descargar la imagen se utiliza el siguiente comando:

```
sudo docker pull mattrayner/lamp
```

Ya descargada la imagen, se utiliza el siguiente comando para desplegar el contenedor:

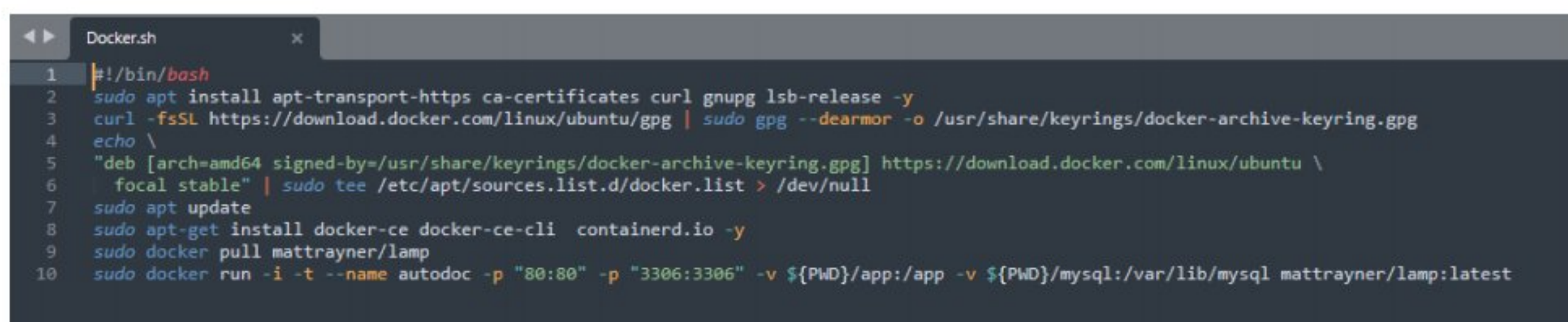
```
sudo docker run -i -t - -name autolamp -p “80:80” -p “3306:3306” -v ${PWD}/app:/app  
-v ${PWD}/mysql:/var/lib/mysql mattrayner/lamp:latest
```

En el comando anterior la opción **-V** es para definir los volumen a montar, en otras palabras es donde se guardarán los archivos de configuración del servidor LAMP, en el directorio **app**, es donde se deben poner los archivos HTML o PHP para que el servidor los muestre al ingresar a el.

El **-p** se utiliza para dar salida a los puertos en la red local, por ejemplo en el primer caso se pone **“-p “80:80”**” debido a que Apache trabaja en el puerto 80 y que que salga a nuestra red por el mismo puerto, el 80 de la izquierda hace referencia al puerto de salida hacia la red y el de la derecha al puerto utilizado por el contenedor. En esta subsección así como en la anterior, *se recomienda copiar los comandos directamente de DockerLAMP en DockerHub por el tema de los caracteres espaciales* y como estos pueden provocar pequeñas variaciones en los comandos.

3.4.3 ESTRUCTURA BASE PARA EL SCRIPT PRINCIPAL

En esta subsección se aborda la creación de los scripts una vez se conoce el procedimiento a realizar, se aplicaron los comandos obtenidos en la subsección 5.3.1 y 5.3.2, para crear el script que instalará Docker y desplegará **DockerLAMP**, se deben de copiar los comandos de las subsecciones mencionadas con anterioridad, en un archivo de texto plano, al principio agregamos la linea **“#!/bin/bash”** y asignamos la extensión **.sh** al archivo, para que el sistema lo detecte como un script, y lo movemos al directorio HOME de nuestro usuario, todo esto con ayuda de Sublime Text, el contenido del script se muestra en la Figura 3.8.



```

1 #!/bin/bash
2 sudo apt install apt-transport-https ca-certificates curl gnupg lsb-release -y
3 curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
4 echo \
5 "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \
6   focal stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
7 sudo apt update
8 sudo apt-get install docker-ce docker-ce-cli containerd.io -y
9 sudo docker pull matttrayner/lamp
10 sudo docker run -i -t --name autodoc -p "80:80" -p "3306:3306" -v ${PWD}/app:/app -v ${PWD}/mysql:/var/lib/mysql matttrayner/lamp:latest

```

Figura 3.8: Estructura base del script

Se añade **-y** al final de cada **install** para que no se le pregunte al usuario si desea instalar o no los paquetes, con esta opción se salta la pregunta y va directo a la instalación. Una

vez tengamos el archivo de texto con los comandos dentro, procedemos a darle permisos, son importantes por que delimitan lo que el script puede y no hacer dentro del sistema operativo.

Tecleamos el siguiente comando para dar premisos a nuestro archivo:

sudo chmod 777 Docker.sh

En este caso se utiliza el **777**, que hace referencia a los máximos privilegios, para mayor información sobre el comando **chmod** consulte aquí. Al ejecutar el comando no habrá ninguna salida como se muestra en la Figura 3.9.

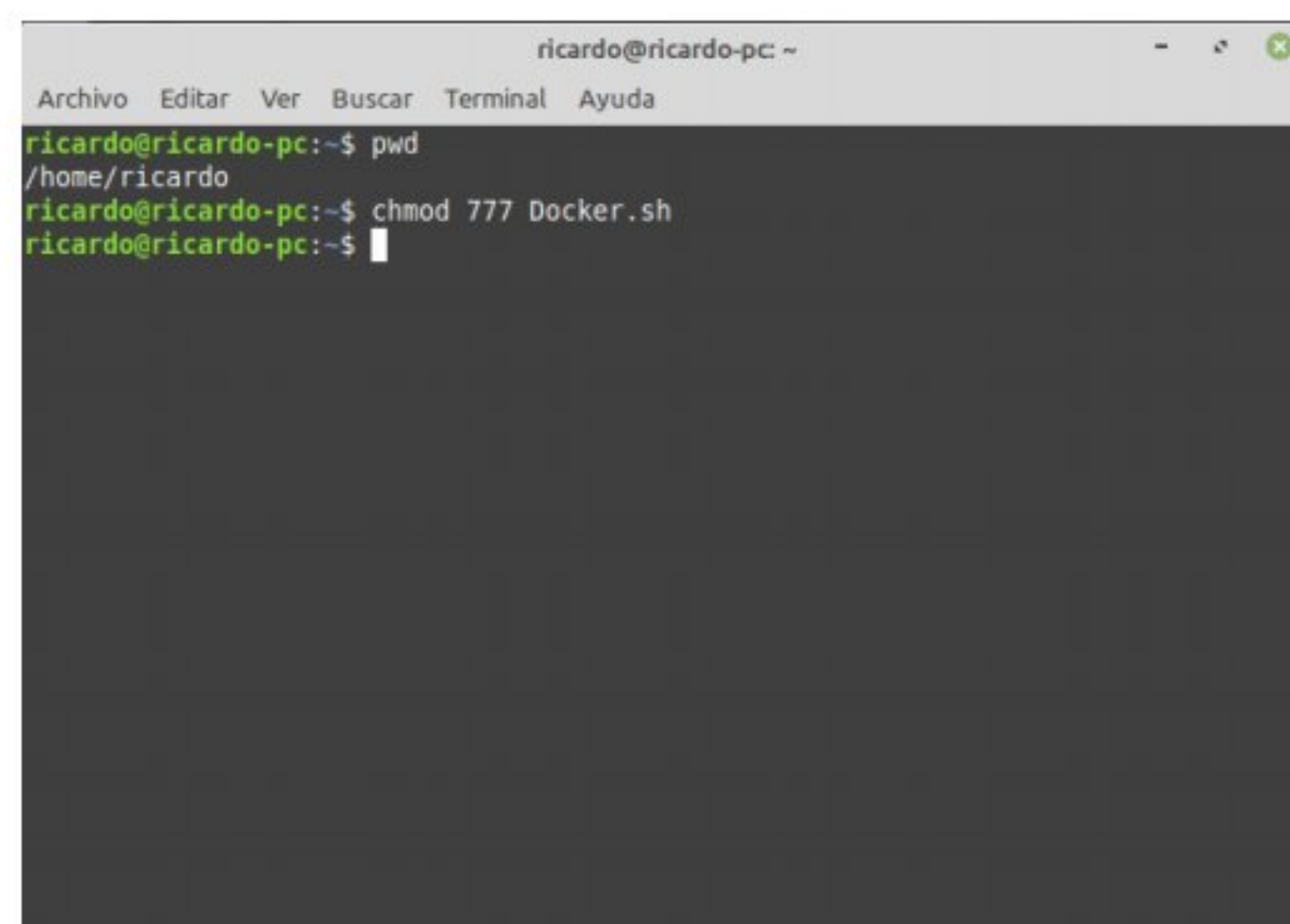
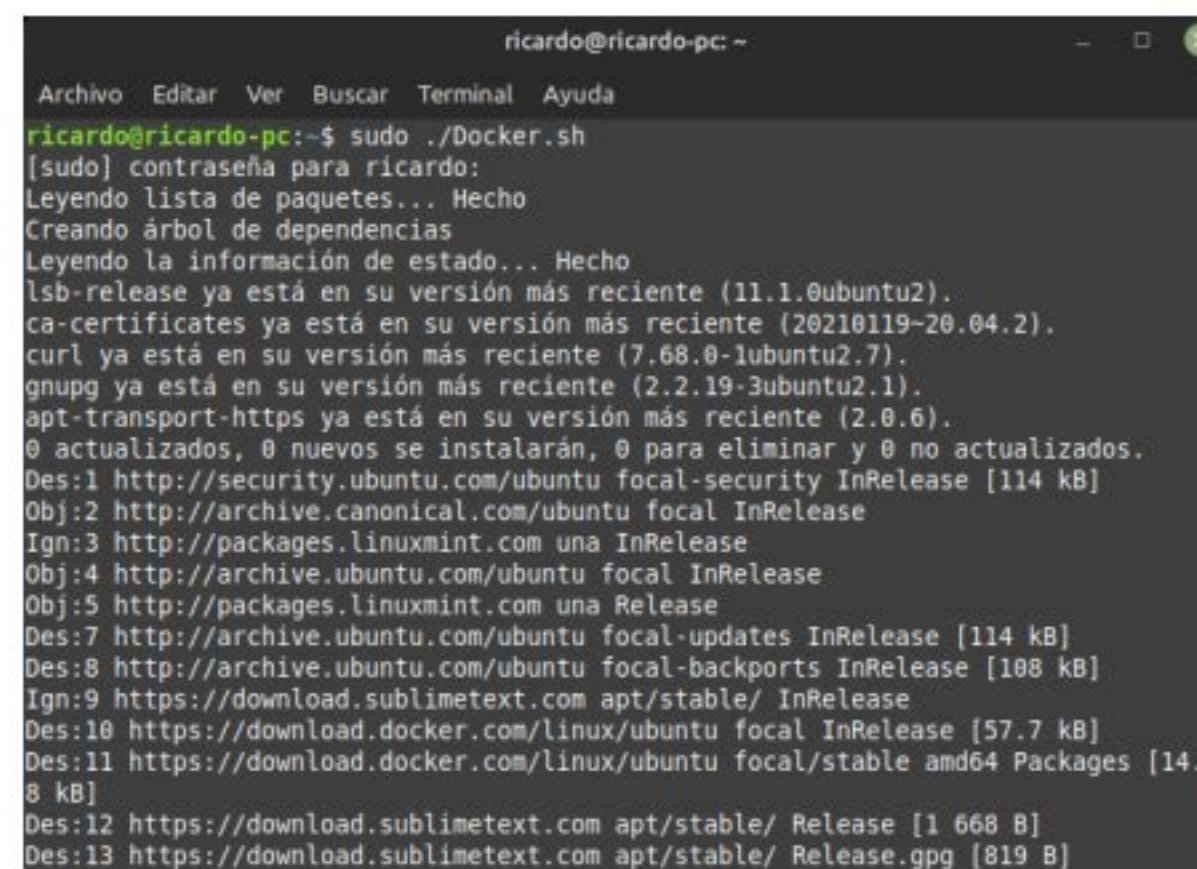
A screenshot of a terminal window titled 'ricardo@ricardo-pc: ~'. The window has a menu bar with 'Archivo', 'Editar', 'Ver', 'Buscar', 'Terminal', and 'Ayuda'. The terminal shows the following commands and output: 'ricardo@ricardo-pc:~\$ pwd' followed by '/home/ricardo', and 'ricardo@ricardo-pc:~\$ chmod 777 Docker.sh'. The prompt 'ricardo@ricardo-pc:~\$' is shown again on the next line, indicating the command was executed successfully without any output.

Figura 3.9: Asignación de permisos (Fuente: Linux Mint)

Ya con los respectivos permisos asignados, solo debemos ejecutar el archivo mediante el siguiente comando:

```
sudo ./Docker.sh
```

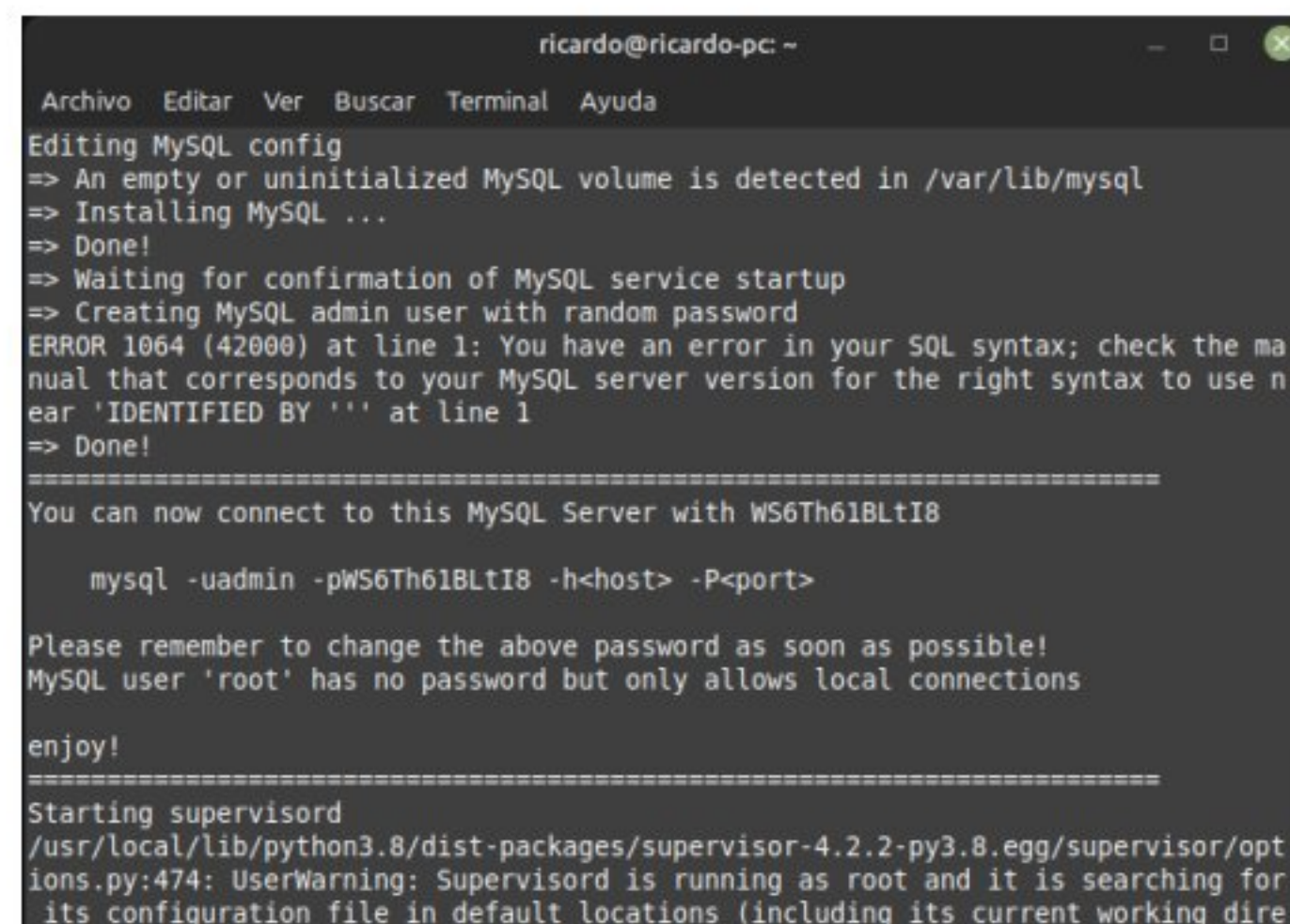
Si hemos hecho todo bien, los paquetes y la imagen comenzaran a configurarse como se muestra en la Figura 3.10.



```
ricardo@ricardo-pc: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
ricardo@ricardo-pc:~$ sudo ./Docker.sh  
[sudo] contraseña para ricardo:  
Leyendo lista de paquetes... Hecho  
Creando árbol de dependencias  
Leyendo la información de estado... Hecho  
lsb-release ya está en su versión más reciente (11.1.0ubuntu2).  
ca-certificates ya está en su versión más reciente (20210119-20.04.2).  
curl ya está en su versión más reciente (7.68.0-1ubuntu2.7).  
gnupg ya está en su versión más reciente (2.2.19-3ubuntu2.1).  
apt-transport-https ya está en su versión más reciente (2.0.6).  
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 0 no actualizados.  
Des:1 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]  
Obj:2 http://archive.canonical.com/ubuntu focal InRelease  
Ign:3 http://packages.linuxmint.com una InRelease  
Obj:4 http://archive.ubuntu.com/ubuntu focal InRelease  
Obj:5 http://packages.linuxmint.com una Release  
Des:7 http://archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]  
Des:8 http://archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]  
Ign:9 https://download.sublimetext.com apt/stable/ InRelease  
Des:10 https://download.docker.com/linux/ubuntu focal InRelease [57.7 kB]  
Des:11 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages [14.  
8 kB]  
Des:12 https://download.sublimetext.com apt/stable/ Release [1 668 B]  
Des:13 https://download.sublimetext.com apt/stable/ Release.gpg [819 B]
```

Figura 3.10: Ejecución de script (Fuente: Linux Mint)

Una vez que el proceso finalice sin errores, ya tendremos la estructura base de nuestro script principal, al finalizar el proceso se mostrara un mensaje como el que se puede apreciar en la Figura 3.11.



```
ricardo@ricardo-pc: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
Editing MySQL config  
=> An empty or uninitialized MySQL volume is detected in /var/lib/mysql  
=> Installing MySQL ...  
=> Done!  
=> Waiting for confirmation of MySQL service startup  
=> Creating MySQL admin user with random password  
ERROR 1064 (42000) at line 1: You have an error in your SQL syntax; check the ma  
nual that corresponds to your MySQL server version for the right syntax to use n  
ear 'IDENTIFIED BY '' at line 1  
=> Done!  
=====
```

Figura 3.11: Ejecución de script con éxito (Fuente: Linux Mint)

Nos cercioramos de que todo se encuentre en orden, colocamos un archivo HTML cualquiera dentro del directorio *app*, que se ha creado en nuestro HOME, le ponemos por nombre index.html y mediante cualquier navegador accedemos a la siguiente dirección: `http://localhost/`, si todo esta bien debemos ver el contenido de el HTML, como se muestra en la Figura 3.12.

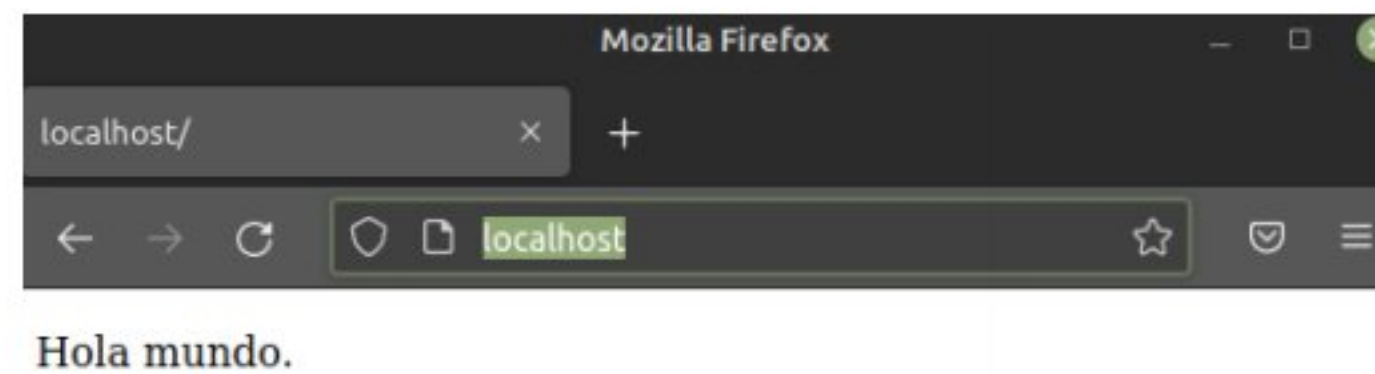


Figura 3.12: Localhost mostrando el contenido del HTML (Fuente: Linux Mint)

Una vez el servidor este en funcionamiento, ya estaría lista la estructura base del script principal.

3.4.4 RETOQUES A LA ESTRUCTURA BASE DEL SCRIPT PRINCIPAL

Para que el script tenga una poco más de validación y en general se vea mejor al ejecutarse, en esta subsección se añaden algunos retoques. El funcionamiento general del script se ve representado con el siguiente pseudocódigo:

```
script Docker;  
chechar si los paquetes necesarios estan instalados  
si lo estan:  
imprimir ("Los paquetes estan instalados...");  
descargar imagen;  
imprimir ("Descargando imagen...");  
desplegar imagen;  
imprimir ("Desplegando imagen...");  
si no lo estan:
```

```

imprimir ("Los paquetes no estan instalados...");
imprimir ("Instalando..."); descargar imagen;
imprimir ("Descargando imagen...");
desplegar imagen;
imprimir ("Desplegando imagen...");

```

Se añadió la validación, para que si los paquetes ya están instalados solo se descargue y configure la imagen, podemos observar esto en la línea 24, contenida en la Figura 3.13, esta línea utiliza el administrador de paquetes **dpkg** para corroborar que los paquetes están instalados, en caso de no estarlos el **if** contenido en la línea 25 retorna un 0, lo cual desencadena el **then**, en el caso contrario se implanta el **else** de la línea 53. También se integraron algunos pequeños detalles visuales, se colocaron algunos **echo** después de cada instrucción, para que el usuario conozca lo que esta pasando en ese momento, mediante un breve mensaje, como se muestra en la línea 30, a esta linea le antecede un **sleep**, el cual se utiliza para esperar determinado tiempo antes de ejecutar el siguiente comando, en este son caso 5 segundos, para que el usuario tenga tiempo de leer los mensajes emitidos.

```

1  #!/bin/bash
2  sudo apt install xdotool -y
3  echo "Instalando xdotool"
4  sleep 5
5  xdotool key alt+F10
6  blanco=$(cat /dev/urandom | tr -dc ' ' | fold -w 100 | xargs printf '%100s' | tr -d '\n')
7  echo -e "${blanco}"
8  echo -e "AUTOLAMP"
9
10
11
12
13
14
15  S{WC}
16  echo "Creado carpetas necesarias"
17  mkdir AutolAMP
18  sleep 5
19  echo "Ubicado la carpeta de trabajo"
20  sleep 5
21  cd AutolAMP
22  sleep 5
23  echo "Checando si los paquetes necesarios estan instalados..."
24  dpkg-query -f='${Package} ${Version} ${Architecture}\n' -W -f='${Package} ${Version} ${Architecture}\n' | grep docker-ce && /dev/null
25  if [ $? -eq 0 ]; then
26
27  then
28      echo "Los paquetes no estan instalados..."
29      sleep 5
30      echo "Instalando..."
31      sleep 5
32      sudo apt install apt-transport-https ca-certificates curl gnupg lsb-release -y
33      echo "Agregando GPG key para docker..."
34      sleep 5
35      curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
36      echo "Agregando repositorio..."
37      sleep 5
38      echo \
39  "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \
40  focal stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
41      echo "Actualizando repositorios"
42      sleep 5
43      sudo apt update
44      sudo apt-get install docker-ce docker-ce-cli containerd.io -y
45      echo "Todos los paquetes han sido instalados correctamente..."
46      sleep 5
47      echo "Obteniendo imagen"
48      sleep 5
49      sudo docker pull matrayner/lamp
50      echo "Configurando imagen"
51      sleep 5
52      sudo docker run -i -t --name autolamp -p "80:80" -p "3306:3306" -v $(PWD)/app:/app -v $(PWD)/mysql:/var/lib/mysql matrayner/lamp:latest
53  else
54      echo "Los paquetes estan instalados..."
55      sleep 5
56      echo "Configurando..."
57      sleep 5
58      echo "Obteniendo imagen"
59      sleep 5
60      sudo docker pull matrayner/lamp
61      echo "Configurando imagen"
62      sleep 5
63      sudo docker run -i -t --name autolamp -p "80:80" -p "3306:3306" -v $(PWD)/app:/app -v $(PWD)/mysql:/var/lib/mysql matrayner/lamp:latest
64  fi

```

Figura 3.13: Script principal con los retoques aplicados

Al principio del script se optó por instalar **xdotool**, nos permite simular pulsaciones de

teclas y movimientos de ratón o clics, en este caso se utilizó para hacer que la terminal se maximice con la combinación de teclas ***alt+F10*** , y que el usuario pueda apreciar de mejor manera el contenido del script al ejecutarse. El ***mkdir*** en la línea 17, es utilizado para crear la carpeta de trabajo del script.

Y por último se integró un tipo letrero con el nombre del proyecto, este letrero se construyó haciendo uso de la web de aquí, en ella solo tecleamos lo que sea de nuestro agrado, en este caso ***“AUTOLAMP”***, posteriormente seleccionamos la fuente que mejor nos parezca, damos clic en ***“Select & Copy”***, en el script se pone un ***echo -e*** y se acomoda tal cual se muestra en la línea 8 a la 15, en las dos líneas anteriores a la 8 se definen los colores para el letrero, para consultar los colores y como se deben implantar consulte aquí.

3.4.5 ESTRUCTURA DE LOS SCRIPTS SECUNDARIOS

Por cuestiones prácticas los scripts se dividieron en dos, el script principal instalan los paquetes necesarios, en caso de ser requerido, descarga y configura DockerLAMP para su posterior despliegue. Los scripts secundarios estan dirijos a crear una base de datos, un usuario con su respectiva contraseña, la tabla y columnas que contendran la información, actualmente se cuenta con 4 script, cada uno diseñado para trabajar con determinado número de variables, como lo indica su nombre, por ejemplo el ***“script1”*** esta diseñado para trabajar con solo una variable, las variables hacen referencia al nombre que tomarán las columnas de la tabla que contendra la base de datos, se muestra el contenido del script mencionado en la Figura 3.14.

```

1  #!/bin/bash
2  echo "Introduzca en nombre para el usuario"
3  read username
4  echo "Introduzca la contraseña para el usuario"
5  read userpass
6  sudo docker exec autodoc mysql -uroot -e "CREATE USER '$username'@'localhost' IDENTIFIED BY '$userpass';"
7  sudo docker exec autodoc mysql -uroot -e "GRANT ALL PRIVILEGES ON *.* TO '$username'@'localhost';"
8  echo "Introduzca en nombre para la base de datos"
9  read dbname
10 sudo docker exec autodoc mysql -uroot -e "create database $dbname"
11 echo "Introduzca en nombre de la tabla"
12 read tname
13 echo "Introduzca en nombre de la variable 1"
14 read var1
15 sudo docker exec autodoc mysql -uroot -e "use $dbname; CREATE TABLE $tname ('id' int(11) PRIMARY KEY NOT NULL AUTO_INCREMENT, $var1 varchar(70) COLLATE utf8mb4_spanish_ci N
16 echo "#####"
17 echo "El nombre para el usuario es: $username"
18 echo "La contraseña para el usuario es: $userpass"
19 echo "El nombre para la base de datos es: $dbname"
20 echo "El nombre para la tabla es: $tname"
21 echo "El nombre de la variable 1 es: $var1"
22 echo "Puede entrar a PHPMYADMIN para ver la estructura y cambiar la configuracion de la base de datos desde el siguiente enlace: http://localhost/phpmyadmin/"
23

```

Figura 3.14: Script secundario con los retoques aplicados

En la línea 3 se muestra como se utiliza *read*, para tomar las entradas del teclado y almacenarlas en una variable, la cuál se encuentra ubicada a la derecha del *read*, llamada *username*, en la línea 6 y 7 se crea el usuario y se le asigna su respectiva contraseña, en la 10 se crea la base de datos cuyo nombre fue asignado en las líneas 8 y 9, en la línea 15 se crea la tabla que almacenara la información, con los datos proporcionados por el usuario en las líneas 11 y 13. La línea 15 no se percibe por completo por lo cual se proporciona un enlace al archivo aquí.

Para entender el funcionamiento de este script secundario de una mejor manera, a continuación se presenta el pseudocódigo del mismo:

```
script1;  
preguntar("¿Cuál será el nombre para el usuario de la base de datos?");  
guardar->username;  
preguntar("¿Cuál será la contraseña para el usuario ingresado  
anteriormente?");  
guardar->userpass;  
crear usuario=userpass;  
asignar contraseña al usuario=userpass;  
preguntar("¿Cuál será el nombre para la base de datos?");  
guardar->bdname;  
crear base de datos=bdname;  
preguntar("¿Cuál será el nombre para la tabla?");  
guardar->tname;  
preguntar("¿Cuál será el nombre para la variable 1?");  
guardar->var1;  
crear tabla=tname y columna 1=var1;  
imprimir username, userpass;  
imprimir bdname, tname , va1;
```

3.5 CONSTRUCCIÓN DE LA PÁGINA WEB

3.5.1 ESTRUCTURA DEL MENÚ DESPLEGABLE

En la Figura 3.15 se muestra el HTML de la página, en las líneas 51 a la 55 se muestran las opciones contenidas en el menú desplegable, cada opción esta ligada a un archivo con extensión *.sh* mediante un *value=*“”, por ejemplo: “value=“script4.sh””, y lo que se presenta en color blanco en esa misma línea sería lo que se mostraría al usuario.

```

50 <h5 class="text-center">Sistema que automatiza la creación de contenedores Docker, para la implementación de redes de sensores en el LIA</h5>
51 <br>
52 <div class="row">
53 <div class="col-md-4 offset-md-4">
54 <div class="card text-center">
55 <div class="card-header">
56 SCRIPTS
57 </div>
58 <div class="card-body">
59 <p class="card-text">Seleccione el Script que necesite según las variables y este se descargara automáticamente</p>
60 <div class="input-group">
61 <select class="col-md-8 offset-md-2" name="download" onChange="download(this.value)">
62 <option value="..index.html">Selecciona..</option>
63 <option value="Docker.sh">Instala Docker para el funcionamiento de los Scripts</option>
64 <option value="script4.sh">Red de sensores (4 variables)</option>
65 <option value="script3.sh">Red de sensores (3 variables)</option>
66 <option value="script2.sh">Red de sensores (2 variables)</option>
67 <option value="script1.sh">Red de sensores (1 variables)</option>
68 </select>
69 </div>
70 </div>
71 <div class="card-footer text-muted">
72 </div>
73 </div>
74 </div>
75 <br>
76 <p class="text-center" class="fw-lighter">Documentación de script:<a href="https://github.com/Ricardo-Gomez-4-113/AutoDoc">Ricardo Gomez</a></p>

```

Figura 3.15: Estructura HTML

La función de descarga en cada opción se añade mediante JavaScript, como se muestra en la Figura 3.16, en la línea 17 se define el directorio de trabajo, en este caso *scripts/*, es aquí donde deberemos poner nuestros scripts para ser descargados.

```

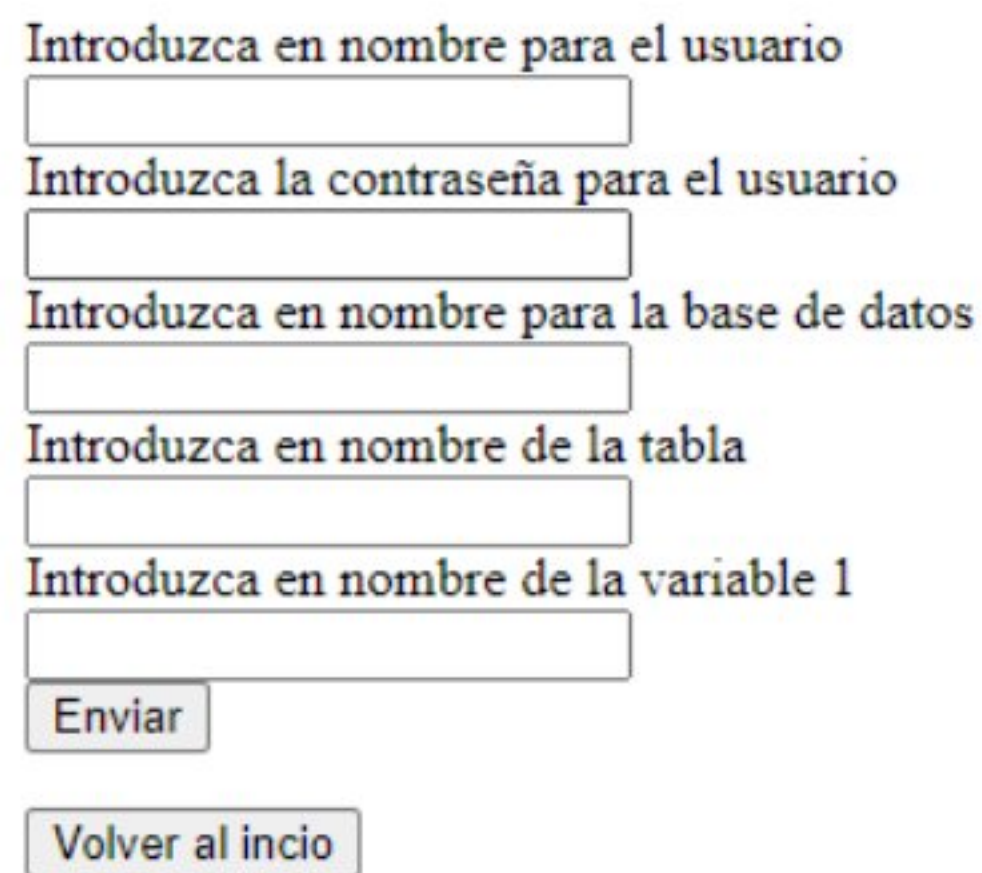
12 <script type="text/javascript" src="estilo/js/mijavascript.js"></script>
13 <script type="text/javascript" src="estilo/js/bootstrap.js"></script>
14 <script type="text/javascript">
15   function download(d) {
16     if (d == 'Select document') return;
17     window.location = 'scripts/' + d;
18   }
19 </script>

```

Figura 3.16: Estructura JavaScript

3.6 GENERADOR DE SCRIPTS

Como función extra se añadió a la pagina la parte para generar scripts para la base de datos, se le proporciona al usuario un formulario, para que proporcione la información requería, y con base en ella pueda generarse el script. El formulario para la creación de los script se encuentra compuesto por HTML básico, no se incluye estilo CSS debió a que se encuentre en una fase funcional no terminada, como podemos observar en la Figura 3.17.



Introduzca en nombre para el usuario

Introduzca la contraseña para el usuario

Introduzca en nombre para la base de datos

Introduzca en nombre de la tabla

Introduzca en nombre de la variable 1

Figura 3.17: Formulario para el generador de scripts en HTML básico

La estructura de estos scripts es la misma que la de los scripts secundarios, al formulario se le añadieron dos botones, el botón “Volver al inicio” redirecciona al usuario a la página principal y el botón “Enviar”, envía los datos contenidos en el formulario a un archivo *des.php*, en el cual son procesados, para ser escritos en el archivo *.sh* que se descargara automáticamente, el contenido de dicho archivo se muestra en la Figura 3.18.

```

1  <?php
2  $username = $_POST["username"];
3  $userpass = $_POST["userpass"];
4  $dbname = $_POST["dbname"];
5  $tablename = $_POST["tablename"];
6  $var1 = $_POST["var1"];
7
8  if (unlink('script1.sh'))
9  {
10     // File was successfully deleted
11
12     $file = fopen("script1.sh", "w");
13
14     fwrite($file, "#!/bin/bash" . PHP_EOL);
15
16     fwrite($file, "sudo docker exec autolamp mysql -uroot -e \"CREATE USER '$username'@'localhost' IDENTIFIED BY '$userpass';\"\" . PHP_EOL);
17
18     fwrite($file, "sudo docker exec autolamp mysql -uroot -e \"GRANT ALL PRIVILEGES ON *.* TO '$username'@'localhost';\"\" . PHP_EOL);
19
20     fwrite($file, "sudo docker exec autolamp mysql -uroot -e \"create database $dbname\"\" . PHP_EOL);
21
22     fwrite($file, "sudo docker exec autolamp mysql -uroot -e \"use $dbname; CREATE TABLE $tablename (\"id\" int(11) PRIMARY KEY NOT NULL AUTO_INCREMENT, $var1 varchar(70) COLLATE utf8mb4_spanish_ci NOT NULL, \"
23     fecha\" datetime NOT NULL DEFAULT CURRENT_TIMESTAMP);\"\" . PHP_EOL);
24
25     fwrite($file, "echo \"El nombre para el usuario es: $username\"\" . PHP_EOL);
26
27     fwrite($file, "echo \"La contraseña para el usuario es: $userpass\"\" . PHP_EOL);
28
29     fwrite($file, "echo \"El nombre para la base de datos es: $dbname\"\" . PHP_EOL);
30
31     fwrite($file, "echo \"El nombre para la tabla es: $tablename\"\" . PHP_EOL);
32
33     fwrite($file, "echo \"El nombre de la variable 1 es : $var1\"\" . PHP_EOL);
34
35     fwrite($file, "echo \"Puede entrar a PHPMYADMIN para ver la estructura y cambiar la configuracion de la base de datos desde el siguiente enlace: http://localhost/phpmyadmin/\"\" . PHP_EOL);
36
37     fclose($file);
38
39     header("Content-disposition: attachment; filename=script1.sh");
40     readfile("script1.sh");
41 }
42 else {
43     echo "Hubo un error";
44 }
45
46 >>

```

Figura 3.18: Algoritmo PHP para generar scripts

Para escribir el archivo de texto línea por línea mediante PHP, se utilizó la función *fwrite*, la descarga se hace mediante *header*, cada vez que se crea un archivo se borra el anterior para no generar archivos basura, esto se realiza en la línea 8 con la función *unlink*.

3.7 EVALUACIÓN

3.7.1 SCRIPT PRINCIPAL

A continuación se muestra la evaluación de los requerimientos del script principal, mediante iteraciones en la Tabla 3.3.

Tabla 3.3: Evaluación de objetivos del script principal		
Completado	Iteración	Objetivo
X	1	Instalar paquetes necesarios.
X	2	Validar si los paquetes necesarios ya están instalados.
X	3	Descargar imagen DockerLAMP.
X	4	Configurar imagen DockerLAMP y desplegar servidor.

3.7.2 SCRIPTS SECUNDARIOS

Por cuestiones prácticas el proceso de automatización se dividió en dos, el script principal prepara y despliega el entorno de desarrollo LAMP, mientras que los scripts secundarios crean la base de datos con la información proporcionada por el usuario, la evaluación de los requerimientos de éstos se muestra en la Tabla 3.4.

Tabla 3.4: Evaluación de objetivos de los scripts secundarios		
Completado	Iteración	Objetivo
X	1	Recopilar datos para la creación de la base de datos.
X	2	Crear base de datos con los datos obtenidos.

3.7.3 PÁGINA WEB

En un principio, la página web fue pensada, para solo proporcionar la interfaz para la descarga de los scripts, pero también se añadió la función para generarlos, los requerimientos se muestran a continuación en la Tabla 3.5.

Tabla 3.5: Evaluación de objetivos de la página		
Completado	Iteración	Objetivo
X	1	Integrar menú desplegable.
X	2	Proporcionar interfaz para la descarga de los scripts.
X	3	Generar scripts para la base de datos.

CAPÍTULO 4

RESULTADOS

Se obtuvieron 5 scripts cuya función se describe a continuación:

- Docker.sh: Instala los paquetes necesarios para el correcto funcionamiento de Docker, descarga la imagen DockerLAMP y despliega el contenedor.
- script1.sh: Contiene las instrucciones necesarias para crear una base de datos, una tabla con una columna, el usuario para dicha base y su respectiva contraseña.
- script2.sh: Contiene las instrucciones necesarias para crear una base de datos, una tabla con dos columnas, el usuario para dicha base y su respectiva contraseña.
- script3.sh: Contiene las instrucciones necesarias para crear una base de datos, una tabla con 3 columnas, el usuario para dicha base y su respectiva contraseña.
- script4.sh: Contiene las instrucciones necesarias para crear una base de datos, una tabla con cuatro columnas, el usuario para dicha base y su respectiva contraseña.

Adicionalmente a las columnas nombradas por el usuario en los scripts, se añade un ***id*** y la ***fecha*** para identificar de mejor manera los datos insertados en la tabla.

Se obtuvo también la página, que proporciona la interfaz para la descarga de los scripts mediante un menú desplegable, y se generaron repositorios en Github tanto para esta como para los scripts, donde se muestran las instrucciones para usar los scripts y/o la página.

Como una funcionalidad extra, se integró la posibilidad de generar scripts para la base de datos, con base en la información proporcionada por el usuario.

CAPÍTULO 5

DISCUSIÓN

Los scripts automatizan la instalación y configuración de un Servidor LAMP, en consecuencia la implementación de una red de sensores se simplifica, y a su vez simplifica el monitoreo de varios cultivos a la vez, esto trae como beneficios un mejor uso de los recursos naturales y cultivos saludables.

Aunque el uso de los scripts facilita el proceso, se requiere un mínimo de conocimiento respecto al uso de una terminal para hacer uso de estos, esta podría ser una desventaja, aunque por lo cual, también se generó el repositorio en Github, en el se incluye el procedimiento a realizar para hacer uso de los scripts, se explican los comandos a utilizar y el porque de cada uno de ellos.

Docker es muy similar a Podman y la diferencia mas notable entre estas dos tecnologías de virtualización mediante contenedores, es la manera en que se administran los servicios en segundo plano, Docker gestiona todo con el mismo servicio, lo cual puede traer algunas desventajas, ya que mientras más contenedores usemos, más grande y complejo se hace el servicio de Docker, mientras que Podman no necesita de un gran servicio en segundo plano para funcionar.

Se utilizó Docker por la experiencia en el uso del mismo, ya se había utilizado antes esta tecnología para llevar a cabo algunos proyectos, se utilizó para ahorrar un poco de tiempo, tiempo que nos tomaría en aprender a utilizar Podman, aunque si bien su uso es muy similar existen conceptos que en Docker no se manejan.

CAPÍTULO 6

CONCLUSIONES

Generar Docker.sh desde la página no tiene mucho sentido práctico, el algoritmo dentro de este no cambia, entonces lo mejor es que se quede igual y que solo se proporcione la descarga mediante el menú desplegable.

Los script que crean las tablas y las columnas de la misma están limitadas a cuatro variables en próximas versiones podrían considerarse un mayor numero de variables.

Las pruebas de software no pudieron realizarse adecuadamente debido a la configuración interna de la red de la institución, el uso de Fortinet limita mucho a la misma e imposibilita hacer uso de los scripts dentro de la red institucional.

Aunque si bien Podman y Docker son tecnologías cuyos propósitos son similares, se opto por utilizar Docker porque la imagen utilizada esta hecha para trabajar con él y a su vez por el hecho de que se tenia un poco mas de experiencia utilizando Docker.

APÉNDICE A

ANEXOS

A.1 ANEXO 1: MANUAL DE INSTALACIÓN Y CONFIGURACIÓN DE LINUX MINT

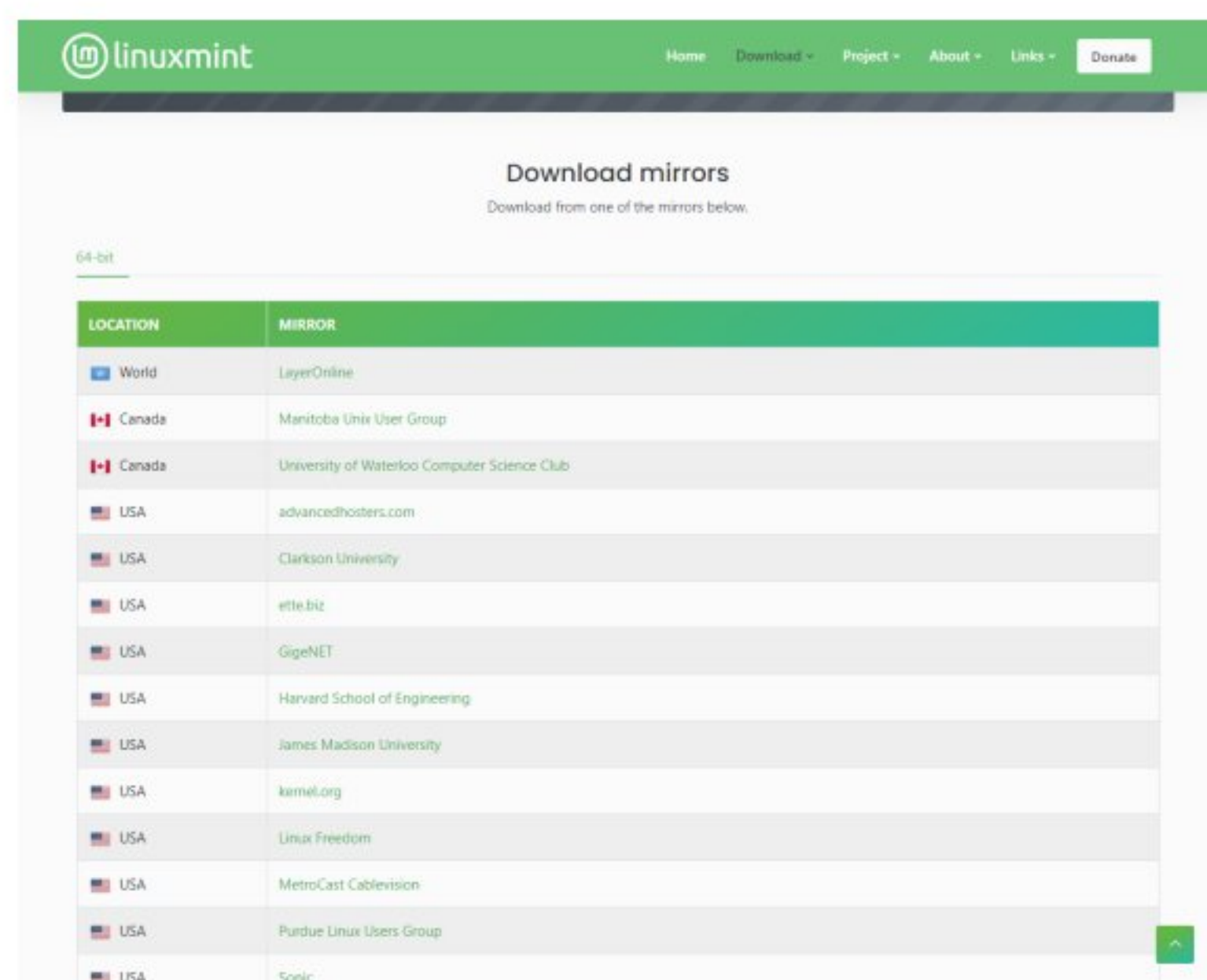
A.2 DESCARGA DE LOS ARCHIVOS NECESARIOS PARA LA INSTALACIÓN DEL SISTEMA OPERATIVO DE TRABAJO

Para realizar la instalación es necesario en mi primer lugar descargar el archivo ISO desde la pagina oficial mediante el siguiente enlace: <https://linuxmint.com/edition.php?id=292>, la Figura A.1 muestra la apariencia del sito de descarga.



Figura A.1: Interfaz página oficial (Fuente: página oficial Linux Mint)

Una vez en el sitio de descarga, procedemos a desplazarnos hacia abajo, hasta observar los servidores espejo, los cuales nos proporcionan un enlace para la descarga del archivo ISO, como se muestra la Figura A.2. De estos servidores podemos elegir el más cercano o el de nuestra preferencia, el archivo es el mismo.



LOCATION	MIRROR
World	LayerOnline
Canada	Manitoba Unix User Group
Canada	University of Waterloo Computer Science Club
USA	advancedhosters.com
USA	Clarkson University
USA	ette.biz
USA	GigaNET
USA	Harvard School of Engineering
USA	James Madison University
USA	kernel.org
USA	Linux Freedom
USA	MetroCast Cablevision
USA	Purdue Linux Users Group
USA	Sonic

Figura A.2: Servidores espejo (Fuente: página oficial Linux Mint)

A.2.1 CREACIÓN DEL MEDIO DE ARRANQUE PARA LA INSTALACIÓN

Para la creación del medio de arranque se hizo uso de Rufus, es un programa utilizado para crear medios de arranque para instalar sistemas operativos en equipos de cómputo, se descargó el programa haciendo uso del siguiente enlace: <https://rufus.ie/es/>.

Una vez en el sitio procedemos a buscar la sección de descargas, en esta sección hay un par de opciones de descarga, la versión portable con la cual podremos trabajar sin instalar nada en nuestro PC y la versión común con la cual necesitaremos instalar para poder trabajar. En la Figura A.3 se muestran las opciones de descarga.

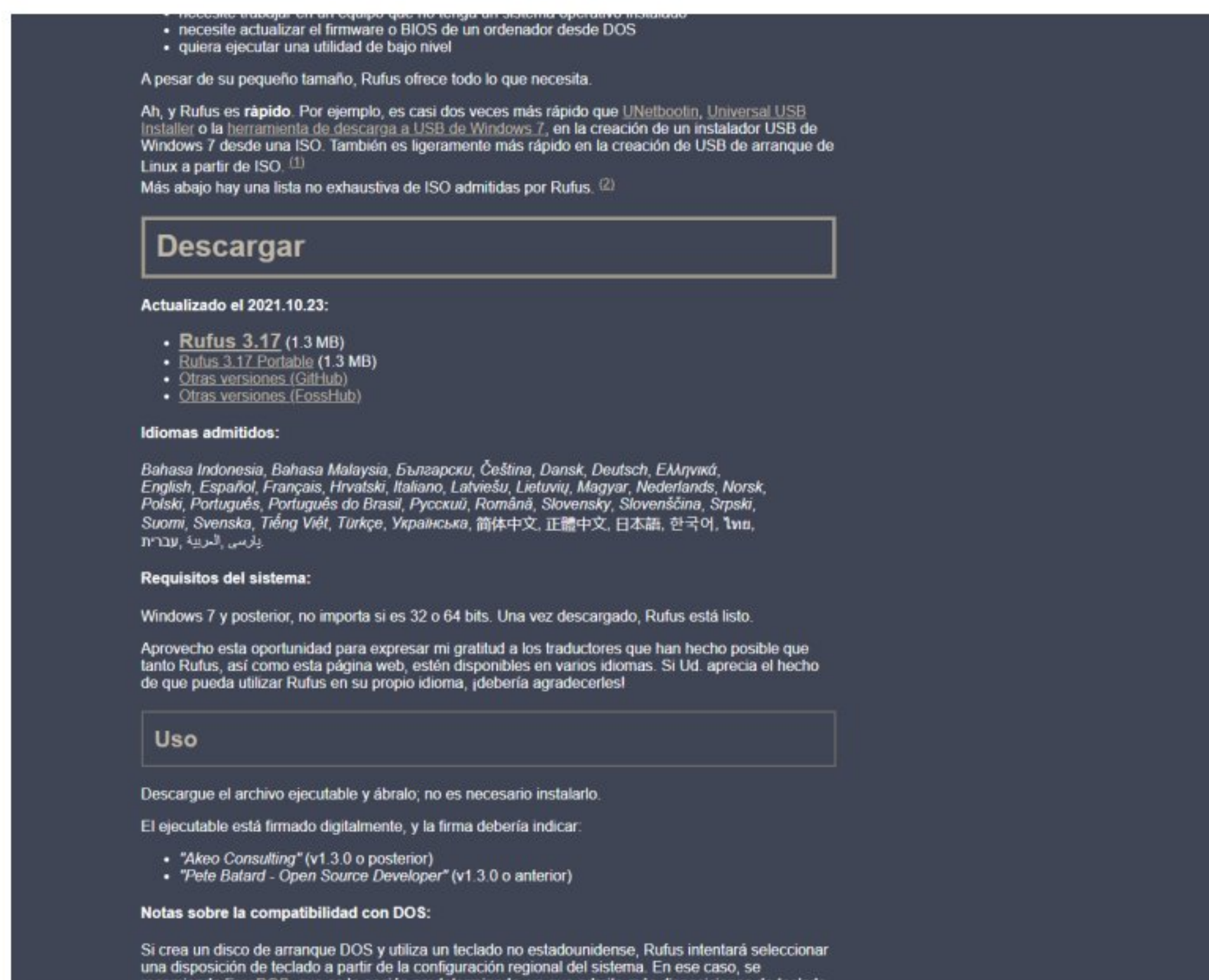


Figura A.3: Opciones de descarga (Fuente: página oficial Rufus)

Ya descargado ejecutamos el programa y procedemos a realizar la creación del medio de arranque. En la Figura A.4 se muestra la interfaz de Rufus para tomar como referencia.

Para crear el medio de arranque solo seleccionamos el dispositivo de almacenamiento que hemos destinado para esto, en el apartado “Elección de arranque” se selecciona “Disco o imagen ISO”, se pulsa en seleccionar para elegir la imagen de “Linux Mint” previamente descargada y por ultimo oprimimos el botón “Empezar” para que el proceso comience.

En este caso el proceso se realizó con un SSD externo conectado por USB, pero se puede hacer uso de cualquier memoria con 4GB de almacenamiento o superior.

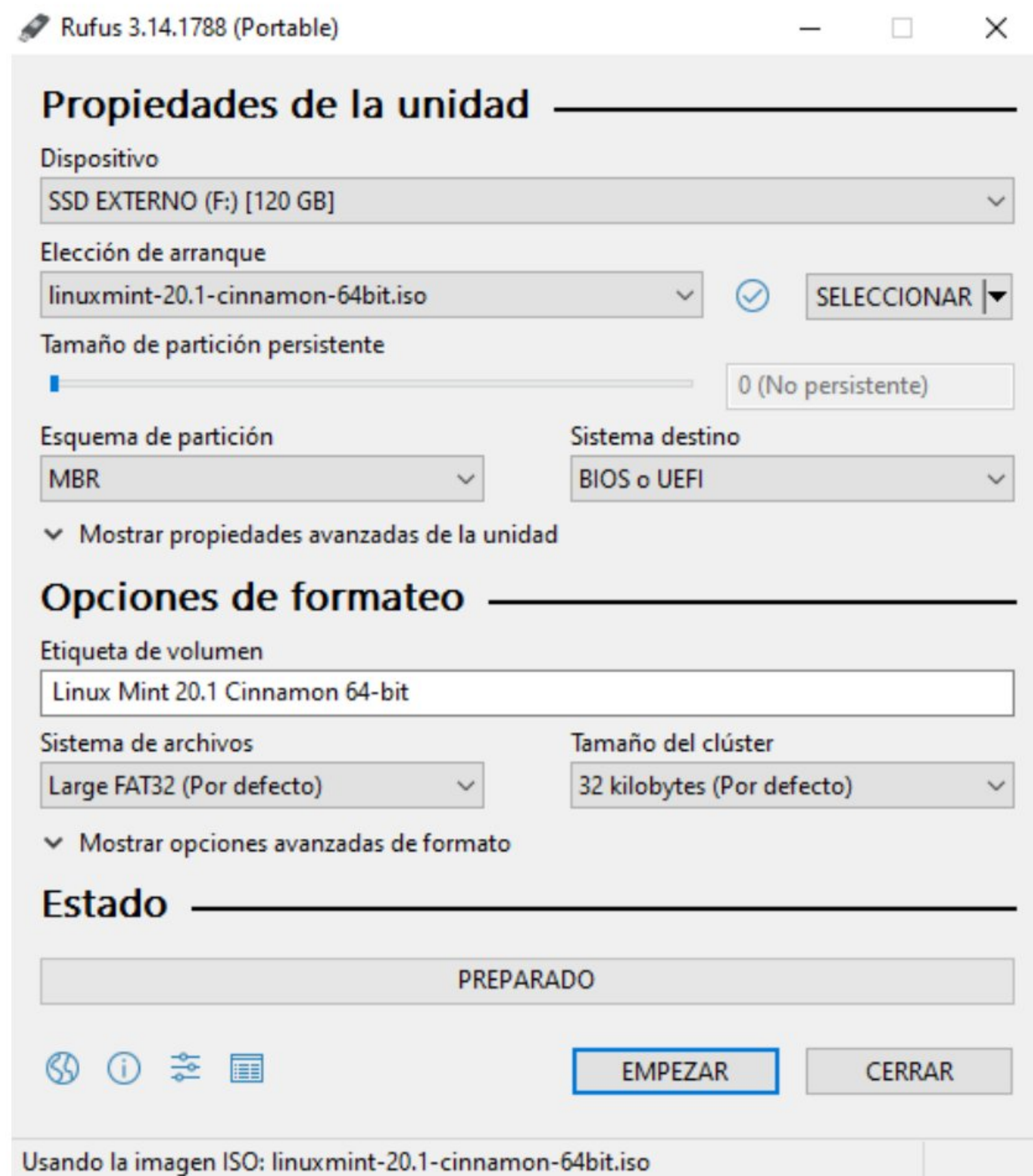


Figura A.4: Interfaz del programa (Fuente: Rufus)

A.2.2 INSTALACIÓN DE LINUX MINT

En esta sección se presenta el procedimiento para realizar la instalación de Linux Mint una vez se tenga el medio de arranque preparado, cabe mencionar que el proceso puede variar un poco dependiendo de la PC en la que se realice el proceso, esto debido a que puede haber pequeñas variaciones en la estructura de la **BIOS** y puede variar también la tecla con la cual se accede a ella.

Conectamos el medio de arranque creado con anterioridad a cualquier puerto USB de nuestra computadora y entramos al “Boot menú” de nuestra computadora, para entrar a dicho menú presionamos la tecla del teclado designada para ello, la tecla puede variar de marca a marca, en este caso se utiliza la tecla “F12”. Procedemos a apagar la computadora, una vez apagada oprimimos el botón de encendido e inmediatamente comenzamos a presionar la tecla en cuestión. Una vez dentro del “Boot menú” nos saldrá una pantalla como la que se muestra en la Figura A.5.



Figura A.5: Interfaz del Boot menú

Seleccionamos nuestro medio de arranque y damos enter, la computadora iniciará desde la USB y podremos ver algo como lo que se muestra en la Figura A.6.

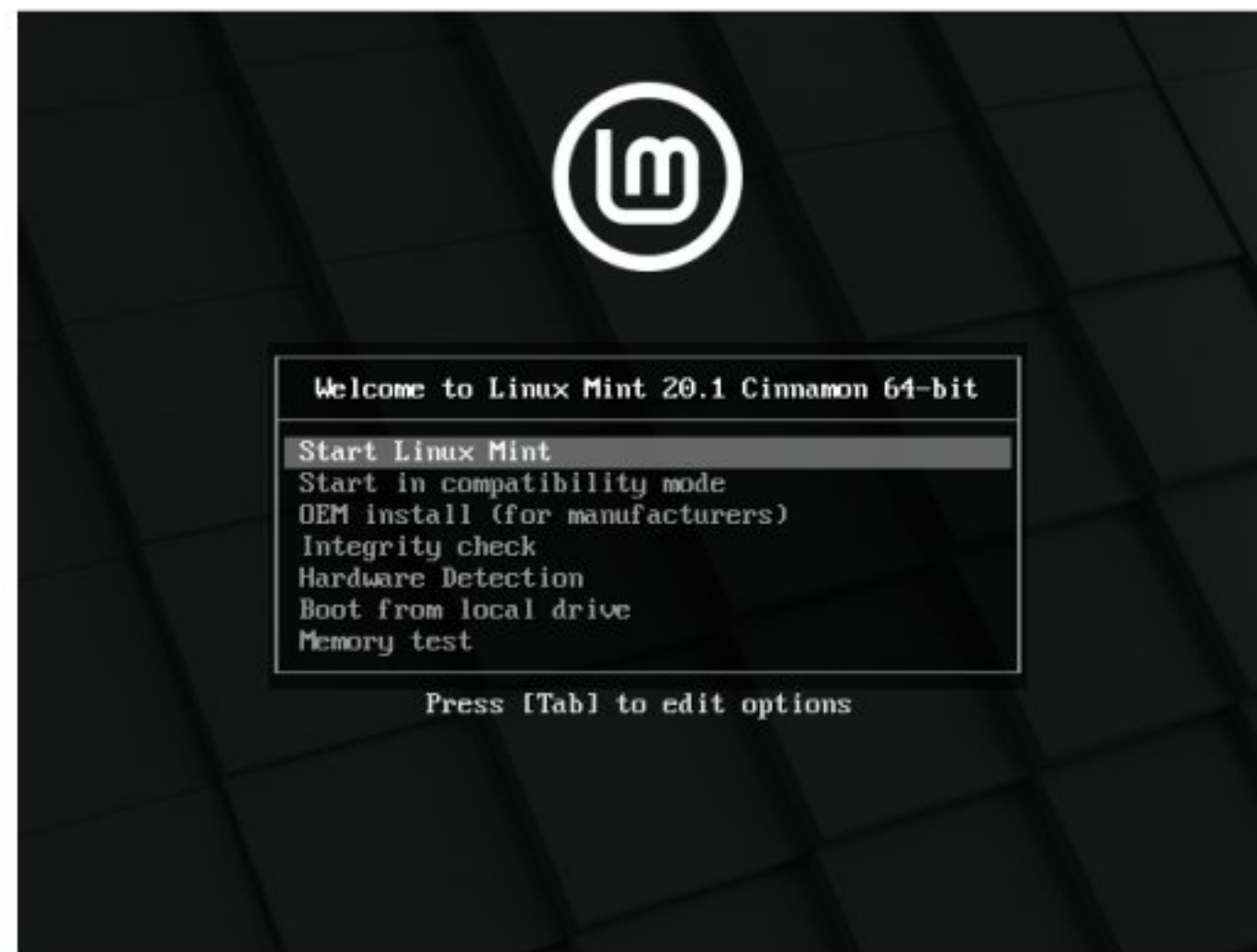


Figura A.6: Pantalla de inicio al arrancar (Fuente: Linux Mint)

Seleccionamos “Start Linux Mint”, para que la distribución inicie, una vez iniciada nos aparecerá el escritorio de la distribución en cuestión, el cual luce como se muestra en la Figura A.7.



Figura A.7: Escritorio de Linux Mint (Fuente: Linux Mint)

Una vez en el escritorio hacemos doble clic en el icono del CD que dice “Install Linux Mint”, se abrirá una ventana, en la cual iremos configurando nuestro sistema. Como se muestra en la Figura A.8.

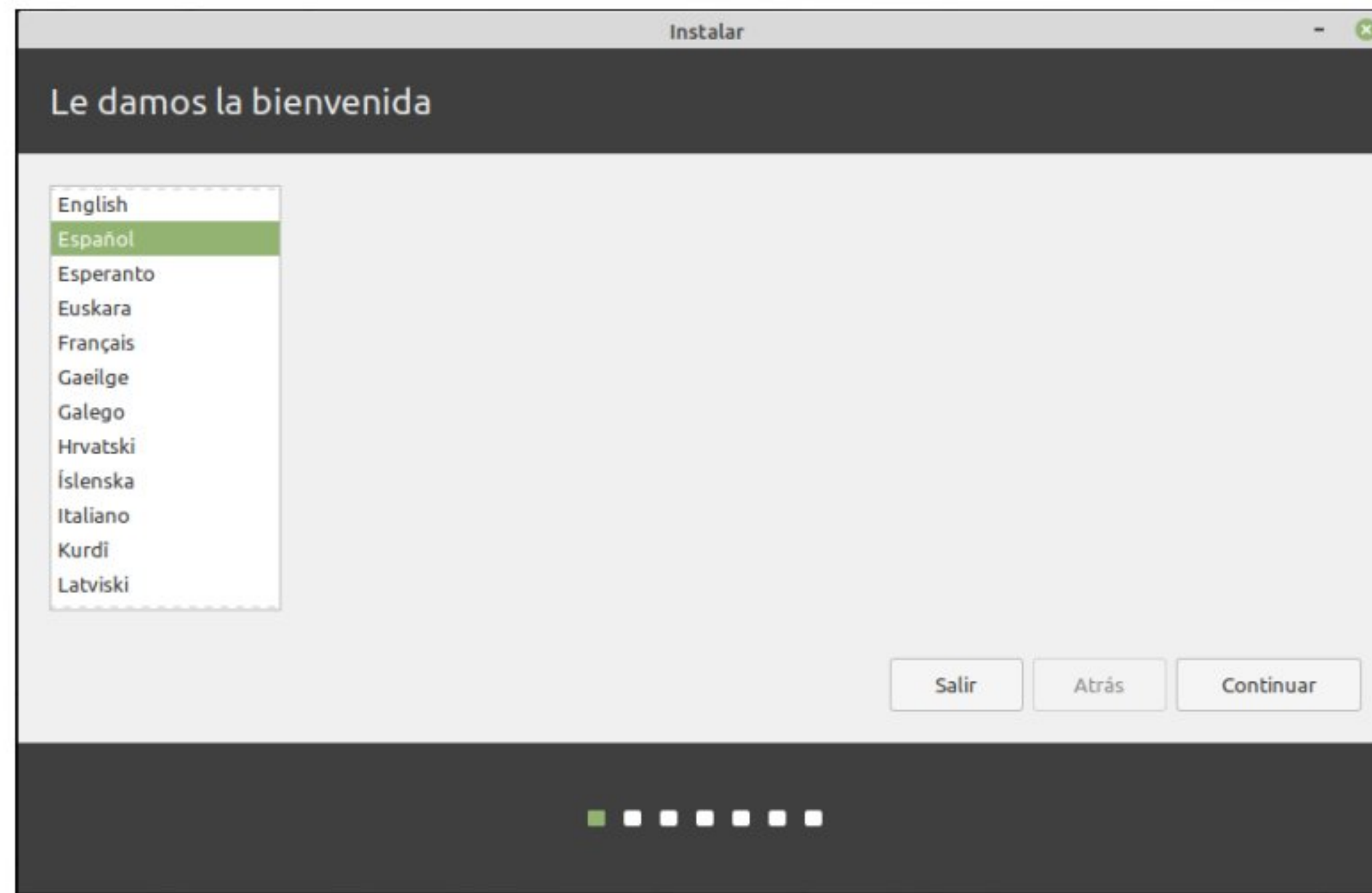


Figura A.8: Selección de idioma (Fuente: Linux Mint)

En el apartado anterior seleccionamos el idioma de nuestra preferencia, en este caso usaremos “Español”, y presionamos en el botón “Continuar”, seleccionamos la distribución de nuestro teclado y volvemos a oprimir “Continuar”, como se muestra en la Figura A.9.

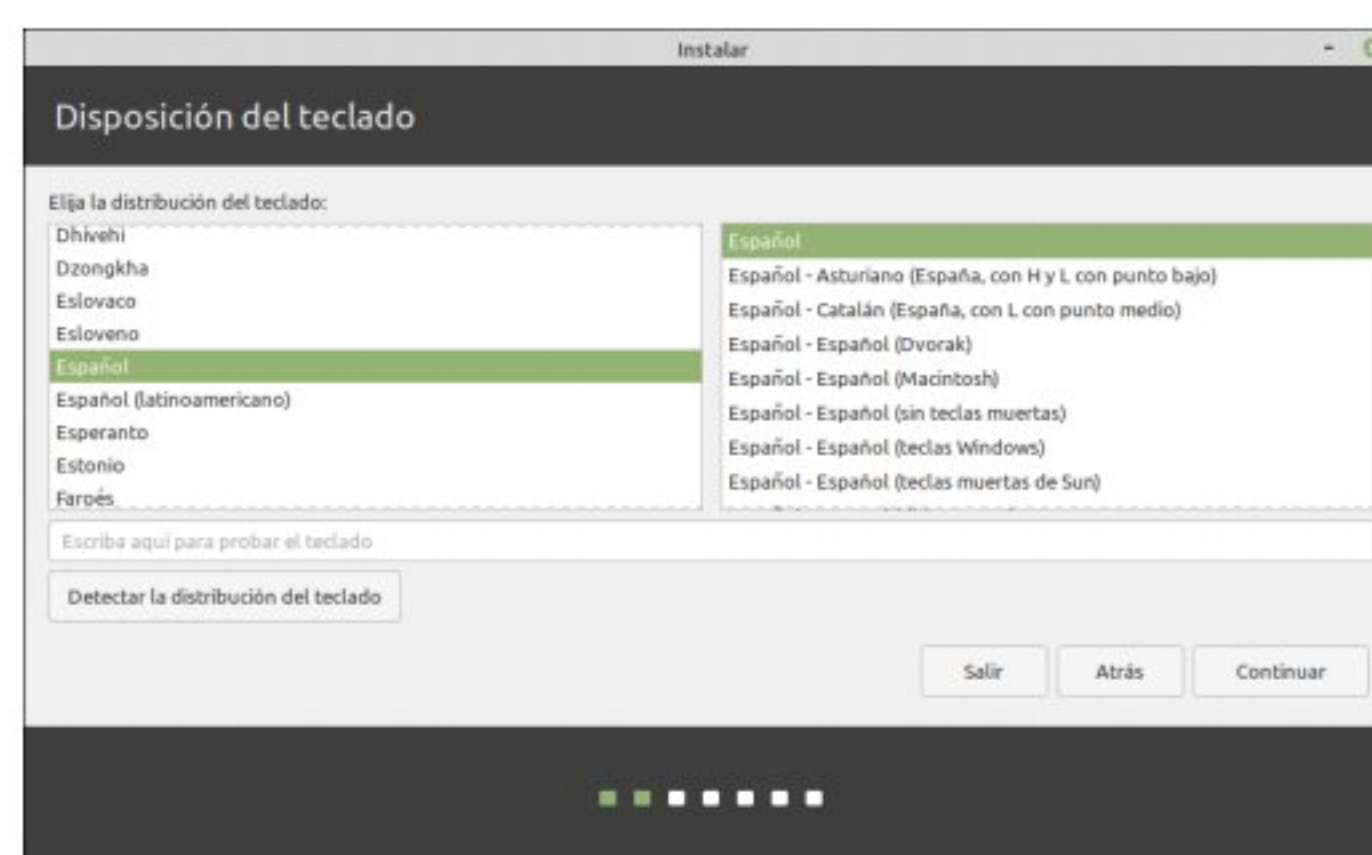


Figura A.9: Distribución de teclado (Fuente: Linux Mint)

En el apartado que se muestra en la Figura A.10, es importante seleccionar la casilla que dice “Instalar los códecs multimedia”, debido a que esto nos ahorrará tiempo después de la instalación, de no marcar dicha casilla conlleva tener que instalarlos manualmente después.

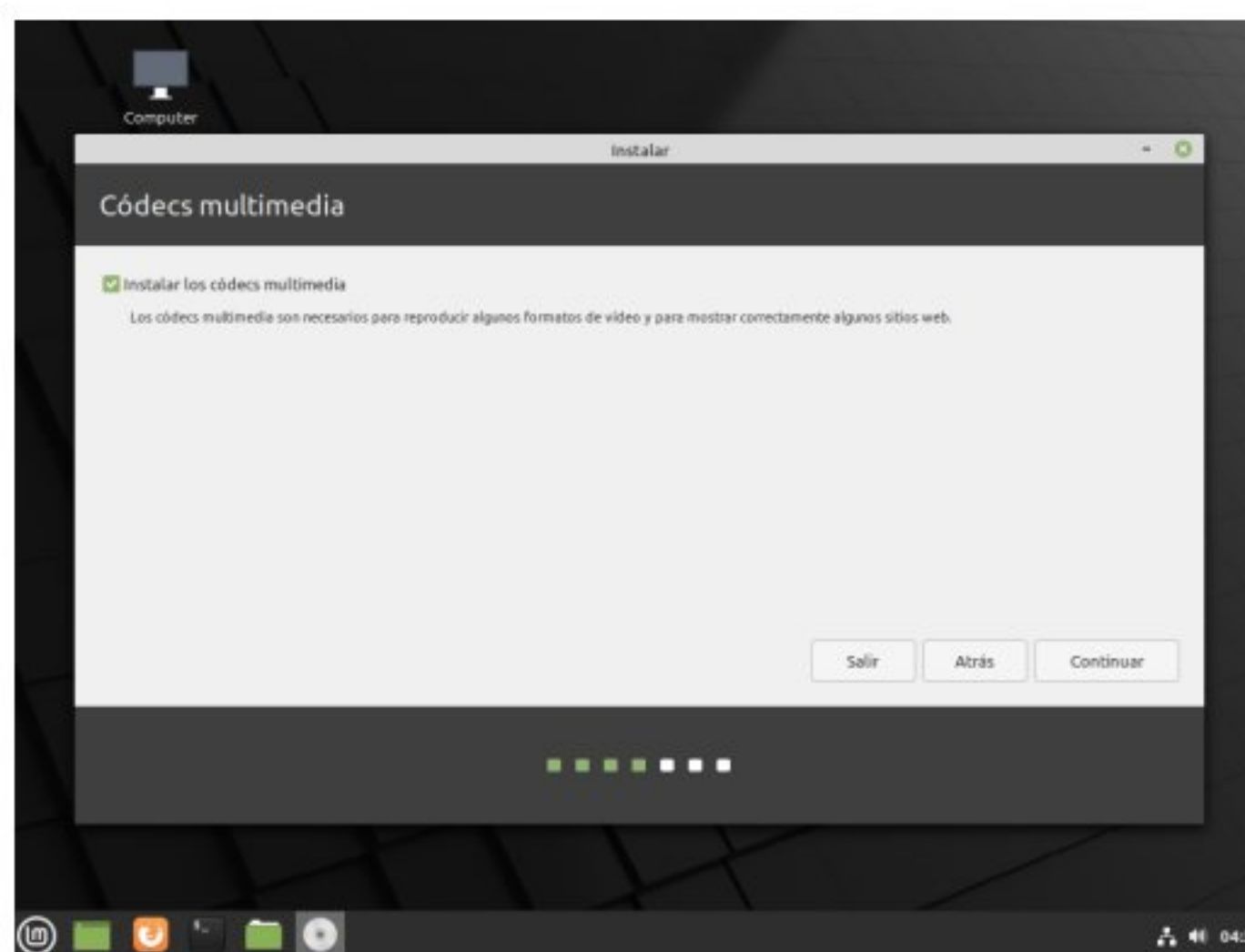


Figura A.10: Códecs multimedia (Fuente: Linux Mint)

Estos Códecs son necesarios para reproducir contenido multimedia como vídeo, audio, entre otros. Seleccionamos “Borrar disco e instalar Linux Mint” como se muestra en la Figura A.11, en caso de tener Windows instalado en el equipo, se recomienda investigar cómo crear un arranque dual para poder conservar ambos sistemas operativos.

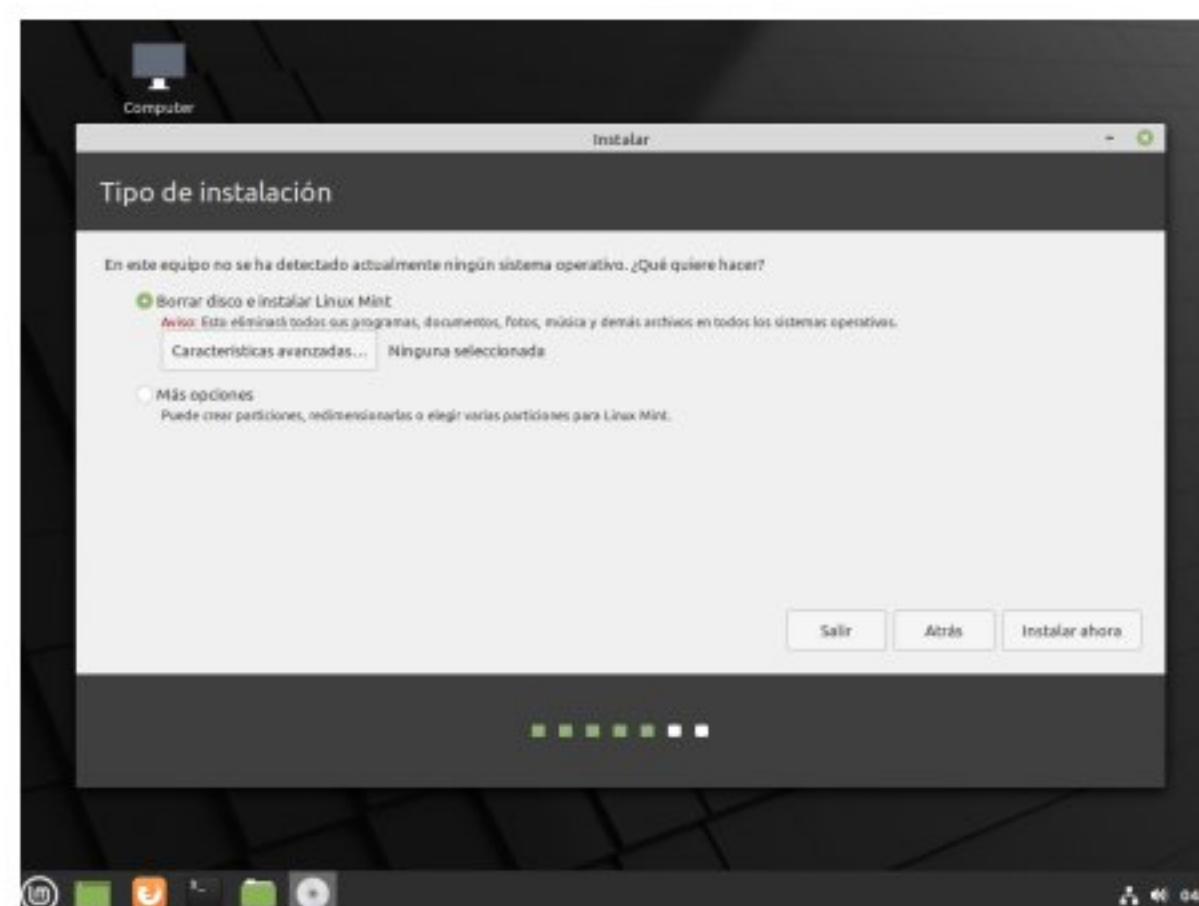


Figura A.11: Tipo de instalación (Fuente: Linux Mint)

Seleccionamos nuestra ubicación y presionamos “Continuar”, para después asignar un nombre y una contraseña a nuestro usuario como se muestra a continuación en la Figura A.12.

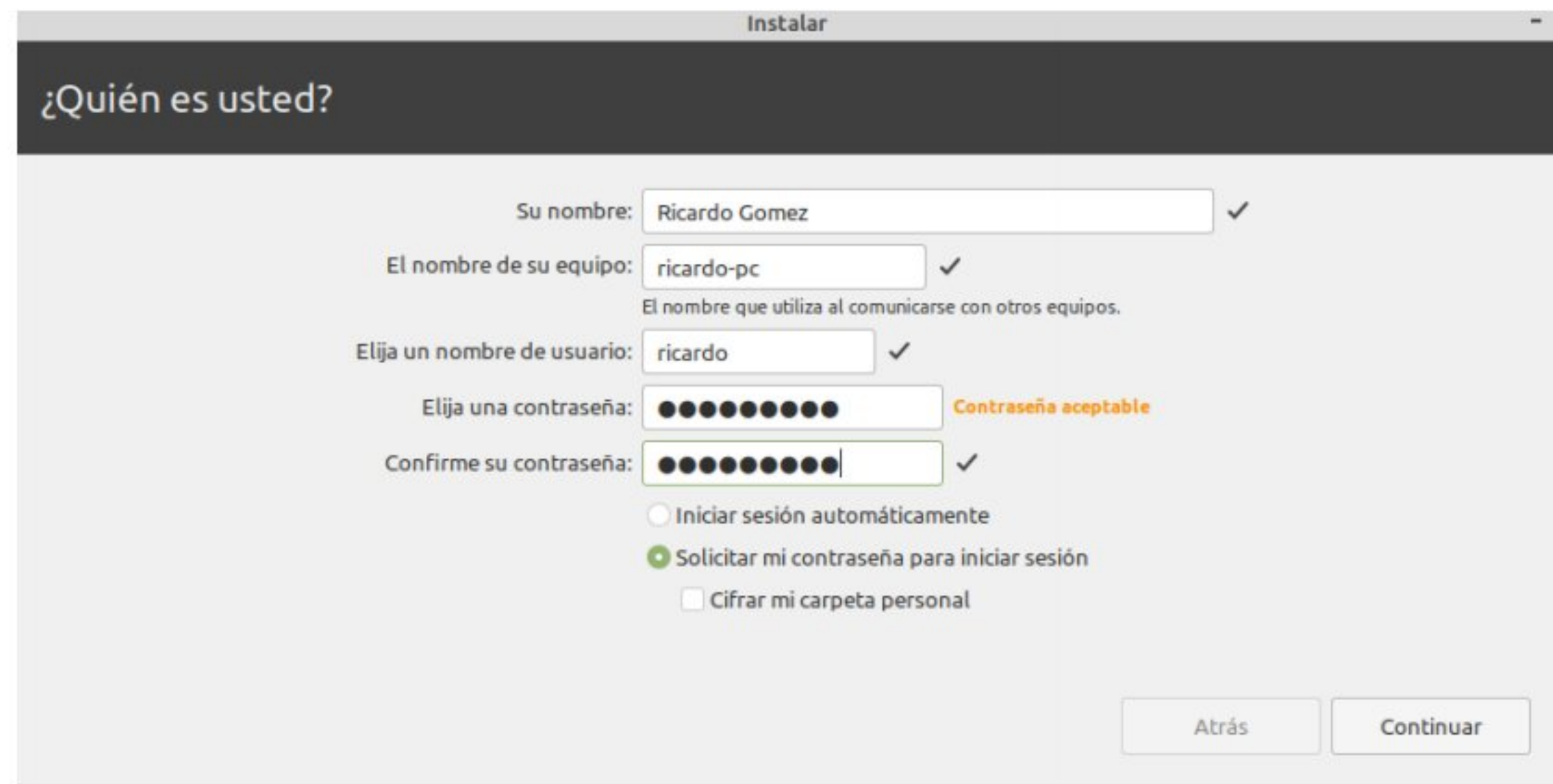


Figura A.12: Asignación de usuario y contraseña (Fuente: Linux Mint)

Una vez rellenos los campos presionamos en “Continuar” y el sistema comenzará a instalarse, como se muestra en la Figura A.13.

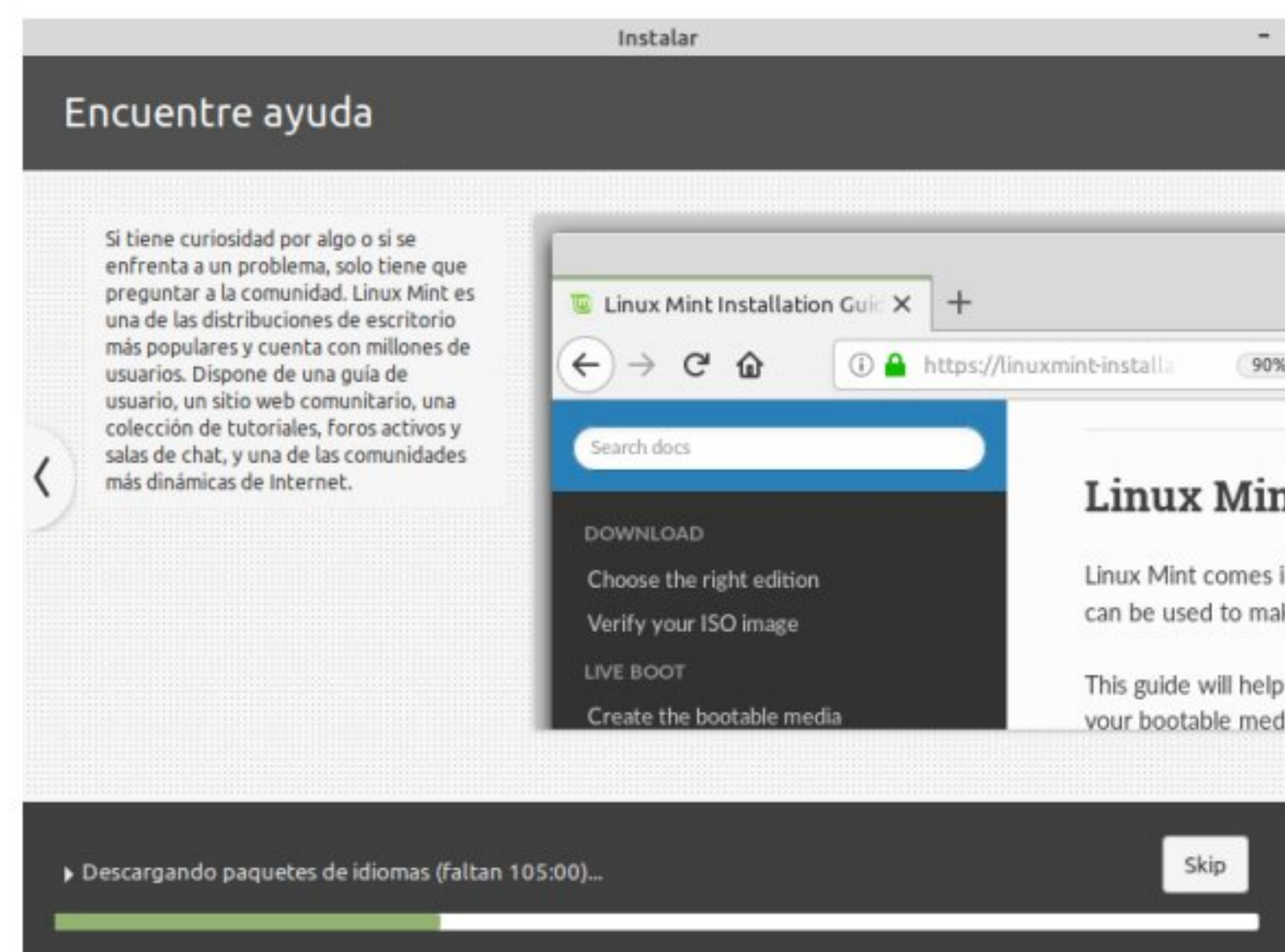


Figura A.13: Proceso de instalación (Fuente: Linux Mint)

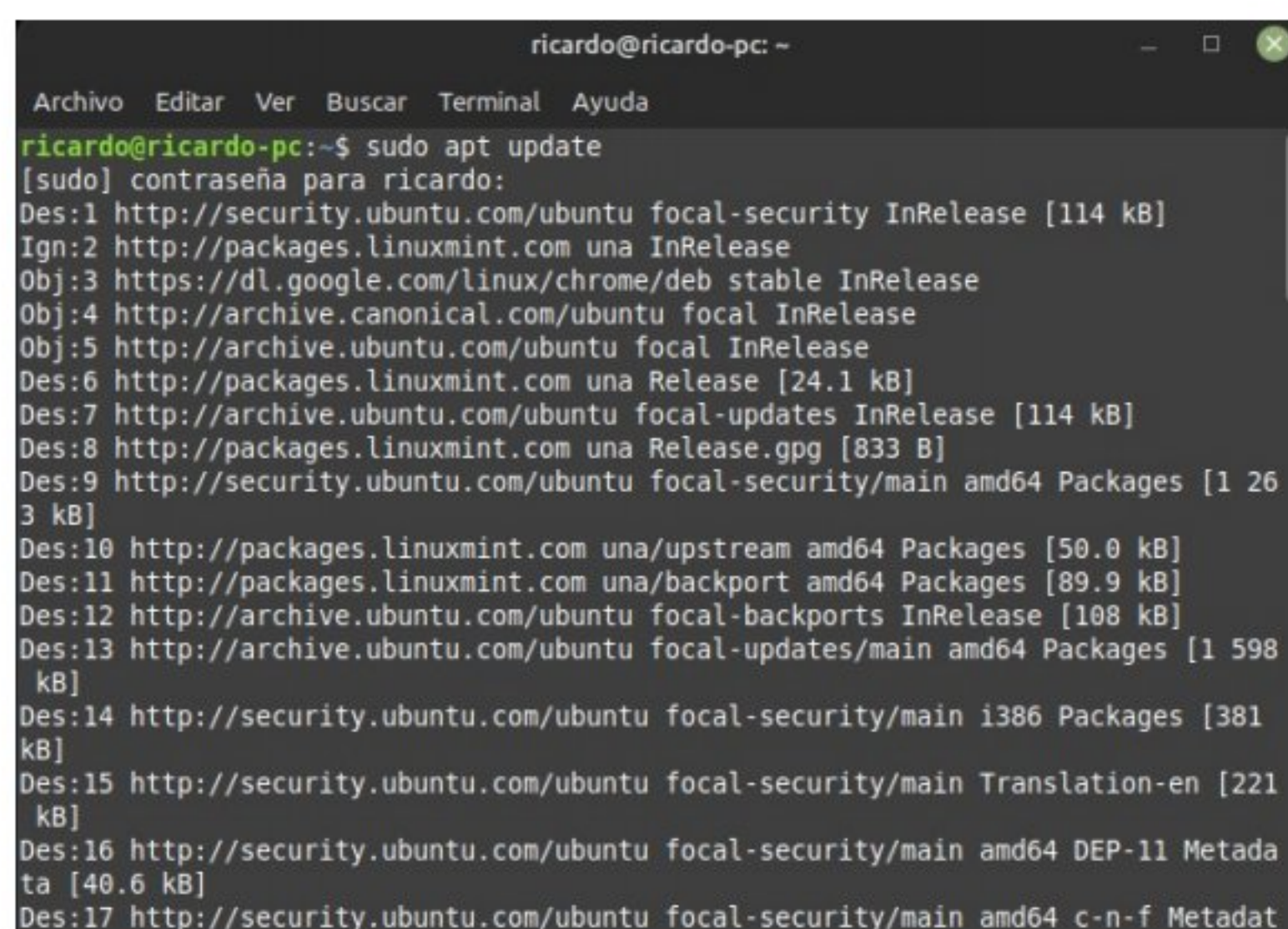
Una vez termine el proceso se nos pedirá reiniciar y ya tendremos Linux Mint instalado.

A.3 INSTALACIÓN Y CONFIGURACIÓN DE LINUX MINT

A lo largo del desarrollo se utilizó Linux Mint como sistema operativo principal, Linux Mint es una distribución GNU/Linux basada en Ubuntu que a su vez está basada en Debian, se caracteriza por ser una distribución fácil de configurar y fácil de utilizar, en el Anexo 1 se presenta el procedimiento llevado a cabo para la instalación y configuración del mismo.

A.3.1 CONFIGURACIÓN DE LINUX MINT PARA EL DESARROLLO DE LOS SCRIPTS

Para iniciar con la configuración ejecutamos una terminal oprimiendo “Ctrl+Alt+T” y en ella tecleamos ***sudo apt update***, se nos pedirá la contraseña del usuario que creamos durante la instalación, este comando actualizará los repositorios de software de nuestra distribución, en la terminal aparecerá el siguiente mensaje mostrado en la Figura A.14.



```
ricardo@ricardo-pc: ~
Archivo Editar Ver Buscar Terminal Ayuda
ricardo@ricardo-pc:~$ sudo apt update
[sudo] contraseña para ricardo:
Des:1 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Ign:2 http://packages.linuxmint.com una InRelease
Obj:3 https://dl.google.com/linux/chrome/deb stable InRelease
Obj:4 http://archive.canonical.com/ubuntu focal InRelease
Obj:5 http://archive.ubuntu.com/ubuntu focal InRelease
Des:6 http://packages.linuxmint.com una Release [24.1 kB]
Des:7 http://archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Des:8 http://packages.linuxmint.com una Release.gpg [833 B]
Des:9 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [1263 kB]
Des:10 http://packages.linuxmint.com una/upstream amd64 Packages [50.0 kB]
Des:11 http://packages.linuxmint.com una/backport amd64 Packages [89.9 kB]
Des:12 http://archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Des:13 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [1598 kB]
Des:14 http://security.ubuntu.com/ubuntu focal-security/main i386 Packages [381 kB]
Des:15 http://security.ubuntu.com/ubuntu focal-security/main Translation-en [221 kB]
Des:16 http://security.ubuntu.com/ubuntu focal-security/main amd64 DEP-11 Metadata [40.6 kB]
Des:17 http://security.ubuntu.com/ubuntu focal-security/main amd64 c-n-f Metadata
```

Figura A.14: Actualización de repositorios (Fuente: Linux Mint)

Cuando termine de ejecutarse en comando anterior ejecutamos *sudo apt upgrade*, el cual actualizará los paquetes instalados en nuestra distribución, al ejecutarlo se percibe un mensaje como el que se muestra en la Figura A.15 . Este par de comandos mantendrán nuestra distribución actualizada, con las ultimas versiones de software disponibles.

```

ricardo@ricardo-pc: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
cryptsetup cryptsetup-bin cryptsetup-initramfs fdisk firefox
firefox-locale-en firefox-locale-es gir1.2-javascriptcoregtk-4.0
gir1.2-nm-1.0 gir1.2-polkit-1.0 gir1.2-webkit2-4.0 google-chrome-stable
initramfs-tools initramfs-tools-bin initramfs-tools-core language-pack-en
language-pack-en-base libarchive13 libbluetooth3 libcryptsetup12
libcscreensaver0 libdrm-amdgpu1 libdrm-common libdrm-intel1 libdrm-nouveau2
libdrm-radeon1 libdrm2 libegl-mesa0 libexpat1 libexpat1-dev libfdisk1
libgbm1 libgl1-mesa-dri libglapi-mesa libglx-mesa0
libjavascriptcoregtk-4.0-18 libmount1 libmysqlclient21 libnm0
libpolkit-agent-1-0 libpolkit-gobject-1-0 libsmartcols1 libsmbclient
libspeex1 libspeexdsp1 libuuid1 libwbclient0 libwebkit2gtk-4.0-37
libxatracker2 linux-firmware linux-generic linux-headers-generic
linux-image-generic linux-libc-dev mesa-va-drivers mesa-udpau-drivers
mesa-vulkan-drivers mount network-manager
network-manager-config-connectivity-ubuntu policykit-1 python-apt-common
python3-apt python3-samba rfc802 samba-common samba-common-bin samba-libs
smbclient sublime-text thunderbird thunderbird-gnome-support
thunderbird-locale-en thunderbird-locale-en-us thunderbird-locale-es
ubuntu-drivers-common util-linux uuid-runtime vim-common vim-tiny
xul-ext-lightning xxd
9 actualizados, 5 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 488 MB de archivos.
Se utilizarán 503 MB de espacio de disco adicional después de esta operación.
Desea continuar? [S/n]

```

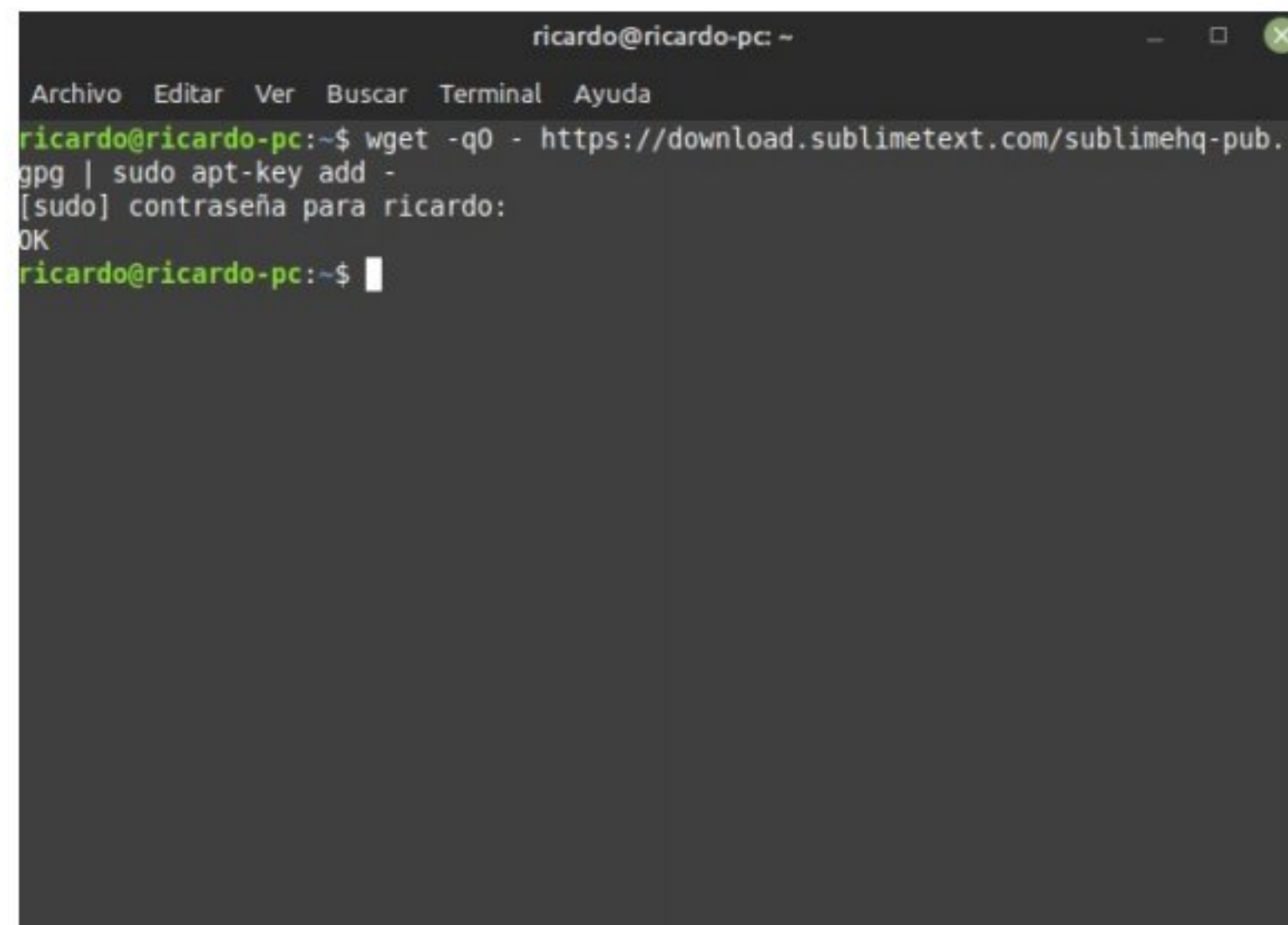
Figura A.15: Actualización de de los paquetes (Fuente: Linux Mint)

Hecho esto, procedemos a instalar *Sublime Text*, que es un editor de texto y sera de gran utilidad para crear nuestros scripts.

Añadimos la clave GPG con el siguiente comando:

```
wget -qO - https://download.sublimetext.com/sublimehq-pub.gpg | sudo apt-key
add -
```

Al introducir el comando anterior se muestra el contenido de la Figura A.16.



```
ricardo@ricardo-pc: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
ricardo@ricardo-pc:~$ wget -qO - https://download.sublimetext.com/sublimehq-pub.  
gpg | sudo apt-key add -  
[sudo] contraseña para ricardo:  
OK  
ricardo@ricardo-pc:~$
```

Figura A.16: Clave GPG (Fuente: Linux Mint)

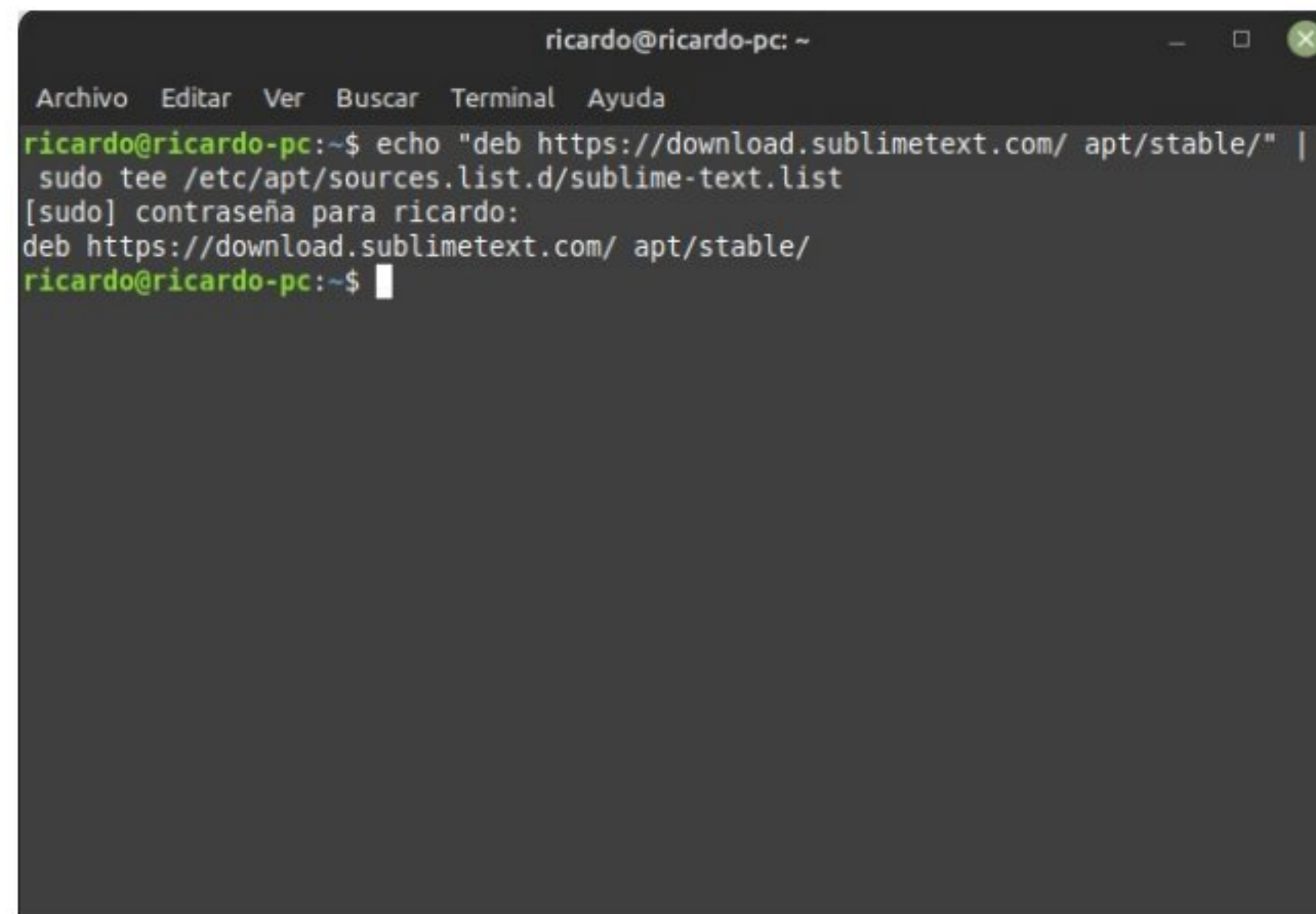
Después nos aseguramos de que nuestro administrador de paquetes esta configurado para trabajar con fuentes https:

```
sudo apt-get install apt-transport-https
```

Saldrá un mensaje donde se nos pregunta si deseamos instalar oprimimos **S** y damos enter.

Tecleamos el siguiente comando para agaragar el repositorio a nuestra lista de fuentes, el resultado es mostrado en la Figura A.17.

```
echo "deb https://download.sublimetext.com/ apt/stable/" |  
sudo tee /etc/apt/sources.list.d/sublime-text.list
```

A terminal window titled 'ricardo@ricardo-pc: ~' with a menu bar (Archivo, Editar, Ver, Buscar, Terminal, Ayuda). The terminal shows the following commands and output:

```
ricardo@ricardo-pc:~$ echo "deb https://download.sublimetext.com/ apt/stable/" |
sudo tee /etc/apt/sources.list.d/sublime-text.list
[sudo] contraseña para ricardo:
deb https://download.sublimetext.com/ apt/stable/
ricardo@ricardo-pc:~$
```

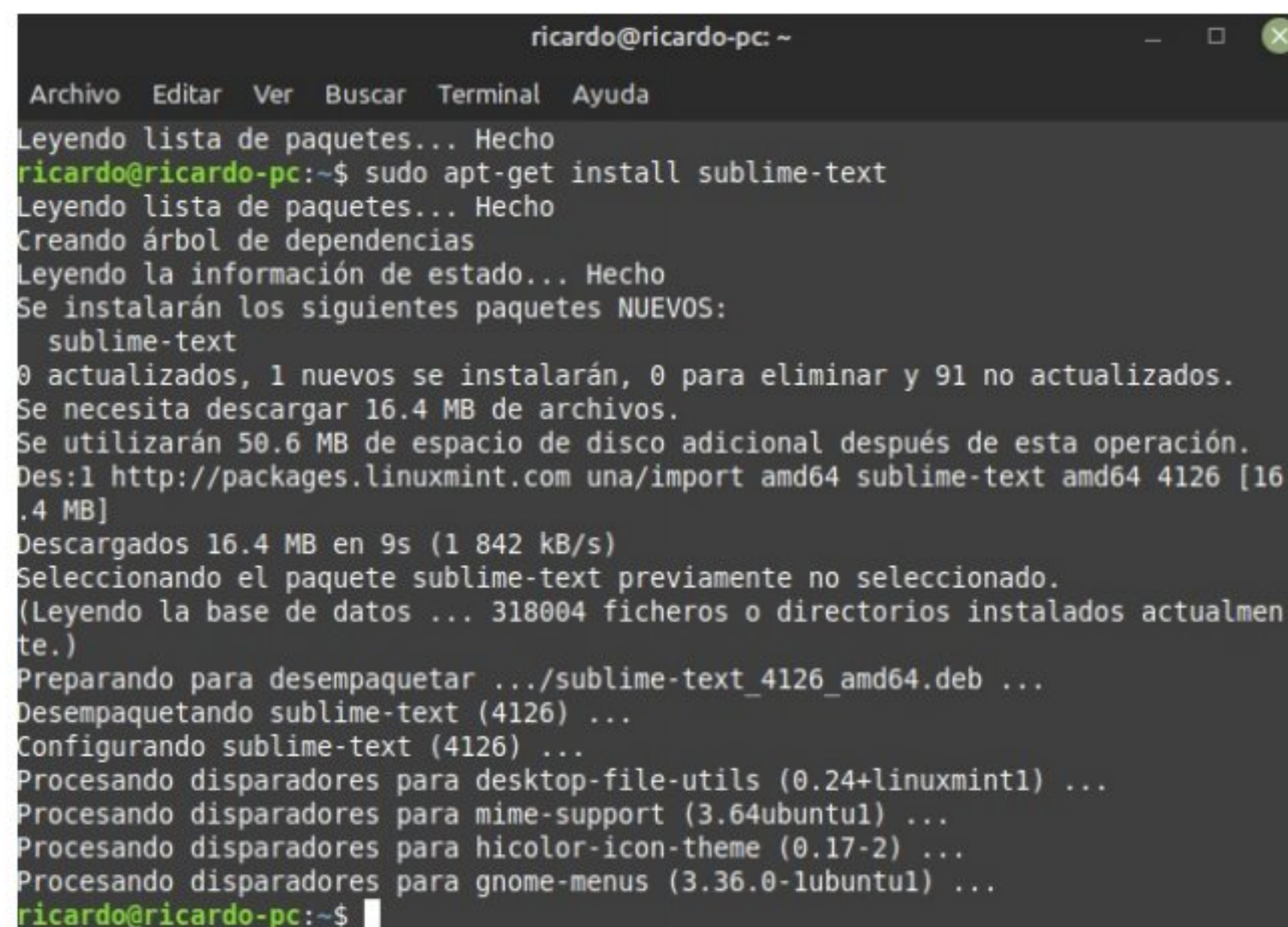
Figura A.17: Integración del nuevo repositorio (Fuente: Linux Mint)

Por ultimo actualizamos nuestros repositorios de software e instalamos *Sublime Text* con los siguientes comandos:

sudo apt-get update

sudo apt-get install sublime-text

El resultado de este comando se muestra en la Figura A.18.

A terminal window titled 'ricardo@ricardo-pc: ~' with a menu bar (Archivo, Editar, Ver, Buscar, Terminal, Ayuda). The terminal shows the output of the command 'sudo apt-get install sublime-text':

```
Leyendo lista de paquetes... Hecho
ricardo@ricardo-pc:~$ sudo apt-get install sublime-text
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes NUEVOS:
  sublime-text
0 actualizados, 1 nuevos se instalarán, 0 para eliminar y 91 no actualizados.
Se necesita descargar 16.4 MB de archivos.
Se utilizarán 50.6 MB de espacio de disco adicional después de esta operación.
Des:1 http://packages.linuxmint.com una/import amd64 sublime-text amd64 4126 [16
.4 MB]
Descargados 16.4 MB en 9s (1 842 kB/s)
Seleccionando el paquete sublime-text previamente no seleccionado.
(Leyendo la base de datos ... 318004 ficheros o directorios instalados actualmen
te.)
Preparando para desempaquetar .../sublime-text_4126_amd64.deb ...
Desempaquetando sublime-text (4126) ...
Configurando sublime-text (4126) ...
Procesando disparadores para desktop-file-utils (0.24+linuxmint1) ...
Procesando disparadores para mime-support (3.64ubuntu1) ...
Procesando disparadores para hicolor-icon-theme (0.17-2) ...
Procesando disparadores para gnome-menus (3.36.0-1ubuntu1) ...
ricardo@ricardo-pc:~$
```

Figura A.18: Instalación Sublime Text (Fuente: Linux Mint)

A.4 ANEXO 2

En este anexo se proporcionan los enlaces a los repositorios:

Repositorio de la página:

<https://github.com/trinat00/PW-Scripts-LIA>

Repositorio de los scripts:

<https://github.com/Ricardo-Gomez-4-113/AutoLAMP>

BIBLIOGRAFÍA

- [1] I. Díaz, “Granjas verticales: una respuesta sostenible al crecimiento urbano,” vol. 7, no. 1, pp. 3–6. Number: 1.
- [2] F. A. Urbano-Molano, “Redes de sensores inalámbricos aplicadas a optimización en agricultura de precisión para cultivos de café en Colombia,” p. 7.
- [3] FAO, *FAO publications catalogue 2021: April*. FAO.
- [4] J. C. Suárez, F. F. Salazar, I. F. Nava, and R. H. Hernández, “Industry 4.0 and digital manufacturing: a design method applying reverse engineering,” vol. 24, no. 1, pp. 6–28.
- [5] R. Muddinagiri, S. Ambavane, and S. Bayas, “Self-Hosted Kubernetes: Deploying Docker Containers Locally With Minikube,” *2019 International Conference on Innovative Trends and Advances in Engineering and Technology (ICITAET)*, 2019.
- [6] A. Khanna and S. Kaur, “Evolution of Internet of Things (IoT) and its significant impact in the field of Precision Agriculture,” *Computers and Electronics in Agriculture*, vol. 157, pp. 218–231, Feb. 2019.
- [7] A. Čolaković and M. Hadžialić, “Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues,” *Computer Networks*, vol. 144, pp. 17–39, Oct. 2018.
- [8] CEPAL, “Tecnologías digitales para un nuevo futuro,” p. 99, 2022.
- [9] G. Nazareno, “Virtualización de servidores. conceptos básicos,” Accepted: 2018-09-25T19:09:49Z.

- [10] C. T. Lizarraga, V. M. Baeza, and C. M. M. Sánchez, “Análisis de las necesidades de una smart city en el marco del desarrollo sostenible,” p. 86, 2020.
- [11] V.-V. C. Daniel, M.-M. Patricia, and P.-H. Javie, “Virtualización de sistemas gestores de cómputo en las organizaciones.,” p. 10, 2020.
- [12] F. Ameijeiras and J. Ángel, “Orquestación de aplicaciones corporativas mediante virtualización de contenedores usando docker,” p. 94, 2016.
- [13] F. Balderas Guzmán, “Método para la supervisión de sistemas de contenedores virtuales basado en multi-modelado,” Master’s thesis, Tesis (MC)–Centro de Investigación y de Estudios Avanzados del IPN Unidad . . . , 2020.
- [14] J. M. Ortega, *DOCKER. Seguridad y monitorización en contenedores e imágenes*.
- [15] M. G. Castillo Ramírez and L. M. Muñoz Valencia, *Implementación de servicios con la tecnología de contenedores PODMAN*. Servicio Nacional de Aprendizaje (SENA). Accepted: 2021-11-23T20:58:04Z.
- [16] J. Nogués García, “Orquestación de contenedores con kubernetes,” B.S. thesis, 2018.
- [17] L. E. G. López and C. A. G. Alarcón, “Extensión de la arquitectura docker para el despliegue automático de contenedores,” no. 29, pp. 11–26. Number: 29.
- [18] J. L. Pacheco Laje, *Estudio comparativo entre una arquitectura con microservicios y contenedores dockers y una arquitectura tradicional (monolítica) con comprobación aplicativa*. PhD thesis, Universidad de Guayaquil. Facultad de Ciencias Matemáticas y Físicas . . . , 2018.
- [19] L. Leite, F. Kon, G. Pinto, and P. Meirelles, “Building a theory of software teams organization in a continuous delivery context,” in *2020 IEEE/ACM 42nd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pp. 296–297, IEEE, 2020.
- [20] E. G. F. Castro, “Implementación de una base de datos heterogénea distribuida entre los sgbd’s oracle, mysql y postgresql con replicación, mediante un script bash implementado en el sistema operativo centos usando software libre,” *INNOVA Research Journal*, vol. 3, no. 2.1, pp. 59–66, 2018.

-
- [21] F. J. Luquin Oroz, “Automatización y configuración de una red de área local en entorno gnu/linux,” 2010.
- [22] J. Ribes Rodriguez Moldes, *Automatización del despliegue, configuración y administración de servicios en GNU/Linux en sistemas de bajo coste*. PhD thesis, Universitat Politècnica de València, 2019.
- [23] I. de Fez Lava, P. Arce Vila, R. Belda Ortega, and J. C. Guerri Cebollada, “Uso de contenedores docker en el entorno educativo,” 2021.
- [24] J. P. Lazarte Mendez, *CONTENEDORES DOCKER COMO ESTRATÉGIA DE VIRTUALIZACION*. PhD thesis, 2019.
- [25] P. C. Guale González, “Implementación de una pantalla informativa en el laboratorio de informática de la UPSE: Modulo medio de difusión mediante raspberry,” Accepted: 2021-03-03T04:37:53Z Publisher: La Libertad: Universidad Estatal Península de Santa Elena, 2021.
- [26] Debian.org, “Debian – acerca de debian [online],” (2022, Feb 10).
- [27] R. Hertzog and R. Mas, “El libro del administrador de debian,” 2013.
- [28] M. Corporation’s, “HTTP | MDN [online],” 2022, Feb 11.
- [29] T. A. S. Foundation, “Welcome! - the apache HTTP server project [online],” (2022, Feb 11).
- [30] L. A. C. Santillán, M. G. Ginestà, and Ó. P. Mora, “Bases de datos en mysql,” *Universitat oberta de Catalunya*, 2014.
- [31] R. C. Casado Vara, *Introducción a HTML*. Ediciones Universidad de Salamanca (España). Accepted: 2019-07-10T08:33:13Z.
- [32] M. P. Duarte and I. M. Pérez, “Programación en php a través de ejemplos,” *Recuperado el*, 2007.
- [33] R. O. DIAZ, “UNA MIRADA HACIA EL INTERIOR DE UNA COMPUTADORA PERSONAL.,”
- [34] M. Corporation’s, “About JavaScript - JavaScript | MDN [online],” (2022, Abr 19).

-
- [35] hostinger.es, “¿qué es CSS? [online,” (2022, Abr 19).
 - [36] J. P. A. Torres and L. F. C. Guarín, “Implementación de plataforma web de reservas de espacios deportivos, usando desarrollo basado en pruebas TDD.,” p. 40.
 - [37] I. V. Arroniz, “Agricultura de precisión, la nueva alternativa sustentable (opinión) [online],” (2022, Jun 15).

RESUMEN AUTOBIOGRÁFICO

Ricardo Guadalupe Gomez Martínez

Candidato para obtener el grado de
Licenciado en Ingeniería en Tecnologías de la Información y las Comunicaciones

Instituto Tecnológico de Pabellón de Arteaga.
TecNM Campus Pabellón de Arteaga

SCRIPTS QUE AUTOMATIZAN LA CREACIÓN DEL ENTORNO LAMP PARA LA INSTALACIÓN DE REDES DE SENSORES EN GRANJAS VERTICALES (VERTICAL FARMS)

Ricardo Guadalupe Gomez Martínez nació el 24 de Noviembre de 1999 en el municipio de Jesús María, Aguascalientes y su lugar de residencia es la comunidad de Gorriones, siendo el hijo mayor de Leonardo Gomez Llamas y Elvira Martínez Jiménez, concluyó su educación media superior en el Telebachillerato Comunitario de Gorriones, la institución en cuestión no contaba con instalaciones propias y ni con los suficientes materiales para trabajar, las clases eran impartidas en una bodega proporcionada por el comisario ejidal, pese a esto, Ricardo destacó por su iniciativa de aprender de manera autónoma e independiente.