MIGRACIÓN DEL SOFTWARE KACTUS-HCM DE UNA ARQUITECTURA CLIENTE-SERVIDOR A UNA ARQUITECTURA CLIENTE-CONTENEDOR

AUTOR

JESUS LEONARDO GARCIA HOYOS

UNIVERSIDAD DE PAMPLONA
INGENIERÍA EN TELECOMUNICACIONES
DEPARTAMENTO ELECTRÓNICA, ELÉCTRICA, SISTEMAS Y TELECOMUNICACIONES
FACULTAD DE INGENIERÍAS Y ARQUITECTURA
PAMPLONA, NORTE DE SANTANDER
2019

MIGRACIÓN DEL SOFTWARE KACTUS-HCM DE UNA ARQUITECTURA CLIENTE-SERVIDOR A UNA ARQUITECTURA CLIENTE-CONTENEDOR

AUTOR

JESUS LEONARDO GARCIA HOYOS

TRABAJO DE GRADO PRESENTADO COMO REQUISITO PARA OPTAR AL TÍTULO DE INGENIERO EN TELECOMUNICACIONES

DIRECTOR

ING. HERNANDO JOSE VELANDIA VILLAMIZAR

CODIRECTOR

ING. EDWIN GIOVANNY BARRETO LIZCANO

UNIVERSIDAD DE PAMPLONA
INGENIERÍA EN TELECOMUNICACIONES
DEPARTAMENTO ELECTRÓNICA, ELÉCTRICA, SISTEMAS Y TELECOMUNICACIONES
FACULTAD DE INGENIERÍAS Y ARQUITECTURA
PAMPLONA, NORTE DE SANTANDER
2019

AGRADECIMIENTOS

Agradezco a Dios por permitirme vivir esta experiencia durante estos 5 años que han sido vitales para mi carrera profesional, por brindarme la sabiduría, constancia y paciencia para poder alcanzar esta meta.

Agradezco también a mi familia y a mi novia que han sido unas personas importantes en este camino de formación profesional, gracias por brindarme todo su apoyo y todo el acompañamiento brindado.

Agradezco a todos mis profesores por todos los detalles que aportaron a mi formación profesional, por todo el conocimiento impartido y los buenos consejos que me brindaron, igualmente agradecer a mi grupo de estudio (Natalia, Angela, Salvador).

Por último, quisiera agradecer a la empresa **Digital Ware** que me abrió las puertas para realizar las practicas, de la misma forma quiero agradecer al Ingeniero Edwin Barreto que depositó su confianza en mí para el nuevo reto que emprendía en la empresa.

A todos y cada una de las personas que hicieron parte de lo que hoy es para mí alcanzar este logro, decirles, ¡Gracias!.

“A Dios por brindarme la sabiduría,
a mis padres, mis hermanos y
mi novia, por toda la confianza y
acompañamiento”.

ÍNDICE

FIGURAS

TABLAS

RESUMEN

El presente proyecto se ha llevado a cabo en la empresa Digital Ware con la colaboración del área de tecnología y del Ingeniero Edwin Barreto, este consiste en realizar la migración de uno de los productos de la compañía, dicho producto cuenta actualmente con una arquitectura de cliente-servidor, y la misión es cambiar el servidor físico o virtualizado, por un contenedor de software que contenga todo lo necesario para realizar el despliegue de las aplicaciones. Lo cual acarrea una investigación, puesto que, es una tecnología emergente, por tanto, se deberá comprender su funcionamiento y conocer las oportunidades que esta brinda. El desarrollo de este se ejecutará en un ambiente de pruebas, además, con este trabajo se pretende brindar a la empresa una herramienta que le permita disminuir los costos en la infraestructura y disminuir los tiempos de despliegue de las aplicaciones, ya sea para ambientes productivos o ambientes de pruebas.

Palabras claves: Virtualización, Software, Cliente, Servidor, Contenedor.

ABSTRACT

The present project has been carried out in the company Digital Ware with the collaboration of the technology area and Engineer Edwin Barreto, this is to perform the migration of one of the products of the company, this product currently has a client-server architecture, and the mission is to change the physical or virtualized server for a software container containing everything necessary to perform the deployment of applications. This entails an investigation, since it is an emerging technology, therefore, its functioning must be understood and the opportunities it offers must be known. The development of this will be executed in a test environment, in addition, this work is intended to provide the company a tool that allows you to reduce infrastructure costs and reduce the deployment times of applications, either for productive environments or test environments.

Keywords: Virtualization, Software, Client, Server, Container.

INTRODUCCIÓN

La tecnología ha estado avanzando a pasos agigantados, cuando en un pasado surgió la virtualización, se presentó como una solución muy buena para las empresas, pues esta disminuía mucho los costos, pero con el pasar del tiempo se ha visto que ya no es una solución tan económica y efectiva, ya que estos ambientes de virtualización en servidores hacen que se necesite un mayor espacio físico, mayores recursos y mayor tiempo de ejecución de tareas y despliegues de las aplicaciones, debido a que estas cada vez son más complejas.

Esa evolución diaria que se da en el ámbito tecnológico hace que se exploren nuevas formas de facilitar la vida a las personas y a las empresas, estas innovaciones buscan una reducción en los tiempos de ejecución de los procesos, dicho progreso hace que unos años atrás haya tomado fuerza la idea de los contenedores de software, basándose en el concepto de los contenedores de Linux, esta tecnología permite ejecutar aplicaciones encapsulados ejecutándose sobre un sistema operativo, además este sistema brinda facilidad de uso y eficiencia. La arquitectura de contenedores permite trabajar topologías de red físicas o virtuales u orquestarlas como una topología de red mixta. Por esto, se propone una migración de arquitectura actual de cliente-servidor a una arquitectura cliente-contenedores, debido que es una tecnología nueva, eficiente, economizadora, fiable y rápida.

Para el desarrollo del presente trabajo se hizo uso de Docker en el desarrollo de los contenedores, ya que es una herramienta que cuenta con una gran cantidad de imágenes en el Docker Hub, de igual forma esta soportada por una gran comunidad de desarrolladores.

1. PROBLEMA

1.1. JUSTIFICACIÓN

Digital Ware es una compañía especializada Software ERP (*Enterprise Resource Planning, Planificador de recursos empresariales),* Software de Nómina y Gestión Humana y Software para IPS (*Instituciones Prestadoras de Servicios*) y Clínicas, lleva más de 25 años al servicio de la comunidad y actualmente es una de las más grandes empresas en el sector de la tecnología en Colombia además una de las más innovadoras, por ser una empresa destacada en este ámbito y por querer estar siempre a la vanguardia de la tecnología la empresa invierte una cantidad considerable en infraestructura tecnológica *(router, switch, servidores, host, etc.),* también el despliegue de sus aplicaciones en un proyecto externo puede tardar mucho tiempo, debido a que en el momento de la implantación se debe contemplar diferentes configuraciones y parametrización de los productos. Entonces, ¿De qué manera se podría agilizar los despliegues de las aplicaciones y a su vez reducir los costos de infraestructura a la compañía?

Una muy buena solución a este tipo de inconvenientes anteriormente mencionados es la migración de los productos a un entorno de cliente-contenedor ya que con esto las aplicaciones podrán ser más portables, además se podrían contar con diferentes contenedores parametrizados según sea la necesidad de implementación que tenga la empresa, esto permitiría que los tiempos de despliegues de los proyectos se reduzcan.

Un contenedor puede alojar todas las herramientas que la aplicación necesita para desplegarse. También en un ambiente de cliente-contenedor se puede manejar una infraestructura bajo código, es decir que los dispositivos de red son simulados y ejecutados en un contenedor, por ende, los costos de infraestructura disminuirán y la funcionalidad de esta estará a la distancia de un solo clic.

1.2. DELIMITACIÓN

1.2.1. OBJETIVOS

1.2.1.1. OBJETIVO GENERAL

- Migrar el software KACTUS-HCM de una arquitectura cliente-servidor a una arquitectura cliente-contenedor por medio de la implementación de contenedores sobre Docker con el fin de agilizar los despliegues y disminuir los costes de infraestructura.

1.2.1.2. OBJETIVOS ESPECÍFICOS

- Realizar revisión documental sobre la arquitectura actual de KACTUS-HCM con apoyo de textos oficiales que la describen y así poder diseñar la arquitectura de contenedores para el software.

- Analizar las herramientas de contenerización a implementar sobre el sistema operativo Windows para el desarrollo de la arquitectura de contenedores.

- Implementar KACTUS-HCM sobre un ambiente de contenedores.

- Validar funcionalmente la migración de KACTUS-HCM mediante pruebas y comparaciones contra la arquitectura de servidores virtualizados.

2. MARCO INSTITUCIONAL

La práctica empresarial fue desarrollada en la empresa DIGITAL WARE en la ciudad de Bogotá con una duración de 6 meses, tiempo ocupado entre el 07 de junio de 2018 hasta el 6 de Diciembre de 2018.

2.1. DIGITAL WARE:

Digital Ware según lo define su página Web Oficial es:

> Empresa del sector de tecnología especializada en *Software ERP, Software* de Nómina y Gestión Humana y *Software* para IPS y Clínicas, con más de 20 años en el mercado, líder en diseño e implantación de soluciones empresariales en las áreas de RRHH, Finanzas, Logística, Manufactura, Seguridad, Petróleos, Energía, Cajas de Compensación, Gobierno, Educación y Salud.
>
> Ganadora de varios premios, entre ellos la Orden de la Democracia SIMÓN BOLÍVAR en el grado de Cruz Comendador otorgada por el Congreso de la República, y la Medalla al Mérito de las Comunicaciones MURILLO TORO otorgada por el Ministerio de Tecnologías de la información y las comunicaciones, además del Premio Portafolio a la Innovación. [32]

2.1.1. MISIÓN

> Mejorar la vida de las personas y ayudar a las empresas a cumplir sus metas, utilizando el conocimiento, el talento y la tecnología, soportados en visión, pasión, conciencia, disciplina e intensidad. [32]

2.1.2. VISIÓN

> Nos convertiremos en la más innovadora, motivadora y servicial empresa de la tecnología en américa latina en el desarrollo de plataformas informáticas con foco en: ***ERP:*** *Enterprise Resource Plannig*, planificación de recursos empresariales. ***HR*****:** Human Resources - Recursos Humanos. ***HIS****: Hospital Information System* - Sistemas de Información Hospitalario. ***BPM:*** *Business Process Management*- Gestión por Procesos Inteligentes. [32]

2.2. CARGO DESEMPEÑADO

El cargo desempeñado durante el tiempo pasante fue de desarrollador de pruebas automatizadas, la labor a cumplir era la de automatizar las pruebas funcionales del producto, el trabajo consistía en recibir unos requerimientos impartidos por el área de calidad de la compañía y el objetivo era que los robots de pruebas funcionaran según las especificaciones solicitadas, de la misma forma dar soporte a estas pruebas y ser el encargado de poner en ejecución los sets de pruebas.

Otro de las labores desempeñadas durante el tiempo de práctica era poner en marcha la integración de las operaciones de un producto de la compañía, el cual consistía en brindar capacitación de una herramienta para el control del código fuente y de la misma forma brindar el soporte, esta herramienta permite llevar un control sobre un activo de la compañía como lo es el código fuente. En la ejecución de esta labor se estuvo trabajando también en el desarrollo de extensiones que permitan ejercer control sobre las fuentes.

El reto académico e investigativo que también se llevó a la par con la ejecución de las tareas antes mencionados era el de realizar una migración de un producto de la compañía de una arquitectura Cliente-Servidor a una arquitectura Cliente-Contenedor.

3. MARCO REFERENCIAL

3.1. ESTADO DEL ARTE

El siguiente estado del arte expone algunas de las investigaciones más relevantes halladas acerca de lo que es la tecnología de contenedores sobre Docker, algunos de los trabajos realizados con ella y el impacto creado en el ámbito empresarial, tomado como referencia para la construcción de un proceso comparativo, analítico y constructivo que pueda aportar a este proyecto.

En primera instancia en su investigación Hurtado, David, "*Tecnología de Contenedores de Software en entornos de Pruebas*", (2018). Expone un documento que es presentado como una tesis en donde realiza una selección entre varias herramientas de contenedores como lo son LXC, OpenStack, CoreOS, Docker y Contenedores en Windows, el desarrollo de la selección de las herramientas se utilizó un método de análisis comparativo basado en los principios de calidad sugeridos por la norma ISO/IEC 9126 estándar para la evaluación de productos de software. Al realizar la investigación los resultados obtenidos fue la selección de dos herramientas: Docker (para ambientes en Linux) y Contenedores en Windows (para ambientes en Windows). En el caso del desarrollo del presente proyecto se elige como herramienta a Docker, ya que se encuentra disponible para Windows y además es una herramienta de contenerización más solidificada del mercado, posee un gran repositorio de Imágenes y una comunidad de desarrollo a través del mundo. [1]

Por otra parte, Redondo de Álvaro, Eloy, *"Implementación de un Sistema de Desarrollo de Aplicaciones en Contenedores Docker y su Automatización a través de un Bot de Slack".* habla de la importancia de implementar un sistema de desarrollo de aplicaciones sobre contenedores Docker, ya que este ayuda a manejar el agilismo en las empresas, de la misma forma habla acerca de la reducción de costos que un proyecto basado en contenedores puede causar a una compañía, esto debido a que sus operaciones se hacen en menos tiempo y además los costos que se deben asumir por infraestructura tecnológica disminuyen. Esto es lo que se busca, disminuir los costos y hacer despliegues de aplicaciones de forma rápida y parametrizable. [2]

Así mismo el autor Fernández, José, "*Orquestación de aplicaciones corporativas mediante virtualización de contenedores usando Docker*", (2017), presenta un estudio acerca de la tecnología de virtualización mediante contenedores de Docker esto con el fin de implementar y ejecutar un sistema el cual permita orquestar el

despliegue de aplicaciones sobre un entorno empresarial, todo con el fin de buscar siempre un rápido despliegue de aplicaciones y servicios y también una reducción de los costos, aprovechando al máximo las tecnologías nacientes y que puede agilizar los procesos. Todas estas integraciones permiten a las empresas tener sus productos en continua ejecución y una mayor tolerancia a fallos, es allí en donde radica la importancia de poner en marcha sus aplicaciones empresariales en un entorno de contenedores. [3]

Por último, Garcés, Manuel, "*Diseño e Implementación del sistema de Gestión de Entornos para la Oficina Asesora de Sistemas de la Universidad Distrital*", (2017), propone una mejora para la Oficina Asesora de Sistemas de la Universidad Distrital haciendo uso de contenedores en Docker crea una automatización que permita una fácil vinculación de los desarrolladores a la oficina y también agilizando el despliegue de una forma controlado de los clientes que hacen parte del proyecto. Es importante destacar que con Docker se pueden hacer integraciones reduciendo así la brecha existente entre las operaciones y el desarrollo. [4]

Como bien se puede evidenciar en las anteriores referencias todas hablan acerca de la automatización de los procesos, la reducción de los costos y la entrega continua de un producto, en muchos de los casos este producto hace referencia a una aplicación corriendo sobre uno o varios contenedores de software, buscando de esta forma simplificar los procesos y el despliegue de cualquier aplicación o servicio.

4. MARCO TEÓRICO

Bien, como ya se mencionó anteriormente el proyecto consiste en realizar una migración de un ambiente cliente-servidor virtualizado a cliente-contenedor. Contextualizando, se hace necesario empezar a definir cada uno de los términos mencionados.

Una migración consiste en realizar o proponer un reemplazo, este caso se habla de tecnología, por tal motivo esta reside en transformar datos o como para el objeto del proyecto, proponer una transformación de los modelos de arquitectura de software de la empresa.

El modelo a reemplazar es el modelo de Cliente-Servidor, así que se empieza por realizar algunas definiciones de lo que involucra este modelo. Se empieza entonces hablando de lo que es un servidor.

En la actualidad se puede hablar de dos tipos de **servidores**, un tipo de servidor ofrece un concepto netamente de software y el otro concepto habla acerca de la parte del Hardware.

4.1. Servidor (Software)

Un servidor basado en software es un programa que ofrece un servicio especial que otros programas denominados clientes pueden usar a nivel local o a través de una red. El tipo de servicio depende del tipo de software del servidor. La base de la comunicación es el modelo cliente-servidor y, en lo que concierne al intercambio de datos, entran en acción los protocolos de transmisión específicos del servicio. [5]

En el mundo de las telecomunicaciones existen una gran cantidad de servicios, los servicios más conocidos actualmente y de mayor demanda por mencionar algunos son: servidor proxy, servidor web http o https, servidor DNS servidor DHCP, servidor de Base de Datos, de correo electrónico, servidor de archivo, servidores de monitoreo, entre otros. [5]. En la Figura 1 se puede observar algunos ejemplos de los servidores de software que existen actualmente.

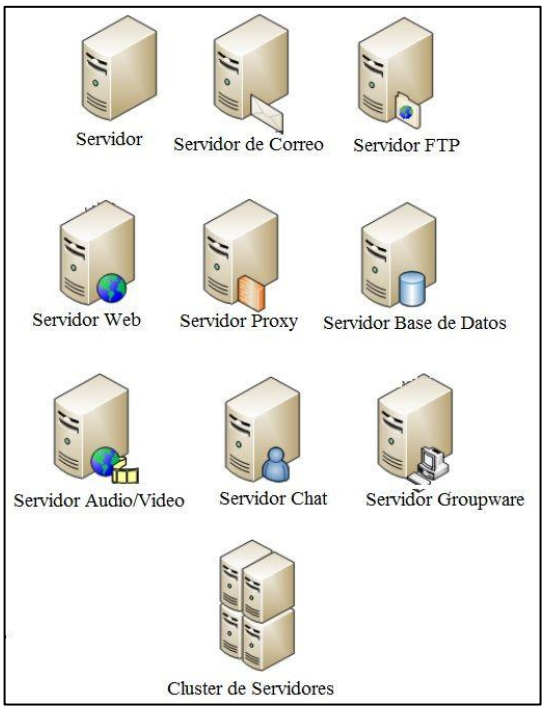


Figura 1. Servicios. Fuente [6]

La anterior evidencia alguno de los servicios que se utilizan a diario, queda claro que para la ejecución de ellos se hace necesario de un hardware, acá es donde interviene el concepto de servidor de hardware.

4.2. Servidor (Hardware)

un servidor basado en hardware es una máquina física integrada en una red informática en la que, además del sistema operativo, funcionan uno o varios servidores basados en software. Una denominación alternativa para un servidor basado en hardware es "host" (término inglés para "anfitrión"). En principio, todo ordenador puede usarse como "host" con el correspondiente software para servidores. [5]

Un servidor por lo general cuenta con grandes prestaciones, debido a que, por lo general, en él se alojan varias máquinas virtuales y ellas necesitan consumir prestaciones del host principal. Un servidor de hardware por lo general se observa de la forma que se muestra en la Figura 2.

Figura 2. Servidor (Hardware). Fuente [7]

Se ha hablado de una arquitectura de cliente-servidor, pero tan solo se ha hablado de la parte del servidor y los tipos de servidores que hay. A continuación, se hablará de la otra parte importante de esta arquitectura, el **cliente**.

4.3. Cliente

Los clientes (o programas que representan entidades que necesitan servicios) y los servidores (o programas que proporcionan servicios) son objetos separados desde un punto de vista lógico y que se comunican a través de una red de telecomunicaciones para realizar una o varias tareas de forma conjunta, Un cliente hace una petición de un servicio y recibe la respuesta a dicha petición; un servidor recibe y procesa la petición, y devuelve la respuesta solicitada. Por decirlo de otra forma, un cliente es el usuario final quien va a gozar de los servicios ofrecidos por el servidor. [8]. En la figura 3 se muestra el cliente final.

Figura 3. Cliente. Fuente [9]

Ya habiendo conjugado estas dos partes del modelo cliente-servidor se presentará lo que es el modelo cliente servidor como tal.

4.4. Modelo Cliente-Servidor

El modelo Cliente/Servidor como se evidencia en la Figura 4 es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Las aplicaciones Clientes realizan peticiones a una o varias aplicaciones Servidores, que deben encontrarse en ejecución para atender dichas demandas. [10]

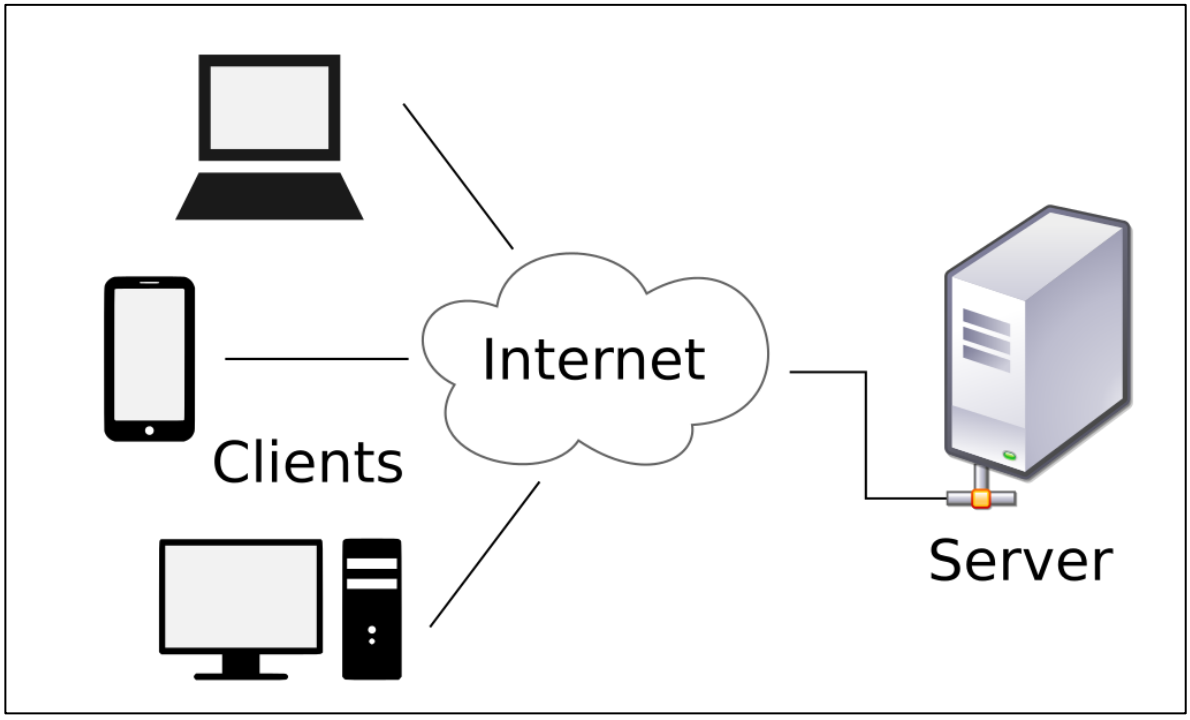


Figura 4. Modelo cliente-servidor. Fuente [11]

Después de haber expuesto lo que es la arquitectura basada en el modelo cliente servidor, se pasa a definir lo que es un contenedor y su arquitectura.

4.5. Contenedor

> Los contenedores son un paquete de elementos que permite crear un entorno donde correr aplicaciones independientemente del sistema operativo. También un contenedor es un entorno ligero de tiempo de ejecución que proporciona a las aplicaciones los archivos, las variables y las bibliotecas que necesitan para ejecutarse, maximizando de esta forma su portabilidad. Además, son una alternativa a las máquinas virtuales porque en lugar de arrancar una máquina entera dentro de otra, los contenedores seccionan los procesos para hacerlos pensar que están en su pequeño mundo. [11]

El contenedor actúa como una funda para el software que lo habilita para funcionar dentro de cualquier entorno. Solo basta con empaquetar el código y las herramientas necesarias para poder ejecutarse dentro de un contenedor. En la Figura 5 se muestran los componentes básicos de los contenedores.

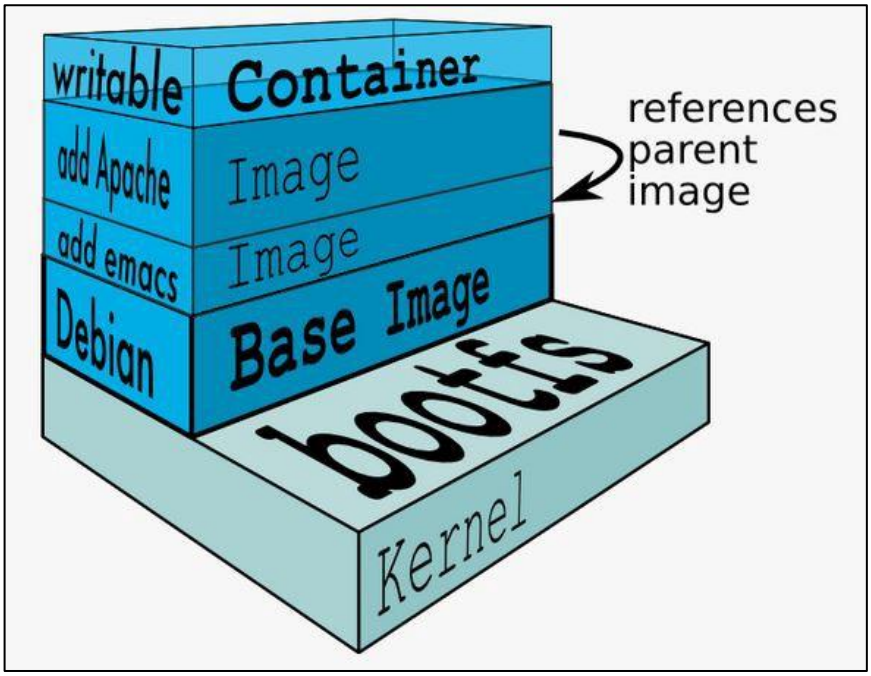


Figura 5. Contenedor. Fuente [12]

4.6. Docker

Docker es un proyecto de código abierto “open source” lanzado por Salomon Hykes el 13 de marzo del 2013, básicamente Docker es una plataforma para la creación de contenedores de software ligeros portátiles y autosuficientes. [11] Esto quiere decir, que permite crear un entorno en el que se pueden correr aplicaciones sin dependencia del sistema operativo. Esto es permitido ya que dentro de él se alojan todos los componentes necesarios para su ejecución, tales como el código, herramientas del sistema y librerías del sistema, etc. Garantizando así la completa ejecución en cualquier maquina solo con el hecho de contar con Docker instalado en las maquinas que se deseen ejecutar. [4]

Docker actualmente ha aumentado mucho las expectativas a futuro algunos analistas prevén que este podría ser el siguiente paso de las “tecnologías de virtualización” y que posiblemente en un futuro los contenedores podrían reemplazar las máquinas virtuales, puesto que los contenedores generan una alta eficiencia y una reducción significativa en los costos.

4.7. Máquinas virtuales VS Contenedores Docker.

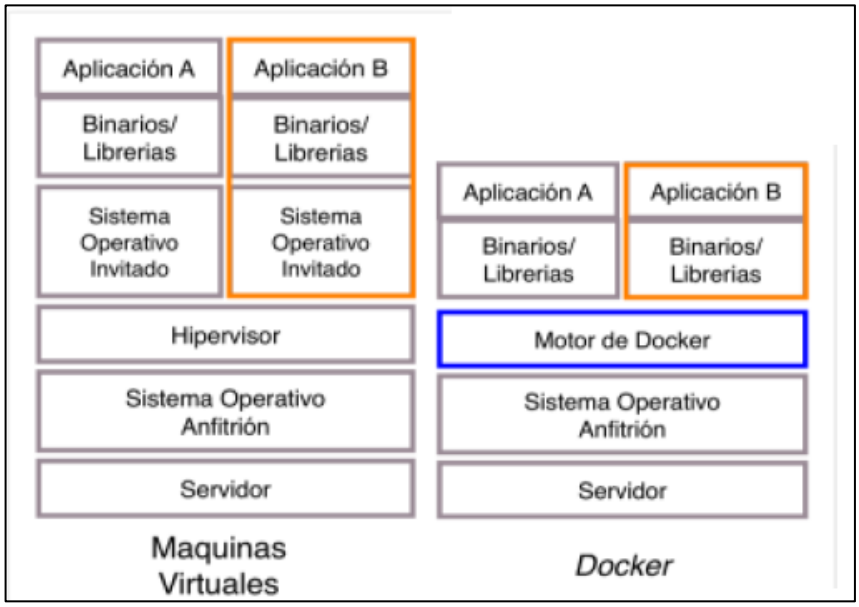


Figura 6. Máquinas virtuales VS Contenedores (Docker). Fuente [4]

En la figura 6 se evidencia la diferencia existente entre las máquinas virtuales y los contenedores.
Una arquitectura de máquina virtual tradicional utiliza un sistema operativo anfitrión y dentro de él se debe hacer uso de un software de virtualización (hipervisor) para

correr los diferentes sistemas operativos que contengan sus librerías y aplicaciones, de esta forma la maquina anfitriona demanda de mucha más capacidad.

El enfoque arquitectónico de Docker es diferente al de las máquinas virtuales tradicionales, siendo la portabilidad el punto más importante de su uso, debido a que son más fáciles de transportar.

A diferencia de las máquinas virtuales los contenedores en Docker corren sobre el Kernel del sistema operativo anfitrión, esto hace que un contenedor Docker sea más flexibles para el empaquetamiento, entrega y despliegue del software y sus aplicaciones.

De un contenedor cabe resaltar que al estar corriendo en un espacio dentro de la maquina anfitriona este puede ser ejecutado en cualquier maquina o infraestructura que cuente con el motor de Docker. [4]

4.8. Arquitectura de Contenedores Docker

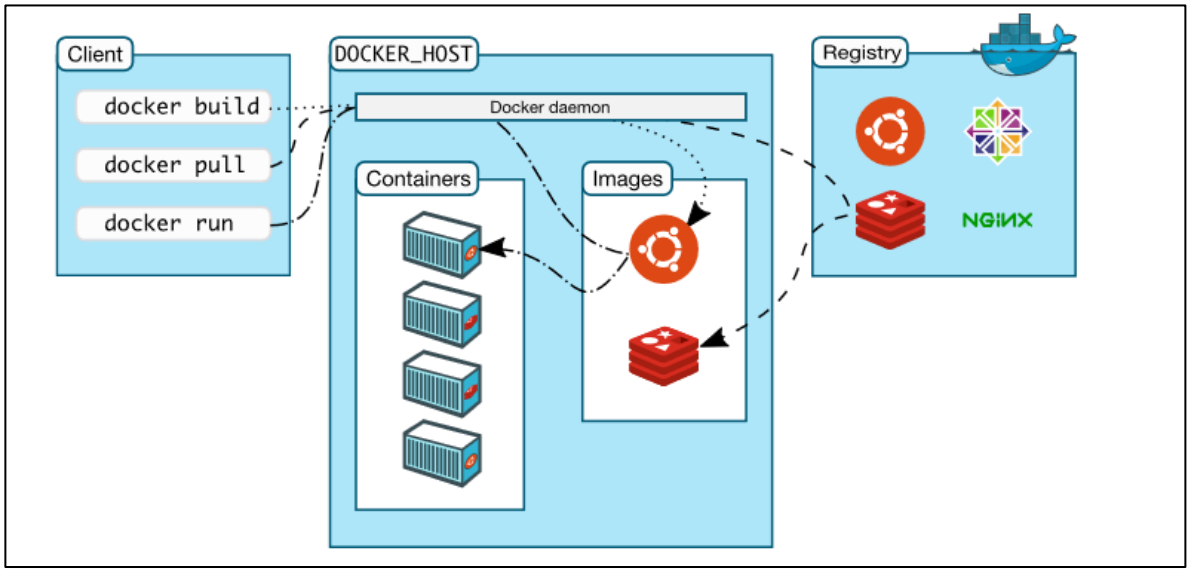


Figura 7. Arquitectura Docker. [13]

En la figura 7 se muestra la arquitectura usada en los contendores. Docker utiliza una arquitectura de cliente-servidor, en donde el cliente se comunica con el demonio de Docker por medio de línea de comandos, que sería la parte del servidor y se encarga de la construcción, funcionamiento y distribución de los contenedores. En el caso de Windows los comandos se ejecutan desde la línea de comandos de PowerShell.

El cliente y el demonio de Docker se pueden ejecutar en el mismo sistema o también conectar un cliente a un demonio Docker que se encuentre a distancia, esta comunicación se realiza a través de sockets o de un API. [4]

Para la creación de los contenedores de Docker se necesita hacer uso de imágenes base (Docker Images), las imágenes no son más que unas plantillas de solo lectura que contienen sistemas operativos base como Windows Server Core, Nano Server, Debian, Ubuntu, etc. Estas imágenes se encuentran situadas en los Registros de Docker (Docker Registries).

Los Registros de Docker son esos repositorios de imágenes públicos o privados al cual se puede acceder para extraer o para subir imágenes de la comunidad. El registro público del proyecto es llamado Docker Hub, allí se pueden encontrar una gran cantidad de imágenes aportadas por los principales distribuidores y por toda la comunidad de desarrolladores de Docker que crea imágenes customizadas pero que también se basan de una imagen oficial aportada por los distribuidores oficiales.

Por último, los contenedores (Docker Containers) es aquel espacio en donde se alojan todos los componentes necesarios para ejecutar una aplicación o brindar algún servicio. Estos contenedores son basados en las imágenes Docker. Cada uno de los contenedores creados es una plataforma aislada, estos solo comparten el Kernel del sistema operativo que los aloja. [13]

4.8.1. Demonio

Un demonio o servicio es un programa que se ejecuta en segundo plano, fuera del control interactivo de los usuarios del sistema ya que carecen de interfaz con estos. [33]

4.8.2. PowerShell

Es un framework de configuración y administración de Windows que permite un shell de línea de comandos y un lenguaje de scripts basado en .NET. [34]. Los comandos en PowerShell son llamados cmdlets, estos hacen que sé que se parezcan más a un lenguaje de programación. PowerShell fue presentada por primera vez junto con el Sistema operativo Windows Vista.

PowerShell a diferencia de CMD permite una ejecución de scripts con mayor complejidad, sus scripts sirven para manejar procesos automatizados y la administración de sistemas Windows.

4.9. Arquitectura de N-Capas

La programación por capas es una arquitectura cliente-servidor cuyo objetivo es separar la lógica de negocios de la lógica de diseño; un ejemplo de esto es separar la capa de datos de la capa de presentación al usuario.
La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado.
Además, permite distribuir el trabajo de creación de una aplicación por niveles; de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles, de forma que basta con conocer la API que existe entre niveles.
En el diseño de sistemas informáticos actual se suelen usar las arquitecturas multinivel o Programación por capas. En dichas arquitecturas a cada nivel se le confía una misión simple, lo que permite el diseño de arquitecturas escalables. [34]

En la Figura 8 se muestra uno de los diseños más utilizados para un diseño de tres niveles.

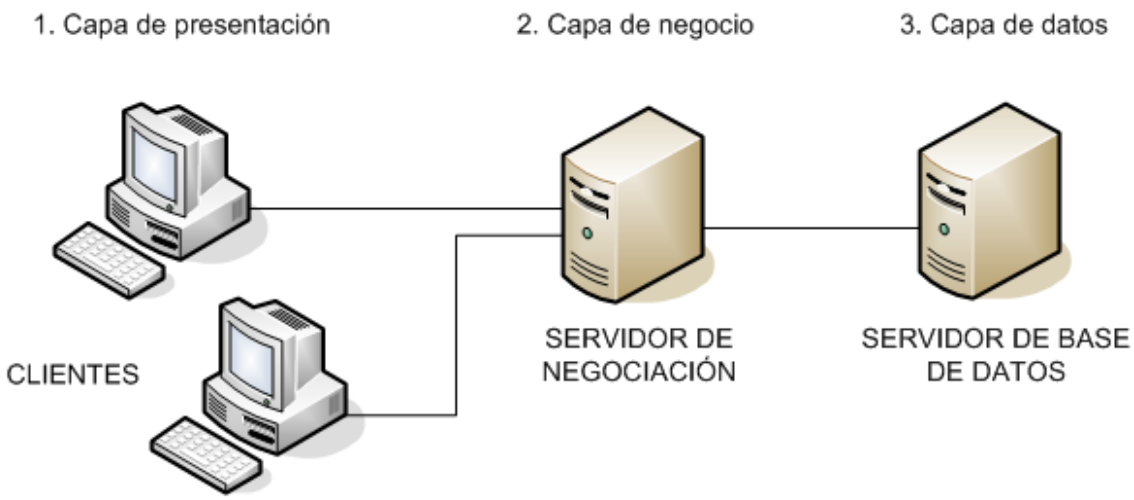


Figura 8. Modelo a 3 capas. Fuente [34].

4.9.1. API

La abreviatura viene del término en ingles Application Programming Interfaces (Interfaces de programación de aplicaciones). Es una especificación formal sobre como un módulo de un software se comunica e interactúa con otro.
Las API son un conjunto de comandos, funciones y protocolos informáticos que permiten a los desarrolladores crear programas específicos para ciertos sistemas operativos. Estas simplifican el trabajo de un programador, esto es debido a que no tienen que escribir código desde cero, puesto que, permiten usar funciones para interactuar con el sistema operativo. [35]

4.10. Arquitectura actual del producto

Digital Ware hace uso de un modelo Multi-Nivel (N-Capas) creado bajo su propio framework de desarrollo. Su arquitectura está basada en unidades lógicas que se ejecutan en una o varias máquinas. Para el transporte de los datos se realiza mediante una red local, una intranet o incluso vía internet haciendo uso de tecnologías estándares de Microsoft. En este modelo de capas se estableció para implementación de la aplicación de las siguientes 3 capas y una explicación grafica similar de este modelo se evidencia en la figura 8. [36]

4.10.1. Nivel de datos.
Provee el sistema de administración de bases de datos relacionales

4.10.2. Nivel de aplicación.
Administra las reglas de negocio y provee servicios de seguridad y acceso a la información.

4.10.3. Nivel de presentación:
Provee el ambiente visual al usuario final. [36]

4.11. Windows Server Core VS Nano Server

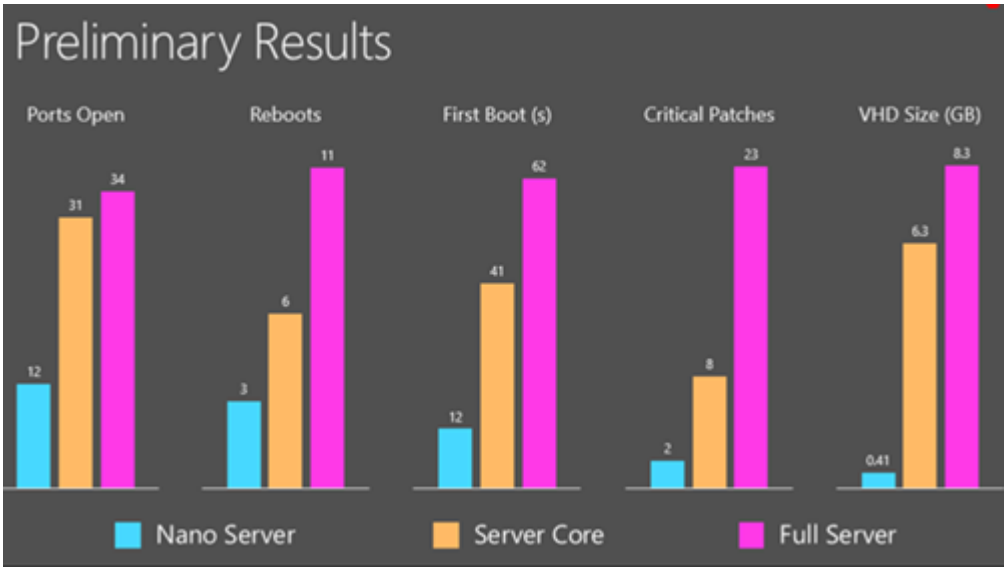


Figura 9. Comparación entre Nano Server, Server Core y Full Server. Fuente [36].

Como se evidencia en la Figura 9, las imágenes de Nano Server ocupan menos espacio en disco duro, el número de puertos es menor, se configura mucho más rápido, requiere menos actualizaciones y reinicios, y los reinicios se ejecutan más rápido debido a la poca cantidad de recursos que este necesita, esto comparado con las imágenes de Server Core y Full Server. [37]

A su vez, una imagen de Nano Server solo puede ser ejecutada en un sistema de 64 bits, no puede actuar como un controlador de directorio activo, no admite directivas de grupo, no se puede utilizar como servidor proxy para el acceso a internet. [38]

Por otro lado, una imagen de Nano Server está ligada con las aplicaciones para la nube basadas en .NET Core, mientras que, la implementación de aplicaciones que necesitan una compatibilidad total con el .NET Framework como lo son las aplicaciones de KACTUS-HCM deben hacer uso de una imagen se Windows Server Core. [37] [38]

5. METODOLOGÍA

5.1. DESCRIPCIÓN Y CARACTERIZACIÓN DEL LUGAR

Digital Ware S.A. es una compañía de desarrollo de software, su oficina principal se encuentra situado en la Avenida Calle Chile en la ciudad de Bogotá D.C y es allí en donde se encuentra toda la infraestructura tecnológica y la casa de desarrollo principal de la empresa, lugar en donde se desarrolló el presente proyecto.

5.2. TIPO DE INVESTIGACIÓN

El presente proyecto tiene un grado de abstracción aplicada, dado que no genera gran cantidad de aportes al conocimiento científico desde el punto de vista teórico, pero si busca resolver problemas prácticos aplicados al desarrollo tecnológico en tiempo real.

Por otro lado, el componente bibliográfico está basado en diferentes referencias, tales como; páginas web, documentos oficiales de la empresa, artículos, libros, trabajos de grados de otras universidades a nivel nacional e internacional y asesorías brindadas por conocedores de la materia.

A partir de la información recolectada se logró estructurar y desarrollar de una forma coherente el presente proyecto y a su vez aportar al desarrollo práctico de las aplicaciones corriendo sobre contenedores.

5.3. DESARROLLO DEL PROYECTO POR FASES

5.3.1. FASE 1: REVISIÓN BIBLIOGRÁFICA Y DE ANTECEDENTES

AUTOR	TITULO	APORTE
David Hurtado	Tecnología de contenedores de software en entornos de pruebas	Realiza una comparación entre herramienta para crear contenedores como lo son: LXC, OpenStack, CoreOS, Docker y Contenedores de Windows, por método comparativo mediante el uso de la norma ISO/IEC 9126.

		Encuentra que las mejores herramientas son Docker y los contenedores de Windows.
Eloy Redondo de Álvaro	Implementación de un sistema de desarrollo de aplicaciones en contenedores de Docker y su automatización a través de un bot de Slack	La implementación de los contenedores basados en Docker agiliza los procesos empresariales, esta agilidad hace que se reduzcan los costos empresariales, así los contenedores causan una disminución en el valor de la infraestructura tecnológica.
José Fernández	Orquestación de aplicaciones corporativas mediante virtualización de contenedores usando Docker	Se realiza el despliegue de aplicaciones haciendo uso de contenedores sobre Docker en ambientes productivos empresariales, permitiendo así un despliegue continuo y gradual, además, aumentar la tolerancia a fallos.
Manuel Garcés	Diseño e Implementación del Sistema de Gestión de Entornos para la Oficina Asesora de Sistemas de la Universidad Distrital	Por medio de los contenedores en Docker crea una automatización que permita una fácil vinculación de los desarrolladores a la oficina y también agiliza el despliegue de las aplicaciones de una forma controlada hacia los clientes.

Tabla 1. Revisión bibliográfica y de antecedentes.

En la tabla 1 se muestra los trabajos tomados como referencia para la construcción del proyecto, en la tabla se realiza un análisis de los resultados obtenidos por cada uno de los autores y que aporte significativo fue tomado para la construcción del proyecto.

5.3.2. FASE 2: ANÁLISIS DE LA ARQUITECTURA ACTUAL DEL PRODUCTO

Como se evidencia en la parte del marco teórico, en donde se habla de la arquitectura actual de producto, se logra identificar tres capas, en las cuales está dividida una aplicación.

Estas capas están divididas en capas lógicas, pueden estar todas implementadas sobre un mismo host o estas pueden ser implementadas en diferentes máquinas.

Partiendo de esto un diagrama aproximado de esta arquitectura se muestra en la figura 10.

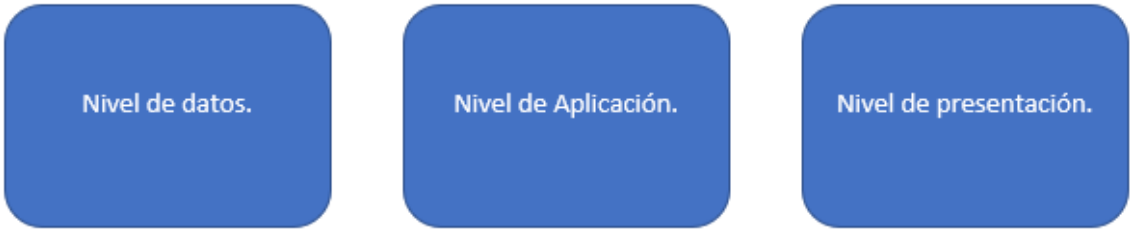


Figura 10. Diagrama de la arquitectura de KACTUS-HCM

Para la implementación de cada una de ellas se establecen unos requisitos mínimos a nivel de Hardware evidenciados en la Tabla 2.

Capa	Procesador	Memoria RAM	Espacio libre en disco duro
Nivel de datos	4 procesadores Intel Quad Core desde 2.6 GHz	Desde 64 GB	250 GB, se recomienda trabajar con Arreglo de discos duros en RAID5.
Nivel de lógica	2 procesadores Intel Quad Core desde 2.6 GHz	Desde 32 GB	100 GB
Nivel de presentación	1 procesador desde 2 GHz	Desde 4 GB	80 GB

Tabla 2. Requerimientos mínimos de hardware para KACTUS-HCM. Fuente [36].

5.3.3. FASE 3: DISEÑO DE LA ARQUITECTURA BASADA EN CONTENEDORES

Se propone la creación del mismo modelo planteado por la empresa, pero en este caso cada uno de los servidores serán reemplazados por un contenedor en el cual será alojado todo lo necesario para su ejecución.

En primer lugar, la capa de datos será implementada sobre un contenedor de base de datos en el cual se hace uso de una imagen de SQL Server Express, esta imagen se encuentra a disposición en el Docker Hub.

Así mismo, la capa de lógica se dividirá en la cantidad de aplicaciones que se desean implementar sobre contenedores. En esta capa se hará la implementación de dos contenedores, cada uno con aplicaciones diferentes. Para la creación de los contenedores se usó una imagen basada en Windows Server Core, el motivo por el cual se eligió esta imagen es que esta permite ejecutar comandos de PowerShell, que permiten la instalación de algunas características, tales como, la característica de Servidor Web, así mismo, las aplicaciones de KACTUS-HCM como están escritas en ASP.NET necesitan una compatibilidad con .NET Framework, esta compatibilidad como se habló en el apartado teórico la ofrece la imagen de Windows Server Core.

El nivel de presentación es el cliente final, quien interactúa con la aplicación que corre sobre contenedores.

El modelo planteado basado en contenedores de Docker es el mostrado en la Figura 11, la comunicación entre ellos se realiza a través de la red nativa de Docker. Una vez el servicio de Docker es iniciado, este configura una interfaz puente virtual en el host. Esto hace que Docker cree una subred virtual para el uso de los contenedores que se ejecutan. Este puente es quien permite la comunicación entre los contenedores y de los contenedores con el host. [39]

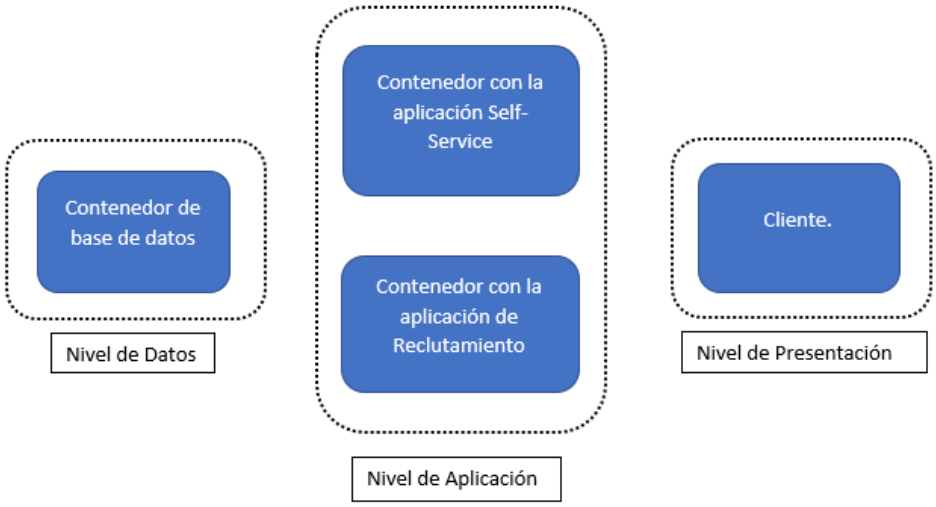


Figura 11. Modelo propuesto sobre contenedores.

5.3.4. FASE 4: IMPLEMENTACIÓN DE CONTENEDORES DE PRUEBAS.

En primera instancia se realizaron unas pruebas básicas para familiarizarse con los comandos básicos de Docker.

5.3.4.1. IMPLEMENTACIÓN CONTENEDOR DE BASE DE DATOS

Se realiza la implementación de un contenedor basado en una imagen de SQL Server Express, dentro del contenedor se crea una base de datos, en ella se crea una tabla y se ejecutan consultas básicas. A continuación, se muestran imágenes relacionadas con el proceso antes mencionado.

Para la obtención de la imagen base que permite la creación del contenedor se debe ejecutar un comando, cabe aclarar que para la ejecución de los contenedores de Docker sobre Windows los comandos deben ser ejecutados desde la línea de comandos de PowerShell, una vez abierta una sesión de PowerShell y con los servicios de Docker en marcha se procede a ejecutar el siguiente comando:

docker pull microsoft/mssql-server-windows-express

este comando nos permite la obtención de la imagen de SQL Server Express extraída directamente desde el Docker Hub. De igual forma si se desea hacer uso de cualquier otra imagen diferente se puede listar los contenidos del Docker hub haciendo uso del siguiente comando. En donde el termino palabraclave sea la coincidencia que se desea buscar en Docker Hub.

docker search palabraclave

Una vez obtenida la imagen se procede a la creación del contenedor, se ejecuta el siguiente comando para crear el contenedor.

docker run -e "ACCEPT_EULA=Y" -e "SA_PASSWORD=Contrasena" -p 1433:1433 --name Nombre -it microsoft/mssql-server-windows-express

docker run: es el comando que genera la creación del contenedor

-e 'ACCEPT_EULA=Y': se utiliza para aceptar el acuerdo de licencia de usuario final. Esta configuración es requerida por la imagen de SQL Server.

-e 'SA_PASSWORD=Contrasena': se utiliza para proporcionar la contraseña de del usuario SA de la base de datos

-p 1433:1433: se asigna un puerto TCP al host y un puerto TCP al contenedor.

--name Nombre: se le asigna un nombre al contenedor.

-it: bandera para que el contenedor sea interactivo, es decir que se pueda entrar mediante un CMD a ejecutar comandos dentro de él.

microsoft/mssql-server-windows-express

una vez creado el contenedor se procedió a la creación de la base de datos de pruebas dentro del contenedor, se ejecutó el siguiente script:

```
CREATE DATA BASE PRUEBAS

GO

USE PRUEBAS

GO

CREATE TABLE UNIVERSIDAD

(NOM_ESTUDIANTE VARCHAR (50),

COD_ESTUDIANTE INT)

INSERT INTO UNIVERSIDAD (NOM_ESTUDIANTE, COD_ESTUDIANTE)
VALUES ('PEPITO PRUEBAS',1234)

SELECT * FROM UNIVERSIDAD
```

5.3.4.2. IMPLEMENTACION CONTENEDOR CON SERVIDOR WEB

En esta etapa se crea un contenedor basado en la imagen de Windows Server Core, dentro de este se habilita la característica de Servidor Web (IIS), para hacer la prueba de que esta característica haya quedado bien instalada se creó una página web básica de HTML que contiene un "Hola desde el mundo de los contenedores".

En primera instancia se tuvo que descargar la imagen del Docker hub al repositorio local de imágenes. Para realizar dicho se procedimiento se ejecutó el comando:

docker pull microsoft/windowsservercore

Para la creación del contenedor se hizo uso del siguiente comando:

docker run –name pruebasiis -p 80:80 -it microsoft/windowsservercore

El contenido del anterior comando ha sido explicado anteriormente. Posteriormente a esto en el contenedor se instaló la característica Servidor Web (IIS), esta característica se instala haciendo uso de la línea de comandos de PowerShell dentro del contenedor, el comando a ejecutar es el siguiente:

Install-WindowsFeature Web-Server

La instalación de esta característica crea en el contenedor la ruta C:\inetpub\wwwroot\ en donde se encuentra la página por defecto configurada para un servidor web de IIS.

Una vez realizado todo lo anterior se procedió, a crea el código HTML básico en donde se implementó un “Hola desde el mundo de los contenedores”. El código que se usó para este propósito es el siguiente:

```
<html>
    <head>
        <title>Hola mundo</title>
    </head>
    <body>
        <p>Hola desde el mundo de los contenedores</p>
    </body>
</html>
```

Este código se creó dentro de un archivo de texto plano al cual se le llamo index.html, posteriormente, el contenedor se detuvo haciendo uso del comando:

docker stop pruebasiis

A continuación, el archivo de index.html situado en la máquina local se alojó dentro del contenedor haciendo uso del siguiente comando.

docker cp C:\index.html pruebasiis:C\inetpub\wwwroot

Este comando copia el archivo de la ruta origen dentro del host local y lo aloja en la ruta destino dentro del contenedor.

Seguido de esto se procedió a iniciar de nuevo el contenedor haciendo uso del comando:

docker start pruebasiis

Por último, se abrió un navegador en la máquina local en donde se introdujo la dirección IP del contenedor seguido de /index.html, esto permitió cargar el contenido del archivo html.

5.3.5. FASE 5. IMPLEMENTACION DEL CONTENEDOR QUE PROVEE LA CAPA DE DATOS DE LAS APLICACIONES

Una vez realizada una etapa de pruebas respecto a la creación de un contenedor de base de datos, se procedió a crear el contenedor que presta el servicio de acceso a datos de la aplicación.

Para la creación de este se hizo necesario contar con una imagen de SQL Server Express. Dicha imagen se extrajo del repositorio de Docker Hub haciendo uso del siguiente comando

docker pull microsoft/mssql-server-windows-express

Una vez extraída la imagen se procedió a la creación del contenedor. Para la creación del contenedor se ejecutó el siguiente comando

docker run -e "ACCEPT_EULA=Y" -e "SA_PASSWORD=Contrasena" -p 1433:1433 --name sqlkactus -it microsoft/mssql-server-windows-express

Posterior a esto el contenedor es detenido haciendo uso del comando:

docker stop sqlkactus

Lo anterior se realizó para poder alojar el archivo del backup a restaurar dentro del contenedor, para este caso el archivo de backup se alojó en la carpeta raíz C:\ del contenedor. Para realizar la copia del backup del host al contenedor se ejecutó el siguiente comando:

docker cp C:\ KACTUS_Backup.bak sqlkactus:C:

Una vez alojado el backup dentro del contenedor, se inicia de nuevo el contenedor usando el comando:

docker start sqlkactus

Para la conexión con el contenedor de base de datos creado anteriormente se procede a extraer la dirección IP, la extracción de la IP se realizó mediante la ejecución del siguiente comando:

docker inspect -f "{{.NetworkSettings.Networks.nat.IPAddress}}" sqlkactus

Después de haber obtenido la dirección IP se procedió a conectarse al contenedor de base de datos haciendo uso la Microsoft SQL Server Management Studio (SSMS), en el tipo de servidor se colocó “Motor de base de datos”, en el Nombre del servidor se introdujo la dirección IP extraída anteriormente, en el tipo de Autenticación se le asigno Autenticación de SQL Server, el usuario para conexión a la base de datos es el SA, por último, se insertó la clave asignada.

Una vez se accedió al motor de base de datos se procedió a hacer la restauración de la base de datos con el backup alojado en el contenedor y haciendo uso del asistente de SSMS, a esta base de datos se le asigno el nombre de KACTUSK. En la Figura 12 se evidencia el asistente para la restauración de la base de datos.

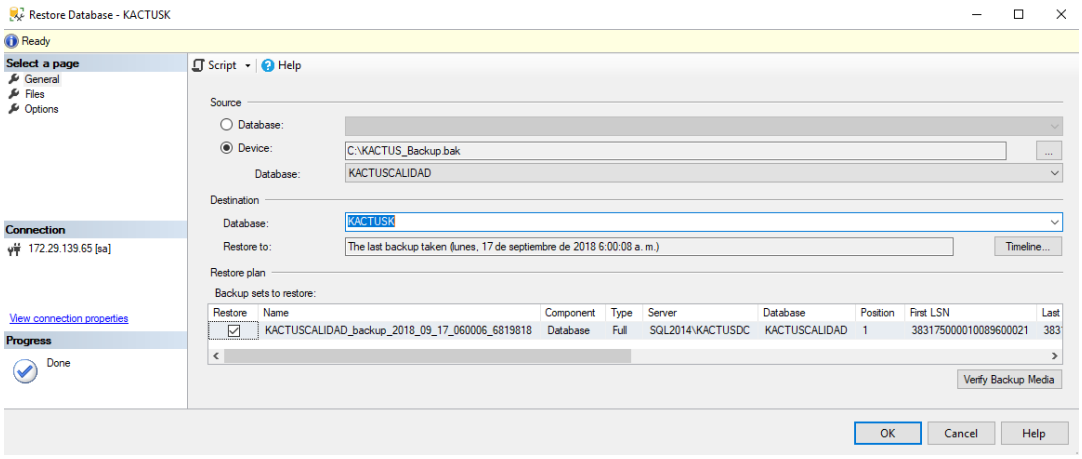


Figura 12. Restauración de base de datos mediante SSMS.

Posteriormente se creó un nuevo Inicio de Sesión evidenciado en la Figura 13. Con el SSMS en el apartado de seguridad del motor se crea un nuevo inicio de sesión, a este inicio de sesión se eligió la autenticación de SQL server se le asigno el nombre de KACTUS y se le asigno una contraseña, de la misma forma se desactivo la opción de Exigir directivas de contraseña, y la base de datos predeterminada se eligió la restaurada KACTUSK.

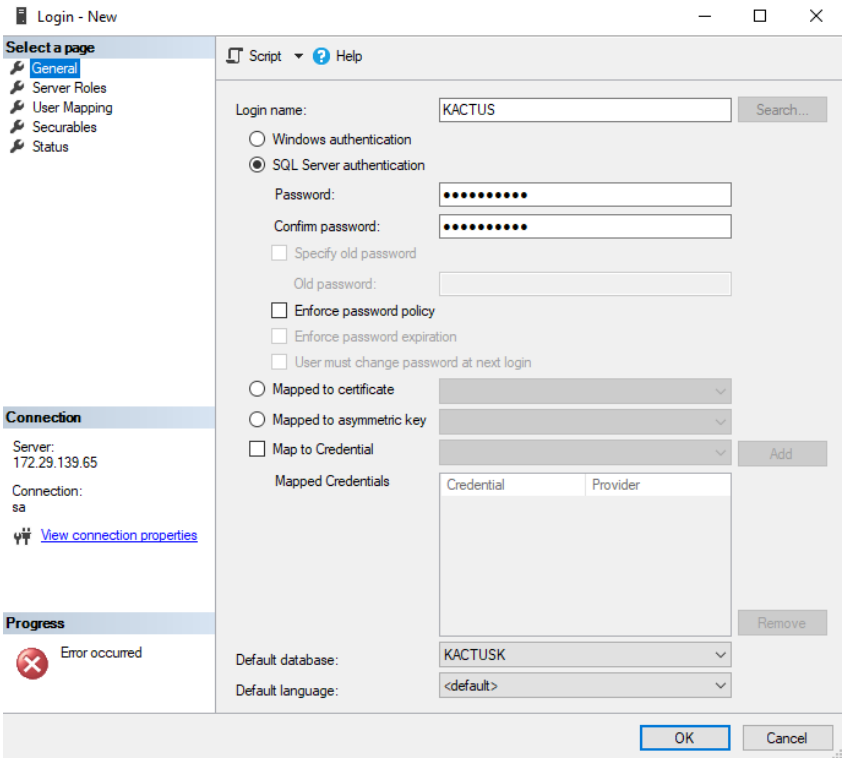


Figura 13. Creación de un nuevo inicio de sesión para la base de datos.

La importancia de creación de este usuario se verá reflejado más adelante en la implementación de los contenedores con las aplicaciones

5.3.6. FASE 6. IMPLEMENTACION DE LOS CONTENEDORES CON LAS APLICACIONES DE SELF-SERVICE Y RECLUTAMIENTO.

Para el despliegue de las aplicaciones en los contenedores se hizo necesario crear dos contenedores con características similares, la diferencia radica en las dos aplicaciones implementadas (Reclutamiento y Self-Service).

En primera instancia se crearon cada uno de los contenedores, como los contenedores se crearon a partir de una imagen base de Windows Server Core, se utilizó el siguiente comando para la extracción de la imagen del Docker Hub al repositorio de imágenes locales:

docker pull microsoft/windowsservercore

Una vez obtenida la imagen se procede a la creación del contenedor, para la creación del contenedor se usó el comando:

docker run –name selfkactus -p 80:80 -it microsoft/windowsservercore

Para acceder a la ejecución de comandos de PowerShell dentro del contenedor se debe ejecutar el siguiente comando:

docker exec -it selfkactus powershell

Ya una vez dentro del contenedor y con la línea de comandos de PowerShell abierta se procedió a hacer la instalación de algunas de las características del servidor para poder realizar el despliegue de la aplicación. Se hizo uso del siguiente comando:

Install-WindowsFeature Caracteristica

La palabra característica fue reemplazada por cada una de las siguientes características de servidor, cada una de ellas es necesaria para la correcta ejecución de la aplicación dentro del contenedor.

Web-Server, Web-Asp-Net45, Web-ASP, Web-CGI, Web-ISAPI-Ext, Web-ISAPI-Filter, Web-Metabase, Web-Lgcy-Scripting, Web-WMI, Web-Scripting-Tools, Web-Mgmt-Service, NET-Framework-45-ASPNET, NET-WCF-HTTP-Activation45, WinRM-IIS-Ext

Después de haber instalado las características se procedió a detener cada uno de los contenedores, el paso a continuación consistió en alojar en el directorio C:\inetpub\wwwroot\ de cada uno de los contenedores los directorios con el contenido de la aplicación, para el caso del Self Service es un directorio con el nombre WebKactus y para el caso del Reclutamiento es una carpeta con el nombre de Reclutamiento.

De la misma forma, en la maquina local se realizó la ejecución de un programa, este programa se encargó de asignar la conexión con la base de datos creada para el funcionamiento de las aplicaciones, es aquí donde interviene el usuario creado en el apartado de la creación de la base de datos, este usuario provee los datos de conexión que necesita la aplicación hacia la base de datos. Al ejecutar el programa este pide el usuario y la contraseña con el cual deseamos acceder, de igual forma, el nombre del servidor y el nombre de la base de datos. Habiendo diligenciado lo anterior este proceso crea un documento de texto plano en el cual se especifican las características mencionadas. Este archivo es el que le indica al contenedor a cuál base de datos debe direccionarse para hacer sus consultas o actualizaciones de datos. En la figura 14 se muestran los datos afectados al ejecutar el archivo.

```
servidorbasedatos=172.29.139.65
basedatos=KACTUSK
puerto=1433

url=jdbc:jtds:sqlserver://172.29.139.65:1433/KACTUSK
```

Figura 14. Archivo con datos de conexión de la aplicación con la base de datos.

Este archivo al igual que las carpetas con el contenido de las aplicaciones se alojaron en cada una de las rutas pertinentes en el contenedor, haciendo uso del siguiente comando:

docker cp RutaOrigenArchivo NombreContenedor:C:

docker cp RutaOrigenDirectorio NombreContenedor:C:\inetpub\wwwroot

De la misma forma en cada uno de los contenedores se alojó el archivo de PowerShell que permitió desplegar cada una de las aplicaciones.

Una vez realizados todos los pasos se volvió a iniciar el contenedor, para realizar el despliegue de la aplicación se ejecutó el script, para ejecutar el script se ejecutó el comando:

C:\DesplegarSitio.ps1

En este momento empezó a ejecutarse cada una de las líneas de código evidenciadas en la figura 15.

En primera instancia se hace un llamado del administrador de IIS, seguido de esto se declaran las variables con la ruta física en donde se encuentra todo el contenido de la aplicación, se declara el path por defecto del servidor web en donde se alojará las aplicaciones y se declara el nombre que llevará la aplicación. Seguido de esto se pregunta si existe el pool de aplicaciones, si este no existe se crea. Luego se pregunta si la aplicación existe o ha sido creada anteriormente, de no existir la aplicación es creada, y es agregada al pool de aplicaciones. De existir la aplicación entonces se convierte a una aplicación web y se le asigna al pool de aplicaciones. Si el pool esta creado y la aplicación también, entonces se escribe en el host que la aplicación ya existe. Por último, se reinicia el pool de aplicaciones.

```
Import-Module WebAdministration;
$physicalPath = "C:\inetpub\wwwroot\WebKactus";
$appPath = "IIS:\Sites\Default Web Site\";
$appName="WebKactus";

if((Test-Path IIS:\AppPools\$appName) -eq 0)
{
    New-WebAppPool -Name $appName -Force;
}

if((Test-Path $appPath$appName) -eq 0 -and (Get-WebApplication -Name $appName) -eq $null)
{
    New-WebApplication -Name $appName -ApplicationPool $appName -Site "Default Web Site" -PhysicalPath $physicalPath;
}
elseif((Get-WebApplication -Name $appName) -eq $null -and (Test-Path $appPath$appName) -eq $true)
{
    ConvertTo-WebApplication -ApplicationPool $appName $appPath$appName;
}
else
{
    Write-Host "$appName la aplicacion existe";
    exit 1
}

Restart-WebAppPool $appName
Restart-WebAppPool $appName
```

Figura 15. Script de PowerShell para el despliegue de las aplicaciones.

5.3.7. FASE 7. VALIDACIÓN FUNCIONAL DE LAS APLICACIONES EJECUTANDOSE SOBRE CONTENEDORES

En la base de datos existe un empleado con código de empleado 5001, este empleado ha sido registrado en el Self Service y posee su contraseña de acceso. Usando el código de empleado y la contraseña pudo acceder y modificar sus datos, a continuación, en la figura 16 se evidencia el registro en un principio. En la figura 17 se realiza el inicio de sesión a la aplicación.

5001	5001	AC 2	UD_11022019_175919	jesusg@digitalware.com.co

Figura 16. Consulta base de datos antes de las modificaciones (Self Service)

Figura 17. Acceso a la aplicación de Self Service.

Una vez accedido se despliega la interfaz mostrada en la figura 18, es allí en donde se modificarán los registros que se mostraron en la primera parte de esta fase. Los recuadros en rojo son los registros modificados.

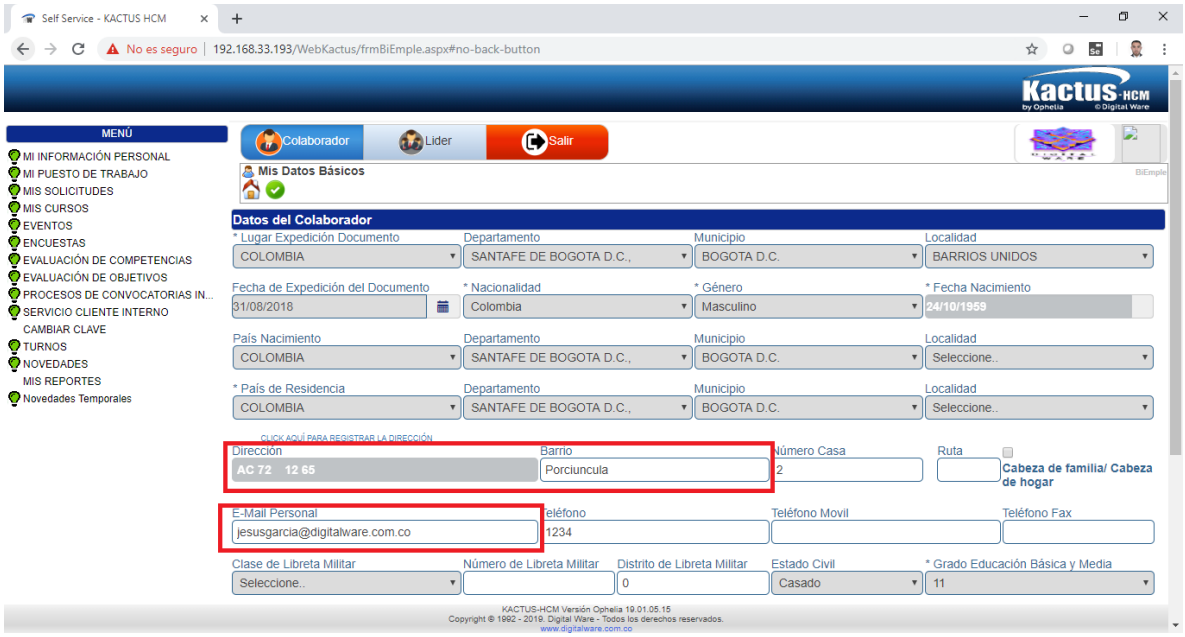


Figura 18. Modificación de registros desde la aplicación.

Una vez se realizó la modificación se procedió a verificar dicha información insertada por medio de la base de datos. En la figura 19 se evidencia la información.

5001	5001	AC 72	12 65	PORCIUNCULA	jesusgarcia@digitalware.com.co

Figura 19. Validación del registro desde la base de datos.

Se continuo entonces a hacer la verificación del correcto funcionamiento de la aplicación de reclutamiento. Para este caso en la base de datos existía un registro con datos iniciales, a continuación, en la figura 20 se muestra la consulta de la base de datos.

1000000	SANJUAN	PEPITO	AC 68	unidos	pepe@gmail.com	5555	3

Figura 20. Consulta base de datos antes de las modificaciones (Reclutamiento).

Para el caso de la aplicación de Reclutamiento el acceso se realiza mediante el correo electrónico de la persona registrada, cabe resaltar que para esta aplicación si existe la opción de registrarse. En la figura 21 se muestra el acceso del usuario a la aplicación para realizar las respectivas modificaciones.

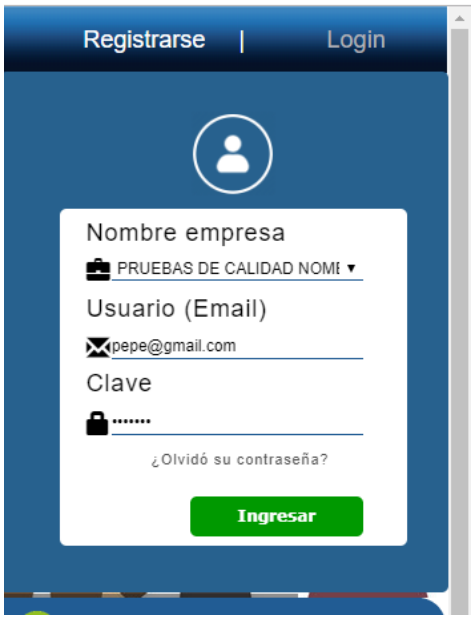


Figura 21. Acceso a la aplicación de Reclutamiento.

Una vez habiendo ingresado a la aplicación se realizó la modificación de los registros, en la figura 22 se encierra en recuadros rojos los registros modificados.

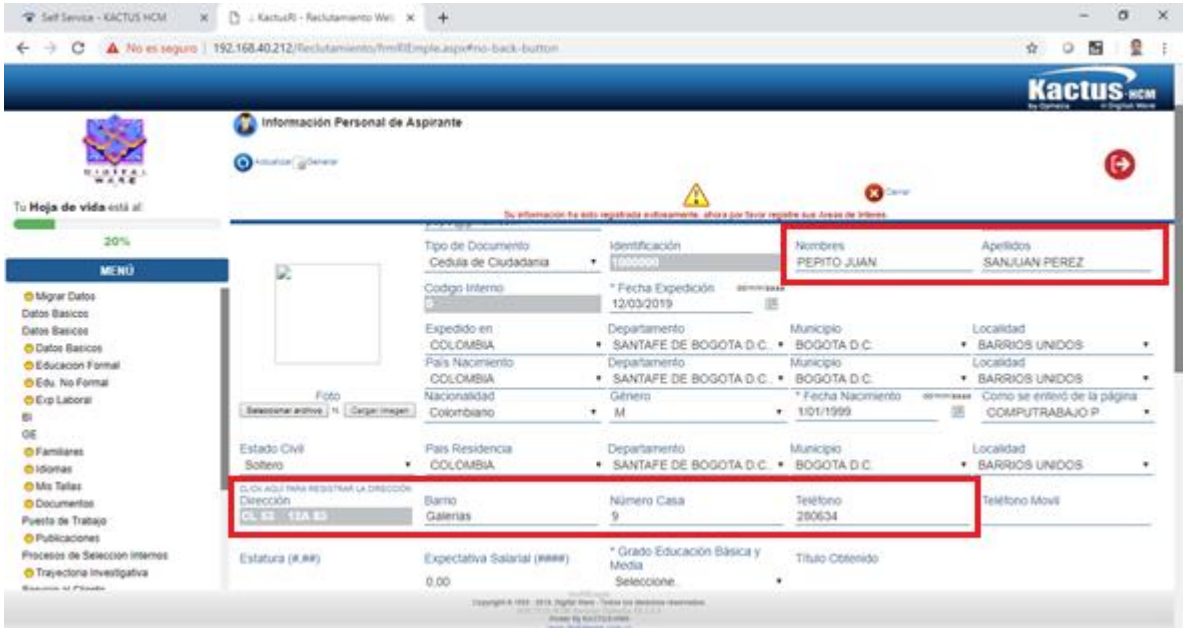
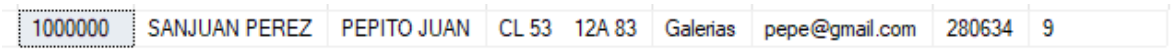


Figura 22. Modificación de registros desde la aplicación.

Una vez más se realiza una consulta a la base de datos filtrando por el correo electrónico del registro y se extrae la siguiente información evidenciada en la figura 23, con esto se puede verificar que la aplicación está funcionando bien.



1000000	SANJUAN PEREZ	PEPITO JUAN	CL 53	12A 83	Galerias	pepe@gmail.com	280634	9

Figura 23. Validación del registro desde la base de datos.

6. RESULTADOS

A partir de la revisión documental se logró identificar que el software de KACTUS-HCM cuenta con un número aproximado de 3000 programas y un total de 41 módulos, estos módulos y programas permiten que el software sea capaz de ejecutar procesos de Nomina y Gestión Humana. Su arquitectura está basada en una arquitectura de cliente-servidor, con un modelo Multi-Nivel de N-Capas, para el despliegue de esta se hace necesario por lo menos contar con un servidor de base de datos, un servidor para aplicaciones y un equipo cliente, todo esto implementado con tecnologías Microsoft.

Con base a lo anterior se creó un modelo basado en contenedores, este consistió en sustituir el servidor virtualizado en diferentes contenedores que ofrecieran los servicios de la aplicación.

Así mismo, esta búsqueda sistemática de varias fuentes de información, referencias aportados por diferentes autores, y a su vez recomendaciones brindadas por conocedores de los contenedores de software, fue posible identificar que el software más apropiado para realizar el despliegue de las aplicaciones en contenedores es Docker, pues está soportada por una gran comunidad de desarrolladores y además sus repositorios virtuales cuenta con una amplia gama de imágenes que permiten la creación de contenedores.

Conociendo el estado del software, habiendo planteado el modelo y una vez elegida la herramienta se procedió a realizar la implementación de los contenedores.

En primera instancia se realizaron unas pruebas básicas para familiarizarse con los comandos básicos de Docker. En este apartado se logró la construcción de un servidor de base de datos basado en una imagen de SQL Server Express, el cual permitió realizar una creación de una base de datos sencilla que se puede ver en la figura 25, en donde se realizaron consultas sencillas evidenciada en la figura 24, este paso se incluyó porque fue de vital importancia para tener un acercamiento con los comandos básicos para la creación de contenedores. A continuación, se muestran imágenes relacionadas con el proceso antes mencionado.

```
C:\>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet 2:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::b194:b2be:4b5:27b7%5
   IPv4 Address. . . . . . . . . . . : 192.168.44.144
   Subnet Mask . . . . . . . . . . . : 255.255.240.0
   Default Gateway . . . . . . . . . : 192.168.32.1
```

Figura 24. Dirección IP Contenedor de base datos de pruebas.

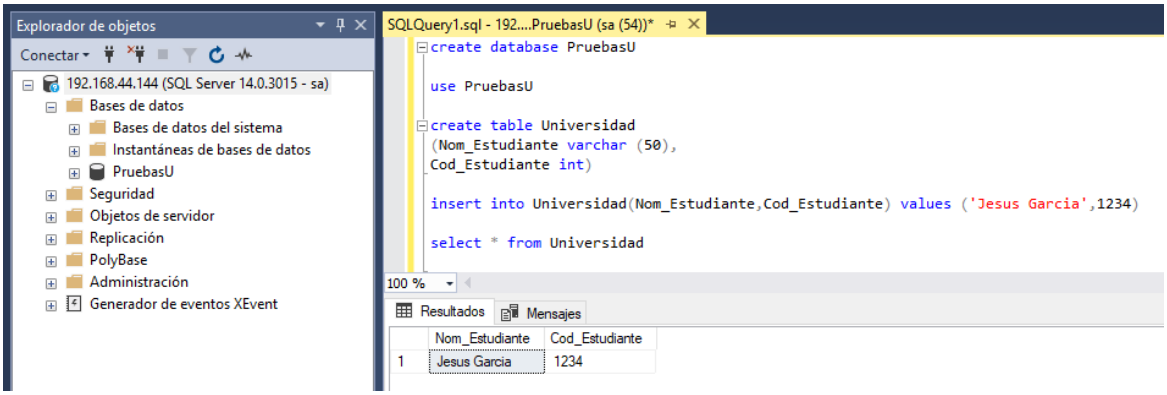


Figura 25. Contenedor de base de datos de prueba.

La Figura 24 evidencia la dirección IP que en el momento de crear el contenedor se le asigna por defecto. La Figura 25 muestra una conexión realizada a la base de datos creada. En ella se puede ver que la dirección o nombre de conexión es la misma dirección IP que se muestra en la Figura 24. Además, se evidencia la prueba realizada, en este caso se creó una base de datos con el nombre **PruebasU** en ella se crea una tabla llamada ***Universidad*** que contiene dos columnas una llamada **Nom_Estudiante** y la otra llamada ***Cod_Estudiante***, posterior a esto se realizó una inserción de datos y por último se ejecutó una consulta de los datos antes creados.

Siguiendo con la secuencia se implementó un contenedor basado en una imagen de Windows Server Core, dentro del mismo se habilita la característica de servidor web (IIS), dentro del contenedor se crea una página HTML básica que contiene un “Hola desde el mundo de los contenedores”.

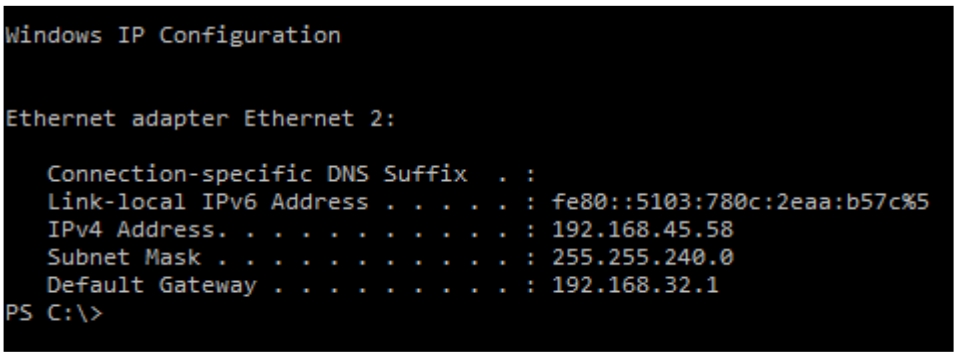


Figura 26. Dirección IP del contenedor con el Servicio Web (IIS).

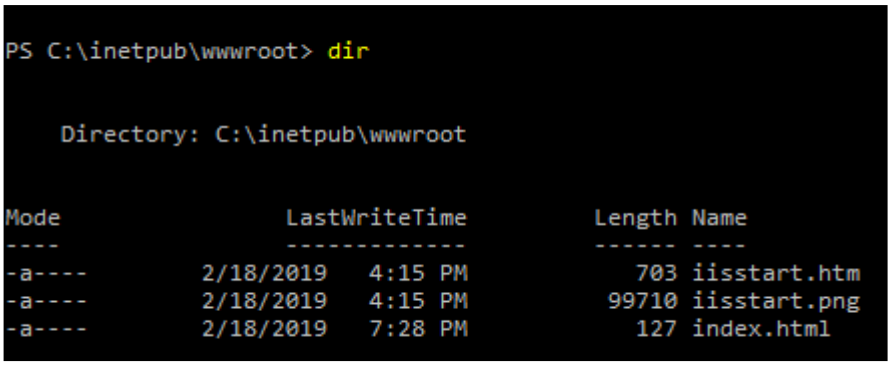


Figura 27. Ruta predeterminada del Servidor Web (IIS).

La Figura 26 muestra dirección IP del contenedor al cual se le instaló la característica de Servidor Web IIS, esta IP se asignó por defecto en el momento que se creó el contenedor.

En el momento que se realizó la implementación de la característica de Servidor Web dentro del contenedor esta creo por defecto la ruta C:\inetpub\wwwroot\ como se observa en la Figura 27. Allí se encuentra alojada la página por defecto del Servidor Web, pero además de esto allí se alojó la página HTML básica que se implementó para verificar el funcionamiento de la característica instalada.

En la figura 28 se puede apreciar el navegador en donde se abrió la página HTML, en ella se evidencia que para abrir la página es necesario escribir la dirección IP del contenedor en el cual fue alojada la página HTML.

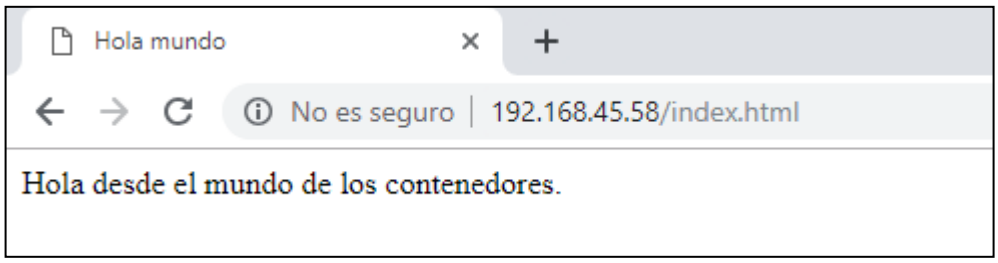


Figura 28. Página HTML alojada en el contenedor.

De esta fase se evidencia en la Figura 29 los contenedores de pruebas creados, la siguiente imagen muestra la lista de los contenedores creados, con su respectiva identificación, de la misma forma la imagen que se utilizó para crear cada uno, la línea de comandos, el tiempo de creación, el lapso de ejecución del contenedor, los puertos y por último el nombre que se le ha asignado al contenedor.

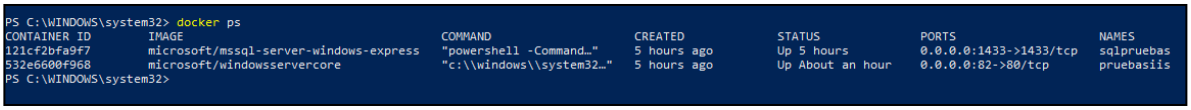


Figura 29. Listar contenedores de prueba.

Una vez realizadas las pruebas y conocer los comandos básicos para la creación de los contenedores, se procedió a implementar los contenedores que contienen las bases de datos y las aplicaciones.

En primer lugar, se implementó un contenedor basado en una imagen de SQL Server Express, en este contenedor se realizó la restauración de una base de datos para la implementación de la capa de datos de las aplicaciones. Se realizó consultas dentro de la base de datos para hacer la verificación de la restauración antes mencionado.

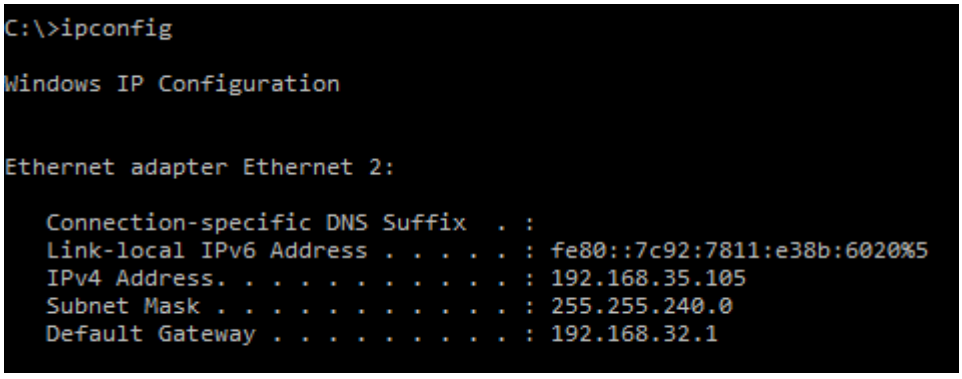


Figura 30. Listar características de red del contenedor de base de datos .

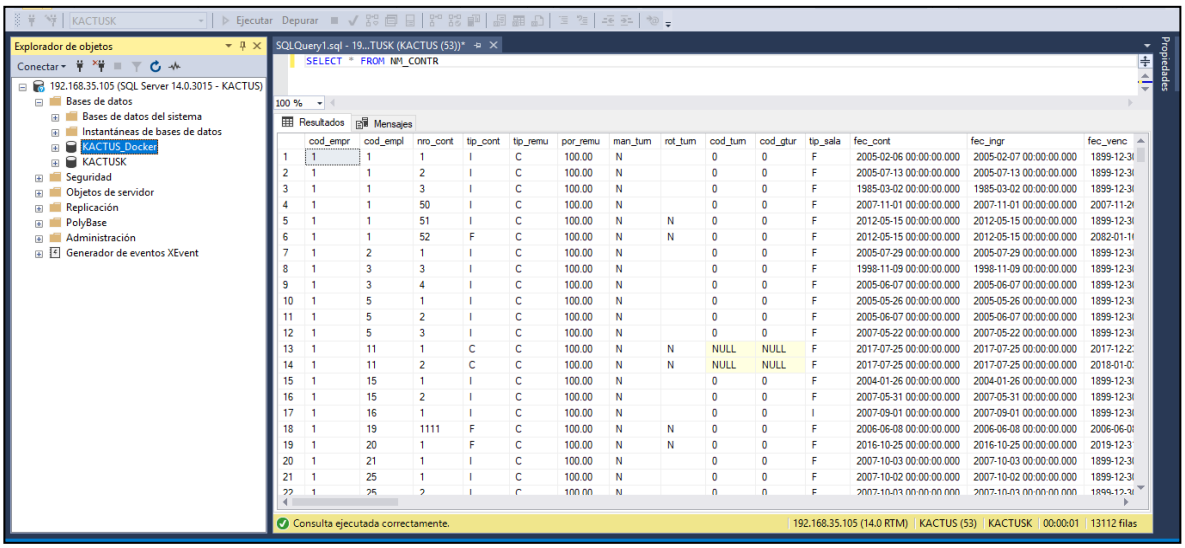


Figura 31. Consulta a la base de datos restaurada.

En la Figura 30, se evidencia la ejecución del comando IPCONFIG esto con el fin de obtener la dirección IP del contenedor de la base de datos, con esta dirección IP se realizó la conexión a la misma haciendo uso de SSMS, en la Figura 31 se puede apreciar los resultados arrojados por la consulta ejecutada, con esto se realizó la comprobación de la restauración de la base de datos del producto.

Para el despliegue de las aplicaciones en los contenedores se hizo necesario la creación de dos contenedores con características similares, la diferencia radica en las dos aplicaciones implementadas (Reclutamiento y Self-Service). En la figura 32 se evidencia la aplicación de reclutamiento ejecutándose dentro de un contenedor. De la misma forma en la figura 33 se evidencia la aplicación del Self Service ejecutándose dentro del contenedor.

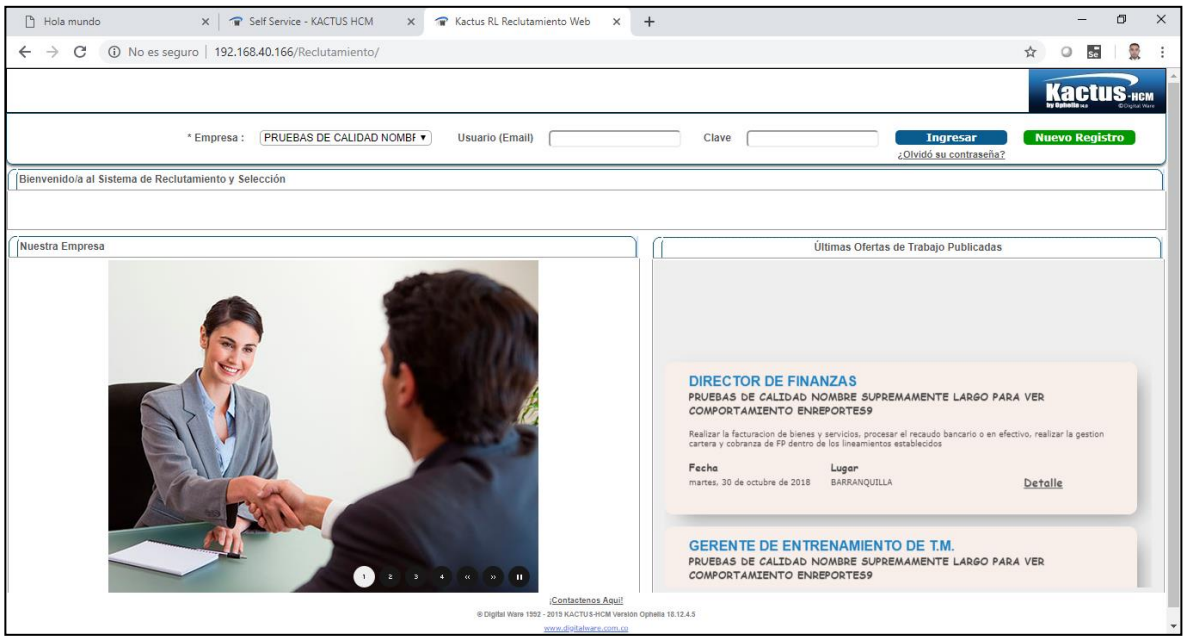


Figura 32. Aplicación de Reclutamiento desplegada en el contenedor.

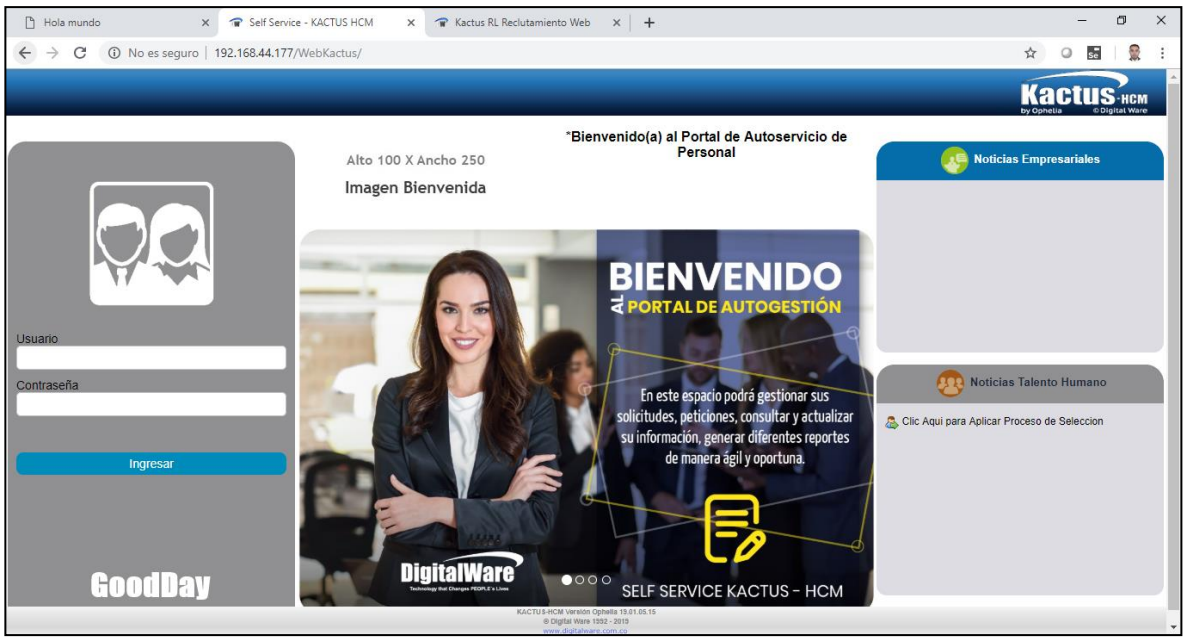


Figura 33. Aplicación de Self-Service desplegada en el contenedor.

7. ANÁLISIS

De acuerdo con los resultados encontrados anteriormente es importante resaltar que KACTUS-HCM es un producto que se compone por tres aplicaciones; Kactus se encarga de todos los procesos de nómina, el Self-Service se encarga de todo el servicio personal y por último la aplicación para el reclutamiento de personal, durante el desarrollo del proyecto surgió un imprevisto al implementar la aplicación de nómina, puesto que esta aplicación es compleja para el despliegue.

Actualmente existe continuidad en el proceso de implementación de esta, el motivo de dicho problema es que todos los ejecutables que se utilizan para instalar la aplicación se deben replicar con comandos de PowerShell, Bash y CMD.

Por otra parte, una implementación básica de KACTUS-HCM sobre un ambiente virtualizado necesita recursos de hardware en un servidor tales como; 2 procesadores Intel Quad Core mínimo 2.6 GHz, una memoria RAM de 32 GB y espacio libre en el disco duro de 100 GB, todos estos recursos son solo para el servidor de aplicaciones.

Así mismo, el servidor de base de datos demanda en prestaciones de hardware aproximadamente 4 procesadores Quad Core desde 2.6 GHz, memoria RAM de 64 GB, espacio libre en disco duro de 250 GB.

Todo esto mencionado anteriormente es lo necesitado para poder la capa de datos y la capa de aplicación de todas las aplicaciones, ahora bien, si comparamos esto con los contenedores se puedo observar una diferencia considerable. Los contenedores fueron construidos sobre un host que cuenta con las siguientes especificaciones; 8GB de memoria RAM, 1TB de disco duro y un procesador Intel® Core™ i5-3470s CPU @2.90GHz, además todos estos recursos no son consumidos en su totalidad si los 3 contenedores creados se ejecutan al tiempo.

El tamaño en disco que ocupa el contenedor con la base de datos restaurada y funcionando es cercano a las 35 GB esto vendría siendo un 14% del mínimo de disco duro requerido para el servidor de base de datos virtualizado.

Ahora si se hace la comparación para el servidor de aplicaciones podemos encontrar también diferencia. El contenedor que contiene la aplicación del Self-Service consume del disco aproximadamente 12 GB y el servidor virtualizado necesita de 100 GB libres en disco duro, el espacio ocupado por el contenedor con la aplicación desplegada es apenas el 12% de lo que necesita el servidor virtualizado. Estos valores son similares para la aplicación de Reclutamiento.

Almacenamiento en disco duro			
	Servidor virtualizado	Contenedor	Porcentaje
Base de datos	250 GB	35 GB	14%
Aplicación	100 GB	12 GB	12%

Tabla 3. Diferencia entre el almacenamiento en disco duro de los servidores virtualizados y los contenedores.

Ahora si se realiza una comparación desde la cantidad de memoria RAM que se utiliza para crear cada uno de los servidores virtualizados VS la memoria RAM que utiliza la máquina en la cual se crearon también se nota una diferencia significativa, a continuación, aparece una tabla que muestra estas diferencias.

Memoria RAM mínima			
	Servidor virtualizado	Contenedor	Porcentaje
Base de datos	64 GB	8 GB	8.3%
Aplicación	32 GB		

Tabla 4.Memoria RAM. Servidor Virtualizado VS Contenedor.

Los recursos de memoria RAM de la maquina tan solo significan un 8.3% de los recursos totales que se usan para la implementación básica de la aplicación KACTUS-HCM

Ahora haciendo un análisis a la cantidad de procesadores que necesita cada uno de los servidores VS la cantidad de procesadores de la máquina que se usó para la creación de los contenedores. Se obtuvo lo siguiente

Cantidad de procesadores			
	Servidor virtualizado	Contenedor	Porcentaje
Base de datos	4 procesadores Intel Quad Core	1 procesador Intel Quad Core	16.6%
Aplicación	2 procesadores Intel Quad Core		

Tabla 5. Procesadores. Servidor Virtualizado VS Contenedor.

Haciendo una similitud solo con la cantidad de procesadores sin tener en cuenta cual es el tipo de procesador. El procesador de la maquina en el cual se creó el contenedor solo significa el 16.6% de la cantidad de procesadores que se usan para la creación del servidor de base de datos y de aplicaciones.

Todas las anteriores cifras mencionadas significarían a la empresa una reducción de los costos bastante altos, como se evidencio, la utilización de los contenedores equivale a un promedio de 13% de los recursos de hardware mínimos necesarios para ejecutar las aplicaciones en servidores virtualizados.

Al realizar una comparación desde el punto de vista de la experiencia de usuario al acceder a las aplicaciones instaladas en los diferentes ambientes no se alcanza a percibir diferencia, una diferencia significativa se logró evidenciar aquí y es que el despliegue de la aplicación en el contenedor se realiza de una forma más rápida que la aplicación desplegada en el servidor.

8. IMPACTO DE LA PASANTIA Y EL TRABAJO REALIZADO

Desde el punto de vista propio la pasantía tuvo un alto impacto, pues aprendí a trabajar en equipo y a cumplir con las metas propuestas, también aprendí mucho sobre una tecnología que era totalmente desconocida para mí y para la compañía.

A nivel empresarial se aporta innovación, pues temas como el desarrollado en el presente proyecto no se habían tratado anteriormente. De igual forma quedan muchos retos por afrontar en la empresa, uno de ellos es la continuación con el proyecto de migración de las aplicaciones, la idea es que el proyecto se ejecute a nivel local y posteriormente llevarlo a los clientes. La idea es ayudar a que los despliegues y las entregas de las implantaciones se realicen en el menor tiempo posible.

9. CONCLUSIONES Y TRABAJOS FUTUROS

Por medio del trabajo se logró determinar que Docker es una herramienta que cada día toma más fuerza y se propone como una herramienta que puede tomar el puesto de las maquinas virtualizadas, esto se debe a que los contenedores consumen mucho menos recursos que una máquina virtual, y de igual forma, en un contenedor se pueden alojar servicios y aplicaciones.

De lo anterior mencionado aún se siguen teniendo muchas discusiones, pues en la actualidad existe una gran cantidad de aplicaciones que aún se siguen desplegando en entornos virtualizados.

Así mismo, haciendo uso de la labor investigativa y práctica, se logró establecer cuál es la arquitectura del producto y en base de esto se pudo plantear la arquitectura basada en contenedores y determinar el número de contenedores a implementar sobre Docker que fue la herramienta elegida para el desarrollo del proyecto.

En cuanto la implementación de las aplicaciones se logró la construcción de dos de las tres aplicaciones que conforman KACTUS-HCM, debido que en el desarrollo de las aplicaciones se presentaron inconvenientes, pues para el despliegue de la aplicación se deben tener en cuenta muchas variables.

Por otra parte, la validación del funcionamiento de la aplicación se realizó accediendo a cada una de las aplicaciones y realizando modificaciones de registros, estos registros modificados fueron validados a través de base de datos y se evidenció el correcto funcionamiento.

Desde el punto de vista comparativo con las aplicaciones sobre servidores virtualizados, la experiencia de usuario ofrecida por los contenedores es similar a la de los servidores. El hecho de implementar contenedores significa para la empresa una reducción de costos, puesto que, los contenedores fueron implementados sobre una sola computadora física, pero en el caso de que se hubiese implementado sobre servidores habría sido necesario por lo menos una máquina virtual para la base de datos y otra máquina virtual para las aplicaciones.

Cabe aclarar que actualmente se sigue trabajando en la implementación de la aplicación de Kactus esto para poder brindar el producto completo corriendo sobre contenedores. De igual forma una vez terminado el despliegue las aplicaciones se piensa crear un contenedor que sirva como servidor DNS que permita una fácil iteración entre las aplicaciones y las bases de datos porque actualmente se necesita de modificar unos ficheros estableciendo la dirección IP de los contenedores para que se permita la iteración sobre ellos.

Otra apuesta ambiciosa de este proyecto es poder llevar cada uno de esos contenedores a un medio portable, por ejemplo, una USB en donde se cuente con cada una de las aplicaciones y esta una vez sea conectada a un host sea capaz de desplegar cada una de ellas.

10. REFERENCIAS

[1] HURTADO CHICHANDE, David Alfredo. *Tecnología de contenedores de software en entornos de pruebas.* Tesis de Grado. Pontificia Universidad Catolica del Ecuador Esmeraldas.

[2] REDONDO DE ÁLVARO, Eloy. *Implementación de un Sistemas de Desarrollo de Aplicaciones en Contenedores Docker y su Automatización a través de un Bot de Slack.* Trabajo fin de grado. Universidad de Valladolid.

[3] FERNÁNDEZ AMEIJEIRAS, José Angel. *Orquestacion de aplicaciones corporativas mediante virtualización de contenedores usando Docker.* Grado de ingeniería informática. Universitat Oberta de Catalunya.

[4] MUÑOZ GARCÉS, Manuel Fernando, et al. Diseño e Implementación del Sistema de Gestión de Entornos para la Oficina Asesora de Sistemas de la Universidad Distrital. 2017.

[5] ¿Qué es un Servidor?. Digital Guide 1and1, ©2018 [consulta: 1 Agosto 2018]. Disponible en: https://www.1and1.es/digitalguide/servidores/know-how/que-es-un-servidor-un-concepto-dos-definiciones/

[6] Que es un servidor y tipos de servidores, ©2018 [consulta: 10 Enero 2019]. Disponible en: https://www.areatecnologia.com/informatica/servidor-y-tipos.html

[7] El coste del servidor en la pyme: el hardware, ©2010 [consulta: 15 Enero 2019]. Disponible en: https://m.pymesyautonomos.com/tecnologia/el-coste-del-servidor-en-la-pyme-el-hardware

[8] E.U. Informática en Segovia. (2005). El Modelo Cliente/Servidor [archivo PDF]. Recuperado de: https://www.infor.uva.es/~fdiaz/sd/2005_06/doc/SD_TE02_20060305.pdf

[9] Diseño de sistemas computarizados, [consulta: 16 Enero 2019] Disponible en: http://luisjmartinezj.freetzi.com/diseno/cliente.html

[10] Emiliano Marini. (Octubre del 2012). El Modelo Cliente/Servidor [archivo PDF]. Recuperado de: https://radiosyculturalibre.com.ar/compartir/biblioteca/REDES/linuxito%20-%20El%20Modelo%20Cliente-Servidor.pdf

[11] PONSICO MARTIN, Pol. *Tecnología de Contenedores Docker*. 2017. Tesis de Licenciatura. Universitat Politècnica de Catalunya.

[12] Introducción a Docker, Carlos Verdes, 2015 [consulta: 12 diciembre 2018]. Disponible en: https://medium.com/@carlosverdes/introducci%C3%B3n-a-docker-34a64fabeeaf

[13] Primeros paso con Docker, ©2019 José Domingo Muñoz,[consulta: 1 enero 2018]. Disponible en: https://www.josedomingo.org/pledin/2016/02/primeros-pasos-con-docker/

[14] About Docker Engine, ©2019 Docker Inc. [consulta: 18 octubre 2018]. Disponible en: https://docs.docker.com/engine/

[15] Docker for Windows, ©2019 Docker Inc. [consulta: 1 septiembre 2018]. Disponible en: https://docs.docker.com/docker-for-windows/

[16] Docker Documentation, ©2019 Docker Inc. [consulta: 30 noviembre 2018]. Disponible en: https://docs.docker.com/

[17] Copiar ficheros locales a un contenedor, Alex Such [consulta: 16 octubre 2018]. Disponible en: https://veamospues.wordpress.com/2017/11/28/copiando-ficheros-locales-a-un-contenedor-docker/

[18] Docker Hub, ©2019 Docker Inc. [consulta: 30 agosto 2018]. Disponible en: https://hub.docker.com/

[19] Quickstart: Run SQL Server container images with Docker, © Microsoft 2019 [consulta: 4 septiembre 2018]. Disponible en: https://docs.microsoft.com/en-us/sql/linux/quickstart-install-connect-docker?view=sql-server-2017&pivots=cs1-powershell

[20] Step bry step guide to run SQL Server in a Windows Docker Container, dwjunkie diciembre 12, 2017 [consulta: 5 septiembre 2018]. Disponible en: https://nexxtjump.com/2017/12/12/step-by-step-guide-to-run-sql-server-in-a-windows-docker-container/

[21] DevOps, © 2019 MediaOps Inc. [consulta: 10 diciembre 2018]. Disponible en: https://devops.com/

[22] Customers Docker, ©2019 Docker Inc. [consulta: 19 noviembre 2018]. Disponible en: https://www.docker.com/customers

[23] Compañías que usan Docker, © 2019 StackShare, Inc. [consulta: 21 diciembre 2018]. Disponible en: https://stackshare.io/docker

[24] Containers on Windows, © Microsoft 2019 [consulta: 4 septiembre 2018]. Disponible en: https://docs.microsoft.com/en-us/virtualization/windowscontainers/about/

[25] Getting Started with Windows Containers, © Microsoft 2019 [consulta: 5 septiembre 2018]. Disponible en: https://devblogs.microsoft.com/aspnet/getting-started-with-windows-containers/

[26] Relays IIS Log Entries To Read Them in Docker, Elton Stoneman [consulta: 14 diciembre 2018]. Disponible en: https://blog.sixeyed.com/build-a-dev-rig-for-running-windows-docker-containers/

[27] 1, 2, 3, GO! Run IIS in Docker on Windows Server 2016, Elton Stoneman [consulta: 15 diciembre 2018]. Disponible en: https://blog.sixeyed.com/1-2-3-iis-running-in-nano-server-in-docker-on-windows-server-2016/

[28] Running Microsoft SQL Server on a Linux container in Docker, Tom Chantler [consulta: 19 diciembre 2018]. Disponible en: https://tomssl.com/2018/01/15/running-microsoft-sql-server-on-a-linux-container-in-docker/

[29] Learn Docker in 12 Minutes, Nelsonic [consulta: 20 agosto 2018]. Disponible en: https://github.com/dwyl/learn-docker/issues/14

[30] Docker for beginners, ©2019 Docker Inc. [consulta: 28 junio 2018]. Disponible en: https://docker-curriculum.com/

[31] Get Started, Part 1: Orientation and setup, ©2019 Docker Inc. [consulta: 25 junio 2018]. Disponible en: https://docs.docker.com/get-started/

[32] Digital Ware Tecnology that changes People´s Lives. [En línea]. Bogotá DC. [Consulta: 20 de septiembre de 2018]. Disponible en Internet: http://www.digitalware.com.co/index.php/acerca-de-digital-ware/nosotros-2

[33] Servicios y demonios en Linux. Jesús Torres [consulta 12 marzo 2019] Disponible en: https://medium.com/jmtorres/servicios-y-demonios-en-linux-a424366336ac

[34] Administración de base de datos. Alfsan [consulta 17 marzo 2019] Disponible en: http://iutll-abdd.blogspot.com/2012/05/arquitectura-de-n-capas.html

[35] ¿Qué es una API y para qué sirve? ABC Tecnología [consulta 17 marzo 2019] Disponible en: https://www.abc.es/tecnologia/consultorio/20150216/abci--201502132105.html

[36] Manual de Arquitectura. ©2019 Digital Ware

[37] Windows Server 2016. Contenedores Nano Server y Multipoint Server. Sidertia [consulta 17 marzo 2019] Disponible en: https://www.sidertia.com/Home/Community/Blog/2017/06/20/Windows-Server-2016-Contenedores-Nano-Server-y-Multipoint-Server

[38] Microsoft Nano Server. Margaret Rouse [consulta 17 marzo 2019] Disponible en: https://searchwindowsserver.techtarget.com/definition/Microsoft-Nano-Server

[39] El Ecosistema de Docker: Creación de Redes y Comunicación. ©2019 Digital Ocean [consultado 17 marzo 2019] Disponible en: https://www.digitalocean.com/community/tutorials/el-ecosistema-de-docker-creacion-de-redes-y-comunicacion-es