PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR ESMERALDAS

ESCUELA DE SISTEMAS Y COMPUTACIÓN.

TESIS DE GRADO

TÍTULO:

"TECNOLOGÍA DE CONTENEDORES DE SOFTWARE EN ENTORNOS DE PRUEBAS"

LÍNEA DE INVESTIGACIÓN:

PROGRAMACIÓN Y DESARROLLO DE SOFTWARE

PREVIO A LA OBTENCIÓN DE TÍTULO DE INGENIERO DE SISTEMAS Y COMPUTACIÓN

AUTOR:

HURTADO CHICHANDE DAVID ALFREDO

ASESOR:

MGT. MARC GROB

FECHA:

ESMERALDAS, ENERO 2018.

Tesis de grado aprobada luego de haber dado cumplimiento a los requisitos exigidos, previo a la obtención del título de INGENIERO EN SISTEMAS Y COMPUTACIÓN.

TRIBUNAL DE GRADUACIÓN

Título: "TECNOLOGÍA DE CONTENEDORES DE SOFTWARE EN ENTORNOS DE PRUEBAS"

Autor: DAVID ALFREDO HURTADO CHICHANDE

Mgt. Marc Grob **f.-……**
Asesor/a

Mgt. José Luis Carvajal Carvajal **f.-…….**
Lector #1

Mgt. Xavier Quiñónez Ku **f.-…….**
Lector #2

Mgt. Xavier Quiñónez Ku **f.-……**
Director de Escuela

Mgt. Maritza Demera Mejía **f.-……....**
Secretaria general PUCE-Esmeraldas

Esmeraldas, Ecuador, Enero 2018.

DECLARACIÓN DE AUTENTICIDAD Y RESPONSABILIDAD

Yo, **DAVID ALFREDO HURTADO CHICHANDE** portador de la cédula de identidad No. **085003568-4** declaro que los resultados obtenidos en la investigación que presento como tesis de grado, previo a la obtención del título de **"Ingeniero en Sistemas y Computación**" son absolutamente originales, auténticos y personales.

En tal virtud, declaro que el contenido, las conclusiones y los efectos legales y académicos que se desprenden del trabajo propuesto de investigación y luego de la redacción de este documento son y serán de mi sola, exclusiva responsabilidad legal y académica.

DAVID ALFREDO HURTADO CHICHANDE

CI 085003568-4

CERTIFICACIÓN

Mgt. Marc Grob, docente investigador de la PUCE-Esmeraldas, certifica que:
La tesis de grado realizada por DAVID ALFREDO HURTADO CHICHANDE bajo el título "TECNOLOGÍA DE CONTENEDORES DE SOFTWARE EN ENTORNOS DE PRUEBAS" reúne los requisitos de calidad, originalidad y presentación exigibles a una investigación científica y que han sido incorporadas al documento final, las sugerencias realizadas, en consecuencia, está en condiciones de ser sometida a la valoración del Tribunal encargada de juzgarla.

Y para que conste a los efectos oportunos, firma la presente en Esmeraldas, a 8 días del mes de enero del 2018.

Mgt. Marc Grob
Asesor

DEDICATORIA

El presente trabajo de investigación está dedicado a Dios, a mis padres que fueron de vital importancia en el transcurso de mi carrera universitaria. A mis hermanos que en cada momento me dieron su apoyo para seguir adelante, gracias.

AGRADECIMIENTO

Agradezco a Dios por darme la oportunidad de vivir esta linda experiencia, por haberme llenado de sabiduría y paciencia para llegar a la meta.

Me gustaría enfatizar el apoyo de mi familia a lo largo de mi camino por esta prestigiosa institución.

Agradecimiento mis padres y abuelos. Por el constante apoyo y animo que me dieron.

RESUMEN

La presente investigación se realizó con el fin de proponer tecnologías de contenedores de software en ambientes de pruebas, haciendo uso de estándares de calidad en el proceso que conlleva la aplicación de las mismas, para ello, se tomó como caso particular el departamento de TIC (Tecnologías de la Información y Comunicación) de la PUCE (Pontificia Universidad Católica del Ecuador) – Esmeraldas, realizando una comparación entre las diversas tecnologías de contenedores de software disponibles hasta el momento para la ejecución de las pruebas, seleccionando así, las que permitan mejorar la calidad del software.

Para ejecutar la investigación, se recolectó información de la problemática mediante entrevistas al Jefe y al desarrollador de software del departamento, dichas entrevistas permitieron la identificación de procesos y actividades que desarrollan, determinando así, las características esenciales que deben poseer las tecnologías que se necesitan en el área de estudio, consiguiendo de tal manera, el procedimiento necesario para la administración del proceso de las pruebas en los ciclos de desarrollo de software.

Los resultados obtenidos de las entrevistas permitieron identificar información de flujo de procesos, ésta información es de vital importancia a la hora de categorizar y asignar prioridades de uso a las tecnologías evaluadas, de tal manera que se proponga el software adecuado en casos determinados, optimizando así el proceso de selección de software para la ejecución de pruebas.

Teniendo clara la situación actual del departamento de TIC, se procedió a identificar las tecnologías con características similares como son: LXC, OpenStack, CoreOS, Docker y Contenedores en Windows, esta selección se realizó mediante el método de análisis comparativo basado en los principios de calidad sugeridos por la norma ISO/IEC 9126 que basa su evaluación en productos de software.

Esta investigación concluye con la selección de dos tecnologías de contenedores de software, las cuales son: Docker (para ambientes en Linux) y Contenedores en Windows (para ambientes en Windows), los cuales trabajan con herramientas de despliegue de software que permiten tener un reporte y control de errores encontrados.

Palabras clave: pruebas, software, contenedores, tecnología, estándares, calidad

ABSTRACT

The aim of the present investigation has been to propose software container technologies in test environments, making use of quality standards in the process that entails the application of them, for this, the ICT department was taken as a particular case (Information and Communication Technologies) of the PUCE (Pontifical Catholic University of Ecuador) - Esmeraldas, making a comparison between different technologies of software containers available so far for the execution of the tests, thus selecting the ones that allow improving the software quality.

To carry out the investigation, information about the topic was collected through interviews with the Chief and the software developer of the department. These interviews allowed the identification of processes and activities that they carry out, thus determining the essential characteristics that the technologies needed in the study area, achieving in this way, the necessary procedure for the administration of the testing process in the software development cycles.

The results obtained from the interviews allowed identifying process flow information, this information is of vital importance at the time of categorizing and assigning priorities of use to the evaluated technologies, in such a way that the appropriate software is proposed in certain cases, thus optimizing the process of selecting software for the execution of tests.

Having clear the current situation of the ICT department, it proceeded to identify technologies with similar characteristics such as: LXC, OpenStack, CoreOS, Docker and Containers in Windows, this selection was made through the method of comparative analysis based on the principles of quality suggested by the ISO / IEC 9126 standard that bases its evaluation on software products.

This research concludes with the selection of two software container technologies, which are: Docker (for environments in Linux) and Containers in Windows (for environments in Windows), which work with software deployment tools that allow to have a report and control of founded errors.

Keywords: testing, software, containers, technology, standards, quality.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Presentación de la investigación

La presente tesis estudia el proceso de aseguramiento de calidad y el manejo de nuevas alternativas al momento de desarrollar aplicaciones, teniendo en cuenta la calidad del producto final, ambientes o entornos de pruebas y requisitos esperados de un software a la hora de entregar a un cliente.

Desarrollar aplicaciones es una tarea compleja que maneja una cantidad de procesos, que en ocasiones no son tomados en consideración en su totalidad por los involucrados. Factores como: aumento del tiempo de entrega de un proyecto, aumento de costo de desarrollo de software, cambios continuos sobre los requisitos iniciales, hacen que los involucrados tengan que priorizar sobre los aspectos importantes de un proyecto.

En los proyectos de desarrollo, uno de los procesos para el aseguramiento de calidad es realizar las respectivas pruebas para la búsqueda de errores, aplicándolas en los módulos de código que tiene el proyecto desarrollado.

En el área de desarrollo se establecen procesos que permitan realizar aplicaciones de calidad y que el tiempo de entrega se reduzca. Aunque se tenga la noción de lo que se tiene que hacer en muchos casos no se realiza, dejando a la deriva los proyectos, permitiendo que los problemas se acumulen. Se debe tener en cuenta que las aplicaciones evolucionan conforme las exigencias de los usuarios por tal motivo siempre habrá nuevas versiones que verificar y entregar.

Planteamiento del problema

Un problema que se presenta en el proceso de desarrollo de software es el aseguramiento de calidad. Esto se debe porque no usan apropiadamente metodologías de pruebas para realizar la validación de proyectos.

En el departamento de TIC de la PUCE-Esmeraldas realizan procesos de aseguramiento de la calidad de los proyectos que desarrollan mediante tecnologías que cubran los requerimientos de estándares internacionales que permitan brindar productos de calidad, evaluándolos desde ambientes de desarrollo hasta producción, donde hay que realizar

pruebas del software para asegurar su calidad según lo que se presenta en los niveles de servicio de QA (Aseguramiento de la calidad por sus siglas en inglés Quality Assurance).

La presente investigación busca disminuir la inversión de dinero, tiempo y que las pruebas de software no se ejecuten de manera improvisada. Por ello se establece la siguiente interrogante: ¿Es adecuada la implementación de tecnologías de contenedores de software para entornos de pruebas mediante estándares de calidad en el departamento de TIC de la PUCE-Esmeraldas?

Justificación

Las pruebas de software forman parte fundamental al momento de asegurar la calidad de los productos pudiendo identificar los defectos evitando que aparezcan en el ambiente de producción. Existen organismos internacionales de estandarización los cuales presentan distintas maneras de hacer la implementación de los procesos de pruebas, enfocándose en ser genéricos en ciclos que realicen la medición y optimización, encargándose de asegurar la calidad del software en sus diferentes etapas de desarrollo.

El desarrollo de proyectos de software en una empresa o institución (ya sea como herramienta interna o producto) obliga a asegurar que lo que se está realizando esté encaminado en cuanto a la calidad y el desarrollo seguro de software, logrando cumplir los objetivos y satisfacción que califica el cliente el cual obtiene el máximo beneficio de un producto terminado.

En una empresa o institución conviene tener presente los entornos de pruebas y un modelo eficiente de calidad de software, porque cuando se concluya el software, el usuario será quien revise detenidamente, y si se presenta alguna duda o desconformidad, el Tester (quien tiene conocimiento del software a fondo) es el encargado de verificar las dudas, hablar con el usuario y en dado caso ayudarlo a aclarar sus ideas. De aquí se puede deducir que los contenedores benefician con su utilidad a mejorar los servicios y prestaciones de la PUCE-Esmeraldas.

Objetivos

General:

- Evaluar diferentes tecnologías de contenedores de software, según la norma ISO 9126 para la implementación conjunta facilitando al desarrollador el despliegue de aplicaciones en un ambiente de pruebas en el departamento TIC de la PUCESE.

Específicos:

- Realizar un estudio de las arquitecturas basadas en los contenedores para entornos de pruebas especificando una comparación sobre sus configuraciones, almacenamiento y velocidad.
- Identificar los servicios más representativos que se brindan en este modelo en base al análisis de aplicaciones.
- Determinar las tecnologías que permitan automatizar el despliegue y la instalación de la arquitectura de contenedores en el análisis del proceso de pruebas.

CAPÍTULO I

MARCO DE REFERENCIA

1.1. Antecedentes

Al referirse a la calidad de software, siguiendo el lineamiento de esta investigación es importante tener en cuenta las pruebas de software, existen investigaciones que apuntan con mucha importancia las pruebas en el proceso en el desarrollo de software. La retroalimentación que se da al revisar los cambios de los proyectos de software; desde el desarrollo hasta la forma en las que se le aplican las pruebas y validaciones.

Según Ciolli (2007), para asegurar la calidad de software se requieren que los casos de prueba puedan ser reusables a través de las distintas versiones operativas del software, por lo que aquellos cuya funcionalidad haya quedado obsoleta deben ser desechados.

En Ingeniería de software, la calidad es un aspecto importante, estipulando que las pruebas deben de ejecutarse desde el desarrollo, formando así, parte de las características evaluadas para garantizar que un producto sea de calidad.

De acuerdo con Mendoza (2010), se debe definir un modelo de referencia para la selección de herramientas de pruebas como soporte en el proceso de desarrollo de software, que contribuyan a mejorar el proceso productivo de las empresas y elevar la calidad de sus productos, tomando como caso de estudio las PYMES (Pequeñas y Medianas Empresas) desarrolladoras de software.

Existen modelos de procesos de pruebas validadas que pueden ser adaptadas al modelo con el que se va a evaluar con la finalidad de cubrir y optimizar todas las actividades y las tareas planteadas. De acuerdo con Akella & Rao (2011), el Modelo-V define un procedimiento al momento de desarrollar aplicaciones presentando el proceso de pruebas y las actividades que se sigue para realizar la verificación y validación (Ver ilustración 1) de todos los niveles de desarrollo para comprobar los resultados de los niveles de desarrollo con lo establecido desde un inicio.

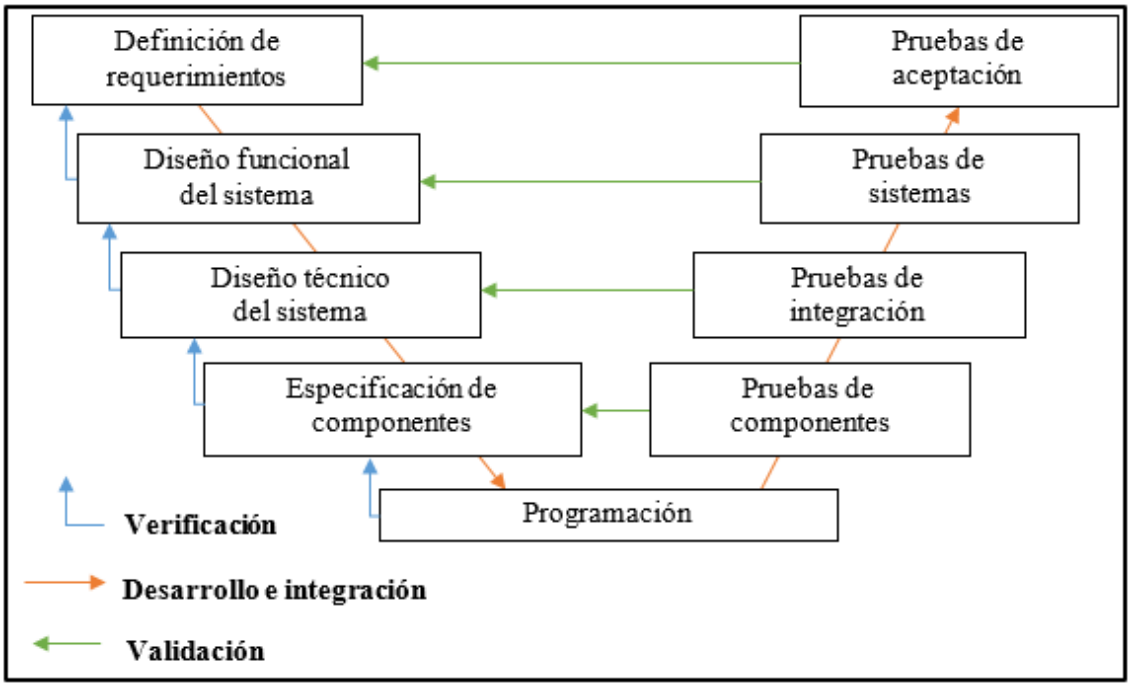

Ilustración 1 Modelo V

Cardona (2009), menciona en su investigación que las pruebas son un medio para asegurar la calidad del producto ya que permite identificar los defectos. Es por ello que las pruebas se toman en cuenta en todas las etapas de desarrollo del software.

Gibert & Peña (2005) mencionan que desde 1985, han ido apareciendo herramientas, metodologías y tecnologías que se presentaban como la solución definitiva al problema de la planificación, previsión de costes y aseguramiento de la calidad en el desarrollo de software.

Es importante resaltar que cada una de las metodologías basadas en pruebas de software existentes, son producto de necesidades y requisitos específicos presentados en diferentes lugares del mundo. Por eso no se tiene un estándar único que se pueda aplicar en cada uno de los procesos al momento de desarrollar un software.

1.2. Bases teóricas - científicas

1.2.1. Calidad de software

La Calidad de Software según Rogers (2005): "es la concordancia con los requisitos funcionales y de rendimiento establecidos con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado de forma profesional".

Existen estándares internacionales como la ISO 25000:2005 SQuaRE (*Software Product Quality Requirements and Evaluation*), que permiten la especificación de los requisitos y evaluación de la calidad del software, proporcionando una guía para el uso de una serie de estándares Internacionales, tales como:

- ISO/IEC 9126 donde establece un modelo para la evaluación de la calidad de software mediante características de la calidad, existe un modelo llamado MECABIC que hace uso de una matriz para evaluar software (Ver Anexo 3) que permite concentrarse en características que dependen exclusivamente de la arquitectura, y al ser un estándar facilita la correspondencia con características de calidad consideradas por los métodos estudiados.
- IEEE 829-2008 sigue un plan adecuado para evitar los posibles problemas que puedan presentarse en los sistemas informáticos y otros tipos relacionados, desarrollando una metodología de prueba, que consta de planes de pruebas para así diseñarlas después, ejecutarlas esperando los resultados para poder en la siguiente fase evaluar las pruebas.
- ISO/IEC 20119 se encarga de definir el proceso de que se debe seguir para la puesta en marcha de las pruebas de software, identificando las estrategias y políticas, recopilar la terminología, documentación y técnicas para todo el ciclo de vida de pruebas del software.

1.2.2. Pruebas de software

Según Myers, Sandler, & Badgett (2011) "las pruebas son un conjunto de procesos ejecutado en un software teniendo como objetivo encontrar errores"; además, de acuerdo con Tian (2005) el realizar pruebas es una característica fundamental para el respectivo

aseguramiento de calidad del software (QA), la cual ayuda a asegurar que el servicio o producto cumpla con todos los requisitos establecidos por el cliente.

La industrialización de software se enfoca en tres características esenciales con el proceso de producción de software, tales como el tiempo, los costos y la calidad; donde al referirse al tiempo y a los costos conceptualiza que al momento de realizar la planificación de proyectos de software ocurren problemas de tiempos estimados, lo cual provoca altos costos; y al hablar de la calidad hace referencia en que la competencia en el mercado de software, hace que este aspecto tome fuerza convirtiéndolo en característica principal a la hora de adquirir un sistema lo que hace que el tiempo en el proceso de mantenimiento disminuya y que presente pocos errores y considerados como los más confiables.

Las pruebas son de importancia según lo que se presentan en las siguientes referencias:

- Según Dustin (2002), las pruebas de software permiten trasladar de manera confiable del ambiente presentado por la ingeniería de software, es decir del ambiente de análisis, diseño y construcción, a entornos de producción reales donde se verifica hacer que el software esté acto.
- Según Gao, Tsao, & Wu (2003), las pruebas de software que se basan en componentes permiten la reutilización y la reducción de las actividades en los ciclos de pruebas, lo cual se denota en la disminución de tiempo y costos.
- Según Burnstein (2006), la necesidad de adquirir software de alta calidad obliga a cuantificar e identificar factores de calidad tales como: capacidad de uso, mantenimiento, prueba, confiabilidad, de ser medible y al desarrollo de prácticas y mejoras en ingeniería que contribuyen a la adquisición de productos con requisitos de alta calidad.

Antes de poner una aplicación en producción hay que verificar que se cumplan con los requisitos planteados, y así evitar y corregir los errores que se presentes en cualquier actividad o etapa del desarrollo.

El objetivo de las pruebas de software consiste en obtener información sobre errores posibles para que no afecten la calidad del producto. Estás pruebas conforman parte del proceso y actividades de control de calidad de software. Las pruebas en sí, son un conjunto de actividades en el proceso de desarrollo de software y según el tipo de pruebas,

se pueden implementar las actividades o procesos necesarias para asegurar la calidad al momento del desarrollo.

¿Cómo y cuándo empezar el proceso de las pruebas?

Según Cardona (2009), teniendo presente el proceso de desarrollo de una aplicación, una vez realizado el análisis y diseño se deben aplicar de forma conjunta las etapas de pruebas(Ver ilustración 2), donde se requiere un ambiente de pruebas, un ambiente distinto al de desarrollo y el de producción cuya finalidad es de proveer una buena infraestructura de software y hardware necesarios para la ejecución de pruebas de integración, sistema, rendimiento y funcionales verificar el comportamiento del software aplicado en escenarios reales. Referencialmente este ambiente de pruebas debe ser similar al ambiente de producción, el cual permitirá obtener los resultados más acertados a la realidad. Se puede apreciar el proceso de pruebas en la ilustración 2.

Según Burnstein (2006), "debe existir un equipo de testeadores que tengan las herramientas para realizar las pruebas de una forma eficiente, y que la búsqueda de fallos trabaje en conjunto con la metodología que se adapte a los requerimientos de la aplicación".

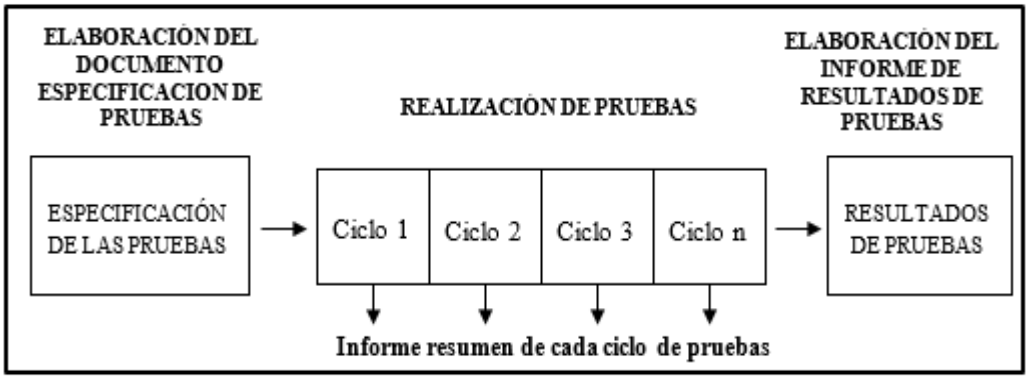

Ilustración 2 Esquema del Proceso De Pruebas. Fuente: Cardona (2009)

1.2.2.1. Entornos de Pruebas

Para la ejecución de las pruebas de una aplicación es necesario tener preparado un entorno de pruebas (Ver ilustración 3). Este entorno de pruebas tiene los siguientes componentes:

- Hardware.
- Software.
- Instalaciones de comunicaciones.

- Herramientas para la creación y el uso de datos de prueba.
- Procedimientos.
- Casos de pruebas

El entorno de pruebas se establece para poder hacer pruebas de forma óptima. Esto influye en la medida de la calidad, duración y costo de los procesos de pruebas. Algunos puntos de importancia en el entorno de pruebas son: roles y responsabilidades, control, disponibilidad suficiente y a tiempo, flexibilidad y representatividad de los entornos reales de producción.

Las pruebas se deben realizar al servidor, a la base de datos y a las comunicaciones, donde las herramientas que participen tengan la capacidad de medir, de presentar métricas y obtención de resultados.

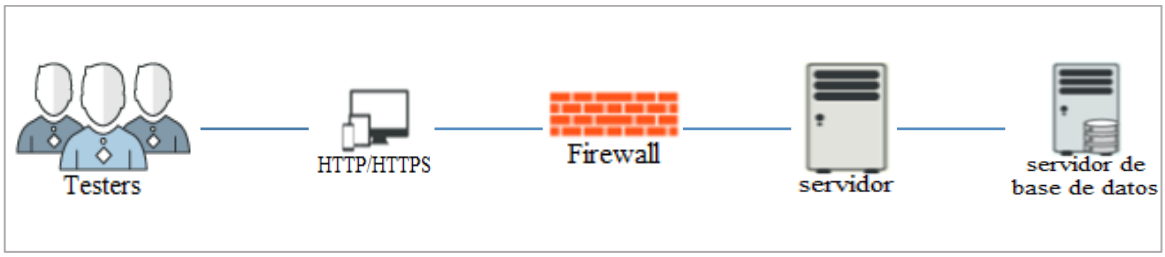

Ilustración 3 Arquitectura estándar de pruebas

1.2.2.2. Principios de diseño en la arquitectura del entorno de pruebas

Las características de calidad de diseño son importantes para probar el diseño de la arquitectura según los siguientes tipos de pruebas:

- Pruebas funcionales o de caja negra
- Pruebas no funcionales o de caja blanca

Las pruebas funcionales o de funcionamiento al momento de ser ejecutadas evalúan cada una de las características del software, buscando si la solución satisface las necesidades del cliente, si realiza correctamente el funcionamiento que se espera y si es compatible a la adaptación de nuevos cambios o versiones.

Las pruebas no funcionales se encargan de la revisión de las características implícitas del software. Enfocándose en las características del software, hay pruebas no funcionales que encargan de la medición de aspectos tales como:

- Pruebas de integración
- Pruebas de rendimiento

Las pruebas de integración y de rendimiento son complementarias a las funcionales, por lo que es necesario probar el funcionamiento del software y verificar diferentes aspectos y características, como pruebas de rendimiento, mantenimiento, instalación y portabilidad.

1.2.2.3. Automatización de pruebas

Las aplicaciones informáticas evolucionan y la capacidad de validación de los componentes de un software tienden a reducirse, por eso se usan herramientas que sean capaces de asegurar que lo que se ha codificado, siga con su funcionamiento, para que cuando se aplique un cambio, actualización o mejora a futuro no alteren el comportamiento de una aplicación de forma inesperada. Cuando se hace una modificación o actualización a un software sin realizarle pruebas, al momento de realizar un despliegue a producción, podrían aparecer errores o daños en cualquier otro componente del sistema.

El mayor aliado de los controles de calidad son las pruebas automatizadas, los objetivos más representativos de las pruebas automatizadas, son los siguientes:

- Las Pruebas ayudan a asegurar la calidad
- Las pruebas ayudan a comprender el sistema que está siendo probado
- Las pruebas reducen (y no introducen) el riesgo de fallos críticos
- Las pruebas deben ser fáciles de ejecutar
- Las pruebas deben ser de fácil lectura y de mantenimiento

1.2.3. Proceso de despliegue

El despliegue se realiza cuando el código haya pasado por todas las pruebas del software, y no contengan errores críticos siendo aprobado para darle de alta, además de haber sido distribuido en el área de producción (Ver ilustración 4).

La mejora y mantenimiento de un software que tiene problemas en el proceso de despliegue, podrían llegar a requerir más tiempo del que inicialmente se había dispuesto para el desarrollo del mismo. En el proceso de solución del problema, la codificación generada podría no ajustarse al diseño inicial, lo que podría resultar en un problema mucho mayor, ya que se debería rediseñar el programa para que los cambios que se hagan con el afán de solucionar los problemas posteriores al desarrollo, se ajusten de manera que el diseño y funcionamiento sea fluido, y esto implica un costo de mantenimiento muy elevado, lo cual puede resultar contraproducente a nivel económico. Se presenta el proceso de despliegue de software presentado por IBM:

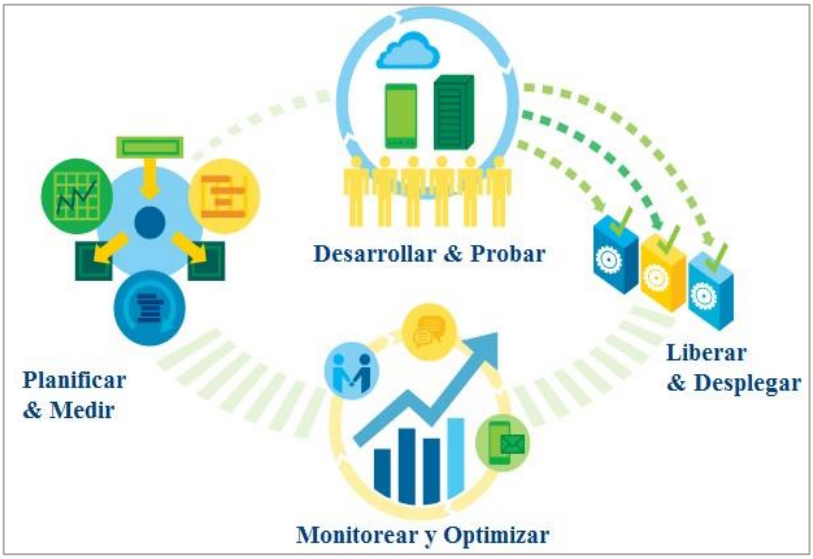

Ilustración 4 Proceso de despliegue de software. Fuente: IBM (2017)

1.2.3.1. DevOps: mejores prácticas en la gestión de software

DevOps es un acrónimo inglés de Development (desarrollo) y Operations (operaciones). En términos presentados por IBM (2017) en un cambio cultural dentro de la empresa y los departamentos de tecnología es:

- Una metodología para creación de software
- Se basa en la integración entre desarrolladores de software y administradores de sistemas
- Permite la fabricación de software de manera más rápida, con mayor calidad, menores costos y una alta frecuencia de entregas

Al adaptarse a esos cambios, se obtiene ventajas agiles tales como:

- Reducción del tiempo al desplegar el software
- Mejoramiento del ciclo de vida del software
- Contribución con la mejora en los ciclos de integración
- Fácil utilización de recursos en procesos de entrega continua
- Reducción de inconsistencias y errores

Pero hay que tener en cuenta que estas ventajas se pueden automatizar con la adquisición de herramientas tecnológicas y técnicas facilitadoras:

- Herramientas que permitan el despliegue automatizado
- Herramientas para dar al desarrollador autoservicios
- Herramientas para la integración de productos y entrega continua
- Herramientas que ayuden en la gestión de la configuración
- Softwares automatizados y catálogo de servicios
- Herramientas de control, monitorización y entornos de colaboración

Prácticamente, DevOps es una solución al desarrollo de aplicaciones, el control de los parámetros de calidad que aseguren su garantía la calidad del producto y las operaciones de TI que intervienen en él. Ayudando así a las organizaciones en los servicios de software y producción de software de forma rápida.

1.2.3.2. Enfoque de Automatización de despliegue de software

La "Automatización de despliegues" busca simplificar, agilizar y automatizar punto a punto el proceso de despliegue de las aplicaciones de negocio, al tiempo que habilita una estrategia de entrega continua desde el ambiente de desarrollo hasta producción, es decir, tener la capacidad de orquestar de manera centralizada todas las herramientas involucradas en el ciclo de vida del software y automatizar la ejecución entre los ambientes.

Humble & Farley (2010), mencionan que el despliegue de aplicaciones es importante porque:

- Evita errores provenientes de las operaciones sobre los sistemas, los cuales no son fáciles de encontrar
- Acorta el tiempo a consumir

- El control de documentación es complejo y consume tiempo al involucrar varias personas. Los procesos automatizados funcionan como la documentación, porque después el despliegue no funcionará
- En la mayoría de los casos el despliegue manual requiere de un personal experto para la ejecución de las acciones. Con la automatización se libera del trabajo excesivo a los operadores de los sistemas y todo el proceso se encuentra explícito
- Comprobar un proceso manual es costoso, la única forma de llevarlo a cabo es ejecutándolo de esta misma forma. La automatización maneja el proceso de forma sencilla
- El proceso automatizado se logra revisar totalmente

Acelerar la entrega de software corresponden a la agilidad de llevar las ideas de innovación a servicios en producción de forma automatizada. Sin embargo, los métodos tradicionales de liberación de servicios poseen procesos intensivos en tiempo, recursos y costos, los cuales se derivan de la dependencia en personas que contienen el conocimiento especializado, extensos procesos y flujos de aprobación, intervenciones manuales y posibles errores humanos, tiempos de inactividad de los equipos de trabajo, documentación desactualizada de los procesos, entre otros. A esto se le suma la complejidad de la arquitectura de las aplicaciones y el trabajo no colaborativo entre los equipos de desarrollo y operación.

1.2.3.3. Ambientes para el proceso de pruebas y despliegue de software

Según Bass (2007), “La arquitectura de software consiste en la estructura o sistema de estructuras, que comprenden los elementos de software, las propiedades externas visibles de esos elementos y la relación entre ellos”.

En la investigación de Kazman, Klein, & Clements (2001), la Arquitectura de Software es una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. Considerada como la estructura del sistema, consistente en elementos y la relación entre

ellos. Al describir las Arquitecturas de Software es importante identificar cuáles son los elementos de interés y seleccionar una manera apropiada para describirlos.

Una Arquitectura de Software ejerce influencia notable sobre la calidad del sistema que se implementa por eso nace la necesidad de evaluarla y determinar si la arquitectura cumpla con los requisitos de calidad que se exige.

1.2.4. Contenedores (Entornos Aislados o Virtualizados)

Los contenedores proporcionan un entorno aislado dentro del sistema operativo del servidor que se conoce comúnmente como la virtualización de ordenadores a nivel de todo un sistema operativo, según Morabito (2016), "Un contenedor es un entorno de ejecución independiente que comparte el kernel del sistema host y que (opcionalmente) está aislado de otros contenedores del sistema".

Un contenedor puede considerarse como un entorno virtual aislado, que incluye un conjunto de dependencias específicas necesarias para ejecutar una aplicación específica.

Uno de los usos principales de los contenedores en lo ligero que son, es su versionamiento aislado.

Las máquinas virtuales pueden crear una nueva instancia de un sistema operativo para cada máquina virtual mientras que los contenedores requieren la aplicación y sus dependencias, mientras que el kernel es compartido entre ellos, donde al iniciar un contenedor tiende a ser mucho más rápido que al iniciar una máquina virtual (Ver ilustración 5).

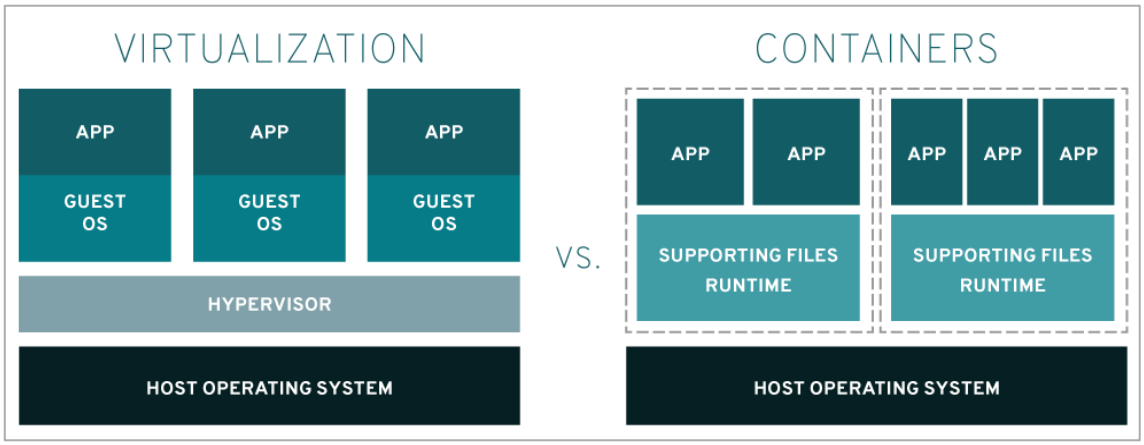

Ilustración 5 Máquina Virtual frente a contenedores. Fuente: Redhat (2017)

A continuación, se compara los contenedores frente a las máquinas virtuales:

Tabla 1 Comparación de Contenedores frente a Máquinas Virtuales

MÁQUINAS VIRTUALES	CONTENEDORES

Beneficios de las máquinas virtuales:
• Mejor uso de los recursos comunes
  ○ Una máquina física dividida en múltiples máquinas virtuales
• Fácil de escalar
• Máquinas virtuales en la nube
  ○ Elasticidad rápida
  ○ modelo de pago por uso
 	
Limitaciones:
• Cada máquina virtual requiere
  ○ Asignación de CPU
  ○ Almacenamiento y RAM
  ○ Un sistema operativo huésped
• Entre más máquinas virtuales ejecute, más recursos necesita
• Un Sistema operativo huésped significa pérdida de recursos
• La portabilidad de la aplicación no está garantizada.
	Beneficios de los contenedores:
	• Separación de preocupaciones
	  ○ Los desarrolladores se enfocan en construir sus aplicaciones
	  ○ Los administradores de sistemas se enfocan en la implementación.
	• Portabilidad de la aplicación
	  ○ Construido en un entorno, entregado en otro
	• Escalabilidad
	  ○ Fácil despliegue de nuevos contenedores si es necesario
	• Ejecución de más aplicaciones en una sola máquina
	• Versionamiento aislado, fácil de crear, remover y duplicar.
	
	Limitaciones:
	• Algunas aplicaciones no se pueden implementar en contenedores
	• Los contenedores pueden no ser adecuados si sus aplicaciones están estrechamente relacionadas con sus datos, o si la administración de datos es un enfoque clave de su aplicación.

CAPÍTULO II

METODOLOGÍA

2.1. Descripción y caracterización del lugar

La PUCE-Esmeraldas es una institución de Educación Superior, se encuentra ubicada en la calle Espejo subiendo a Santa Cruz, centro de la ciudad de Esmeraldas del cantón Esmeraldas, de la Provincia de Esmeraldas, donde funciona el departamento de TIC, en el cual se llevó el enfoque del proceso de pruebas de software para esta investigación.

2.2. Tipos de investigación

Según los objetivos planteados, se considera el tipo de investigación tecnológica, debido que, para el uso del servicio de contenedores de software para un ambiente de pruebas, es necesario partir de contenidos teóricos-prácticos realizados en diferentes investigaciones, que permiten la implementación de un contenedor para las respectivas pruebas.

Es bibliográfica por lo que está basada en las fuentes extraídas de: artículos científicos, tesis de grado en maestrías, entre otros medios, que permitieron estructurar y desarrollar el marco teórico del trabajo de investigación, en el cual se detallan los beneficios de implementar contenedores de software y la importancia de estos para mejorar la calidad de las aplicaciones, llegando a la conclusión, de acuerdo al método inductivo, que todas las entidades que desarrollan software para su propio uso o venta del mismo deben contar con un software que permita optimizar los recursos.

La investigación exploratoria forma parte, ya que permite conocer sobre el uso de la ingeniería del software, contemplando dentro de éste la organización, las métricas, técnicas, estándares y las prácticas enfocadas en la producción de software de calidad, la adquisición de software es un proceso que se trabaja como parte de las tecnologías de la información y las comunicaciones, al igual que la accesibilidad en el software y el estudio de la interacción, teniendo lineamientos para acceder al uso de tecnologías de información a una cantidad considerable de personas sin importar si tienen alguna limitación.

2.3. Métodos y técnicas

2.3.1. Métodos

Para la presente investigación se emplearon los métodos: inductivo, deductivo y cuali-cuantitativo con propósitos que se detallan a continuación:

- Método Inductivo: Este método permitió mediante la observación de los hechos para su registro; la clasificación y el estudio; la derivación inductiva que parte de los hechos específicos y permite llegar a una generalización eficaz sobre la investigación para determinar la propuesta de modelo, proceso y herramienta para el departamento de TIC de la PUCE Esmeraldas.
- Método Deductivo: Este método permitió la validación del modelo, proceso y herramienta acorde a los requerimientos obtenidos a través de las distintas técnicas utilizadas en esta investigación.
- Método cuali-cuantitativo: Es cualitativa porque se realizó la interpretación de los datos de una forma particular, es decir la información que se recopiló fue con el propósito de analizar a los beneficiarios para conocer la forma en que ven los problemas o fenómenos que se dan en la institución objeto de estudio.

  Es cuantitativa porque los datos obtenidos fueron procesados y analizados de forma numérica a través del uso de herramientas estadísticas para realizar los procedimientos de medición y control. Es sustancial recalcar que los modelos estadísticos de estos procedimientos permitieron explicar desde otro punto de vista parte de lo que aconteció con la investigación.

2.3.2. Técnicas e Instrumentos

Para la recolección de datos del presente trabajo de investigación, según la investigación de campo se emplearon algunas técnicas definidas por Sampieri, Collado, & Lucio (2003):

- Entrevistas

Se llevaron a cabo entrevistas al área de calidad y otras áreas (desarrollo, infraestructura y control de software) según documento que se basa en la ISO/IEC/IEEE 29119 Software Testing para la verificación de calidad de software que se desarrolla en una empresa y

así recopilar información de forma directa a través de un diálogo que tuvieron preguntas cualitativas y cuantitativas, algunas de las preguntas con respuestas de escala tipo Likert de 5 a 1, donde el 5 es la valoración más alta y el 1 la más baja, para poder obtener valores estadísticos tabulados y procesados.

Las respuestas a las preguntas son escalables ya que tiene el riesgo de que las respuestas o palabras usadas no sean claras, es por ello que las preguntas deben plantearse de acuerdo a la naturaleza de la investigación y a las personas que serán entrevistadas. (Ver Anexo 1)

- Ficha de Observación

Según Sampieri, Collado, & Lucio (2003), "La observación consiste en el registro sistemático, válido y confiable del comportamiento o de la conducta manifiesta, la cual puede utilizarse en muy diversas circunstancias".

Con la técnica de observación, se puede realizar una revisión, registrando, controlando y analizando los hechos de interés.

En esta investigación para obtener información de la calidad del software se hizo uso de una matriz con el método MECABIC(Anexo 2) que está basada en los aspectos de evaluación de calidad estipulados en la ISO/IEC 9126, para obtener resultados del proceso que lleva cada una de las tecnologías de contenedores de software al momento del desarrollo del software y del ambiente de pruebas, y aplicando casos de pruebas utilizando la tecnología de contenedores que sea elegida para usar en el departamento de sistemas, según ambientes en Linux y Windows que utilizan.

2.4. Población y muestra de estudio

La población que se eligió corresponde al número de personas que integraron los miembros administrativos entrevistados, es decir un total de 2 personas, el jefe del departamento y al desarrollador de las aplicaciones; por lo tanto, debido a que la población es muy pequeña no hubo necesidad de aplicar la técnica de muestreo.

2.5. Descripción y validación del instrumento

Los datos obtenidos de las entrevistas fueron analizados rigurosamente elaborando una síntesis con puntos claves para la presente investigación; las respuestas de los datos obtenidos a través de la aplicación de este instrumento, incluyeron preguntas específicas y relevantes considerando inquietudes propias sin dejar de lado los objetivos claros de la entrevista. La entrevista acorde al modelo descrito anteriormente facilitó la obtención de requerimientos e información para establecer una propuesta que supla con las inquietudes y que ayude a la mejora de los procesos al momento de montar un ambiente de pruebas.

El proceso de validación de los instrumentos fue desarrollado con el asesor del presente proyecto antes de su aplicación, para verificar si la información concluida complementa con los objetivos previamente planteados.

2.6. Técnicas de procesamiento y análisis de datos

Se llevaron a cabo 2 entrevistas, las cuales fueron efectuadas al encargado del desarrollo de las aplicaciones y al Jefe del departamento de TIC de la PUCE Esmeraldas los días 4 y 5 de julio del 2017.

Además, se analizó y evaluó 5 tecnologías, 4 de ellos para distribución Linux, y 1 de distribución Windows, mediante la matriz MECABIC basada en la norma ISO/IEC 9126; donde están incluido tanto las características como las sub-características de la norma.

De tal modo que sirva para determinar la selección de las tecnologías necesarias para las necesidades del departamento de TIC de la PUCESE.

2.7. Normas éticas

Se utilizaron entrevistas donde la información fue utilizada únicamente para el desarrollo de la investigación. Los datos y los resultados obtenidos en esta investigación guardan absoluta reserva y no serán divulgados sin consentimiento expreso de las personas involucradas bajo ningún concepto.

Este documento fue desarrollado bajo las normas éticas de la universidad y también guardan absoluta relación con el reglamento de grado.

CAPÍTULO III

RESULTADOS

3.1. Análisis e interpretación de datos

3.1.1. Análisis de las entrevistas

Las entrevistas al jefe y al desarrollador del departamento de TIC de la PUCE-Esmeraldas, permitieron desarrollar el marco teórico, observar la problemática y así permitir la elaboración de la propuesta.

Según la entrevista realizada al desarrollador del departamento, elaboran 2 tipos de aplicativos en web y escritorio con lenguajes de programación open source en ambientes de Linux y de paga en ambientes de Windows.

Los tipos de pruebas que se realizan en el departamento son funcionales y unitarias mediante la herramienta PHPUnit y con dicha herramienta realizan las pruebas de aceptación en conjunto con la tecnología de despliegue automatizado llamada Jenkins.

No cuentan con documentos que informen o detallen el proceso de pruebas, solo obtienen los requerimientos del cliente, como el tema de pruebas es nuevo para el equipo se tiene un escenario de prueba realizado por la persona que se encarga del desarrollo.

No existe clasificación de errores por documentación o detalles de incidentes, pero comenzaron a realizar el control de versiones con la herramienta que les permite el despliegue.

Según la entrevista realizada al Jefe del departamento se tomaron las preguntas cuantitativas de las que se obtuvo los siguientes resultados:

En el departamento de TIC se encargan de 60 aplicaciones entre web y de escritorios, donde las aplicaciones web tienen su información almacenada en SQLServer y MySQL, y las de escritorio para Windows donde sus datos se encuentran en Microsoft Access y SQLServer.

La universidad cuenta con 38 aplicaciones web, donde 3 de éstas aplicaciones han sido sometidas a pruebas, y de las 22 aplicaciones de escritorio, ninguna se ha sometido a pruebas, a pesar de que se han puesto a prueba 3 aplicaciones ninguna cuenta con la documentación de resultados de pruebas, pero cuentan con un respaldo con versionamiento en un repositorio donde tiene la información de forma privada.

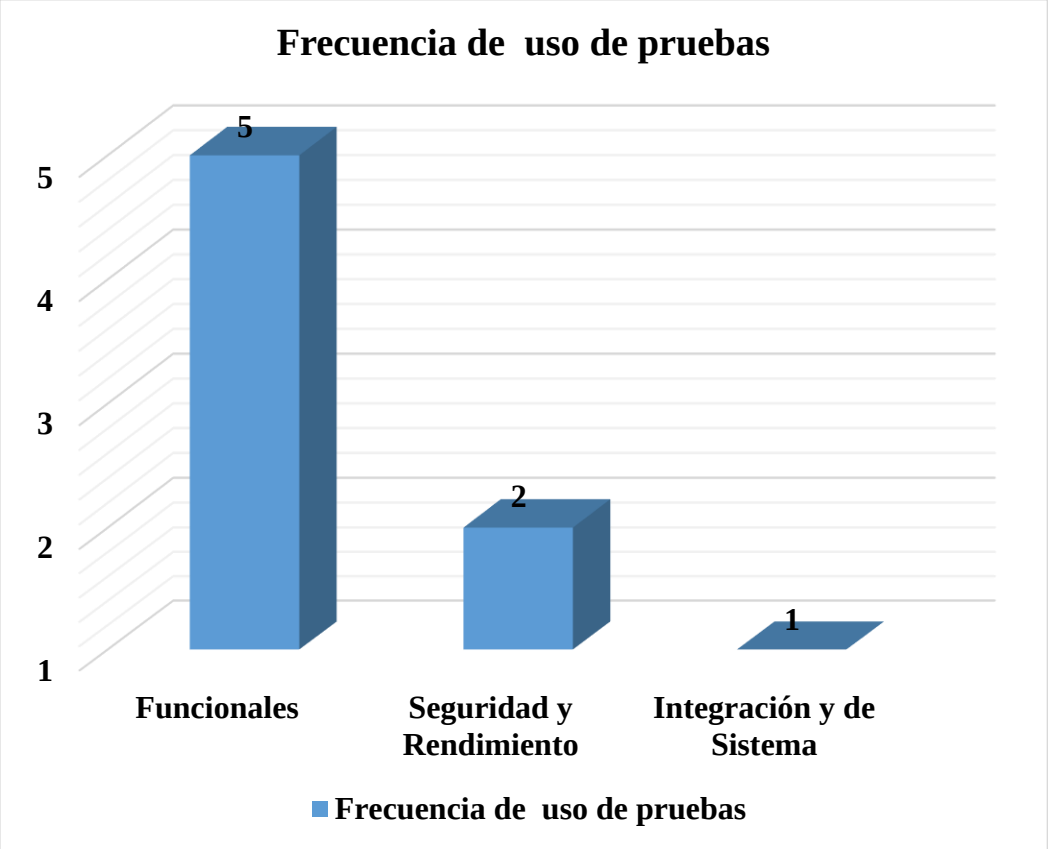

Ilustración 6 Frecuencia de uso de pruebas en el departamento de TIC

Según la ilustración 6, con la calificación de 5, siempre se ha realizado pruebas Funcionales, pero de una manera empírica, es decir no controladas. Con la calificación de 2, es decir rara vez se ha sometido a pruebas de seguridad y rendimiento, una vez se sometió la plataforma de Moodle de la universidad a este tipo de pruebas, pero no se obtuvo los resultados esperados. La tercera barra del gráfico con la calificación de 1 demuestra que nunca se ha realizado pruebas de integración y de sistemas.

3.1.2. Preparación de los ambientes de prueba

Para realizar la selección de la tecnología de contenedores se usaron 2 equipos del departamento de TIC que usan para realizar desarrollo y pruebas de software, que soporten las siguientes características:

Tabla 2 Características de los contenedores de software

Características	***LXC***	***Windows Container***	***OpenStack***	***CoreOS RKT***	***Docker***
SO compatibles	Linux	Windows	Linux	Linux	Linux/Windows
Versión	2.0	10.0	15.0.6	1.29.0	17.03.0
Licencia	Open Source	Software Propietario	Open Source	Open Source	Open Source

Fuente: Páginas oficiales de cada tecnología.

Para configurar los contenedores, se instaló sistemas operativos compatibles en cada uno de los equipos con las siguientes características:

Tabla 3 Características de los sistemas operativos utilizados.

Características	***Windows***	***Linux***
Sistema operativo	Windows Server 2016 GA 64bits	Ubuntu Server 16 LTS 64bits
Versión	10.0.1	16.04.3
Licencia	Software Propietario	Open Source

Se preparó 2 ambientes (Windows y Linux) en cada uno de los equipos, donde la configuración de cada servidor fue la siguiente:

Tabla 4 Características de los equipos de prueba.

Características	***Equipo 1***	***Equipo 2***
Procesador	Core i7 3,60 GHz	Core i5 2,5 GHz
RAM	6 Gb	4 Gb
Disco Duro	500 Gb	500 Gb

3.1.3. Análisis de las tecnologías de contenedores según ISO/IEC 9126

Se clasificó las características de las tecnologías de contenedores según el requerimiento de sus configuraciones, la velocidad y almacenamiento para su funcionamiento.

A partir de la ejecución de cada tecnología, se evaluó la parte técnica de éstas, mediante la matriz que se propone el método MECABIC basado en la ISO 9126.

En la tabla 5 se observan porcentajes de ponderación, los cuales dan como resultado 100% para cada característica, estipulando los porcentajes para cada sub-características:

Funcionalidad

El sistema provee las funciones necesarias que responden a las necesidades para cumplir con los requisitos funcionales.

- Adecuación: 20% por los recursos que se adhieren a los procesos de las tecnologías con respecto al SO
- Exactitud: 20% por la capacidad de presentar resultados funcionales coherentes
- Interoperabilidad: 20% por el control de sistemas múltiples
- Seguridad: 40% por el asegurar que los componentes no pierdan datos

Usabilidad

El sistema es interactivo, lo cual permite el fácil uso del manejo de la aplicación.

- Documentación: 40% por el soporte para la implementación de los servicios
- Operabilidad: 40% por la capacidad de utilizar el sistema sin mucho esfuerzo
- Interfaz Gráfica: 20% por el uso amigable de la interfaz de la aplicación

Fiabilidad

El sistema puede mantener un nivel de rendimiento, sea que esté bajo ciertas condiciones, partiendo de la base que este sistema se puede adaptar a cualquier necesidad que requiera.

- Tolerancia a fallos: 60% por basarse en que las operaciones sean correctas bajo condiciones adversas
- Capacidad de recuperación: 40% por la capacidad solucionare problemas con los servidores, haciendo que los componentes continúen prestando sus servicios

Eficiencia

El sistema tiene buen un rendimiento dependiendo de las características del hardware

- Comportamiento en el tiempo: 50% por la capacidad de respuesta del sistema
- Utilización de recursos: 50% por controlar los componentes de forma eficiente

Mantenibilidad

Adaptación a nuevos requisitos y especificaciones de software.

- Acoplamiento: 40% por la capacidad de funcionamiento a pesar de tener cambios
- Modularidad: 40% por permitir que el sistema sea estudiado, visto y entendido
- Facilidad de pruebas: 20% por la capacidad de permitir pruebas de software

Portabilidad

El software puede ser transferido a otro entorno.

- Adaptabilidad: 20% por soportar otros entornos similares
- Facilidad de Instalación: 35% por la capacidad de ser instalado con normalidad
- Coexistencia: 25% por adaptar su funcionamiento con otros sistemas
- Reemplazamiento: 20% por administrar la facilidad de reemplazo de software con otro similar

Según lo evaluado en la tabla 5, se detalla las sub-características con cada tecnología abarcada en las características que se presentan a continuación:

Funcionalidad

- Adecuación: LXC puede afectar los recursos para otros contenedores alojados en la misma máquina necesitando de servicios adicionales para equilibrar los recursos, igual que OpenStack el cual se ejecuta con LXC como hipervisor donde sus servicios están en la nube, haciendo que algunas funciones de control de pruebas falten.

  CoreOS RKT ejecuta sus entornos en paralelos basado en Docker bajo el mismo kernel, con otros recursos aislados por el contenedor.

Contenedores en Windows basado en Docker, utiliza las librerías de estos contenedores para implementarlos en su SO, adecuando sus recursos según los entornos que se ejecuten.

- Exactitud: LXC excederse en con el uso de recursos para demanda de resultados, utilizando su propia librería (liblxc) la cual no trabaja con procesos en paralelos, haciendo que sus datos de entrada se pierdan, OpenStack al trabajar con LXC mejoró la librería para sus operaciones ya que están destinadas para operaciones privadas necesitaban que los procesos se ejecuten en paralelo para que los resultados sean exactos.

  Docker basado en LXC creó su propia librería (libcontainer) que proporciona un entorno de ejecución en múltiples operaciones, pero al tener ocupado un proceso por algún otro contenedor puede confundir el llamado, tomándolo como inconsistente.

  CoreOS RKT usa un modelo de proceso en contenedor similar al modelo de proceso Docker.

  Contenedores en Windows manejan procesos basados en Azure (en la nube) como la hace Docker, pero apila las librerías con mayor frecuencia en los procesos de los entornos.

- Interoperabilidad: LXC por diseño no se pueden tener interacción con servidores en modo kernel, solo maquinas con OS compatible con el núcleo del anfitrión, mientras que OpenStack actúa con componentes que utilizan recursos de infraestructura en nube para proporcionar mayor funcionalidad a aplicaciones de más alto nivel, pero no al nivel dedicado de un contenedor en Docker, el cual actúa como un motor de contenedor portátil para empaquetar aplicaciones y dependencias en contenedores fácilmente implementables en cualquier sistema.

  CoreOS RKT es una alternativa más segura, interoperable y abierta a Docker.

  Contenedores en Windows es seguro, al modo que mejoraron la seguridad de los contenedores en Docker, ya que sus desarrolladores vieron ciertas falencias en la tecnología. En Windows mediante la nube Azure se pueden interactuar con otros sistemas.

- Seguridad: LXC no es segura para entornos de múltiples usuarios, haciendo que las vulnerabilidades existentes dentro de los contenedores podrían potencialmente

otorgar privilegios de superusuario al atacante, a diferencia de OpenStack que, al ser una tecnología con privilegios de contenedores en nube, mantiene una protección potente con los servicios.

CoreOS RKT se caracteriza por enfocar los servicios a la seguridad de gestión de infraestructuras de las imágenes mientras que Docker no tenía forma de verificar la autenticidad de una imagen de servidor para firmar y verificar automáticamente la firma de un editor.

Contenedores en Windows es seguro, al modo que mejoraron la seguridad de los contenedores en Docker, ya que sus desarrolladores vieron ciertas falencias en la tecnología, haciendo que cada proceso herede servicios de los módulos de seguridad dedicado.

Usabilidad

- Documentación: LXC no tiene una comunidad de usuarios tan prolífica o receptiva como los demás tipos de contenedores, haciendo que en OpenStack la comunidad use su documentación, por ser una mejora de LXC, lo cual requiere una alta gama de soporte, en comparación con Docker por la más usada, tiene mayor documentación que proporcionan distribuciones oficiales, por otro lado, CoreOS mantiene un centro activo de recursos de la comunidad; de manera similar a Docker, el portal de la comunidad y los foros son recursos populares de autoservicio entre los usuarios de Docker y CoreOS RKT.

  El soporte documentado para Contenedores en Windows está establecido en la página de Microsoft y en la nube Azure para cada configuración o levantamiento de servicios que se requieran.
- Operabilidad: LXC principalmente se mantiene y desarrolla en la plataforma Ubuntu, lo que demanda de un nivel medio-avanzado en manejo de comandos por consola mientras que OpenStack comenzó de igual forma, pero cuenta ahora con la implementación gráfica, pero tiene la desventaja de que se generan muchas configuraciones predeterminadas, lo cual hay que revisar una a una mediante un terminal, en cambio CoreOS RKT no fue creado para principiantes, pero la necesidad de operar con facilidad los servicios ahora se recurre a una herramienta visual para sus operaciones, en Docker su comunidad mantiene un conjunto de configuraciones que optimizan el uso de ésta tecnología, pero no es minuciosa

como CoreOS RKT, pues dentro de esas configuraciones pueden tener bugs de versiones anteriores, mientras que con los Contenedores en Windows como es una adaptación reciente de lo que son los contenedores ya existentes como Docker y RKT, aún tiene problemas al utilizar los servicios, pero estos son controlados por usuarios que tengan el nivel necesario.

- Interfaz Gráfica: En LXC tiene una interfaz que no permiten crear los contenedores, solo iniciarlos, detenerlos y ver los servicios, lo que hace que el usuario siga interactuando con comandos en el terminal para establecerle los recursos, no es el caso en OpenStack, ya que este permite configuración base de los ambientes que se deseas trabajar en la nube, dando el servicio básico configurable, donde sí se requiere de un mejor rendimiento hay que acceder por medio de la consola; con CoreOS RKT utiliza una plataforma llamada Tetonic para la administración visual de contenedores y clústeres, mientras que Docker utiliza Kitematic para la administración de contenedores de forma más fácil que CoreOS RKT, los Contenedores en Windows se caracterizan por facilitar visualmente y accesos rápidos a los usuarios que usan esta tecnología.

Fiabilidad

- Capacidad de recuperación: LXC en caso de rotura todas las máquinas virtuales dependen del mismo hardware haciendo que se pueda recuperar solo maquinas con OS compatible con el núcleo del anfitrión, en el caso de OpenStack los procesos se recuperan mediante respaldos en la nube de terceros, pero perdiendo procesos que solo pueden ser generados por el propio contenedor, mientras que CoreOS como antes mencionado se caracteriza por la seguridad que brinda, esto involucra a la forma de evitar que existan problemas, y en el caso de haberlos se crea un respaldo automático del ultimo estado en funcionamiento usando un formato de contenedor de código abierto llamado appc, mientras que Docker usa su propio formato de imagen para estos procesos, de igual forma los Contenedores en Windows tiene el solucionador de estos problemas que antes que aparezcan como problemas primero prueban en estados anteriores para detectar la falla.
- Tolerancia a fallos: LXC tiene problemas de acceso a hardware con múltiples máquinas virtuales, sin embargo, OpenStack presenta procesos de mejora a la hora de hacer usos con varias máquinas virtuales, pero hay que detener todos los

procesos para levantar 1 solo, mientras que CoreOS RKT tiene capacidad de adaptar y dividir los fallos en procesos de esperas hasta que logren a un estado donde no fallaban, es un proceso adaptado de Docker, pero está mejorado, con los contenedores en Windows se toleran fallos en su totalidad, puesto que se generan duplas de cada proceso, donde si falla uno existe el respaldo para la continuación del servicio, y así al mismo tiempo creando una nueva dupla por si llegue a fallar nuevamente.

Eficiencia

- Comportamiento en el tiempo: LXC al momento de crear el servicio, se requiere de doble de tiempo, para levantar el servicio y para activarlo, en OpenStack cuando se crea el servicio se levantan procesos bases, lo cual hace que, al querer ejecutar los otros procesos, hagan que disminuyan en tiempo de espera estos levantamientos, mientras que Docker, CoreOS RKT responden de igual forma según en el kernel donde se encuentran ejecutándose, ya sea para preparar, levantar o detener los servicios, de igual manera los Contenedores en Windows cuenta con un nivel de respuesta aceptable y un gran desempeño en las tareas de cada uno de los módulos, cabe recalcar que no es igual el tiempo al usarlo directo desde el servidor base, que desde la nube.
- Utilización de recursos: LXC está diseñado para ejecutar "contenedores de sistema completo", que generalmente consisten en una imagen de sistema operativo completa, lo que se asimila a una máquina virtual que adapta los recursos de hardware completos de un SO, de tal forma trabaja OpenStack pero controla los recursos llamando a los necesarios, para aplicaciones que se ejecutan en masa, mientras que CoreOS RKT no está diseñado para ejecutar "contenedores de sistema completo" sino aplicaciones individuales como aplicaciones web, bases de datos o caché, lo que con Docker se presenta una forma simple de empaquetar y entregar aplicaciones y todas sus dependencias necesarias, de esta forma Windows adaptó ese concepto el cual antes tenía la necesidad de tener todas los recursos de hardware, haciendo que los procesos del SO anfitrión disminuyan.

Mantenibilidad

- Acoplamiento: LXC tiene soporte de características incoherentes en diferentes distribuciones de Linux, pero esto fomenta a un cambio directo desde el Core de la tecnología, mientras que OpenStack puede ser puesto con cambios en los servicios que ofrece, dejando los servicios con bugs, por los nuevos cambios, haciendo que los servicios continúen, de la misma forma pasa con Docker, sus cambios modulares no interrumpen servicios terceros, en el caso de CoreOS RKT los cambios pueden afectar servicios de alto consumo de recursos, disminuyendo los procesos para brindárselos a los nuevos cambios.

  Los Contenedores en Windows se caracterizan por adaptar las nuevas versiones o cambios que su proveedor agrega a las funcionalidades, sin perder los servicios que fallaban, cruzando solo por un proceso de mejora.
- Modularidad: LXC cuenta con módulos para cada proceso prestado, pero pueden ser estudiados por ser open source, pero hay que tener el nivel de entendimiento en esta distribución para implementar cambios, de la misma forma OpenStack al utilizar LXC, el cual mejoró ciertos módulos, como el de seguridad e integridad, pero no son tan accesibles ni modulares como en CoreOS RKT, los cuales son capaces de ser estudiados pero desde dentro del sistema, mientras que Docker es modular por permitir que proveedores externos se encarguen de optimizar servicios como hacen los Contenedores en Windows al permitir accesos con otras tecnologías a sus procesos para entender sus servicios.
- Facilidad de pruebas: LXC es una tecnología que para ser estudiada y probada requiere de tener conocimiento en la plataforma que se ejecuta, lo cual el acceder a probar sus módulos, no imposible sino dificultoso, en el caso de OpenStack admite pruebas en cualquier plataforma en la que se encuentre, pero la desventaja es que no se pueden modificar sus librerías en el caso de hallar algún error, solo ay que informar al proveedor directo, sin embargo con CoreOS RKT y Docker se manejan versiones de entornos junto con el código de las aplicaciones, lo cual admite pruebas, en el caso de los contenedores en Windows permite pruebas modulares de forma interna y externa en configuraciones bases.

Portabilidad

- Adaptabilidad: LXC se puede usar para ejecutar (pero no descargar) contenedores de aplicaciones, pero este uso requiere una mayor comprensión de los detalles del sistema operativo de bajo nivel, siendo una práctica menos común, sin embargo, OpenStack perfeccionó este detalle para que los contenedores sean adaptados desde en todas sus aplicaciones, ya sean desde la base hasta la nube.

  Docker actuando como su librería (libcontainer) que proporciona el entorno de ejecución en múltiples distribuciones que permitan esta tecnología, permitiendo una movilidad y portabilidad de aplicaciones sin interrupciones.
- Facilidad de Instalación: En LXC cada requisito del sistema es especifico por lo que hay que hacerlo de formar manual, en OpenStack se presenta 2 formas de hacerlo, ya sea manual o automática, pero al hacerlo manualmente hay que tener en cuenta las dependencias con requisitos de sistemas de terceros para su funcionamiento, en CoreOS RKT la instalación es fácil desde que mejoraron la accesibilidad para usuarios novatos, de igual forma Docker el cual maneja automatizaciones en procesos de instalación con pocos pasos para su preparación, con los contenedores en Windows en ambos servicios, ya sea por consola o por la interfaz la instalación es fácil y entendible.
- Coexistencia: LXC soporta solo arquitectura basada en hardware y la virtualizada, mientras que OpenStack cuenta con múltiples suministros adecuados en la plataforma cloud escalable, sin tener que empezar de cero, sin embargo, Docker proporciona un paquete de software determinista y se adapta muy bien al modelo de infraestructura inmutable de otros sistemas con proveedores de tercero. CoreOS RKT proporciona a cada contenedor exactamente lo que necesita donde se necesita, sin perder ni un ciclo de CPU en tareas de coexistencia entre host y contenedor.

  Los Contenedores en Windows permiten que la infraestructura maneje la escalabilidad, haciendo que los desarrolladores puedan enfocarse en crear la funcionalidad de la aplicación en lugar de como sus servicios manejan la coexistencia de múltiples instancias de ellos mismos.
- Reemplazamiento: En LXC solo se puede reemplazar por tecnologías apegadas a los procesos que estos demandan por ejemplo como lo hace OpenStack, el cual

puede ser reemplazado por sistemas como que permitan manejo de entornos en la nube, en el caso de CoreOS RKT sus funcionalidades pueden ser reemplazadas por Docker, de igual forma recíproca Docker puede ser reemplazado por CoreOS RKT, mientras que en los Contenedores en Windows se admite el reemplazo total en sus funciones por Docker, el cual maneja el Core otorgado por Microsoft.

Tabla 5 Calidad Externa e Interna deseada aplicando ISO/IEC 9126

CARACTERÍSTICA	SUB-CARACTERÍSTICA	PONDE-RACIÓN	LXC		OpenStack		CoreOS RKT		Docker		Contenedores Windows	
			Peso	**Pond**	**Peso**	**Pond**	**Peso**	**Pond**	**Peso**	**Pond**	**Peso**	**Pond**
Funcionablidad	Adecuación	20%	4.00	0.80	4.00	0.80	5.00	1.00	5.00	1.00	5.00	1.00
	Exactitud	20%	3.00	0.60	4.50	0.90	4.50	0.90	5.00	1.00	5.00	1.00
	Interoperabilidad	20%	2.50	0.50	3.75	0.75	3.75	0.75	5.00	1.00	5.00	1.00
	Seguridad	40%	2.50	1.00	4.25	1.70	5.00	2.00	4.25	1.70	4.50	1.80
Total Funcionalidad		**100%**		**2.90**		**4.15**		**4.65**		**4.70**		**4.80**
Usabilidad	Documentación	40%	3.75	1.50	3.75	1.50	4.25	1.70	5.00	2.00	5.00	2.00
	Operabilidad	40%	3.00	1.20	4.25	1.70	5.00	2.00	4.50	1.80	4.75	1.90
	Interfaz Gráfica	20%	2.50	0.50	3.75	0.75	4.75	0.95	5.00	1.00	5.00	1.00
Total Usabilidad		**100%**		**3.20**		**3.95**		**4.65**		**4.80**		**4.90**
Fiabilidad	Recuperabilidad	40%	2.50	1.00	4.25	1.70	4.50	1.80	5.00	2.00	5.00	2.00
	Tolerancia a Fallas	60%	2.25	1.35	3.75	2.25	4.60	2.76	4.25	2.55	5.00	3.00
Total Fiabilidad		**100%**		**2.35**		**3.95**		**4.56**		**4.55**		**5.00**
Eficiencia	Comportamiento en el tiempo	50%	3.50	1.75	4.00	2.00	5.00	2.50	5.00	2.50	5.00	2.50
	Utilizacion de Recursos	50%	4.00	2.00	5.00	2.50	4.50	2.25	5.00	2.50	5.00	2.50
Total Eficiencia		**100%**		**3.75**		**4.50**		**4.75**		**5.00**		**5.00**
Mantenibilidad	Acoplamiento	40%	3.25	1.30	5.00	2.00	4.50	1.80	5.00	2.00	5.00	2.00
	Modularidad	40%	3.75	1.50	3.75	1.50	4.50	1.80	4.50	1.80	5.00	2.00
	Facilidad de pruebas	20%	3.75	0.75	3.75	0.75	5.00	1.00	5.00	1.00	5.00	1.00

Total Mantenibilidad		**100%**		**3.55**		**4.25**		**4.60**		**4.80**		**5.00**
Portabilidad	Adaptabilidad	20%	2.50	0.50	3.74	0.75	4.50	0.90	5.00	1.00	4.50	0.90
	Instalabilidad	35%	2.25	0.79	3.00	1.05	4.75	1.66	5.00	1.75	5.00	1.75
	Coexistencia	25%	3.00	0.75	4.00	1.00	4.50	1.13	4.50	1.13	4.75	1.19
	Reemplazabilidad	20%	3.75	0.75	3.75	0.75	4.00	0.80	4.50	0.90	4.50	0.90
Total Portabilidad		**100%**		**2.79**		**3.55**		**4.49**		**4.78**		**4.74**
TOTAL (PROMEDIO)				**3.09**		**4.06**		**4.62**		**4.77**		**4.91**

Tabla 6 Datos porcentuados de la tabla 5

	PONDERACIÓN	**LXC**		**OpenStack**		**CoreOS RKT**		**Docker**		**Contenedores Windows**	
		Pond	%	Pond	%	Pond	%	Pond	%	Pond	%
Funcionalidad	**16.67%**	2.90	**48.33**	4.15	**69.17**	4.65	**77.50**	4.70	**78.33**	4.80	**80.00**
Usabilidad	**16.67%**	3.20	**53.33**	3.95	**65.83**	4.65	**77.50**	4.80	**80.00**	4.90	**81.67**
Fiabilidad	**16.67%**	2.35	**39.17**	3.95	**65.83**	4.56	**76.00**	4.55	**75.83**	5.00	**83.33**
Eficiencia	**16.67%**	3.75	**62.50**	4.50	**75.00**	4.75	**79.17**	5.00	**83.33**	5.00	**83.33**
Mantenibilidad	**16.67%**	3.55	**59.17**	4.25	**70.83**	4.60	**76.67**	4.80	**80.00**	5.00	**83.33**
Portabilidad	**16.67%**	2.79	**46.46**	3.55	**59.13**	4.49	**74.79**	4.78	**79.58**	4.74	**78.96**
TOTAL (PROMEDIO)	100.00%		51.49		67.63		76.94		79.51		81.77

ANÁLISIS GENERAL

Se definieron los pesos para las tecnologías de contenedores de acuerdo a las características y sub-características de la norma ISO/IEC 9126; partiendo de la base de los análisis realizados, lo que permitió tener un criterio formado y sustentado para poder dar valores a cada herramienta.

De tal manera que el valor vertical de todos los pesos que corresponden a cada software de acuerdo a sus características es el promedio de sí mismas. Los valores ponderados en cada sub-característica equivalen al producto del porcentaje de ponderación por el peso de cada sub-característica. El resultado ponderado de cada característica es igual la suma de las ponderaciones de las sub-características, el cual llega a un valor máximo de 5.

Se consideró que todas las características son igual de importante. Por ende, se los pondero todos con el mismo peso, donde el valor total de cada software es el promedio del resultado de los valores de las características ponderadas, donde estas son divididas para 6 que contemplan el 100%.

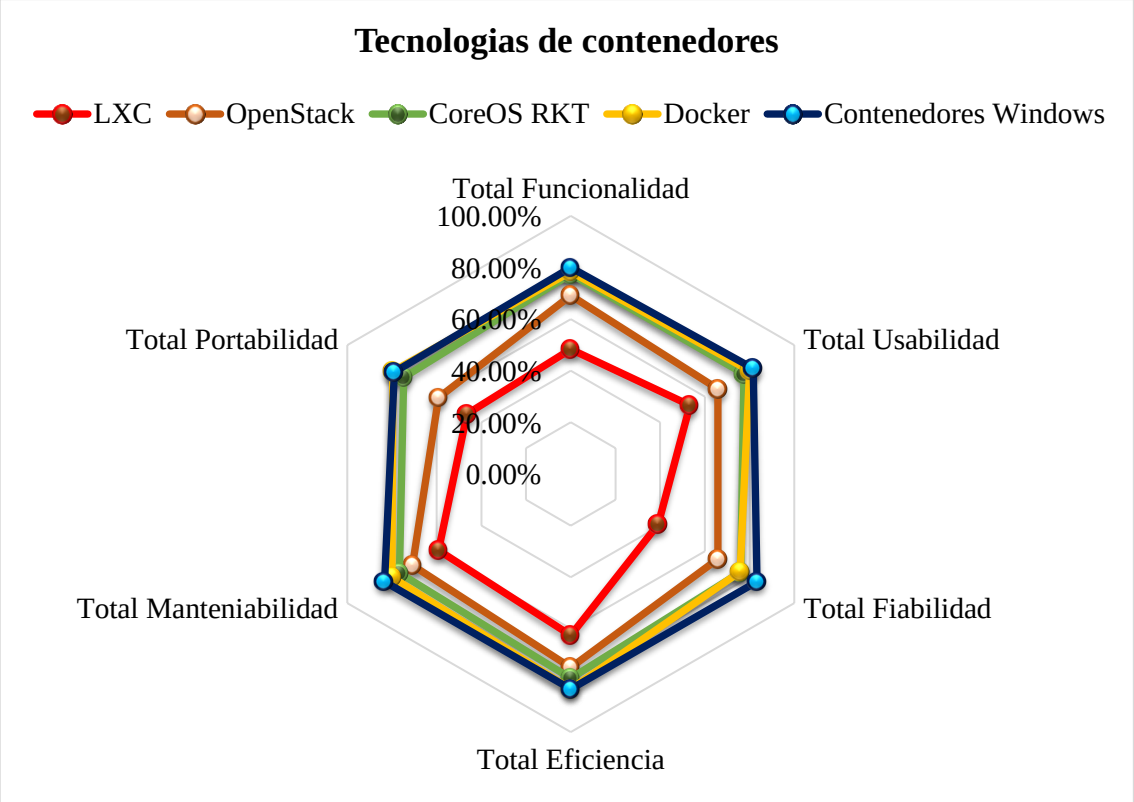

Ilustración 7 Resumen de la Matriz de evaluación según ISO 9126

En la ilustración 7, se resume de los datos tabulados en la tabla 6, donde se los valores están representados por datos ponderados en la tabla 5, lo cual se aclara lo siguiente:

- En la Funcionalidad las tecnologías de Docker, CoreOS RKT, OpenStack y los Contenedores en Windows superan el 60% en funcionalidades, ya que son dedicadas a entornos pre-configurados.
- En la Usabilidad las mismas 4 tecnologías mencionadas dedican más del 65% de su uso para armar ambientes de desarrollo, lo que involucra que, en su Fiabilidad, se presenten resultados similares por tener servicios dedicados a configuración de entornos aislados, mientras que LXC, no es tan fiable por la pérdida de datos en sus procesos a tal punto que no alcanza ni el 40% de fiabilidad, aunque otras tecnologías gozan de sus servicios, lo hacen, pero una vez que reparen los bugs que estos presentan.
- En la Eficiencia, todas las tecnologías superan el 60% usando los recursos necesarios de hardware donde ejecuta las aplicaciones, siendo la característica representativa entre las demás.
- En la Mantenibilidad la tecnología LXC no se adapta con facilidad a cambios, ya que su soporte no mide las versiones anteriores, por otro lado, CoreOS RKT, OpenStack, Docker y los Contenedores en Windows cuentan con soportes, permitiendo que los cambios sean probados antes de ser implementados, haciendo que sus versiones sean estudiadas.
- Con la Portabilidad las tecnologías como LXC y OpenStack presentan características específicas para ser transferidos a otros ambientes por el hecho de compartir las mismas librerías, mientras que CoreOS RKT, Docker y los contenedores en Windows con más del 75% son transferibles a ambientes que permitan entornos aislados.

Se puede observar que entre los Open Source más sobresale Docker en muchas de sus características de la ISO/IEC 9126 en comparación de los otros tres Open Source, demostrando que es un sistema que brinda garantía para poder funcionar en cualquier ambiente que requiera de sus procesos y servicios. Y en ambientes para Windows, se pueden implementar contenedores Docker, pero por ser nativos de Linux, son limitados, por eso la mejor opción es el uso de los contenedores en Windows.

CAPÍTULO IV

DISCUSIÓN

4.1. Discusión

Un estándar, es el conjunto de normas pre establecidas que sirven como modelo para llegar a un objetivo. En la informática se aplican una serie de modelos y estándares internacionales para cada una de las diferentes áreas. Para el control de calidad de software está la ISO 9126 la cual propone un modelo detallado, en el que describe 6 características para la selección de un software por su calidad, las cuales son: Funcionalidad, Usabilidad, Fiabilidad, Eficiencia, Mantenibilidad, Portabilidad (ISO 9126, 2000).

En la investigación de Cardona (2009), resalta que las pruebas en el proceso de desarrollo aseguran la calidad del producto ya que identifica los defectos de una aplicación, esto permite que se reduzca el riesgo de poner en marcha software con errores.

Con la ISO 9126 se definieron mejores prácticas; tales como: la documentación en general según la IEEE 829 y planeamiento del proceso de desarrollo y de pruebas según la ISO 29119 de la aplicación para el aseguramiento de su calidad dentro de un tiempo establecido y acorde a los requerimientos solicitados por el cliente. Al seguir esta normalización se puede trabajar con diferentes proyectos, ya sean éstos grande o pequeños, lo cual permite que se tengan definidas desde un inicio las pruebas que se van a llevar a cabo, identificando y definiendo los escenarios de pruebas, para que la ejecución de las pruebas sea más eficiente.

Lo investigado por Gilbert & Peña (2005), se menciona el uso de herramientas, metodologías y tecnologías como solución a los problemas de planificación y aseguramiento de calidad en el proceso de desarrollo de software.

Según los estudios realizados por Mendoza (2010), en la cual realiza la selección de herramientas de pruebas como soporte para el proceso de desarrollo de software, que contribuyan a mejorar el proceso de las empresas para elevar la calidad de sus sistemas, por lo que esta investigación se centra en las pruebas de software y la implementación de herramientas que faciliten la automatización de las mismas.

En esta investigación se realizó comparaciones entre tecnologías que cumplen la misma función para luego establecer cuál es la más adecuada para cubrir las necesidades del departamento de TIC de la institución.

En el contexto de la PUCE-Esmeraldas requería de tecnologías que agilicen sus procesos de desarrollo y pruebas, automatizándolos y desplegándolos reduciendo el tiempo de entrega, para mejorar la planificación y aseguramiento de calidad en el proceso de desarrollo de software.

Según el análisis de los resultados, las tecnologías de contenedores estudiadas, en su totalidad, cubren con las características que plantea la ISO 9126, determinadas para el departamento.

Finalmente se puede decir que los contenedores son software que cualquier empresa de desarrollo puede utilizar, teniendo en cuenta que, si alguno de ellos se vuelve obsoletos o complicados por su manejo, pueden ser reemplazados por otros, teniendo en cuenta que son tecnologías que viven cambiando a cada rato y que fueron creados para automatizar procesos, facilitando la administración y preparación de ambientes de pruebas.

CAPÍTULO V

PROPUESTA DE INTERVENCIÓN

5.1. Título

Tecnologías de Contenedores en entornos de pruebas en Linux y Windows, automatizando el proceso de despliegue de aplicaciones, desde las pruebas hasta su producción.

5.2. Descripción

La propuesta se centra en una solución destinada a mejorar muchos de esos puntos, a través de estándares, modelos y herramientas tecnológicas que permitan agilizar y automatizar el proceso de pruebas para el aseguramiento de servicio QA.

5.3. Desarrollo

Se seleccionó las herramientas de tecnología de contenedores de software óptimas y adecuadas para la integración continua en el despliegue de aplicaciones al momento de preparar entornos de pruebas. Analizado y discutido los resultados se presenta a Docker como una herramienta para los ambientes de pruebas en Linux y a los Contenedores en Windows en desarrollos para Windows ayudando a la automatización de las pruebas de software en el proceso de desarrollo, estas tecnologías permiten el aseguramiento de la calidad de los productos que se desarrollan en las empresas o instituciones.

Al utilizar contenedores en entornos de pruebas se obtienen beneficios tales como:

1. Acelerar el proceso de mantenimiento y desarrollo, de manera que realizar una copia de los sistemas que están en producción y ejecutarlos en otro equipo, sea una tarea sencilla, reduciendo así el tiempo y costo que implica preparar un ambiente.
2. Las aplicaciones se ejecutan tal y como fueron creadas y no importa el equipo ni el ambiente (pruebas o producción).
3. En un escenario típico el cliente instalaría y configuraría SQL y posteriormente la aplicación. Con un contenedor ejecutando se mantiene una configuración simplificada.

4. En DevOps, tanto los desarrolladores como los administradores de sistema pueden probar aplicaciones en un entorno seguro y exactamente igual en todos los casos.
5. Fácil versionamiento, cuando se finaliza con ellos, se pueden desechar.
6. En microservicios, los contenedores abastecen la velocidad de despliegue y separación ideales para dicha arquitectura.

Para la administración de despliegues automatizados se optó la selección de los contenedores compatibles con Jenkins para agilizar el proceso de puesta en producción de las aplicaciones desde sus pruebas, verificación de errores, versionar aplicaciones y promover agilidad en los procesos.

Al trabajar en conjunto con herramientas que permitan el aseguramiento de la calidad, y se cumpla con los tiempos establecidos, la operación del sistema se convierte en estable, los riesgos probables son menores ya que no interfieren ni comprometen la funcionalidad principal del sistema, adicional a esto no hay comprometidos bienes materiales ni daño físico. El sistema entrará en un proceso de mejora continua (Ver ilustración 8).

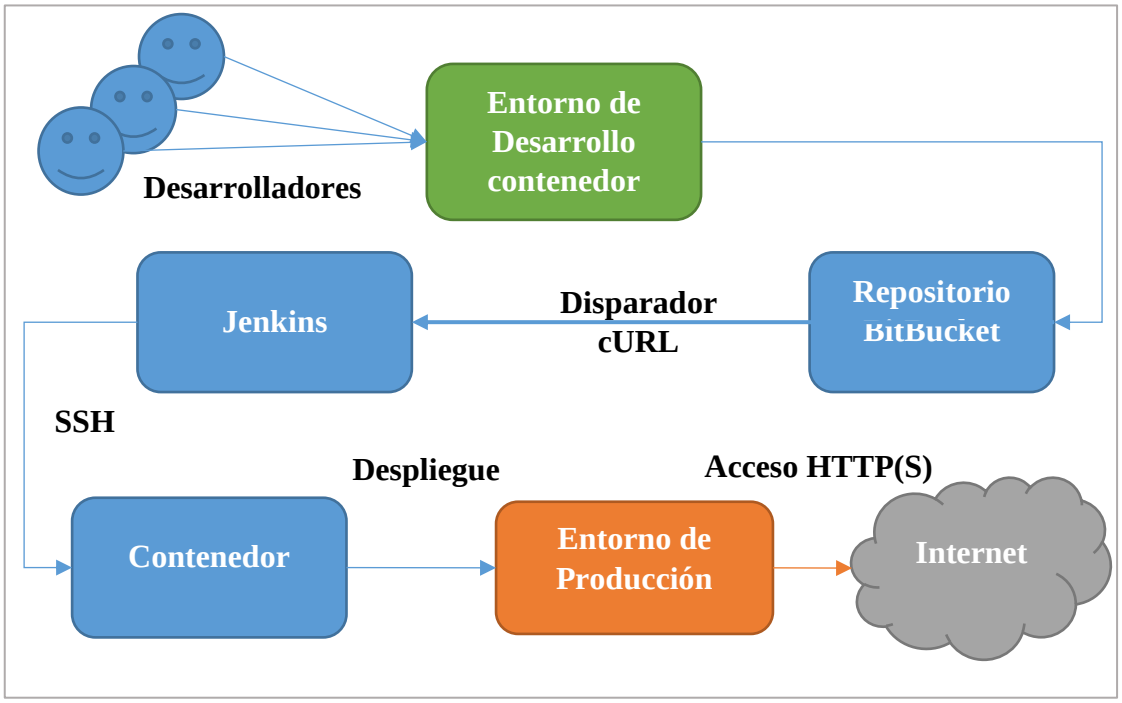

Ilustración 8 Proceso de pruebas mediante el despliegue con Jenkins.

Tras el estudio que se ha detallado en este documento, se ha podido demostrar que se cumplió los objetivos propuestos. Ha sido posible poner en contexto el conjunto de herramientas elegidas llevando a cabo un montaje de cada una de las tareas permitiendo

procesar una aplicación de ejemplo en Wordpress para preparación para la puesta en producción, facilitando además una rápida retroalimentación de los resultados de pruebas y el estado de la aplicación a todos los miembros de los equipos que intervienen en el proceso de creación software a través del sistema de notificaciones provisto por el servidor de entrega continua (Jenkins).

Esta propuesta puede servir para cualquier empresa o institución que se dedican al desarrollo de software, podrá monitorizar los proyectos automáticamente y se podrá tener una mayor calidad del producto que se ofrece.

CAPÍTULO VI

CONCLUSIONES Y RECOMENDACIONES

6.1. Conclusiones

Una vez terminadas las etapas que demandó el desarrollo de esta investigación, se llegó a las siguientes conclusiones:

1. Al realizar el estudio de las arquitecturas basadas en contenedores, se optó por elegir a las que se acoplen y cubran las necesidades del departamento, utilizando las herramientas existentes.
2. El uso de los contenedores permite acelerar los procesos ayudando a la preparación de entornos de forma rápida y corrección de incidentes que son detectados en etapas tempranas, asegurando de esa forma la calidad y reducir la probabilidad de riesgos que se puedan presentar en el desarrollo del software y su puesta en producción.
3. Los contenedores han revolucionado el mundo de la integración continua, ya que teniendo instalada únicamente dicha herramienta se podrá compilar y construir entornos ejecutables de cualquier proyecto, así como poder automatizar un escenario real con múltiples contenedores y ver su funcionamiento.

6.2. Recomendaciones

1. Por cumplimiento de los tiempos establecidos es recomendable que los desarrolladores realicen también pruebas a sus códigos con el fin de mejorar la calidad interna del código, pero que no sea éste quien dé el visto bueno que de las pruebas se han ejecutados con satisfacción por la simple razón de que ya sea de forma consciente o inconscientemente, el programador desea que su programa funcione sin ningún inconveniente, para esto se debe contar con un grupo de testeadores que se encarguen de preparar los ambientes de pruebas, y cumplan con la revisión del software antes de la puesta en producción.
2. No poner en marcha escenarios sin documentación, porque la aplicación debe pasar por varias pruebas, ya que resulta necesario tener documentado qué funciona y qué no funciona y así se evita incurrir y repetir pruebas ya realizadas.
3. Realizar el despliegue automatizado en el proceso de desarrollo junto con entornos de pruebas automatizados desde un inicio hasta su puesta en producción y así mejorar la calidad del producto final y ahorrar en la solución de problemas con los errores encontrados en la última etapa de desarrollo del software.

REFERENCIAS

Glosario

APA: Estándar elaborado por la Asociación Americana de Psicología que los autores utilizan al momento de presentar sus documentos

CPU: Central Processing Unit, unidad de proceso central

GB: GigaByte es una unidad de almacenamiento de información

GHz: Gigahertz, gigahercio es la unidad de medida de frecuencia hercio

HD: Hard Disk, Disco duro

IBM: International Business Machines, empresa multinacional estadounidense de tecnología y consultoría

IEC: International Electrotechnical Commission, en español Comisión Internacional Electrotécnica

IEEE: Institute of Electrical and Electronics Engineers, en español El Instituto de Ingeniería Eléctrica y Electrónica

ISO: International Organization for Standardization, en español Organización Internacional de Estándares

Kernel: núcleo del sistema operativo

LXC: Linux Container

MECABIC: es un método que sirve para analizar y evaluar Arquitecturas de Software Basadas en Componentes basado en el modelo de calidad ISO/IEC 9126

PYMES: acrónimo de pequeñas y medianas empresas

QA: Quality Assurance, garantía de calidad

RAM: Random Access Memory, memoria de acceso aleatorio

Rkt: Rocket, cohete

SO: Sistema Operativo

TI: Tecnología de la Información

TIC: Tecnología de la Información y la Comunicación

Referencias Bibliográficas

Akella, P. S., & Rao, K. N. (2011). *Effective Independent Quality Assessment Using IV&V*. International Journal of Computer Science and Information Technology (IJCSIT), 3(3). 320-332.

Alshammri, M. (2013). *Problems in Software Quality Assurance and Reasons*.

Architecture Working Group of the Software Engineering Committee. (2000). *Recommended Practice for Architectural Description of Software Intensive Systems*. *IEEE Standards Department*.

Azab, A., & Domanska, D. *Software Provisioning Inside a Secure Environment as Docker Containers using STROLL File-system*.

Bass, L. (2007). *Software architecture in practice*. Pearson Education India.

Blaxter, L., Hughes, C., & Tight, M. (2000). *Cómo se hace una investigación*. Barcelona: Gedisa.

Burnstein, I. (2006). *Practical software testing: a process-oriented approach. Springer Science & Business Media*.

Cardona, C. J. (2009). *Propuesta metodológica para la realización de pruebas de software en un ambiente productivo*. Medellín, Colombia.

Ciolli, M. (2007). *Testing de migración de aplicaciones distribuidas a entornos Web*. La Plata, Argentina: Universidad Nacional de la Plata. Tesis para optar el grado de Magíster.

CoreOS Rkt. (10 de Junio de 2017). CoreOS Rkt. Obtenido de https://coreos.com/rkt/

Docker. (15 de Abril de 2017). *Docker*. Obtenido de https://www.docker.com/what-docker

Doña, J. M., García, J. E., López, J., Pascual, F., & Rubén F Pascual, (2012). *Virtualización de Servidores - Una Solución de Futuro*. Málaga, España.

Dustin, E. (2002). *Effective Software Testing: 50 Ways to Improve Your Software Testing*. Addison-Wesley Longman Publishing Co., Inc.

Gao, J., Tsao, H. S., & Wu, Y. (2003). *Testing and quality assurance for component-based software*. Artech House.

Gibert, M., & Peña, Á. (2005). *Ingeniería del Software en entornos SL*.

Hale, J. S., Li, L., Richardson, C. N., & Wells, G. N. (2016). *Containers for portable, productive and performant scientific computing*. arXiv preprint arXiv:1608.07573.

Humble, J., & Farley, D. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation (Adobe Reader)*. Pearson Education.

IBM. (17 de mayo de 2017). *IBM*. Obtenido de https://www.ibm.com

ISO25000. (s.f.). ISO 25000 *Calidad del Producto*. Recuperado el 4 de septiembre de 2013, de www.iso25000.com

IEEE Std 610.12. (1990). *IEEE Standard Glossary of Software Engineering Terminology*.

IEEE Std 829. (2008). *IEEE Standard for Software and System Test Documentation.*

ISO / IEC 9126. (2000). *Information technology - Software*. ISO.

ISO/IEC 29119. (2013). *Software Testing Standards*. Londres: ISO.

Kazman, R., Klein, M., & Clements, P. (2001). *Evaluating Software Architectures-Methods and Case Studies.*

Lofstead, G. F., Jimenez, I., Maltzahn, C., Moody, A., Mohror, K., & Arpaci-Dusseau, R. (2015). *The Role of Container Technology in Reproducible Computer Systems Research (No. SAND2015-0271C)*. Sandia National Laboratories (SNL-NM), Albuquerque, NM (United States).

Losavio, F., Chirinos, L., Lévy, N., & Ramdane-Cherif, A. (2003). *Quality characteristics for software architecture*. Journal of Object Technology, *2*(2), 133-150.

Mendoza, M. (2010). *Modelo de Referencia para la selección de herramientas de pruebas como soporte al proceso de desarrollo de software en PYMES venezolanas*. Caracas, Venezuela: Universidad Católica Andrés Bello. Tesis para optar el grado de Magíster

Merkel, D. (2014). *Docker: Lightweight Linux Containers for Consistent Development*. Houston: Linux Journal. Recuperado el 17 de octubre de 2016, de http://dl.acm.org/

Microsoft. (17 de mayo de 2017). *Microsoft*. Obtenido de https://www.microsoft.com/

Muñoz, R. C. (1998). *Cómo elaborar y asesorar una investigación de tesis*. México.

Murugesan, S., Deshpande, Y., Hansen, S., & Ginige, A. (2000). *Web Engineering: A New Discipline for Development of Web-Based Systems*. Recuperado de http://www-itec.uniklu.ac.at/~harald/proseminar/web11

Myers, G. J., Sandler, C., & Badgett, T. (2011). *The art of software testing*. John Wiley & Sons.

Nishi, Y. (Abril de 2015). *In Software Testing, Verification and Validation Workshops* (ICSTW), 2015 IEEE Eighth International Conference on (pp. 1-4. Design principles in test suite architecture. (IEEE, Recopilador)

OpenStack. (17 de mayo de 2017). *OpenStack*. Obtenido de https://www.openstack.org/

Redhat. (2017). *Virtualization*. Obtenido de https://www.redhat.com/en/topics/virtualization

Rogers, P. (2005). *Ingeniería de Software un Enfoque Práctico*. Editorial McGraw-Hill, Madrid.

Morabito, R. (2016). *A Performance Evaluation of Container Technologies on Internet of Things Devices* Ericsson Research, NomadicLab, Jorvas, Finland. Recuperado el 31 de octubre de 2016, de https://arxiv.org/ftp/arxiv/papers/1603/1603.02955.pdf

Sampieri, H., Collado, F., & Lucio, B. (2003). *Metodología de la investigación*. La Habana: Editorial Félix Varela, 2.

Tian, J. (2005). *Software quality engineering: testing, quality assurance, and quantifiable improvement.*

Valdivia Espinoza, D. R., & Valdivia Espinoza, E. G. (2005). *Estándares de calidad para pruebas de software.*

Wolf, G. (2014). *Programar es un modo de vida*. Software Guru, 50-51. Recuperado el 17 de Octubre de 2016, de SG Buzz: https://sg.com.mx/revista/52/contenedores-un-poco-historia#.WAV9yyjhDIU

Anexos

Anexo 1. Modelo de entrevistas

Entrevista a los desarrolladores de software del departamento de TIC.

El presente documento se basa en una ISO/IEC/IEEE 29119 Software Testing para la verificación de calidad de software que se desarrolla en una empresa:

1. ¿Qué tipo de aplicativos desarrollan aquí la universidad?

2. ¿Qué tipos de pruebas se realiza y que herramientas usan?

3. ¿Qué documentos se trabajan durante el ciclo de pruebas?

4. ¿Quién realiza los escenarios de pruebas?

5. ¿Cómo clasifica los errores en los escenarios de pruebas?

6. ¿Cómo desarrollan las pruebas de software?

7. ¿Usan alguna herramienta para el despliegue automático del software?

8. ¿Considera que la herramienta que ha mencionado es una plataforma de uso sencillo?

9. ¿Cuáles son las ventajas de utilizar esa herramienta?

10. ¿Qué herramienta usa para testear sus servidores?

11. ¿Con que regularidad usa pruebas de stress a su sistema?

12. ¿Qué herramientas usan para probar las aplicaciones?

Entrevista al Jefe del departamento de TIC.

El presente documento sirve para obtener información cuantitativa de lo que posee en aplicaciones que se encuentran en la universidad.

Algunas de las preguntas tienen una respuesta con una escala tipo Likert de 5 a 1, entendiéndose el 5 con la valoración más alta y el 1 la más baja

1. ¿Cuántas aplicaciones existen actualmente en la PUCE-Esmeraldas?, ¿Cuántos de desarrollo propio y cuantos adaptados de softwares estándares?

2. ¿Cuántas aplicaciones son web, de escritorio y/o móviles?

3. ¿Cuántas de las aplicaciones han sido sometidas a pruebas de software?

4. ¿Cuántas aplicaciones tienen su respectiva documentación de pruebas?

5. ¿Cuántas aplicaciones tienen control de versiones para su respectivo soporte al momento de encontrar fallos en las pruebas?

6. ¿Con que frecuencia realizan los siguientes tipos de pruebas?
   a) Pruebas funcionales
   b) Pruebas de seguridad y rendimiento
   c) Pruebas de integración y de sistema

Rango puntual
Nunca: 1
Rara vez: 2
A veces: 3
Casi siempre: 4
Siempre:5

7. ¿Existe algún documento validado para establecer el resultado de las pruebas?

8. ¿Realiza control y seguimiento de software en procesos de desarrollo?, ¿lo hacen manual o automatizado?

Anexo 2. Matriz de evaluación de software

TIPO DE CALIDAD	CARACTERÍSTICA	SUB-CARACTERÍSTICA	PUNTAJE	Software 1	Software 2
				PUNTAJE	**PUNTAJE**
Calidad Interna y Externa	Funcionablidad	Adecuación			
		Exactitud			
		Interoperabilidad			
		Seguridad			
	Usabilidad	Documentación			
		Operabilidad			
		Interfaz Gráfica			
	Fiabilidad	Recuperabilidad			
		Tolerancia a Fallas			
	Eficiencia	Comportamiento en el tiempo			
		Utilizacion de Recursos			
	Manteniabilidad	Acoplamiento			
		Modularidad			
		Facilidad de pruebas			
	Portabilidad	Adaptabilidad			
		Facilidad de Instalación			
		Coexistencia			
		Facilidad de Reemplazamiento			
Calidad en uso		Poductividad			
		Seguridad			
		Satisfacción			
TOTAL					