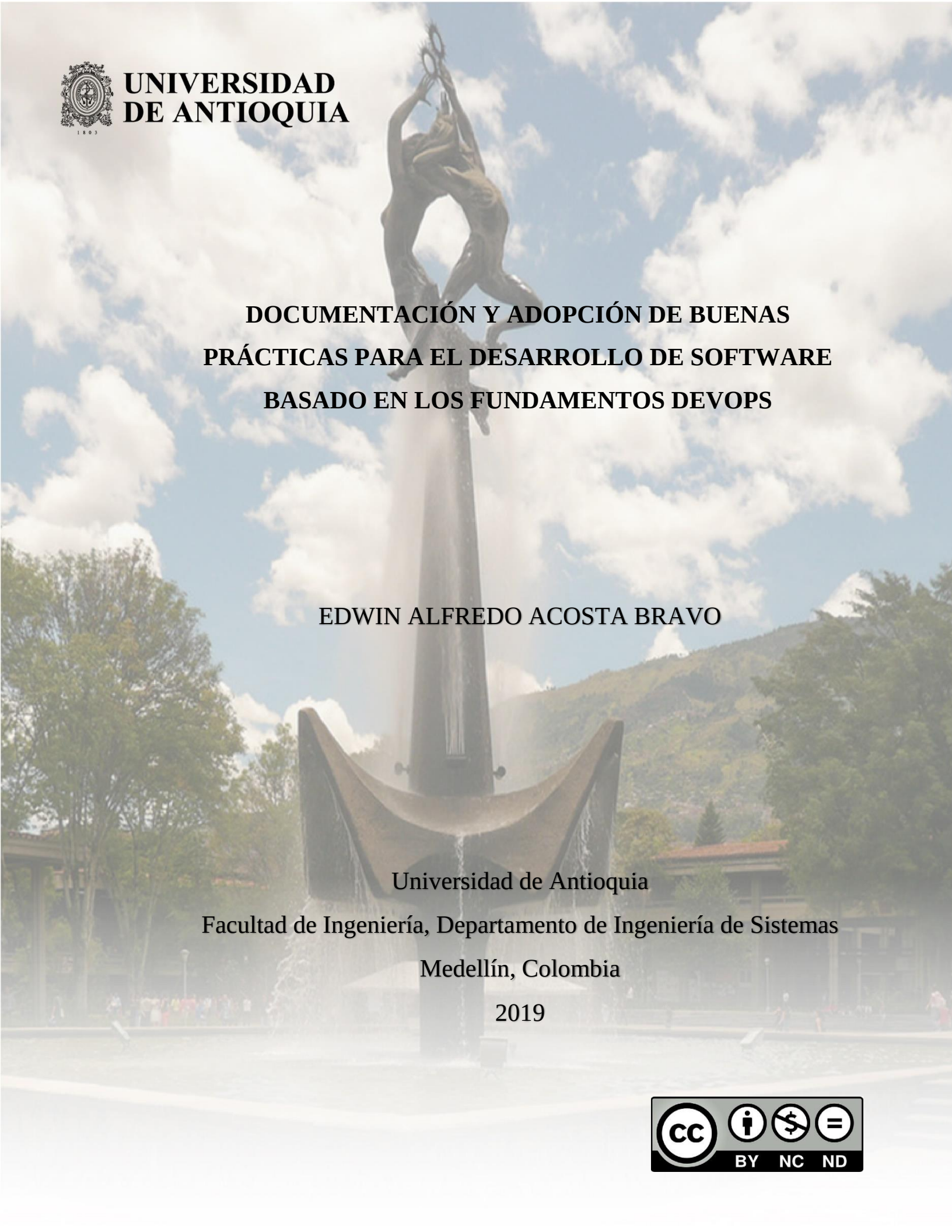


UNIVERSIDAD DE ANTIOQUIA

DOCUMENTACIÓN Y ADOPCIÓN DE BUENAS PRÁCTICAS PARA EL DESARROLLO DE SOFTWARE BASADO EN LOS FUNDAMENTOS DEVOPS

EDWIN ALFREDO ACOSTA BRAVO

Universidad de Antioquia

Facultad de Ingeniería, Departamento de Ingeniería de Sistemas

Medellín, Colombia

2019

DOCUMENTACIÓN Y ADOPCIÓN DE BUENAS PRÁCTICAS PARA EL DESARROLLO DE SOFTWARE DE UNA COMPAÑÍA E IMPLEMENTACIÓN DE DEVOPS

EDWIN ALFREDO ACOSTA BRAVO

Informe de trabajo de grado como requisito para optar al título de:

Ingeniero de Sistemas

Asesor:

José Ignacio López Vélez

Ingeniero de Sistemas

Universidad de Antioquia
Facultad de Ingeniería, Departamento de Sistemas
Medellín – Colombia
2019

DEDICATORIA

A mi familia que han sido un ejemplo a seguir y de quienes siempre he recibido un apoyo incondicional.

A Antonio José, quien hasta el último momento me brindó una amistad sincera y me apoyó en cada uno de mis proyectos.

AGRADECIMIENTOS

Quiero expresar mis más sinceros agradecimientos a mi asesor, Ing. José Ignacio López, por sus conocimientos, su experiencia y en especial por toda la paciencia durante el desarrollo de este trabajo de grado.

Además, mi agradecimiento a Diego Quiroz por ser esa luz que guio e iluminó mi camino de formación y quien brindó muchas ideas para que este trabajo de grado se hiciera realidad.

Resumen

En una sociedad tan competitiva es indispensable emplear herramientas que permitan cumplir con las exigencias del mercado. Lo más importante para una empresa es entregar productos de calidad mediante la conservación de las tareas involucradas en los procesos de las etapas del proceso de software. Sin embargo, al adoptar nuevas herramientas o metodologías es necesario conocer los riesgos e implicaciones que este cambio conllevan.

Uno de los principales inconvenientes que se presentan en las empresas es la falta de comunicación entre los equipos de desarrollo y de los equipos de operaciones, lo cual incurre en retrasos frecuentes por los largos tiempos de respuesta entre las peticiones generadas por cada área.

Como solución a estos problemas, existe DevOps que es un movimiento cultural y profesional que hace hincapié en la comunicación, colaboración e integración entre los desarrolladores de software y los profesionales de las operaciones de TI.

Este trabajo de grado, se centra en la evaluación de cuan preparadas están las empresas para iniciar un proceso de adopción e implementación de DevOps; con lo cual, se brindan unas pautas para la evaluación de las categorías principales de una empresa permitiendo así obtener una valoración cuantitativa con margen de error reducido.

Palabras claves:

Devops, best practices devops, software development, software testing, CI/CD, continuos integration, continuos delivery, continuos deployment, test automation, agile.

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

1. INTRODUCCIÓN

En una fábrica de *software*, es muy común encontrar problemas en los procesos, ya sea porque no están claramente definidos o no existen políticas lo suficientemente sólidas que soporten los procesos críticos en el proceso de desarrollo de *software*. Actualmente, lo más importante para una empresa que se encarga del desarrollo de aplicaciones es entregar productos de calidad optimizando el esfuerzo requerido en cada una de las fases del ciclo de vida del desarrollo de *software*. En la mayoría de las empresas, las áreas de desarrollo y de operaciones[1] se mantienen aisladas y poco interactúan la una con la otra, y cuando lo hacen se pueden presentar conflictos tanto en las peticiones como en la atención de las solicitudes, lo que genera retrasos para las entregas.

DevOps es un nuevo movimiento que trata de mejorar la agilidad en la prestación de servicios. Fomenta una mayor colaboración y comunicación entre los equipos de desarrollo y operaciones y evita que los problemas y las necesidades operacionales se mantengan subexpuestas en el proyecto, afectando a la calidad del *software*. Por otra parte, los cambios y las nuevas implementaciones conllevan a grandes inversiones tanto en equipos como en personal especializado para la implementación y puesta en marcha de las nuevas estrategias. El objetivo principal de DevOps es crear sinergia de los procesos entre el desarrollo y los equipos de operaciones con el fin de hacerlos más eficientes.

En este trabajo, se presentará una guía que proporcionará herramientas conceptuales útiles que ayudarán a las empresas a encaminarse e implementar un correcto proceso DevOps para el proceso de desarrollo de *software*. La

[1] *Los equipos de operaciones pueden referirse a infraestructura, pruebas, despliegues, monitoreo, entre otras áreas involucradas en el ciclo de vida del proyecto.*

metodología empleada para la implementación del modelo DevOps, se centra en la recopilación de las mejores prácticas con los fundamentos necesarios para iniciar y mantener el proceso; teniendo en cuenta unos parámetros específicos, ya que ninguna empresa no es igual a otra aunque compartan el mismo contexto, objetivos y procesos.

2. SITUACIÓN PROBLEMÁTICA

Desarrollar un buen software depende de un sinnúmero de actividades y etapas, donde el impacto de elegir la mejor metodología para un equipo, en un determinado proyecto es trascendental para el éxito del producto. El papel preponderante de las metodologías es sin duda esencial en un proyecto y en el paso inicial, que debe encajar en el equipo, guiar y organizar actividades que conlleven a las metas trazadas en el grupo.

Si los profesionales de desarrollo y operaciones usan metodologías tradicionales[2], es probable que trabajen en dos equipos separados con poca integración o interacción hasta que el equipo de desarrollo entregue los proyectos al equipo de operaciones. Sin una comunicación constante, los equipos de operaciones prueban los productos y envían los cambios a los equipos de desarrollo para que los incorporen. El resultado: gran cantidad de información y de trabajo sin estimar los tiempos y los costos; además de los objetivos y las tecnologías, que también pueden cambiar durante el ciclo del proyecto. Reporte Digital (2019).

Conforme a la investigación realizada por Apiumhub[3], expresa: *"El mayor desafío al que se enfrentan las empresas de tecnología está relacionado con la capacidad de entregar software que funciona mientras la cantidad de features a entregar está al límite. Este desafío es seguido de cerca por el compartir conocimientos."*

---

[2] *Las metodologías de desarrollo de software tienen como objetivo presentar un conjunto de técnicas tradicionales y modernas de modelado de sistemas que permitan desarrollar software de calidad, incluyendo heurísticas de construcción y criterios de comparación de modelos de sistemas.*

[3] *Apiumhub es un hub tecnológico especializado en arquitectura de software, aplicaciones móviles y desarrollo web.*

Gran parte de los fracasos se encuentran expresados en atrasos, en la no estimación de costos reales y a la no incorporación de prácticas esenciales y estandarización en los procesos de *software*; así mismo, por no llevar a la práctica las acciones correctas, no tomando decisiones en el momento oportuno, y no comprometiéndose. Uno de los mayores retos a los que se enfrentan muchas de las empresas de *software,* es hacer que estas prácticas esenciales hagan parte de su estrategia y el centro de todas sus actividades.

Una de las mayores dificultades a las que se enfrentan aquellos que se involucran en proyectos de desarrollo de *software*, es la gran diversidad de conceptos que resultan necesarios incorporar para adentrarse en el tema y el hecho de que coexisten; a su vez un gran número propuestas.

Pueden ser muchos los motivos por los cuales no se está generando *software* de calidad; sin embargo, por motivos del enfoque de este trabajo de grado solo se mencionarán aquellos que están relacionados directamente con el objeto de estudio; estos son:

- Uso de herramientas no estandarizadas.
- Desarrolladores heterogéneos.
- Metodologías de desarrollo no definidas.
- Procesos mal definidos, sin documentación y/o no auditados.
- Ausencia de roles, responsabilidades no definidas o flujo de trabajo no definido.

Con el objetivo de ser más competitivos, algunas empresas de *software* están implantando y brindando importancia a los procesos usados para el desarrollo y mantenimiento del *software*. A través de la mejora de sus procesos, estas organizaciones han obtenido una significativa mejora en la calidad de sus productos y buenos resultados en sus negocios.

3. OBJETIVOS

3.1. Objetivo General

Proponer un marco de trabajo (*framework*) que permita adoptar y documentar las buenas prácticas en el desarrollo de *software* empleando los fundamentos DevOps y los principios de las metodologías tradicionales, con el fin de definir y establecer procesos que evolucionen en el tiempo y así generar aplicativos de alta calidad.

3.2. Objetivos Específicos

- Elaborar un manual general de procesos de desarrollo de *software* con su respectivo diagrama bajo la notación BPM con los roles, procesos y tareas a ejecutar.

- Establecer un método permita caracterizar las empresas a través de la evaluación de diferentes categorías y así determinar si son aptas o no para proseguir con el proceso de adopción e implementación de DevOps.

- Definir y documentar, a través de un manual técnico, los *pipelines* de las fases de desarrollo, pruebas y producción acorde a la arquitectura de referencia del proyecto.

4. METODOLOGÍA

Para el desarrollo del presente trabajo de grado, se ha optado por emplear una metodología investigación descriptiva para la recolección de información y evidencias. En un sentido más específico, se seguirán los siguientes pasos:

- Definición de las palabras claves.
  Devops, fundamentos devops, best practices devops, software development, software testing, continuos integration, continuos delivery, continuos deployment, test automation, agile, devops trend,

- Reconocimiento de los distractores.
  Buenas prácticas, administrador de proyectos, integración, pruebas de software.

- Identificar fuentes de información alrededor del tema.

  Las fuentes de información principales han sido *papers* o publicaciones realizadas por otros investigadores sobre el tema puntual o temas de soporte alrededor del enfoque principal.

  Como referentes bibliográficos, se han empleado libros especializados en ingeniería de software. Por otra parte, se han consultado los artículos y publicaciones de empresas que se han dedicado al negocio de la implementación de DevOps para la estandarización y automatización de procesos para empresas dedicadas a la producción de software.

- Agrupar por tema o enfoque.
  Teniendo en cuenta los referentes, se establecieron varios grupos o categorías para agrupar cada una de las fuentes de información. Las

principales categorías son: Ingeniería de Software, Testing, DevOps, Arquitectura de software y desarrollo de software.

- Analizar las propuestas encontradas.

- Consolidar los resultados de la investigación.

- Elaboración de la propuesta.
  Finalmente, se procederá a construir la propuesta con la cual se buscará brindar una solución a la problemática planteada. Para un mejor entendimiento de la propuesta, su desarrollo se dividió en varias secciones ordenadas en una secuencia lógica que luego se convertirá en la estrategia metodológica para la ejecución de futuros proyectos de implementación.

5. ALCANCE

Este proyecto estará demarcado por la propuesta metodológica y la documentación de los procesos involucrados en la implementación de DevOps para una empresa de desarrollo de *software*.

Condiciones, Restricciones y limitaciones

DevOps es una metodología flexible con recursos suficientes para poder adoptarse a cualquier empresa sin importar su tamaño, el tipo de aplicaciones que desarrollan e incluso independientemente de las metodologías empleadas para el desarrollo de *software*. Sin embargo, la adopción y el cambio de cultura a DevOps implican una serie de compromisos y asignación de recursos económicos, de infraestructura y sobre todo recursos humanos. Entre estos se pueden destacar:

- Establecer un presupuesto para la contratación de recursos especializados y la tecnología.
- Nombrar a un director de proyecto *DevOps* a nivel ejecutivo.
- Los programas de capacitación del personal y las nuevas tecnologías serán fundamentales para el funcionamiento de *DevOps*.
- Crear cultura organizacional sobre las tecnologías relacionadas con la entrega a producción, los servicios de virtualización, la automatización de tareas y la gestión de versiones. Estas sólo pueden ser eficaces si al personal competente le importa el proyecto.

6. MARCO TEÓRICO

“El software es un lugar donde se siembran sueños y se cosechan pesadillas, una ciénaga abstracta y mística en la que terribles demonios luchan contra panaceas mágicas, un mundo de hombres lobos y balas de plata.”

Brad J. Cox

En el ámbito de las tecnologías de información, es indispensable que existan respuestas cada vez más rápidas a las necesidades de los clientes y servicios que sean estables, seguros y predecibles; esto establece un compromiso entre estabilidad y cambio. Trabajar en una solución que entregue cambios de forma frecuente, como pueden ser las metodologías ágiles, sin tener en cuenta la fiabilidad en la gestión operacional o la comunicación entre desarrollo y operaciones hace que en muchas ocasiones no sea suficiente.

El proceso de *software* es una estructura para las actividades, acciones y tareas que se requieren a fin de construir *software* de calidad. Un proceso del *software* define el enfoque adoptado mientras se hace ingeniería sobre el *software*. Pero la ingeniería de *software* también incluye tecnologías que pueblan el proceso: métodos técnicos y herramientas automatizadas. En el proceso de *software* encontramos unos elementos fundamentales, los cuales son: el proceso, el cual es un conjunto de actividades, acciones y tareas que se ejecutan cuando va a crearse algún producto del trabajo. Una actividad busca lograr un objetivo amplio y se desarrolla sin importar el dominio de la aplicación, tamaño del proyecto, complejidad del esfuerzo o grado de rigor con el que se usará la ingeniería de *software*. Una acción es un conjunto de tareas que produce un producto importante de trabajo. Una tarea se centra en un objetivo pequeño pero bien definido que produce un resultado tangible.

Las condiciones del mercado cambian con rapidez, las necesidades de los usuarios finales se transforman y emergen nuevas amenazas competitivas sin

previo aviso. En muchas situaciones no será posible definir los requerimientos por completo antes de que el proyecto comience.

De acuerdo a la Alianza Ágil[4], existen cuatro valores fundamentales para el desarrollo de *software*:

- ***Individuos e interacciones***, sobre procesos y herramientas.
- ***Software que trabaja,*** sobre una amplia documentación.
- ***Colaboración con el cliente,*** sobre negociación de contratos.
- ***Respondiendo al cambio***, sobre seguir un plan.

Estos cuatro valores fundamentales fueron definidos en el manifiesto ágil escrito en el 2001 por un grupo de profesionales de *software* de mentalidad independiente; que, aunque no estaban de acuerdo acerca de muchas cosas, encontraron un consenso común.

Por otra parte, los doce principios fundamentales (Agile Alliance) que se basan en el manifiesto ágil son:

1. La máxima prioridad es satisfacer al cliente a través de la entrega temprana y continua de *software* valioso.
2. Los procesos ágiles aprovechan el cambio para obtener ventajas competitivas del cliente. Es por esto, *"Bienvenidos los requisitos cambiantes, incluso tarde en el desarrollo"*.
3. Entregar *software* que trabaja con frecuencia desde, un par de semanas hasta un par de meses, con una preferencia a la escala de tiempo más corto.
4. La gente de negocios y desarrolladores deben trabajar juntos todos los días durante todo el proyecto.

---

[4] *Alianza Ágil. Es una organización sin ánimo de lucro dedicada a promover los conceptos de desarrollo de software ágil como se indica en el manifiesto ágil.*

5. Construir proyectos en torno a individuos motivados; darles el medio ambiente y el apoyo que necesitan, y confiar en ellos para hacer el trabajo.
6. El método más eficiente y eficaz de transmitir información hacia y dentro de un equipo de desarrollo es la conversación cara a cara.
7. El *software* que funciona es la principal medida de progreso.
8. Los procesos ágiles promueven el desarrollo sostenible. Los patrocinadores, desarrolladores y usuarios deben ser capaces de mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica y el buen diseño mejora la agilidad.
10. Simplicidad – el arte de maximizar la cantidad de trabajo no realizado – es esencial.
11. Las mejores arquitecturas, diseños y requisitos emergen de equipos auto-organizados.
12. A intervalos regulares, el equipo reflexiona sobre ser más eficaces, en consecuencia, ajuste y sintonice el comportamiento del equipo.

"Las organizaciones se enfrentan a la necesidad de adaptarse a un entorno empresarial complejo, en continuo cambio y transformación. Bajo estas circunstancias, la agilidad de la organización es un elemento clave para obtener ventajas estratégicas y éxito en el mercado. Lograr y mantener la agilidad, requiere arquitecturas, técnicas, métodos y herramientas ágiles, capaces de reaccionar en tiempo real para cambiar los requisitos. Los modelos de desarrollo son varios procesos o metodologías seleccionadas para desarrollar el proyecto conforme a sus propósitos y objetivos. Los modelos de desarrollo de software ayudan a mejorar tanto la calidad del software como los procesos de desarrollo en general. Hay varios modelos para el ciclo de vida del desarrollo de software, cada uno desarrollado para ciertos objetivos."(Stoica, 2013).

El ciclo de vida del desarrollo de *software* – SDLC[5] es un entorno que describe las actividades realizadas en cada etapa del proceso de desarrollo de *software*. SDLC consiste de un plan detallado que describe como el desarrollo, el mantenimiento y el reemplazo de *software* específico. Esto también se conoce como proceso de desarrollo de *software*.

El estándar internacional para SDLC es ISO/IEC 12207, que ayuda a definir todas las actividades requeridas para el desarrollo y mantenimiento de *software*. A continuación se muestra el ciclo de vida del desarrollo de *software*.

Imagen 1. Ciclo de vida del desarrollo de software

- *Planeación y análisis de requerimientos.* El análisis de requerimientos es la etapa más importante del SDLC. Es realizada por miembros sénior del equipo, utilizando insumos de los clientes, departamento de ventas, investigación de mercados y expertos en la industria. Esta información se

[5] *SDLC, por sus siglas en inglés. Sofware Development Life Cycle.*

utiliza para un plan de proyecto básico y un estudio de viabilidad desde el punto de vista económico, operativo y técnico.

- *Definición de los requerimientos.* Tan pronto los requisitos son analizados, los requerimientos del producto son claramente definidos y documentados. Estos deben ser aprobados por el cliente o por los analistas de mercado a través de un SRS – *Software* Requirement Specification (Especificación de requerimientos de *software*). El documento SRS lista todos los requerimientos del producto que deben ser diseñados y desarrollados durante todo el ciclo de vida del proyecto.
- *Diseño de la arquitectura del producto.* El SRS es la referencia básica desde la cual los arquitectos se disponen a crear la mejor arquitectura para el producto. Generalmente se propone, al menos una, propuesta de arquitectura del producto y es documentado en un DDS – Design Document Specification (Especificación del Documento de diseño). Este DDS es revisado por todas las partes interesadas y la mejor propuesta es seleccionada, se establecen algunos parámetros como: evaluación de riesgo, robustez del producto, método de diseño, presupuesto y limitaciones de tiempo.
- *Implementación o desarrollo del producto.* En esta etapa del SDLC inicia el desarrollo. El código fuente es generado durante esta etapa. Si el diseño fue realizado de manera detallada y organizada, el código fuente podrá ser realizado sin complicaciones. Los desarrolladores deben seguir los lineamientos de su organización. Para generar el código utilizan herramientas de programación como compiladores, Entornos de desarrollo integrado – IDEs, depuradores, etc.
- *Pruebas del producto.* Esta etapa suele ser un subconjunto de todas las etapas en los modelos modernos de SDLC, ya que las pruebas involucran todas las etapas. Aun así, en esta etapa es donde las fallas del producto

son reportadas, rastreadas, corregidas y re-analizadas hasta que cumplan con los requisitos de calidad del SRS.

- *Mantenimiento y operaciones de mercadeo.* Una vez el producto ha sido probado, está listo para lanzarse al mercado. Puede ser lanzado en un segmento limitado y probado en un entorno real de negocio, así, basados en la realimentación recibida, se puede lanzar al mercado sin cambios o con las mejoras sugeridas por los clientes involucrados en las pruebas.

DevOps responde a la necesidad experimentada por el sector tecnológico de dar una respuesta más rápida a la implementación y operación de aplicaciones. Al momento de adoptar el cambio cultural, las empresas pueden reducir el tiempo que toma cumplir ese ciclo de vida de sus aplicaciones; DevOps es una metodología de trabajo basada en el desarrollo de código que usa nuevas herramientas y prácticas para reducir la tradicional brecha entre los desarrolladores y el personal de TI. Este nuevo enfoque de colaboración permite a los equipos trabajar de manera más cercana, aportando mayor agilidad al negocio y notables incrementos de productividad.

6.1. ¿Qué es DevOps?

Al momento de definir el significado de DevOps, nos encontramos con diferentes definiciones, que, aunque puedan variar conservan ciertas similitudes. A continuación, se exponen diferentes conceptos de DevOps de diferentes autores.

"*DevOps es un conjunto de prácticas que automatizan los procesos entre los equipos de desarrollo de software y TI para que puedan compilar, probar y publicar software con mayor rapidez y fiabilidad. El concepto de DevOps se basa en establecer una cultura de colaboración entre equipos que, tradicionalmente, trabajaban en grupos aislados. Entre las ventajas que promete, se incluyen el aumento de la confianza y de la velocidad de publicación de software, la*

capacidad de solucionar incidencias críticas rápidamente y una mejor gestión del trabajo imprevisto."(Attlasian, 2018)

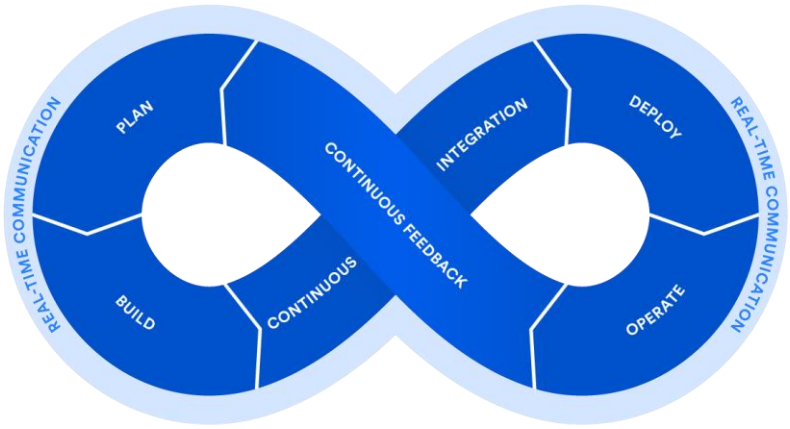


Imagen 2. Diagrama DevOps

En el artículo ***DevOps: Qué es y cómo lo aplicamos***; ClaraNet[6] (2016), presenta su propia definición:

"Hay cientos de definiciones de DevOps. Sin embargo, la idea de base detrás de todas ellas es la misma: la de una organización alineada e integrada que facilita la aceleración del ciclo de vida de aplicaciones. Por eso DevOps no es algo que puedas comprar y llevarte puesto, implica una actitud, unos tiempos y un soporte completamente nuevos."

Por otra parte, IBM en su artículo ***DevOps. The IBM Approach. Continuos delivery of software-driven innovation***, concibe a DevOps como:

"El espíritu de DevOps es la colaboración ampliada entre todos los interesados, no solo entre los equipos de desarrollo y operaciones, sino también entre las líneas de negocio, proveedores involucrados en la entrega del software y

[6] *Claranet es uno de los mayores proveedores de servicios gestionados de Europa. Ofrecemos soluciones de cloud, hosting y redes a miles de clientes empresariales de todos los sectores de actividad.*

los propios consumidores. En este amplio sentido, DevOps incluye prácticas de gobierno empresarial alrededor de la seguridad y el cumplimiento, y todos los aspectos del proceso de entrega tales como el suministro múltiple."

Uno de los objetivos de DevOps es reducir los tiempos de cada ciclo de desarrollo, es decir, reducir el tiempo necesario para que cada uno de los requisitos solicitados por los usuarios finales llegue a sus manos. Pressman (2010), expresa que:

> *La existencia de un proceso del software no es garantía de que el software se entregue a tiempo, que satisfaga las necesidades de los consumidores o que tenga las características técnicas que conducirán a características de calidad de largo plazo. […]Los patrones de proceso deben acoplarse con una práctica sólida de ingeniería de software. Además, el proceso en sí puede evaluarse para garantizar que cumple con ciertos criterios de procesos básicos que se hayan demostrado que son esenciales para el éxito de la ingeniería de software.* (pag. 58).

> *En el contexto de la ingeniería de software, un proceso no es una prescripción rígida de cómo elaborar software de cómputo. Por el contrario, es un enfoque adaptable que permite que las personas que hacen el trabajo busquen y elijan el conjunto apropiado de acciones y tareas para el trabajo. […] La estructura del proceso establece el fundamento para el proceso completo de la ingeniería de software por medio de la identificación de un número pequeño de actividades estructurales que sean aplicables a todos los proyectos de software, sin importar su tamaño o complejidad. Además, la estructura del proceso incluye un conjunto de actividades sombrilla que son aplicables a través de todo el proceso del software. Una estructura de proceso general para la ingeniería de software consta de cinco actividades: Comunicación,*

planeación, modelado, construcción y despliegue […] (Pressman, 2010, pag. 39).

Adicionalmente, DevOps reorganiza los departamentos de desarrollo y operaciones para eliminar los límites operativos que actualmente generan ineficiencia en términos de tiempo de comercialización de nuevas características y calidad de *software*. Incluye la implementación de la entrega continua, la integración continua, las pruebas automatizadas, el monitoreo de las aplicaciones y otras buenas prácticas en el desarrollo y las operaciones de *software*. Es por esto, que DevOps propone la máxima automatización de cada uno de estás etapas de forma tal que la intervención humana sea mínima y solo cuando la situación lo amerite.

La integración, el despliegue y la entrega continua, son algunas técnicas importantes que soportan el concepto de DevOps. A continuación, se brinda una breve descripción:

- La **integración continua** es una práctica de desarrollo de *software* mediante el cual los desarrolladores combinan los cambios en el código en un repositorio central de forma periódica, tras lo cual se ejecutan versiones y pruebas automáticas. La integración continua se refiere en su mayoría a la fase de creación o integración del proceso de publicación de *software* y conlleva un componente de automatización (servicios de versiones) y un componente cultural (aprender a integrar con frecuencia). Los objetivos claves de la integración continua consisten en encontrar y corregir errores con mayor rapidez, mejorar la calidad del *software* y reducir el tiempo que se tarda en validar y publicar nuevas versiones de este.

- El **despliegue y la entrega continua** son técnicas complementarias a la integración continua que se centran en el despliegue de las aplicaciones. Se debe tener en cuenta, que la entrega continua no consiste en desplegar en producción cada cambio lo antes posible, sino que cada cambio debe estar disponible para ser desplegado en cualquier momento.

Estos paradigmas se centran en automatizar lo máximo posible todas las acciones necesarias para implantar una nueva versión de la aplicación y todas las tareas necesarias para validar esa nueva versión. La principal diferencia entre estos paradigmas es que la entrega continua requiere de una validación manual antes de la implantación en producción; por otra parte, el despliegue continuo se realiza de forma automática una vez que se cubren todos los criterios definidos para la entrada en producción para una aplicación. (DEVOPSTI, 2014).

6.2. Historia de DevOps.

Los antecedentes de DevOps se remontan a mediados del siglo pasado, cuando en 1956 aparece el modelo en cascada, el primer modelo de estandarización del proceso de desarrollo. Este engloba los procesos de análisis, diseño, implementación, pruebas y mantenimiento. Y, debido a las limitaciones tecnológicas del momento, ese modelo se mantuvo en uso durante un periodo largo de tiempo. Con el paso del tiempo aparecieron problemas con el modelo en cascada, las indecisiones y contradicciones del cliente suponían un obstáculo y la necesidad de una solución se resolvió cuando apareció la metodología Agile.

Una explicación rápida y sencilla sería imaginarse que AGILE es como un modelo en cascada en tamaño reducido, más corto y con más repeticiones, constantes, hasta el momento de entrega del producto. Con este cambio se ejercía mayor presión hacia los equipos de desarrollo, pues cada vez se encontraban con la

necesidad de hacer *despliegues* (poner la aplicación en producción) más a menudo.

El movimiento DevOps empezó a fusionarse entre el 2007 y el 2008, cuando las comunidades de operaciones de TI y desarrollo de *software* manifestaron claramente lo que veían como un nivel fatal de disfunción del sector.

Se alzaron contra el modelo tradicional de desarrollo de *software*, que exigía que los que escribían el código se mantuvieran al margen, en términos de organización y operación, de los que desplegaban y mantenían tal código.

Los desarrolladores y profesionales de TI/Operaciones tenían objetivos distintos (y, a menudo, rivales), direcciones de departamento independientes, indicadores clave del rendimiento diferentes por los que se les evaluaba y, con frecuencia, trabajaban en plantas separadas o, incluso, en edificios separados. El resultado eran equipos aislados únicamente preocupados por sus propios dominios, largas jornadas, publicaciones chapuceras y clientes insatisfechos.

Lo que empezó en foros de internet y reuniones locales es ahora uno de los aspectos principales del ámbito del *software* actual, y seguramente lo que te ha traído hasta aquí.

6.3. CALMS. Mejores prácticas para DevOps. (Cameron, 2018)

Por sus siglas en ingles *Culture, Automation, Lean, Measurement and Sharing*. Es particularmente útil para una estructura DevOps de una organización, y que últimamente, es útil para cualquier organización. El *framework*[7] CALMS cubre

[7] *Framework (Entorno o marco de trabajo). Es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar. Consultado de https://es.wikipedia.org/wiki/Framework.*

todos los involucrados en DevOps, incluyendo los equipos de negocio, operaciones TI, QA, seguridad de la información y desarrollo, y como entregan de forma colectiva.

Aplicando CALMS.

Aquí se definen algunas formas en las que CALMS puede aplicarse a DevOps para empresas, ya sea pensando en cambiar a una estructura DevOps o buscar mejoras en una implementación existente. Attlasian (2019) define CALMS, como:

Culture (Cultura). Si la cultura de DevOps se pudiera resumir en una palabra, esta sería ‘colaboración’ y, mejor aún, ‘colaboración transversal’ […] Todas las herramientas y automatizaciones del mundo no sirven para nada si no van acompañadas de una voluntad real de trabajar juntos por parte de los profesionales de Desarrollo y TI/Operaciones, porque DevOps no soluciona los problemas relacionados con herramientas, sino los problemas humanos.

Automation (Automatización). Invertir en automatización suprime el trabajo manual repetitivo, genera procesos reproducibles y crea sistemas fiables. Compilar, probar, desplegar y automatizar son los puntos de partida típicos de los equipos que aún no la han puesto en práctica […]. Los equipos que son nuevos en la automatización suelen empezar con la entrega continua: la práctica de ejecutar cada cambio en el código mediante un puñado de pruebas automatizadas, a menudo facilitadas por una infraestructura basada en la nube; a continuación, empaquetar compilaciones satisfactorias y promoverlas hasta producción con despliegues automatizados.

Lean (Apoyo / infalible). Cuando nos dicen “infalible” en un contexto de *software*, solemos pensar en suprimir actividades de escaso valor y avanzar rápido: ser enérgico, ser ágil. Más oportunos aún para DevOps son los conceptos de mejora

continua y aceptación de los errores. Una mentalidad de DevOps ve oportunidades de mejora continua en todas partes. Algunas resultan obvias, como mantener retrospectivas periódicas para mejorar los procesos del equipo. Otras son más sutiles, como las pruebas A/B en distintos métodos de incorporación para nuevos usuarios del producto.

Measurement (Medición). Sin datos, es difícil demostrar que su esfuerzo continuo por mejorar esté mejorando algo en efecto. Por suerte, hay un montón de herramientas y tecnologías para medir el rendimiento: cuánto tiempo pasan los usuarios con su producto, si esa entrada del blog ha generado alguna venta o con cuánta frecuencia aparecen alertas críticas en los registros. Aunque se puede medir casi todo, eso no quiere decir que lo tengas que medir todo. Sigue el ejemplo del desarrollo ágil y empieza por lo básico:

- ¿Cuánto tiempo llevó pasar del desarrollo al despliegue?
- ¿Con qué frecuencia tienen lugar errores o fallos?
- ¿Cuánto se tarda en recuperarse tras un fallo del sistema?
- ¿Cuántas personas utilizan su producto en este momento?
- ¿Cuántos usuarios has ganado o perdido esta semana?

Sobre una base sólida implantada, cuesta menos detectar métricas más sofisticadas del uso de las funcionalidades, el recorrido de los clientes, y los SLA (acuerdos de nivel de servicio). La información que se obtiene viene muy bien a la hora de confeccionar hojas de ruta y especificar el siguiente gran paso.

Share (Compartir). Los equipos que adoptan DevOps suelen tener una función rotativa en la que los desarrolladores tratan las incidencias detectadas por los usuarios finales, mientras que solucionan problemas de producción al mismo tiempo. Esta persona da respuesta a las incidencias urgentes que los clientes han notificado, creando parches cuando haga falta, y trabaja con el *backlog* de los defectos notificados por clientes. El “desarrollador de apoyo” aprende mucho sobre el uso que se le da a la aplicación en la vida real. Y gracias a su gran

disponibilidad para con el equipo de operaciones, los equipos de desarrollo fomentan la confianza y el respeto mutuo. […] El *feedback* positivo de los compañeros nos motiva tanto como nuestras ambiciones profesionales o la nómina. Reconocer públicamente a un compañero de equipo que ha detectado un molesto error antes de que salga a producción es muy importante.

A través de DevOps, se intenta simplificar y unificar el trabajo al equipo de desarrollo, evitar que un desarrollador tenga que ejercer roles de programación y sistemas a la vez, lo cual resulta muy dispendioso; aunque se debe tener en cuenta que, actualmente, en muchas empresas continúa ocurriendo. En cambio, con una metodología DevOps estos problemas se solucionan o pueden gestionarse con mayor agilidad. Esta evolución ha conllevado a que cada vez sean más las empresas del sector que se enfocan en la búsqueda de personal con perfil DevOps para incorporarlos en sus empresas. Es decir, un trabajador con habilidad para la gestión de sistemas, despliegue de aplicaciones, gestión de base de datos, ejecución de scripting, etc.

6.4. Ventajas de implementar DevOps en una organización.

La compañía Aplyca[8] (2018), comenta: “Algunas de las principales ventajas de adoptar las prácticas de CI/CD incluyen las siguientes:

- Entregar o liberar código de forma inmediata, sin esperar a acumular muchas funcionalidades, tickets o desarrollos independientes.
- Menor esfuerzo y más confianza al momento de desplegar código a la etapa de producción.

[8] *Aplyca es una empresa de consultoría y servicios profesionales en tecnologías de información en dos grandes áreas: desarrollo ágil de software e infraestructura de alto rendimiento. Especializada en soluciones Cloud y desarrollo de aplicaciones.*

- Se facilita el trabajo en equipo y la comunicación entre los desarrolladores de *software*.
- Mayor visibilidad del código que se está desarrollando.
- Se pueden identificar errores antes de publicar el código.

En algunos casos es posible que los miembros más antiguos del equipo de trabajo se resistan a la adopción de CI/CD en el flujo de trabajo, pues la necesidad de reorganizar algunos roles, de utilizar nuevas herramientas y de poner al descubierto debilidades técnicas llevan a que se perciba el cambio como una amenaza.

Por esta razón es fundamental definir desde el inicio del proyecto la forma en que la metodología CI/CD puede beneficiar al equipo, teniendo claridad respecto a cuáles serán las acciones, responsabilidades y funciones de cada uno de los integrantes del mismo. A través de la metodología CI/CD el equipo tiene completo control del proceso de desarrollo y producción, probando e integrando cambios para verificar que todo funcione correctamente. De esta forma es posible identificar fácilmente cualquier error que impida su funcionamiento correcto.”.

El nivel de esfuerzo requiere integrar y desplegar código pone en evidencia el nivel de madurez de los procesos técnicos de integración continua y entrega continua (CI/CD). Hacer una sola integración de código al final del proceso de desarrollo, acumular código para desplegar cada semana o cada quince días o en general una tasa alta de fallas en los despliegues son un indicador importante de una oportunidad de mejora.

La Integración Continua (CI) es una práctica que incrementa la eficacia y la eficiencia de los resultados del equipo de desarrolladores de *software*. Se trata de combinar, de forma periódica y en un repositorio central, los cambios realizados en

el código de un proyecto, para luego ejecutar pruebas, detectar y reparar errores lo antes posible.

Esta metodología permite mejorar la calidad del código, entregar avances al cliente más a menudo y trabajar ágilmente con nuevos participantes del equipo de desarrollo, incluso si estos no conocen en profundidad el proyecto completo.

6.5. El ciclo DevOps y los componentes requeridos para su implementación.

6.5.1. Definición del ciclo DevOps.

Imagen 3. Ciclo DevOps

6.5.2. Componentes recomendados para la automatización del proceso de *software*.

- **Herramientas de seguimiento y administración de tareas de proyectos.**

Estas herramientas son usadas para ayudar a simplificar estos procesos, son plataformas dedicadas que permite a los equipos rastrear proyectos desde el inicio hasta la entrega, desde generar un error hasta la resolución. Las herramientas de seguimiento de proyectos ayudan a mantener a los equipos enfocados, proporcionando líneas de tiempo visuales en cuanto al progreso de la carga de trabajo, que se puede actualizar en segundos. Permiten la estructura y la entrega mediante la eliminación de las dependencias y unificar el progreso del proyecto en un solo lugar, eliminando la necesidad de correos electrónicos torpes y hojas de cálculo.

En esencia, las herramientas de seguimiento de proyectos están diseñadas para simplificar las tareas que conforman el día a día de la vida laboral. Se esfuerzan por permitir que el usuario sea más productivo sin tener que esforzarse más. Esto, en su mayor parte, se logra al permitir que los equipos agreguen más atención a su agenda. Las líneas de tiempo, scrum personalizable y tableros Kanban son características que permiten a los equipos dividir sus entregas en pequeños trozos de trabajo iterativos, fácilmente manejables, que agregan valor al producto central. Muchos también cuentan con retrasos compartidos, que permiten a los equipos individuales visualizar el alcance, manteniendo el proyecto enfocado y ágil.

Las herramientas de seguimiento de proyectos ofrecen un nivel de racionalización, que elimina la necesidad de dependencias de software y ofrece una ubicación más centralizada para el seguimiento de proyectos. Tener un punto de referencia singular para el estado de un proyecto, que se puede actualizar en tiempo real, otorga claridad en cuanto al progreso de cada equipo involucrado. Las herramientas, algunas de las cuales son de código abierto, ofrecen diferentes

niveles de funcionalidad, lo que le permite elegir la solución que mejor se adapte a su organización.

Algunos ejemplos de este tipo de herramientas son: Jira, MantisBt, Trello, entre otros.

- **Repositorios de código y control de versiones.**

Un repositorio es un espacio centralizado donde se almacena, organiza, mantiene y difunde información digital, habitualmente archivos informáticos, que pueden contener trabajos científicos, conjuntos de datos o *software*. Los repositorios tienen sus inicios en los años 90, en el área de la física y las matemáticas, donde los académicos aprovecharon la red para compartir sus investigaciones con otros colegas. Este proceso era valioso porque aceleraba el ciclo científico de publicación y revisión de resultados.

Los datos almacenados en un repositorio pueden distribuirse a través de una red informática, como Internet, o de un medio físico, como un disco compacto. Pueden ser de acceso público o estar protegidos y necesitar de una autentificación previa. Los repositorios más conocidos son los de carácter académico e institucional. Los repositorios suelen contar con sistemas de respaldo y mantenimiento preventivo y correctivo, lo que hace que la información se pueda recuperar en el caso que la máquina quede inutilizable. A esto se lo conoce como preservación digital, y requiere un exhaustivo trabajo de control de calidad e integridad para realizarse correctamente (Wikipedia, 2018).

El control de versiones es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante. A pesar de que los ejemplos de este libro muestran código fuente como archivos bajo control de versiones, en

realidad cualquier tipo de archivo que encuentres en un ordenador puede ponerse bajo control de versiones.

Si eres diseñador gráfico o web, y quieres mantener cada versión de una imagen o diseño (algo que sin duda quieres), un sistema de control de versiones (Version Control System o VCS en inglés) es una elección muy sabia. Te permite revertir archivos a un estado anterior, revertir el proyecto entero a un estado anterior, comparar cambios a lo largo del tiempo, ver quién modificó por última vez algo que puede estar causando un problema, quién introdujo un error y cuándo, y mucho más. Usar un VCS también significa generalmente que, si fastidias o pierdes archivos, puedes recuperarlos fácilmente. Además, obtienes todos estos beneficios a un coste muy bajo. (Gitlab, 2015).

- **Sistemas de control de versiones locales**

Un método de control de versiones usado por mucha gente es copiar los archivos a otro directorio (quizás indicando la fecha y hora en que lo hicieron, si son avispados). Este enfoque es muy común porque es muy simple, pero también tremendamente propenso a errores. Es fácil olvidar en qué directorio te encuentras, y guardar accidentalmente en el archivo equivocado o sobrescribir archivos que no querías.

Para hacer frente a este problema, los programadores desarrollaron hace tiempo VCSs locales que contenían una simple base de datos en la que se llevaba registro de todos los cambios realizados sobre los archivos (véase Figura 1-1).

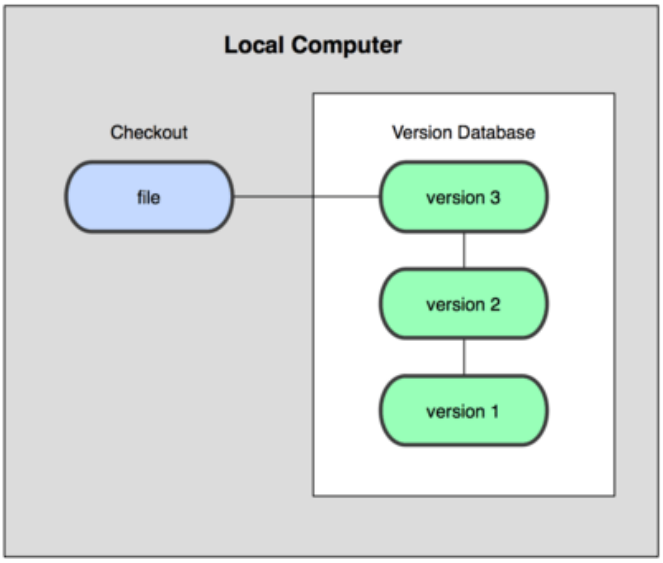


Imagen 4. Esquema de control de versiones local

- **Sistemas de control de versiones centralizados**

El siguiente gran problema que se encuentra la gente es que necesitan colaborar con desarrolladores en otros sistemas. Para solventar este problema, se desarrollaron los sistemas de control de versiones centralizados (Centralized Version Control Systems o CVCSs en inglés). Estos sistemas, como CVS, Subversion, y Perforce, tienen un único servidor que contiene todos los archivos versionados, y varios clientes que descargan los archivos desde ese lugar central. Durante muchos años éste ha sido el estándar para el control de versiones (véase Figura 1-2).

Esta configuración ofrece muchas ventajas, especialmente frente a VCSs locales. Por ejemplo, todo el mundo puede saber (hasta cierto punto) en qué están trabajando los otros colaboradores del proyecto. Los administradores tienen control detallado de qué puede hacer cada uno; y es mucho más fácil administrar un CVCS que tener que lidiar con bases de datos locales en cada cliente.

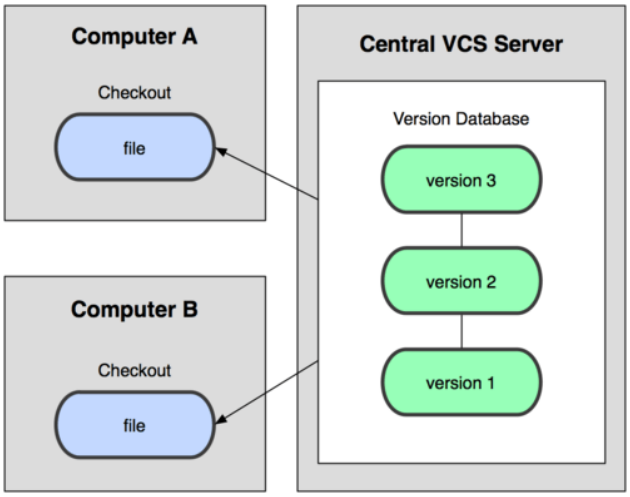


Imagen 5. Esquema VCS Centralizado

Sin embargo, esta configuración también tiene serias desventajas. La más obvia es el punto único de fallo que representa el servidor centralizado. Si ese servidor se cae durante una hora, entonces durante esa hora nadie puede colaborar o guardar cambios versionados de aquello en que están trabajando. Si el disco duro en el que se encuentra la base de datos central se corrompe, y no se han llevado copias de seguridad adecuadamente, pierdes absolutamente todo —toda la historia del proyecto salvo aquellas instantáneas que la gente pueda tener en sus máquinas locales. Los VCSs locales sufren de este mismo problema— cuando tienes toda la historia del proyecto en un único lugar, te arriesgas a perderlo todo.

- **Sistemas de control de versiones distribuidos.**

Es aquí donde entran los sistemas de control de versiones distribuidos (Distributed Version Control Systems o DVCSs en inglés). En un DVCS (como Git, Mercurial, Bazaar o Darcs), los clientes no sólo descargan la última instantánea de los archivos: replican completamente el repositorio. Así, si un servidor muere, y estos sistemas estaban colaborando a través de él,

cualquiera de los repositorios de los clientes puede copiarse en el servidor para restaurarlo. Cada vez que se descarga una instantánea, en realidad se hace una copia de seguridad completa de todos los datos (véase Figura 1-3).

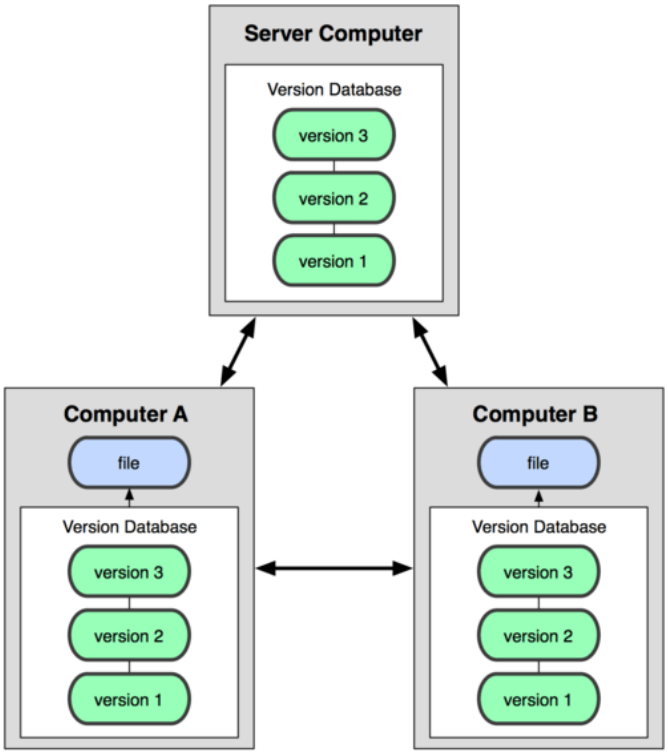


Imagen 6. Esquema VCS Distribuido

Es más, muchos de estos sistemas se las arreglan bastante bien teniendo varios repositorios con los que trabajar, por lo que puedes colaborar con distintos grupos de gente simultáneamente dentro del mismo proyecto. Esto te permite establecer varios flujos de trabajo que no son posibles en sistemas centralizados, como pueden ser los modelos jerárquicos.

- ***Software* para la gestión y construcción de proyectos.**

Build automation es el proceso de automatización, la creación de construcción de software y los procesos asociados incluyendo: la compilación del código fuente de

computadora en código binario, el empaquetamiento del código binario y la ejecución de las pruebas automatizadas.

Historicamente, *build automation* se lograba a través de archivos *makefiles* o archivos de creación. Actualmente, hay dos categorías generales de herramientas:

Utilidades Build-automation. Permite la automatización simple de tareas repetibles. Cuando usas la herramienta, calculará como alcanzar los objetivos mediante la automatización de tareas en el orden correcto y específico y la ejecución de cada tarea. Las dos formas en la que las herramientas de construcción difieren son orientadas a tareas y orientada a productos. Las herramientas orientadas a tareas describe la dependencia o redes en términos de un conjunto de tareas específicas y las herramientas orientadas a proyectos describen cosas en términos de los productos que ellos generan.

Esto incluye utilidades como *Make, Rake, Cake, MSBuild, Ant, Maven o Gradle* (Java), etc. Su propósito general es generar artefactos (*build artifact*) a través de actividades como compilación y el enlace del código fuente.

Servidores Build-automation. Aunque los servidores de compilación existían mucho antes que los servidores de integración continua, generalmente son sinónimos; sin embargo, un servidor de construcción también puede ser incorporado en una herramienta ARA[9] o una herramienta ALM[10].

Son herramientas generales basadas en la web que ejecutan utilidades *build-automation* de forma programada o por disparadores (*triggers*) básicos. Un servidor de integración continua es un servidor de tipo build-automation.

---

[9] Application-release automation. Se refiere al proceso de empaquetamiento y despliegue de una aplicación o actualización de una aplicación desde desarrollo a través de varios ambientes y finalmente a producción.

[10] *Application lifecycle management. Es el administrador del ciclo de vida del producto, engloba administrador de requerimientos, aruitectura de software, desarrollo, pruebas de software, mantenimiento de software, administrador de cambios, integración continua, administración de proyectos y administrador de despliegues.*

- **Automatización de Pruebas.**

La automatización de pruebas consiste en el uso de *software* especial (casi siempre separado del *software* que se prueba) para controlar la ejecución de pruebas y la comparación entre los resultados obtenidos y los resultados esperados. La automatización de pruebas permite incluir pruebas repetitivas y necesarias dentro de un proceso formal de pruebas ya existente o bien adicionar pruebas cuya ejecución manual resultaría difícil. Algunas pruebas de *software* tales como las pruebas de regresión intensivas de bajo nivel pueden ser laboriosas y consumir mucho tiempo para su ejecución si se realizan manualmente. Adicionalmente, una aproximación manual puede no ser efectiva para encontrar ciertos tipos de defectos, mientras que las pruebas automatizadas ofrecen una alternativa que lo permite. Una vez que una prueba ha sido automatizada, ésta puede ejecutarse repetitiva y rápidamente en particular con productos de *software* que tienen ciclos de mantenimiento largo, ya que incluso cambios relativamente menores en la vida de una aplicación pueden inducir fallos en funcionalidades que anteriormente operaban de manera correcta. Existen dos aproximaciones a las pruebas automatizadas:

- **Pruebas manejadas por el código**: Se prueban las interfaces públicas de las clases, módulos o bibliotecas con una variedad amplia de argumentos de entrada y se valida que los resultados obtenidos sean los esperados.
- **Pruebas de Interfaz de Usuario**: Un marco de pruebas genera un conjunto de eventos de la interfaz de usuario, tales como teclear, hacer click con el ratón e interactuar de otras formas con el *software* y se observan los cambios resultantes en la interfaz de usuario, validando que el comportamiento observable del programa sea el correcto.

La elección misma entre automatización y ejecución manual de pruebas, los componentes cuya prueba será automatizada, las herramientas de automatización y otros elementos son críticos en el éxito de las pruebas, y por

lo regular deben provenir de una elección conjunta de los equipos de desarrollo, control de calidad y administración. Un ejemplo de mala elección para automatizar sería escoger componentes cuyas características son inestables o su proceso de desarrollo implica cambios continuos.

Pruebas manejadas por el código

En el desarrollo contemporáneo de *software* existe una tendencia creciente a usar *Frameworks* como los denominados XUnit (por ejemplo, JUnit y NUnit) que permiten la ejecución de pruebas unitarias para determinar cuándo varias secciones del código se comportan como es esperado en circunstancias específicas. Los casos de prueba describen las pruebas que han de ejecutarse sobre el programa para verificar que éste se ejecuta tal y como se espera. La automatización de pruebas es una característica clave del desarrollo ágil de *software* en donde se le conoce como "desarrollo guiado por pruebas". En ellas, las pruebas unitarias se escriben antes que el código que genera la funcionalidad. Sólo cuando el código pasa exitosamente las pruebas se considera completo. Cuando hay cambios, el programador descubre inmediatamente cualquier defecto que rompa los casos de prueba lo cual baja el costo de la reparación. Dos inconvenientes de este estilo de trabajo son:

1. Algunas veces se "desperdicia" la capacidad del programador escribiendo las pruebas unitarias. El entrecomillado se debe precisamente que asegurar la calidad del producto no es desperdicio alguno.
2. Normalmente se prueban los requerimientos básicos o el flujo normal del caso de uso en vez de todos los flujos alternativos, dado que extender las pruebas más allá de la prueba base eleva el costo del producto. En algunas ocasiones los flujos alternativos son probados por un equipo de pruebas más o menos independiente del equipo de desarrollo.

Pruebas de Interfaz de Usuario.

Muchas herramientas de automatización de pruebas proveen características para grabar y reproducir acciones del usuario para posteriormente ejecutarlas un número indefinido de veces, comparando resultados obtenidos con resultados esperados. La ventaja de esta aproximación a la automatización es que requiere de menos desarrollo de *software*, sin embargo el confiar en éstas características del *software* lo hace menos confiable en la medida que muchas veces dependen de la etiqueta o posición del elemento de interfaz, y, al cambiar, el caso de prueba debe ser adaptado al cambio o probablemente fallar. Una variante de estas pruebas es la prueba de sistemas basados en la web en las que la herramienta de prueba ejecuta acciones sobre el navegador e interpreta el HTML resultante. Una variación más es la automatización sin scripts, que no usa grabación y reproducción de acciones, sino que construye un modelo de la Aplicación Bajo Prueba ABP (AUT en sus siglas en inglés) que permite a la persona que prueba ("tester") que cree pruebas simplemente editando parámetros y condiciones

- **Herramienta de Orquestación.**

A medida que la velocidad se vuelve cada vez más vital, las organizaciones continúan adoptando más y más herramientas para obtener alguna ventaja. De hecho, a medida que un negocio crece puede absorber nuevas compañías junto con sus diferentes conjuntos de herramientas y procesos. Rápidamente esto puede conducir al fenómeno conocido como “expansión de cadena de herramientas”. La expansión de cadena de herramientas trae una variedad de riesgos y desafíos de gestión, incluida la falta de control y visibilidad. Y si sus herramientas DevOps no están correctamente organizadas, pueden retrasarlo y dificultar la calidad de sus lanzamientos; lo opuesto a sus intenciones. Las herramientas de orquestación integran dentro de sus sistemas existentes que le

permite administrar y alinear con éxito su cadena de herramientas DevOps de una manera eficiente y transparente, todo desde una interfaz centralizada.

Las herramientas de orquestación son una capa automatizada adicional que se asienta sobre la cadena de herramientas pre-existentes. Le permiten integrar y organizar su cadena de herramientas DevOps en un proceso de extremo a extremo. Esto no significa que tenga que “extraer y reemplazar” o eliminar alguna de sus herramientas actuales, sino más bien proporcionar visibilidad y administración e todo el proceso. Esto crea más tiempo para su personal y brinda agilidad a todo el ciclo de vida de la versión; mejorando la velocidad, la calidad y la escalabilidad.

Las herramientas de orquestación conectan equipos variados e aislados en un proceso armonioso totalmente automatizado. Ofrecen control y visibilidad, lo que permite administrar un ciclo de vida de *release* cada vez más complejo y proporciona una estrategia de entrega continua coherente para su organización. Además, admiten iniciativas de entrega continua con niveles adicionales de confiabilidad, audibilidad y escalabilidad, agregando velocidad y calidad a todos sus *releases*. Esto se logra eliminando los procesos manuales y las comprobaciones, eliminando así una gran cantidad de tiempo de gestión que puede consumir miembros importantes de un equipo y liberarlos para que sean más productivos e innovadores. En última instancia, las herramientas de orquestación brindan una mayor colaboración a una organización, permitiendo agilidad y facilitando DevOps, lo que se acerca al objetivo final de la entrega continua.

- **Herramienta para despliegue automatizado.**

Hay muchas formas diferentes de lograr la automatización de la implementación, desde soluciones basadas en scripts hasta herramientas de implementación

dedicadas, hasta métodos de implementación continua. Sin embargo, en última instancia, cualquiera que sea el medio que emplee, la automatización de la implementación debería otorgarle la capacidad de entregar, actualizar, monitorear y asegurar aplicaciones distribuidas e infraestructuras globales en todos los entornos, de forma automática. La automatización de la implementación no solo se puede implementar en servidores locales o en la nube, sino que también se puede publicar en diferentes entornos, incluido producción. De hecho, permite que los equipos de desarrollo y operaciones, así como las legendarias configuraciones de DevOps, sean lo más productivos posible.

Al eliminar la necesidad de un aporte humano significativo, la automatización de implementación elimina el riesgo del proceso de implementación. Está estrechamente relacionado con las plataformas de automatización de lanzamiento y permite lanzamientos frecuentes a cualquier entorno y a cualquier tipo de servidor. Las empresas pueden lanzar con mayor frecuencia y de manera más segura, lo que a su vez significa que pueden lanzar nuevas características, parches y actualizaciones a sus clientes. Como la automatización de la implementación permite a los desarrolladores lanzar a cualquier entorno desde el desarrollo, facilita la entrega continua, ya que los cambios de código se lanzan automáticamente.

Básicamente, la automatización de implementación elimina los riesgos de la etapa de implementación de la versión del software. Asegura que las implementaciones se vuelvan repetibles, eliminando las variables, asegurando efectivamente que cada versión permanezca igual. Esto se logra eliminando el potencial de errores, minimizando el nivel de entrada humana requerido. La automatización de la implementación también reduce los costos; a medida que surgen menos errores y se necesitan menos aportes humanos, se deben dedicar menos recursos al proceso. Ya no se requieren especialistas en implementaciones; de hecho, cualquier persona puede activar las implementaciones. Incluso puede aumentar

los ingresos, ya que permite a su equipo ser más innovador y lanzar con más frecuencia, lo que le permite introducir nuevas características y abordar las necesidades y demandas de los clientes a una mayor velocidad.

Algunos ejemplos de este tipo de herramientas son: Puppet, Chef, Juju, entre otros.

- **Gestores de Artefactos de *software*.**

Son herramientas de *software* diseñadas para optimizar la descarga y almacenamiento de archivos binarios usados y producidos en el desarrollo de *software*. Centraliza todos los artefactos binarios generados y usados por la organización para superar la complejidad que surge desde la diversidad de tipos de artefactos binarios, su posición en el flujo de trabajo general y la dependencia entre ellos.

Un repositorio binario es un repositorio de *software* para paquetes, artefactos y sus metadas correspondientes. Puede ser usado para almacenar archivos binarios producidos por la misma organización, tales como *releases* y productos compilados, o para binarios de terceros los cuales deben ser tratados de forma diferente tanto por razones técnicas como legales.

Su relación con la integración continua. Como parte del ciclo de vida del desarrollo, el código fuente continuamente se está construyendo en artefactos binarios usando integración continua. Esto puede interactuar con un gestor de repositorios binarios, al igual que lo haría un desarrollador al obtener artefactos de los repositorios y colocar las compilaciones allí. La estrecha integración con los servidores de CI permite el almacenamiento de *metadata* importante, tales como:

- Cual(es) usuario(s) activaron la compilación (ya sea, manualmente o por "*commit*" para el control de revisión.
- Cual modulo fue construido.
- Cuales códigos fuentes fueron usados (Id del *commit*, la revisión, *Branch*).

- Dependencias usadas.
- Variables de entorno.
- Paquetes instalados.

Artefactos y paquetes. Los artefactos y paquetes intrínsecamente significan cosas diferentes. Los artefactos son simplemente una salida, una colección o archivos (por ejemplo: jar, war, dll, etc.) y uno de estos archivos pueden contener *metadata*. Además, los paquetes son un único archivo en un formato bien definido que contiene los archivos apropiados para el tipo de paquetes. Muchos artefactos resultan desde compilaciones o *builds*, pero otros tipos también son cruciales. Los paquetes son esencialmente una o dos cosas: una librería o una aplicación. Comparado con los archivos fuentes, los artefactos binarios son con frecuencia más grandes en órdenes de magnitud, ellos son raramente borrados o sobre-escritos (excepto por casos extraños como *snapshots* o compilaciones nocturnas), y usualmente están acompañados por pérdida de metadatos tales como: id, nombre del paquete, versión, licencia, entre otros.

Metadatos. Los metadatos describen un artefacto binario, está almacenado y especificado separadamente desde sus propios artefactos, y puede tener varios usuarios adicionales.

Tipo de metadato	**Usado para**
Versiones disponibles	Actualización y degradación (*downgrading*) automáticos
Dependencias	Especifica otros artefactos que dependen del artefacto actual.
Dependencias *Downstream*	Especifica otros artefactos que dependen del artefacto actual.
Licencia	Conformidad legal
Fecha y hora de	Trazabilidad

compilación	
Documentación	Provee disponibilidad fuera de línea por documentación contextual en IDE's
Información de aprobación	Trazabilidad
Métricas	Cobertura de código, conformidad de reglas, resultado de pruebas.
Metadatos de usuarios creados	Reportes personalizados y procesos.

Tabla 1. Metadata de artefactos y paquetes

Aspectos claves del gestor de repositorios.

Las características y factores claves cuando se considera la adopción de un administrador de paquetes, incluye:

- **Almacenamiento en caché.** El almacenamiento en caché simplemente almacena copias locales de los paquetes. Esto incrementa el rendimiento para conexiones de internet lentas permitiendo que los usuarios extraigan del repositorio local en lugar de unos externos. El almacenamiento local en caché permite que los paquetes usados con frecuencia estén disponibles incluso en momentos de interrupciones externas del repositorio.
- **Políticas de retención.** Los gestores de repositorios pueden ser usados y configurados para respaldar las políticas de depuración de la organización para garantizar un uso razonable del espacio en disco. Los repositorios locales para los artefactos de terceros también pueden ser depurados después que no hayan sido usados por ninguna versión durante un tiempo específico.
- **Filtrado de licencias.** Los artefactos de terceros pueden estar sujetos a procesos de aprobación debido a licencias y problemas legales. Los

administradores de paquetes permiten la restricción de solo los artefactos aprobados en la implementación.

- **Alta disponibilidad.** Desde el gestor de repositorios binarios se mantienen todas las dependencias de desarrollo, siempre es vital mantener el acceso a estos artefactos. Cualquier caída del administrador de repositorios puede detener el desarrollo con todas las consecuencias significantes a la organización. Una instancia de alta disponibilidad permite a la empresa superar el riesgo asociado con el tiempo de inactividad a través de la conmutación automática por error. Esto se logra al tener un conjunto redundante de gestores de repositorios que trabajen contra la misma base de datos y el mismo almacenamiento de archivos.
- **Restricciones de usuario.** Los gestores de repositorios pueden estar integrados con otros sistemas organizacionales tales como LDAP o servidores de inicio de sesión simple para centralizar y simplificar la administración de usuarios. Esto da a la empresa un control granular sobre quien tiene acceso a componentes vitales de *software*.

Algunos ejemplos de estas herramientas son: Nexus, Jfrog artifactory, Archiva.

6.6. DEVOPS Y LA AUTOMATIZACIÓN DE PROCESOS.

6.6.1. Integración Continua.

Con la integración continua los desarrolladores integran frecuentemente su código a la rama principal de un repositorio común. En lugar de desarrollar partes del código de manera aislada e integrarlas al final del ciclo de producción, un desarrollador contribuirá con cambios a una parte del repositorio varias veces al día.

La idea aquí es reducir los costos de integración, haciendo que los desarrolladores realicen integraciones más rápidamente y con mayor frecuencia. En la práctica, un desarrollador a menudo descubrirá conflictos entre el código nuevo y el existente en el momento de la integración. Si esta se realiza temprano y con frecuencia, la expectativa es que la resolución de conflictos será más fácil y menos costosa.

Requisitos.

- El equipo debe crear pruebas automatizadas para detectar inconsistencias en los nuevos desarrollos.
- Necesitará un servidor de integración continuo que permita el monitoreo y la ejecución de las pruebas de forma automática para cada nueva confirmación que se logre.
- Los desarrolladores deben fusionar los cambios de forma rápida y continua, por lo menos una vez por día.

Beneficios.

- No existen tantos errores y se evita que estos se envíen a producción. Las pruebas automáticas captan las regresiones muy rápido.
- Desarrollar las versiones es más fácil. Los problemas que ocurren en la integración se solucionan mucho más rápido.
- Las pruebas ya no son un gran costo para el equipo, pues el CI permite realizar muchas pruebas en cuestión de segundos.
- El control de calidad es mucho mejor.

6.6.2. Entrega Continua.

La entrega continua es en realidad una extensión de la Integración Continua, en la cual el proceso de entrega de *software* se automatiza para permitir implementaciones fáciles y confiables en la producción, en cualquier momento.

Un proceso de Entrega Continua (CD) bien implementado requiere una base de código que se pueda desplegar en cualquier momento. Con la Entrega Continua*, los lanzamientos de nuevos cambios ocurren de manera frecuente y rutinaria.* Los equipos continúan con las tareas diarias de desarrollo con la confianza de que pueden mandar a producción un lanzamiento de calidad, en cualquier momento que deseen, sin una complicada implementación y sin tener que seguir pasos de manuales ejecutados por un especialista. Se considera que la entrega continua es atractiva principalmente porque automatiza todos los pasos que van desde la integración del código en el repositorio base, hasta la liberación de cambios totalmente probados y funcionalmente adecuados.

El proceso consiste en una compleja automatización de los procesos de compilación (si es necesaria), pruebas, actualización de servidores de producción y ajuste de código usado en nuevas máquinas. En todo momento, se mantiene la autonomía del negocio de decidir qué modificaciones se publican y cuándo deben hacerse.

Requisitos.

- Base sólida de Integración Continua. Su entorno de pruebas debe cubrir una gran cantidad de código base.
- Las implementaciones que se requieren deben ser automatizadas. Aunque se activa de forma manual, una vez inicia no necesita de presencia humana.

Beneficios.

- Mayor velocidad y menor complejidad en el proceso de desarrollo.
- Lanzamientos mucho más seguidos, proporcionando realimentación inmediata.

6.6.3. Despliegue Continuo.

Esta es una modalidad más avanzada de Entrega Continua en la medida en que los despliegues a producción no pasan por una validación humana sino que están totalmente automatizadas.

La única forma de detener nuevos desarrollos en producción es a través de una prueba que falla e identifica los errores. La implementación es una manera de acelerar la realimentación, haciendo más eficiente el trabajo del equipo.
Gracias a este tipo de despliegue, los desarrolladores pueden llegar a ver su trabajo funcionando poco tiempo después de haberlo realizado, pero requiere el mayor esfuerzo en buenas prácticas y procesos automatizados.

Requisitos.

- Mientras mejor sea la calidad de las pruebas, mejor será la calidad de cada uno de los lanzamientos.
- Se debe mantener actualizado el proceso de documentación, coordinando la comunicación con otros departamentos como “soporte” o “marketing”.

Beneficios

- Todo se activa de forma automática para cada cambio.
- Cada nueva versión tiene menos riesgos, pues a medida que ocurren cambios se identifican los problemas que surgen y se pueden solucionar más fácilmente.
- El flujo es continuo y de alta calidad. Este es el beneficio más importante para los clientes.

7. ESTADO DEL ARTE, CASOS DE EXITO Y MOTIVOS DE FRACASO

7.1. Estado del Arte.

De acuerdo a la publicación realizada por la empresa Pink Elephant, menciona que: “DevOps se integra con otros marcos de referencia y mejores prácticas de TI de forma que se aprovechan las fortalezas de cada modelo. Incluso con los modelos de gestión de servicios de TI que pudieran presentar aparentes contradicciones, DevOps puede integrarse para generar estructuras robustas que aprovechen lo mejor de cada uno.” (pink Elephant, 2018)

A continuación de describen la relación con los principales marcos de referencia y mejores prácticas:

Disciplina	**Marco de referencia / Mejores prácticas**	**Relación con DevOps**
Gestión de servicios de TI	Information Technology Infraestructure Library (ITIL)	ITIL provee a DevOps un modelo operativo de la entrega y soporte de los Servicios de TI. La integración con los procesos de Gestión de Cambios, Gestión de Liberación, Planeación de la Transición y Soporte, Validación del Servicio y Pruebas, Evaluación, Gestión de Activos y Configuraciones, favorece el logro de transiciones más efectivas y eficientes.
Desarrollo de *software* ágil	Agile *Software* Development	Scrum plantea métodos y dinámicas de colaboración para lograr que los

	Practice - Scrum	desarrollos se liberen en trabajos pequeños más frecuentemente y alineados a los requerimientos de los clientes. Tiene una fuerte integración en la entrega de los trabajos a Operaciones (pruebas, requerimientos, criterios de aceptación).
Mejora	Lean IT	Lean IT contribuye en identificar mejoras que permiten hacer más eficientes los pasos de desarrollo, pruebas, liberación y entregas, contribuyendo con sus herramientas a alcanzar consistencia en los resultados a corto y largo plazo.
Relacionamiento y Alineación con el Negocio	Business Relationship Management – BRM	BRM facilita el análisis de la identificación sobre quién es el cliente y qué requiere, además de definir la demanda estratégica y mediciones de valor. También aporta técnicas para fomentar una adecuada relación y alineación con los clientes del Negocio.
Arquitectura empresarial	TOGAF	TOGAF aporta a DevOps un orden y organización sobre los puntos de mejora en eficiencia y eficacia. Provee elementos en cuanto a procesos, información, datos, tecnología.
Seguridad de la información	ISO/IEC 27001	ISO/IEC 27001 aporta la identificación, gestión y mitigación de riesgos en la

		seguridad de la información, lo cual propicia un enfoque integral basado en riesgos desde etapas tempranas hasta el desarrollo.

Tabla 2. Relación de DevOps con otras disciplinas

7.2. Casos de Éxito.

Según el artículo "*El uso de agile y DevOps genera un aumento de hasta un 60% en las ganancias de las productoras de software*" (Colombia Digital, 2019). Aunque las empresas están comprometidas con la adopción de las metodologías Agile y DevOps, todavía no aprovechan los beneficios que estas prácticas pueden proporcionar. Esta es una de las principales conclusiones del estudio realizado por CA Technologies, que entrevistó a más de 1.200 ejecutivos de TI sobre el uso de Agile y DevOps en la transformación digital. Entre los datos globales, se destaca que el 75% de los entrevistados reconocen que los enfoques Agile y DevOps colaboran para el éxito significativo en los negocios cuando se implementan juntas.

Francisco Dal Fabbro, VP Latam de Agile de CA Technologies (2019), comenta "*El aumento de la demanda en el mercado de aplicaciones no justifica la entrega de productos de baja calidad. Para ello, prácticas como Agile son esenciales para que las Modernas Fábricas de Software atiendan a las exigencias de sus clientes*"; según él, la flexibilidad es esencial para adaptarse a las transformaciones del cliente, expectativas de los usuarios, cambios regulatorios y oportunidades de negocio.

El estudio también muestra que las organizaciones tienen desafíos similares: cultura, capacidades, inversión en programas y alineación del liderazgo. El estudio

destaca un reconocimiento general de que la adopción de prácticas Agile y DevOps a lo largo del ciclo de vida del software no es sólo una cuestión de nuevas capacidades y estándares de trabajo. Para algunos, esto también requiere un cambio importante en la mentalidad y comportamiento; es decir, estos cambios son una cuestión relacionada con las personas, incluso en el nivel de la dirección.

Las tendencias mundiales en relación con las principales prioridades para mejorar la eficacia identificada por los encuestados incluyen:

- Mejorar la cultura de la organización para incentivar y recompensar la colaboración (84%).
- Más apoyo y compromiso por parte de la administración en todos los niveles (82%).
- Entrenamiento para los equipos de TI sobre cómo colaborar e incorporar las mejores prácticas en sus actividades diarias en el trabajo (78%) y más soporte y compromiso por parte de la gestión (75%).
- Aliviar las presiones de tiempo para que los equipos puedan adoptar prácticas Agile y DevOps efectivas (74%).

Los entrevistados también dijeron que es muy difícil o desafiante encontrar a profesionales familiarizados con los métodos Agile (68%), que tengan experiencia con DevOps (77%) y/o tuvieron experiencia de trabajo colaborativo en su equipo (67%). Esto indica claramente una falta de capacidades en la mayoría de las organizaciones, lo que requiere recursos disponibles, principalmente entrenamientos.

Según Fabbro, cuando hay un aumento en la demanda para atender una nueva tendencia de mercado, las empresas tienen dificultad para encontrar profesionales capacitados. *“Los cursos de formación no se adaptan tan rápido en cuanto a las empresas. Por ello, estas también necesitan apostar en sus talentos, desarrollar a profesionales y aumentar su capacidad de atender al cliente final”*, explica.

La conexión entre los enfoques Agile y DevOps y los resultados de los negocios se concentra en el ciclo de *feedback* continuo de experiencias de clientes en tiempo real para la ingeniería de requisitos, mostrando el rendimiento de entrega de *software* y apoyando el propio negocio. De este modo, para aprovechar más aún los beneficios de los enfoques Agile y DevOps, las organizaciones también deben utilizar la rapidez y flexibilidad de los ambientes en la nube, en contenedores y otras nuevas arquitecturas de desarrollo y entrega de códigos, con un ligero cambio en todas las actividades – como pruebas continuas – y granularidad más fina de la interacción en todo el ciclo de operaciones y entrega del software.

7.3. Mejores prácticas para implementar DevOps.

En la publicación *"Top 7 best DevOps practices for active collaboration and development" (Courseing, 2018),* mencionan que la tecnología DevOps es útil para proporcionar la integración continua, la entrega continua y la implementación del *software*. El entorno de trabajo colaborativo de DevOps permite rastrear el estado actual de la implementación del *software* y la integración simultánea con el progreso del desarrollo. Hay muchas herramientas disponibles para rastrear todo el proceso de entrega, sin embargo, los ingenieros de DevOps deben seguir algunas métricas y prácticas básicas para mantener la entrega continua. Estás buenas prácticas consisten en:

1. Entender las herramientas de colaboración entre los equipos de desarrollo y los equipos de QA – *Quality Assurance* (Aseguramiento de la calidad).

   Los equipos DevOps y aseguramiento de la calidad, deben tomar una decisión sobre las herramientas adecuadas para desarrollo, pruebas y despliegue.

2. Implementar automatización de pruebas.

Las herramientas de automatización de pruebas tienen la capacidad de tomar código fuente y ejecutar procedimientos de prueba estándar para verificar y garantizar la excelencia del código.

3. Adquirir herramientas que permitan rastrear cualquier solicitud.

   El proceso DevOps debe contener herramientas para capturar cada petición realizada al *software*. Ninguna implementación o cambio necesario debe funcionar fuera del proceso de trabajo de DevOps.

4. Implementar pruebas de aceptación para cada herramienta de despliegue.

   Los procedimientos de pruebas de DevOps deben definir pruebas de aceptación[11] y asegurarse de que estas pruebas alcancen los objetivos de los criterios seleccionados.

5. Confirmar realimentación entre los equipos para hallar problemas.

   La realimentación continua a través de la colaboración entre los equipos hace posible la detección de problemas y debemos apoyarnos en las herramientas de pruebas para solucionarlos.

6. Soporte para producción.

   En la mayoría de los ambientes iniciales, los equipos de desarrollo trabajan en nuevas entregas de una solución que ya existe en producción.

7. Despliegue continuo.

   El despliegue continuo reduce los tiempos de los desarrolladores para identificar y desplegar nuevas características en el ambiente de producción.

Las actividades estimadas para lograr los objetivos propuestos son:

[11] Las pruebas de aceptación (User Acceptance Testing, UAT) pertenecen a las últimas etapas previas a la liberación en firme de versiones nuevas a fin de determinar si cumplen con las necesidades y/o requerimientos de las empresas y sus usuarios.

- Modelar procesos para la adopción de buenas prácticas en el desarrollo de *software*.
- Diseñar estrategias para el uso adecuado del sistema de repositorios de código con un sistema de control de versiones para administrar, controlar y centralizar el código fuente de los proyectos de *software*.
- Investigar, analizar y clasificar métricas de calidad que permitan garantizar productos certificados y mejorar el rendimiento de los equipos de desarrollo; y que permitan seguir las buenas prácticas de programación y estándares definidos por la empresa.
- Diseñar procesos estratégicos que ayuden a la implementación y ejecución de un sistema de automatización de pruebas.
- Documentar *pipelines* para orquestar cada uno de los demás servicios de integración continua.
- Diseñar un catálogo de métricas de calidad que se puedan implementar en un cualquier analizador de código estático con el fin de aplicarlo a proyectos categorizados con base a su arquitectura de referencia.
- Plantear estrategias para la creación y administración de entornos virtualizados a partir de contenedores, con el fin de optimizar los recursos tanto de TI como de fábrica de *software* y que faciliten el despliegue de las aplicaciones en entornos de desarrollo, pruebas y preproducción.
- Diseñar un modelo conceptual de integración y despliegue continuo que permita a una empresa a mejorar sus procesos de desarrollo de *software*.

En un sentido generalizado, DevOps busca crear un clima de cooperación y comunicación. Desarrollar *software* implica la participación del departamento de operaciones y de desarrollo, y aunque tengan misiones diferentes, el objetivo final es el mismo.

En una empresa, el departamento de operaciones cubre todas aquellas actividades que tienen relación con las áreas de la misma que generan el producto o servicio que se ofrece a los clientes. Podríamos decir que son la “forma de hacer las cosas dentro de la empresa”, tal que sus actividades permitan prestar el servicio o producir el producto que se da o entrega a los clientes para cumplir sus expectativas. Las actividades que contemplan la Dirección de Operaciones existen siempre en cualquier empresa, tanto si se trata de una fábrica, un hospital, un hotel o la conducción de un autobús. No importa si se trata de una empresa industrial o de servicios: todas poseen la función de Operaciones. Por tanto, las Operaciones abarcan todas las actividades que van desde una idea hasta un cliente satisfecho. La satisfacción del cliente, que se consigue al superar las expectativas que éste espera de la empresa (del producto o servicio) a corto, medio y largo plazo, se convierte en un objetivo operativo: que éste vuelva a comprar (Gómez, 2016).

7.4. Principales errores al implementar DevOps.

Muchas organizaciones que buscan implementar estas metodologías fracasan en el proceso por no tener personas dentro del equipo con los conocimientos necesarios para el diseño de la solución CI/CD adecuada. Implementar este proceso es difícil si no se cuenta con una guía que contenga la información adecuada. Leer sobre el tema puede ayudar mucho, sin embargo, es mucho mejor y más eficiente contar con la asesoría de un experto en el tema. Uno de los errores más frecuentes en la implementación de estas metodologías es la correcta automatización que requiere la fase de prueba. Una de las preguntas más comunes que surgen es: “¿cómo puedo integrar pruebas automatizadas en mi solución CI/CD?” Automatizar las pruebas de código es complejo pues requiere contar con herramientas adecuadas para el equipo y el tipo de proyecto,

automatizar tareas repetitivas y tener procesos para revisar, planear y re-evaluar cada paso de la metodología.

De acuerdo con el artículo ***"5 biggets ways to fail at DevOps"***, publicado por la comunidad DevOps.com el 19 de diciembre de 2017; mencionan cinco razones esenciales por la cual puede fallar la implementación de DevOps. "Cada día más organizaciones se están cambiando a DevOps para entregar versiones de *software* más frecuente, con el resultado de que la entrega ágil del *software* está ganando terreno más rápido de los que se esperaba.

- **Fallo No. 1. No definir lo que DevOps significa para su organización.**
  La definición de DevOps no siempre es clara, y puede tener diferentes significados de una organización a otra. Los desarrolladores pueden equiparlo con un enfoque específico para construir *software* usando herramientas populares Chef, Jenkins, Git o Docker. Mientras tanto, los administradores de TI pueden verlos como una continuación de los procesos existentes con un énfasis en comercialización más rápido y los procesos de liberación mucho más livianos.

- **Fallo No. 2. Enfocarse en herramientas y técnicas, mientras se descuidan los equipos.**
  El segundo error de muchas empresas es elevar la tecnología como el principal impulsor de DevOps a costa de procesos importantes que garantizan la calidad de *software* para que cumpla con las expectativas del cliente. No es suficiente contratar pocos ingenieros de implantación, proporcionarles máquinas virtuales (VM's) y otorgarles permisos para instalar Jenkins o Puppets. Para cualquier iniciativa, la tecnología no debería ser la única consideración a tener en cuenta, las personas y los procesos también necesitan ser tenidos en cuenta.

Siempre se pueden crear más entornos, pero no espere que los equipos se amplíen de la noche a la mañana. Muchas empresas adoptan DevOps para pasar a versiones más frecuentes y más rápidas impulsadas por las necesidades de proyectos individuales, sin darse cuenta que el aumento de la frecuencia de entrega de *software* conlleva a un agotamiento en el control de calidad y en el gestor de versiones.

- **Fallo No. 3. Ignorar las políticas completamente.**
  Gran parte de la retórica de DevOps está arraigada en la idea que los desarrolladores necesitan tomar un enfoque proactivo para evadir a los administradores que son reacios al cambio. A menudo, los DevOps dentro de la empresa emergen desde unos o dos grupos que deciden organizar una "revolución" en contra de organizaciones de TI ineficientes. Cuando una compañía tiene un gran sistema y liberaciones intratables que toman meses, a menudo los equipos pierden la paciencia con los procesos y están tentados a quebrar el molde y se cambian rápidamente.

  Una empresa con docenas de equipos independientes creando *pipelines* de despliegue continuo puede sonar muy bien en teoría, pero no funciona muy bien en la práctica. DevOps no es acerca de reducir el número de puertas de gobierno; por el contrario, permite que las puertas de gobierno sean más efectivas y con frecuencia iluminan los complejos retos de orquestación.

- **Fallo No. 4. No tener en cuenta los riesgos.**
  Muchas compañías primero se zambullen en DevOps sin un entendimiento completo de como afectaran los riesgos asociados con las versiones de *software*. Cuando una empresa pasa desde un proceso más lento basado en ITIL, a una realidad más ágil basada en DevOps, a menudo se espera

que los administradores de los despliegues “avancen” hacia el final del ciclo de despliegue. Muchas empresas olvidan de tomar en cuenta el riesgo de las decisiones de sobre producción. El *software* puede ser liberado más rápido, pero la empresa aun requiere el portal de gobierno.

Los cambios en los sistemas de producción requieren una gestión de cambios rigurosa, junto con la función de gestión de liberación para rastrear conflictos y riesgos. Una forma de simplificar el proceso es a través de la entrega continua, la cual permite a los equipos realizar múltiples liberaciones de forma simultánea mientras que integra con las herramientas existentes, que las operaciones esperan tener en su lugar, para rastrear y administrar los riesgos.

- **Fallo No. 5. Ejecutar DevOps sin métricas.**
  A menudo los equipos DevOps empiezan a trabajar con entusiasmo para realizar los cambios en una infraestructura de una organización y liberar procesos, pero también pueden morder más de lo que pueden masticar.
  A pesar de que los equipos DevOps quieran reinventar un proceso de liberación toda la noche, los administradores de TI aún pueden definir las métricas para evaluar si las iniciativas son un éxito. Administrar la transición lenta a técnicas y herramientas DevOps en varios trimestres puede ser difícil al priorizar el desarrollo continuo de *software* y la administración de versiones.

7.5. Motivos de Fracaso.

De acuerdo al artículo *5 retos al implementar DevOps* (PSL, 2018), publicado por la empresa PSL[12], menciona diferentes aspectos que deben ser tenidos en cuenta al momento de iniciar las implementación de DevOps y los cuales es muy probable deban enfrentarse los responsables de dicha implementación. A continuación de describen las barreras y las posibles soluciones, según el punto de vista de PSL:

- **Superando la resistencia al cambio.**

  La aceptación de las partes interesadas es increíblemente importante para mantener el impulso de su transformación. Muchas de estas partes son resistentes al cambio, especialmente frente a los sistemas y procesos actualmente en funcionamiento, incluso si no se consideran eficaces. Después de todo, si no está roto, ¿por qué arreglarlo?

  Es importante que se abra camino en la adopción de DevOps, seleccionando un pequeño producto o aplicación que se puede remodelar para trabajar con los procesos existentes. La idea es que tu equipo y su organización puedan acostumbrarse lentamente a los nuevos protocolos de desarrollo.

  Un cambio importante y exitoso a DevOps no puede ocurrir de la noche a la mañana, ya que necesita ser fluido, optimizado y respetado por todos. La mejor manera de obtener ese tipo de compromiso y apoyo es facilitar el proceso de desarrollo, en lugar de empujarlo sin ninguna advertencia.

- **Eliminar las herramientas y sistemas que son incompatibles.**

  Un problema común con muchas organizaciones es que han establecido barreras entre departamentos como Operaciones y Desarrollo. Como

[12] PSL. Es una empresa de desarrollo de software ágil, se especializa en ofrecer servicios de TI desde sus centros de desarrollo en Latinoamérica. Es pionero en la adopción de mejores prácticas de ingeniería de Software (desarrollo ágil, SCRUM) y es el principal exportador Colombiano de Servicios de software hacia EEUU.

resultado, los equipos separados pueden tener herramientas y procesos separados para completar el trabajo, lo que puede chocar cuando se intenta adoptar sistemas más eficientes. Varias plataformas de métricas heredadas pueden causar algunos problemas atroces al tratar de trabajar en colaboración en una organización. Esto es especialmente cierto si su equipo ha estado operativo durante bastante tiempo, con sus propios métodos, herramientas y sistemas de desarrollo comprobados.

Para empeorar las cosas, cuando haces cambios radicales, hay algunas partes que son más resistentes al cambio y se niegan a desprenderse de las herramientas heredadas.

La solución es implementar lentamente DevOps, como se describe en el paso anterior. Lo que es más importante, necesitaras sentarte con sus diferentes equipos, incluidos Operaciones y Desarrollo, para extraer las herramientas y las funciones que se necesitan. De esta forma, cuando finalmente eliges una solución integral, puedes estar seguro de que todos obtienen lo que quieren o necesitan, y ese desarrollo se desarrollará sin problemas.

- **Implementación de automatización.**

  Desafortunadamente, muchas de las herramientas y sistemas heredados que usan los equipos actualmente no son propicios para la automatización y la colaboración. Los procesos existentes también pueden ser cruciales porque las pruebas y el desarrollo continuos son necesarios para una implementación fluida. El personal también puede ser resistente a la adopción de procesos más nuevos y automatizados, y muchos están preocupados por su seguridad laboral y como esos procesos de automatización los van a afectar.

La solución a esto puede requerir bastante paciencia y tiempo para implementarla, y comprender que el proceso de adopción puede no ser fluido o rápido. Las pruebas continuas, la integración, la comunicación, la entrega y la colaboración entre departamentos son fundamentales para el éxito futuro, pero todos son conceptos en los que querrá enfocarse desde el principio. Esto incluye la planificación de la seguridad y los procesos más nuevos de antemano, para asegurarse de que no estés agregando nada a la fuerza más adelante. Intentar cambiar las cosas en una etapa posterior no solo es una molestia, sino que también puede ralentizar significativamente las operaciones.

- **¡Cuidado con el presupuesto!**

  Cualquier administrador ingenioso le dirá que se necesita dinero para ganar dinero, y esto es especialmente cierto en un proyecto transformador como es la adopción de DevOps. Esto puede atribuirse a los cambios generalizados que tal transformación puede requerir, como la implementación de nuevas herramientas, habilidades y capacitación, estructuras y procesos, y mucho más.

  Un cambio radical también puede provocar que el rendimiento de los empleados disminuya, al menos inicialmente, lo que significará que la productividad en toda su organización va a sufrir momentáneamente. Esto puede resultar en un golpe financiero aún mayor. Debes asegurarte de que tu equipo y tu organización estén preparados para esto. En las etapas iniciales, querrá estar atento a su presupuesto y abstenerse de realizar algún otro movimiento costoso. Los beneficios de costo de un enfoque de DevOps definitivamente lo valen, pero hay que tener en cuenta que los resultados no se van a ver en el corto plazo.

- **El proceso toma tiempo.**

Como ya dijimos, hacer el cambio a DevOps no ocurrirá de la noche a la mañana, y no todos van a estar muy contentos. De hecho, es probable que tengas que lidiar con la resistencia durante bastante tiempo incluso después de la adopción.

La solución es continuar educando e informar a todos los involucrados. Háblalo y comunícate con tus equipos todo el tiempo. Explica por qué se está realizando el cambio y lo que todos pueden esperar lograr una vez que esté completo. Los casos de uso y ejemplos de otros en el mercado muestran cómo funcionaban las cosas en el pasado y cómo esto se puede aplicar a tu empresa en específico. Tendrás que venderlo, como un producto. Si quieres de verdad implementar DevOps debes prepararte para invertir el tiempo y los recursos necesarios para adoptar DevOps y actualizar tus equipos.

8. PROPUESTA

En el mercado existen muchas soluciones que ofrecen servicios para la adopción e implementación de DevOps, sin embargo, no todas las empresas están preparadas para dicha iniciar dicho proceso.

El objetivo principal de toda empresa dedicada al desarrollo de software, es crear productos con tiempos de entrega prudentes y entregar productos de calidad a sus clientes. Sin saberlo, una empresa puede incurrir en gastos exagerados por el afán de incorporar DevOps.

Con el fin de brindar una solución a esta problemática, la propuesta de este trabajo de grado se basa en presentarle a las empresas una herramienta que les permita auto-evaluarse antes de tomar la decisión de implementar DevOps, de

igual forma, se realiza una propuesta metodológica para prepararse para dicha adopción. Según la metodología empleada, la propuesta se divide en diferentes etapas, las cuales se describen a continuación:

8.1. Caracterización de una empresa.

La factibilidad técnica, operacional y económica, son los factores más importantes que deben ser tenidos en cuenta a la hora de tomar la decisión de implementar DevOps.

Para obtener una caracterización con un alto de grado de exactitud, se ha optado por emplear una valoración cuantitativa basada en aspectos cualitativos, cuyos datos se obtendrán por medio de un *test* de valoración donde se evaluarán los siguientes grupos:

- Factibilidad y disponibilidad Económica.
- Disponibilidad de infraestructura.
- Recursos humanos.
- Ingeniería de software: Cobertura, alcance y madurez de los proyectos.
- Certificaciones.

Una vez obtenida la información de la empresa, se asignarán pesos a cada una de las respuestas con el fin de obtener un puntaje ponderado por cada una de las dimensiones evaluadas y así determinar el estado actual de la empresa y emitir un concepto con las recomendaciones suficientes para iniciar o no la implementación de DevOps.

8.2. Flujo de trabajo, tareas, roles y responsables.

- *Flujo de trabajo.*

  En el anexo 1, se propone el flujo de trabajo para el ciclo de un proyecto de software el cual inicia en la fase de análisis y levantamiento de requisitos hasta la fase de despliegue en los ambientes de producción. Dado que es

un modelo propositivo, se puede ajustar y adaptar a las necesidades de la empresa; sin embargo, cabe anotar, que existen tareas y roles que no deben ser obviados debido a que forman parte fundamental del proceso y de las buenas prácticas propuestas en el presente documento.

En las secciones siguientes se describirán las tareas y los roles involucrados en el flujo de trabajo propuesto.

- *Tareas.*
    - Levantamiento de requisitos.
    - Creación de HU y/o casos de uso.
    - Creación de requerimientos.
    - Revisión y aprobación de requisitos.
    - Revisión y aprobación de prototipos.
    - Definir y documentar la arquitectura del proyecto.
    - Crear estructura base y creación de repositorios de código.
    - Elaboración de prototipos.
    - Publicación de los prototipos.
    - Conformación equipos de desarrollo.
    - Asignación de tareas y estimación de tiempos.
    - Despliegue del proyecto.
    - Ejecutar y documentar pruebas funcionales.
    - Ejecutar y documentar pruebas de aceptación.

- *Roles.*

  A continuación se proponen los siguientes roles que serán los responsables de la ejecución de ciertas tareas definidas en el flujo de trabajo. Dependiendo de las necesidades de la empresa, uno o más de estos roles podrían estar unidos; sin embargo, se proponen mantenerlos individualizados para que las personas que ejecutan dichos roles no se vean afectados por alguna posible sobrecarga laboral y así se vayan a

producir futuros retrasos en las peticiones y/o en el flujo de trabajo como tal. Los roles analizados y propuestos para el flujo de trabajo, son los siguientes:

- **Analista de requisitos.** Rol encargado del análisis y levantamiento de requisitos ante las necesidades del cliente.
- **Product Owner / comité de aprobación.** Siguiendo las bases de la metodología *scrum*, el *product owner* es un actor clave en el desarrollo de un proyecto. Una de sus responsabilidades es tener una visión de lo que desea construir, y transmitir esa visión a todo el equipo. En caso de no tener prácticas estables de agilísimo, este rol puede ser reemplazado por un comité de aprobación conformado por una o varias personas que tengan una amplia visión del proyecto, del negocio y como tal del producto que se va a construir.
- **Arquitectura de software.** Rol encargado de definir la arquitectura del proyecto, seleccionar el lenguaje de programación adecuado, aplicar los patrones de arquitectura y diseño requeridos para seguir con las buenas prácticas de software; así mismo, se encarga de seleccionar los *frameworks* adecuados, y definir la estructura y los estándares de programación para cumplir con las buenas prácticas de codificación.
- **Diseño.** Se encargan de crear los prototipos empleados para las vistas o *frontend* del proyecto. Están sujetos a las definiciones establecidas en el DDS.
- **Desarrollo.** Equipo encargado de crear los componentes de software requeridos para el proyecto; su responsabilidad principal es la de mantener las buenas prácticas de programación de acuerdo a lo establecido por el área de arquitectura.
- **Tester / QA.** Este rol se encarga de realizar las pruebas funcionales del producto. Se encargan de aplicar y mantener los estándares de

validación y verificación del producto. Con el fin de la automatización de los procesos DevOps, se recomienda realizar pruebas automatizadas.

- **Infraestructura TI.** Encargados de instalar, configurar y mantener los servidores físicos, ya sea en el centro de datos propios o en su defecto de la administración de los servicios virtualizados o servicios *cloud* (cuando la situación lo amerite). Además, deben velar por la creación y administración de los ambientes requeridos para el despliegue del producto (desarrollo, pruebas, preproducción, producción, etc).
- **Cliente.** Es el actor principal, es el dueño del producto; por lo que se requiere una comunicación constante para analizar, evaluar y aprobar las propuestas o prototipos del producto.

8.3. Metodología.

La metodología propuesta para la implementación de DevOps se basa en las metodologías tradicionales, específicamente la metodología en espiral la cual se basa en un proceso evolutivo que conjuga la naturaleza iterativa de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial. Además, proporciona el potencial para el desarrollo rápido de versiones incrementales del *software*.

Mediante el modelo propuesto, los procesos están sometidos a una serie de versiones incrementales. Durante las primeras iteraciones, la versión incremental podría ser un modelo en papel o un prototipo. Durante las últimas iteraciones, se producen versiones cada vez más completas del sistema diseñado.

Dentro de las ventajas que se pueden resaltar de la adaptación del modelo en espiral, se pueden señalar:

- Es una metodología adaptable para la implementación de procesos y proyectos ya sean pequeños o de gran cobertura.
- La metodología es flexible y permite una alta cohesión entre cada una de las etapas.
- La evolución constante es una de las principales virtudes de la metodología, por lo tanto, ofrece grandes ventajas para la estabilización de los procesos para empresas que apenas están iniciando y una madurez de procesos para las empresas con más experiencia.
- Se tiene un mayor control de los riesgos generales de la implementación de DevOps, así mismo, se puede hacer una mayor trazabilidad de los riesgos identificados en cada etapa, permitiendo así tomar medidas oportunas y mitigar los riesgos en los ciclos posteriores.
- Utiliza la construcción de prototipos como mecanismo de reducción de riesgos.
- Permite a quien lo desarrolla aplicar el enfoque de construcción de prototipos en cualquier etapa de evolución del producto.
- Mantiene el enfoque sistemático de los pasos sugeridos por el ciclo de vida clásico, mediante el marco de trabajo iterativo se pueden realizar proyecciones en el tiempo con el fin de mejorar los procesos y así alcanzar una madurez en los procesos.

Las fases que componen la metodología son:

- **Conceptualización:** En esta etapa se realizará la sensibilización y conceptualización de todos los aspectos, las metodologías y estrategias de trabajo que se llevarán a cabo a lo largo del ciclo actual de la metodología. La empresa es libre de adoptar cualquier tipo de estrategia de sensibilización, metodología para las capacitaciones y diseñar *test* o pruebas para evaluar el impacto de la conceptualización hacia sus empleados.

- **Selección de proyecto.** Debido a que inicialmente la implementación de DevOps se encuentra en etapas tempranas, se recomienda la selección de proyectos pequeños, con procesos estables y con una arquitectura bien definida con el fin de ir adquiriendo experiencia en cada uno de los procesos, flujos de trabajo, responsables, entre otros. De igual forma, con proyectos pequeños se pueden ir estableciendo métodos y tiempos de desarrollo, pruebas y despliegues.
- **Selección de personal.** Para la primera etapa del ciclo, se recomienda involucrar, tanto como sea posible, el personal con más conocimiento, experiencia y/o que hayan demostrado mayor capacidad de adaptación a DevOps durante la etapa de conceptualización y adaptabilidad.
- **Entrenamiento.** Tan pronto se tenga seleccionado el personal que hará parte de DevOps se procederá con la capacitación pertinente.
- **Infraestructura.** En esta etapa, se seleccionarán cada una de las herramientas de automatización que proveerán los servicios requeridos para el entorno DevOps. De igual forma, esta etapa involucra la selección de los servidores, la definición de los ambientes de trabajo, configuraciones generales y de seguridad, instalación, configuración y pruebas de las herramientas de automatización seleccionadas.
- **Ejecución.** Es la puesta en marcha del sistema DevOps, tanto a nivel de procesos como en automatización se refiere. Si el proyecto es nuevo, se iniciará desde la definición de los requisitos del proyecto, siguiendo así a la fase de desarrollo, pruebas hasta llegar al ambiente de producción. Si el proyecto ya está iniciado, se incorporará las fases previas y se iniciará desde la etapa de desarrollo debido a que se deben realizar ajustes y revisiones de código.
- **Documentación.** Aunque se define como una etapa independiente, realmente funciona como una actividad transversal donde cada etapa documentará los procesos que pertenecen a ella. En esta etapa se

recopilan, clasifican, analizan, procesan, ordenan y evalúan toda la documentación generada por las etapas del ciclo. Con la información centralizada, será mucho más fácil identificar aquellos aspectos críticos que representan riesgos potenciales para el proyecto y para el sistema DevOps.

Para la documentación, es importante definir formatos sencillos, intuitivos y estandarizados donde se pueda recopilar la información de manera precisa y coherente; se deben evitar formatos extensos que exijan, por parte del responsable, gran cantidad de tiempo para diligenciarlos.

Se recomienda el uso de repositorios para la administración, almacenamiento y control de versiones de la documentación, donde todo el personal de la empresa pueda acceder a ella en cualquier momento.

- **Evaluación.** Se realizará una evaluación del ciclo con el fin de tomar acciones preventivas y/o correctivas con el fin de ir madurando y evolucionando el proyecto y los procesos comprometidos con DevOps.
- **Transferencia.** En esta etapa, se tomarán las experiencias y lecciones aprendidas de cada una de las etapas del ciclo, se aplicaran acciones correctivas y preventivas según sea el caso y, para la siguiente fase, se realizará rotación e incorporación del personal con el fin de realizar transferencia de conocimiento; esta rotación e incorporación, se realizará especialmente con el personal de desarrollo y operaciones si la situación así lo amerita.

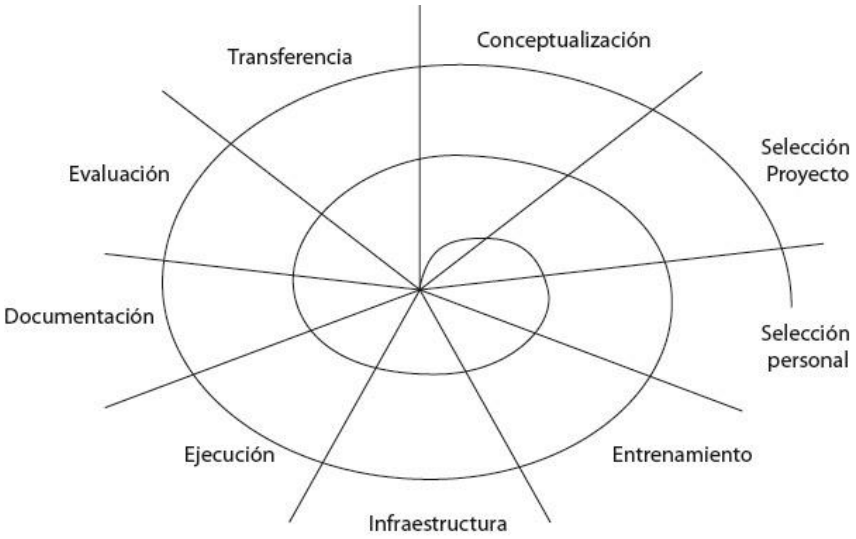


Imagen 7. Ciclo de vida implementación DevOps

9. RESULTADOS Y ANÁLISIS

Conforme al propósito de este trabajo, se esperan obtener los siguientes resultados.

- Diagrama de procesos BPM para la adopción e implementación de DevOps y buenas prácticas en el desarrollo de *software* en una organización.
- Metodología para la categorización de las empresas para determinar cuan preparadas se encuentran para la adopción de DevOps con sus respectivas recomendaciones para lograr, en un futuro, una implementación exitosa.
- Prototipos de trabajo que ayudaran a las empresas a implementar procesos para el área de fábrica de *software*.

9.1. Estudio de caso.

Con el fin de proporcionar un ejemplo de ejecución de la propuesta, y con fines prácticos, se hará uso de una empresa ficticia denominada "*Edalac Software*; la

descripción de esta empresa proporcionará los insumos suficientes requeridos para que sean sometidos a la evaluación de criterios, tal y como ocurriría con una empresa real.

Edalac Software, es una empresa dedicada al desarrollo de software que lleva más de 15 años de trayectoria. A lo largo de su existencia la empresa ha venido creciendo de forma significativa en cuanto a la generación de nuevos proyectos y al incremento de sus recursos económicos. De igual forma, la empresa ha ido mejorando su infraestructura tecnológica y ha incorporado profesionales en sus diferentes áreas, con un crecimiento en su recurso humano 10 veces mayor al que poseía en sus primeros años (en estos momentos, un total de más de 100 personas).

Actualmente, la empresa cuenta con cinco productos líderes, estables, que están diseñados bajo estándares y normativas legales, lo cual los convierten en productos genéricos para empresas con la misma naturaleza. Los productos se encuentran desplegados en ambientes de producción, y ya se han logrado convertir en el *core* de negocio de varios clientes. Conforme a las cambiantes normativas, las necesidades de los clientes y la evolución propia de las nuevas tecnologías, los productos han evolucionado en el tiempo, en el cual han alcanzado una etapa de madurez.

A nivel de infraestructura (TI), la empresa ha evolucionado considerablemente. En un principio el código fuente era almacenado en repositorios SVN, lo que dificultaba su versionamiento pero con el tiempo, lograron implementar un sistema de repositorios GIT y definieron un sistema de control de versiones estandarizado para cada uno de los proyectos migrados y el cual se fue replicando hacia los nuevos productos. De forma paralela, también fueron implementando un gestor de repositorios de software con el fin de administrar cada uno de los componentes creados y así reutilizar los artefactos cuando fuese necesario.

De igual forma, la empresa ha creado otros productos que, aunque más jóvenes que los anteriores (alrededor de tres años en el mercado), también se encuentran en etapa de producción, y aunque fueron diseñados bajo unos lineamientos genéricos, son unos productos que han sido ajustados a las necesidades de los clientes, por lo que se han convertido en *software* “a la medida”.

Por otra parte, en los últimos meses, la empresa ha participado en la licitación de nuevos contratos para brindar soluciones a las necesidades de algunas empresas con la misma naturaleza comercial a la que pertenecen sus productos líderes, pero con funcionalidades adicionales que deben ser desarrolladas e integradas a la solución actual.

La empresa se encuentra divida en diferentes áreas: RRHH, ingeniería, I+D, Operaciones y la parte administrativa; cada una con roles y perfiles definidos para cada uno de los cargos. El área de ingeniería se encuentra compuesta por Arquitectos de software, desarrolladores, analistas funcionales/requisitos, analistas QA, implementadores; en algunos casos, existen personas que desempeñan doble rol sin que estos causen conflictos. Por ejemplo, un analista QA también puede desempeñar funciones de un analista de requisitos.

Adicionalmente, el grupo de arquitectura se mantiene en una evolución constante con el fin de mantener los productos actualizados, procurando mantener las buenas prácticas arquitectónicas y de desarrollo de software.

Aunque son productos estables y cuyos incidentes han sido relativamente bajos, las pruebas de calidad (QA) son ejecutadas de forma manual, lo que dificulta un poco las entregas tempranas de las actualizaciones y/o las nuevas funcionalidades de un producto. Como complemento, los despliegues de actualizaciones a producción son algo lentos (entre 4 a 12 horas); la instalación y configuración de los ambientes de producción aún son un dolor de cabeza.

Debido a las exigencias de los clientes y con el fin de ser más competitivos en el mercado, en los últimos años, la empresa ha obtenido certificaciones como ISO 9001, ISO 27001, CMMI-Dev.

Dentro de los objetivos de la empresa, se encuentra la implementación de DevOps para integrar cada una de las etapas de proceso de software; es por esto, que los datos extraidos del estudio de caso, han sido sometidos a la evaluación de los criterios propuestos en este trabajo de grado, para determinar si la empresa es apta o no para adoptar la metodología DevOps y poder emitir un reporte con las mejoras, ajustes o acciones a tomar para la implementación de DevOps según sea el caso.

9.2. Asignación de pesos por preguntas y categorías.

En el anexo 3, se relacionan los pesos que han sido asignados a cada una de las preguntas por categorías. El valor máximo esperado por categoría se calculó mediante la sumatoria del peso máximo que puede ser asignado a cada pregunta. Por ejemplo, la pregunta número 18 *"¿Qué tipo de metodología aplican a sus proyectos de software?"*, tiene un peso total de 20 pero es de opción múltiple con única respuesta, y los pesos asignados a cada respuesta ha sido 0, 3, 5, 5, 7, por lo tanto, el puntaje máximo que se puede obtener de esa pregunta es 7.

Los valores ponderados se calcularon teniendo en cuenta los valores obtenidos (sumatoria de los valores por preguntas) divido por el valor máximo esperado de su categoría. Los criterios de evaluación se relacionan en la siguiente tabla:

CATEGORIZACION		
Estado	**Minimo**	**Maximo**
Rechazo	0%	25%
Parcialmente Rechazado	26%	50%
Parcialmente Aceptado	51%	90%
Aceptado	91%	100%

Tabla 3. Escala de valoración

9.3. Informe de análisis del estudio de caso.

Como puede observarse en la tabla 4, la cual es un resumen de los puntajes registrados en el anexo 4, se han obtenido los porcentajes ponderados por cada una de las dimensiones evaluadas:

Dimensión	Puntaje Obtenido	Puntaje Máximo	Ponderado	Descripción
Infraestructura	40	80	50%	Parcialmente rechazado
RRHH	43	70	61%	Parcialmente Aceptado
Ingeniería de software	151	203	74%	Parcialmente Aceptado
Certificaciones	60	100	60%	Parcialmente Aceptado
Financiera	42	70	60%	Parcialmente Aceptado
PUNTAJE GENERAL			**61%**	**PARCIALMENTE ACEPTADO**

Tabla 4. Resumen puntajes evaluación

Ver resultado completo de respuestas

Como puede observarse, la empresa del estudio de caso obtuvo un puntaje general del 61%, lo que corresponde a una aceptación parcial (Tiene muchos avances en los componentes requeridos para la implementación de DevOps); sin embargo, el puntaje se encuentra muy cerca del límite inferior de esta categorización, por lo tanto se deben

identificar, analizar y brindar opciones de mejoras de esas dimensiones que obtuvieron un bajo puntaje.

- **La dimensión de Infraestructura**, obtuvo un puntaje del 50%, para lo cual se realizan las siguientes recomendaciones:
    - Aunque en ocasiones no resulte tan relevante poseer servicios en la nube, se debe aclarar son muchas las ventajas obtenidas al implementarlas, dentro de estas: seguridad, velocidad, accesibilidad, disponibilidad, escalabilidad, mitigación de riesgos, entre otras.
    - Las buenas prácticas en el desarrollo de software es una de las principales características para hacer software de calidad, pero se debe tener en cuenta que el código fuente va a estar expuesto a diferentes personas. En un código fuente podemos encontrar diferentes estilos al momento de hacer las cosas; muchos desarrolladores siguen sus propios conceptos y hábitos de programación, otros simplemente no tienen la experiencia suficiente para escribir código optimizado. Es por esto, que los analizadores de código estático son muy importantes, nos brindan herramientas útiles que ayudan a identificar fluctuaciones en el código cuando no se han seguido las buenas prácticas. Tener una herramienta y no utilizarla, es como no tenerla.

      De igual forma, es muy importan definir unas métricas claras en el analizador de código, y deben estar alineadas con los objetivos de calidad que la empresa tenga definidos.
- **Recursos humanos - RRHH**. Se realizan las siguientes recomendaciones:
    - El objetivo principal de DevOps es realizar entregas de software en menor tiempo pero con una calidad garantizada; las metodología ágiles comparten el mismo objetivo por lo que son herramientas compatibles. Tener profesionales especializados en metodologías ágiles y/o scrum y que puedan transmitir el conocimiento a los demás miembros de la empresa, es una gran ventaja al momento de emplear DevOps.
    - En un sentido general, se debe reconocer la importancia del recurso humano en la empresa. En la mayoría de los casos, el éxito o el fracaso de

un proyecto depende de la aceptación y la adaptación del personal de la empresa. Se deben realizar campañas amigables en donde cada persona se sienta comprometida, empoderada y con sentido de pertenencia.

- **Ingeniería de software.** La empresa obtuvo un buen puntaje, dado que no se cuenta con un analizador de código estático es recomendable incluir la fase de revisión par en su proceso de desarrollo de software, especialmente cuando se tiene desarrolladores junior. Los aspectos relacionados con el factor tiempo, mejorarán con el tiempo cuando la implementación de DevOps ya se encuentre en ejecución y en un estado estable.
- **Certificaciones.**

  El que la empresa tenga certificaciones, no es un aspecto relevante dentro del estándar Devops; sin embargo, el hecho de poseer certificaciones quiere decir que la empresa ya ha pasado por procesos de cambios y adopción de ciertos paradigmas y metodologías que han definido estándares.
- **Dimensión Financiera.** Para muchas empresas este puede ser el factor de mayor relevancia, y no se equivocan, la dimensión financiera es muy cambiante y, generalmente, dependiente de las otras dimensiones. Cuando se realiza esta primera valoración, por cada puntaje bajo en alguna de las primeras dimensiones, se requerirá de un mayor compromiso económico por parte de la empresa. Esto es, si la empresa cuenta con un buen centro de datos y con recursos disponibles, el compromiso económico para la adquisición de equipos, recursos y/o servicios va a ser relativamente baja. Por otra parte, si la empresa no cuenta con el suficiente personal o, tal vez, no son idóneos, entonces, se debe incurrir en gastos económicos para la incorporación de nuevos recursos humanos, capacitaciones, etc.

10. CONCLUSIONES Y TRABAJO FUTURO

10.1. CONCLUSIONES

Una vez recopilada, analizada, clasificada y procesada la información de este trabajo de grado, se puede concluir que es común encontrar muchas empresas que quieran adoptar la metodología DevOps. Sin embargo, se debe tener en cuenta que adoptar una tecnología sin estar preparados, es lo mismo que fracasar sin haber iniciado. Aunque son muchas las ventajas que se obtienen a través de DevOps, también es importante conocer todas las implicaciones y los riesgos que conllevan su implementación. Es por esto que se ha propuesto una metodología que categoriza las empresas mediante la evaluación de los pilares fundamentales para una fábrica de software. Conforme a los resultados obtenidos se podrán emitir unos juicios de hecho que permitirán a la empresa prepararse adecuadamente para la adopción de DevOps. Por otra parte, para aquellas empresas que superen el puntaje mínimo de categorización, la propuesta contiene una metodología de implementación de DevOps, la cual fue concebida mediante las virtudes de otras metodologías y experiencias recopiladas de otras empresas que han sido casos de éxitos. Además, se proponen flujos de trabajo tanto para el ciclo de vida del proyecto como para el proceso de desarrollo de software.

En este trabajo de grado, me he reservado el derecho de mencionar, seleccionar o sugerir algún tipo de herramienta de software requerida para el proceso y/o automatización de DevOps. Esta tarea será responsabilidad de los trabajos futuros que puedan derivarse de este y dicha selección estará sujeta a la disponibilidad económica y de infraestructura de la empresa en mención.

10.2. TRABAJO FUTURO

Conforme al alcance de este trabajo de investigación y los temas tratados, se definen las siguientes acciones a seguir:

- Definición del tiempo para el retorno a la inversión.
- Implementación técnica del sistema DevOps.
- Evaluación de la puesta en marcha del sistema DevOps.

11. REFERENCIAS

Agile Alliance. 12 Principles behind the agile manifestó. (2018). Tomado de https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/; consultado el 25 de febrero de 2019.

Amazon. ¿Qué es la integración continua?. (2017). Tomado de https://aws.amazon.com/es/devops/continuous-integration/. Consultado el 20 de octubre de 2018.

Apiumhub. Estudio sobre la situación actual del software. (2019). Tomado de https://apiumhub.com/es/tech-blog-barcelona/situacion-actual-del-software/. Consultado el 15 de Octubre de 2019.

Aplyca. Integración continua y entrega continua. Publicado en el año 2018. Tomado de https://www.aplyca.com/es/blog/integracion-entrega-continua-ci-cd, consultado el 18 de febrero de 2019.

Attlasian. DevOps. Como romper la barrera entre desarrollo y operaciones. Publicado en https://es.atlassian.com/devops. Consultado el 12 de Octubre de 2018.

Attlasian. DevOps: Cómo romper la barrera entre Desarrollo y Operaciones. (2018). Tomado de https://es.atlassian.com/devops, consultado el 14 de febrero de 2019.

AWS Amazon. ¿En qué consisten las operaciones de desarrollo? Tomado de https://aws.amazon.com/es/devops/what-is-devops/, consultado el 12 de febrero de 2019.

B. Cameron Gain (2018). Using CALMS to Assess an Organization's DevOps. Recuperado de https://devops.com/using-calms-to-assess-organizations-devops/

CA Technologies. Build better apps faster. Gain insights to deliver superior experiences. (2019). Tomado de https://www.ca.com/us/why-ca/devops.html?intcmp=headernav. Consultado el 30 de Julio de 2019.

CA Technologies. Embeddable Lead Generation Tools (2019). Tomado de https://assessment-tools.ca.com/tools/continuous-delivery-tools/en?embed. Consultado

el 30 de abril de 2019.

CircleCI. A brief history of devOps. Tomado de https://www2.circleci.com/rs/485-ZMH-626/images/HistoryDevOps-eBook-v11.pdf. Consultado el 12 de febrero de 2019.

ClaraNet. DevOps: Que es y como lo aplicamos. Publicado en año 2018 en https://www.claranet.es/devops-que-es-y-como-lo-aplicamos-como-proveedor-de-cloud-hosting. Consultado el 15 de febrero de 2019.

Corporación Colombia Digital. El uso de Agile y DevOps genera un aumento de hasta un 60% en las ganancias de las productoras de software. (2019). Publicado en https://colombiadigital.net/actualidad/bytes/item/9971-el-uso-de-agile-y-devops-genera-un-aumento-de-hasta-un-60-en-las-ganancias-de-las-productoras-de-software.html.

Courseing. Top 7 best devops practices for active collaboration and development. (2017). Tomado de https://www.courseing.com/devops/blogs/devops-best-practices. Consultado el 25 de Noviembre de 2018.

DevOps.com. 5 biggest ways to fail at DevOps. Publicado el 17 de diciembre de 2017 en https://devops.com/5-biggest-ways-fail-devops/. Consultado el 18 de febrero de 2019.

DevOps.com. DevOps Challenges and versión control: The 2018 report. Publicado en Noviembre de 2018 en https://library.devops.com/devops-challenges-and-version-control-the-2018-report, consultado el 12 de febrero de 2019.

DEVOPSTI. Integración continua (CI), entrega continua (CD) y despliegue continuo (CD). Tomado de https://devopsti.wordpress.com/2014/09/26/integracion-continua-ci-entrega-continua-cd-y-despliegue-continuo-cd/. Consultado el 20 de octubre de 2018.

Gitlab. Empezando acerca del control de versiones. Tomado de https://git-scm.com/book/es/v1/Empezando-Acerca-del-control-de-versiones. Consultado el 10 de enero de 2019.

GitLab. Empezando acerca del control de versiones. Tomado de https://git-scm.com/book/es/v1/Empezando-Acerca-del-control-de-versiones. Consultado el 12 de diciembre de 2018.

Gómez, Emilio. Operaciones en empresas de servicio (2016). Publicado en https://www.eoi.es/blogs/emiliogomez/2016/02/18/operaciones-en-empresas-de-servicio/. Consultado el 24 de Agosto de 2019.

IBM. DevOps: El enfoque de IBM. Publicado en mayo de 2013. ftp://public.dhe.ibm.com/la/documents/imc/la/commons/RAW14323_HR.pdf. Consultado el 12 de Octubre de 2018.

Pink Elephant. Habilitando el alto desempeño en las organizaciones de TI. (2019). Consultado en https://pinkelephant-latam.com/informacion/devops_3.pdf, el 15 de marzo de 2019.

PSL. 5 retos al implementar DevOps. (2018). Publicado por PSL el 27 de abril de 2018. Tomado de https://www.psl.com.co/blog/5-retos-al-implementar-devops-en-tu-organizacion.html

Quint Group. Adoptar la transformación digital con DevOps. https://www.quintgroup.com/es-es/insights/white-paper-transformacion-digital-devops/

Reporte Digital. ¿Cómo el DevOps puede ayudar a unificar los equipos de trabajo?. (2019). Tomado de https://reportedigital.com/iot/devops/. Consultado el 10 de Mayo de 2019.

Stoica, Marian & Mircea, Marinela & Ghilic-Micu, Bogdan. (2013). Software Development: Agile vs. Traditional. Informatica Economica. 17. 64-76. 10.12948/issn14531305/17.4.2013.06.

TechBeacon. 6 common test automation mistakes and how to avoid them. Tomado de https://techbeacon.com/app-dev-testing/6-common-test-automation-mistakes-how-avoid-them. Consultado el 27 de febrero de 2019.

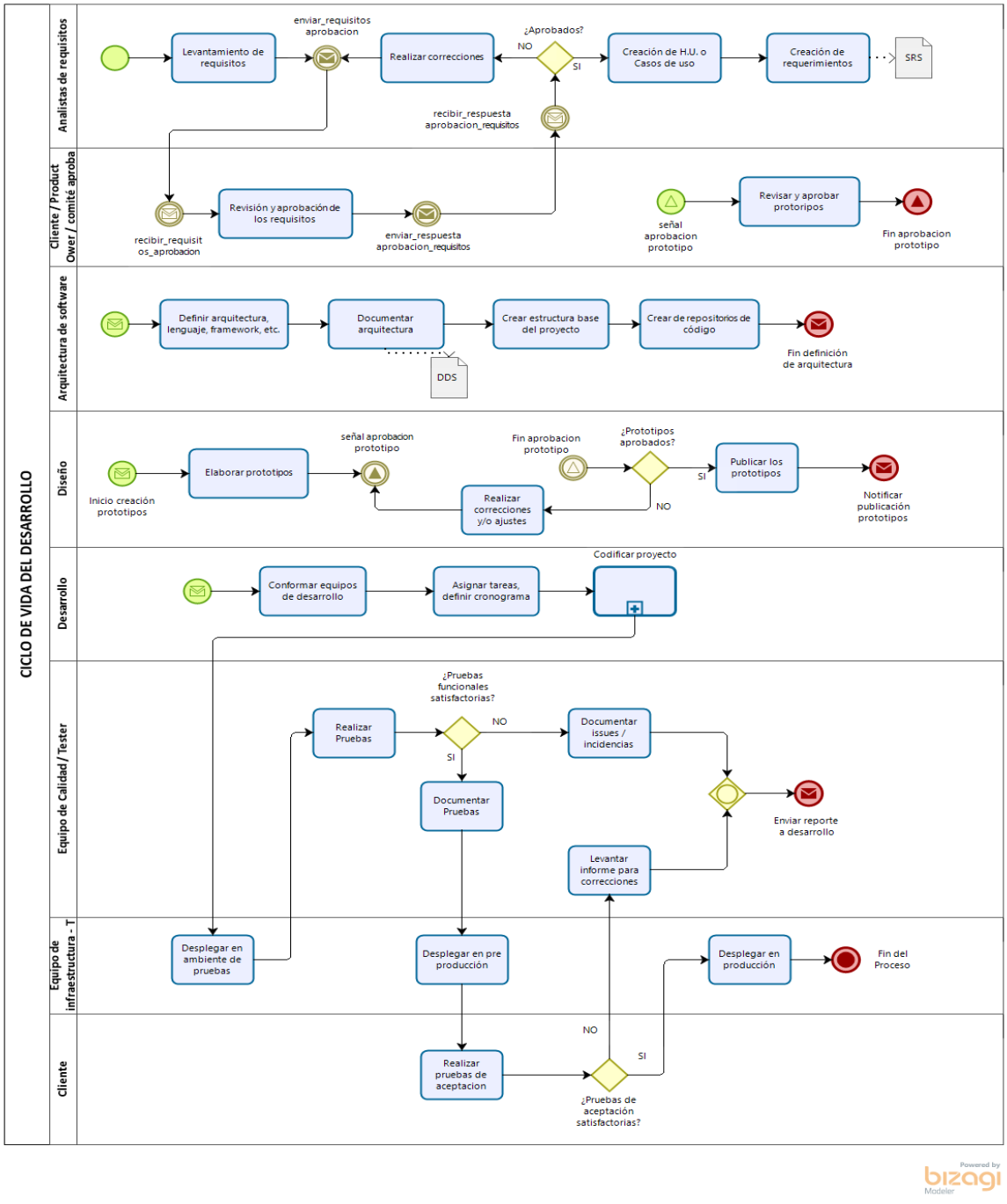
Test Automation vs. Automated Testing: The Difference Matters, tomado de https://www.qasymphony.com/blog/test-automation-automated-testing/#. Consultado el 27 de febrero de 2019.

Wikipedia. Repositorio. Tomado de https://es.wikipedia.org/wiki/Repositorio, consultado

el 13 de noviembre de 2018.

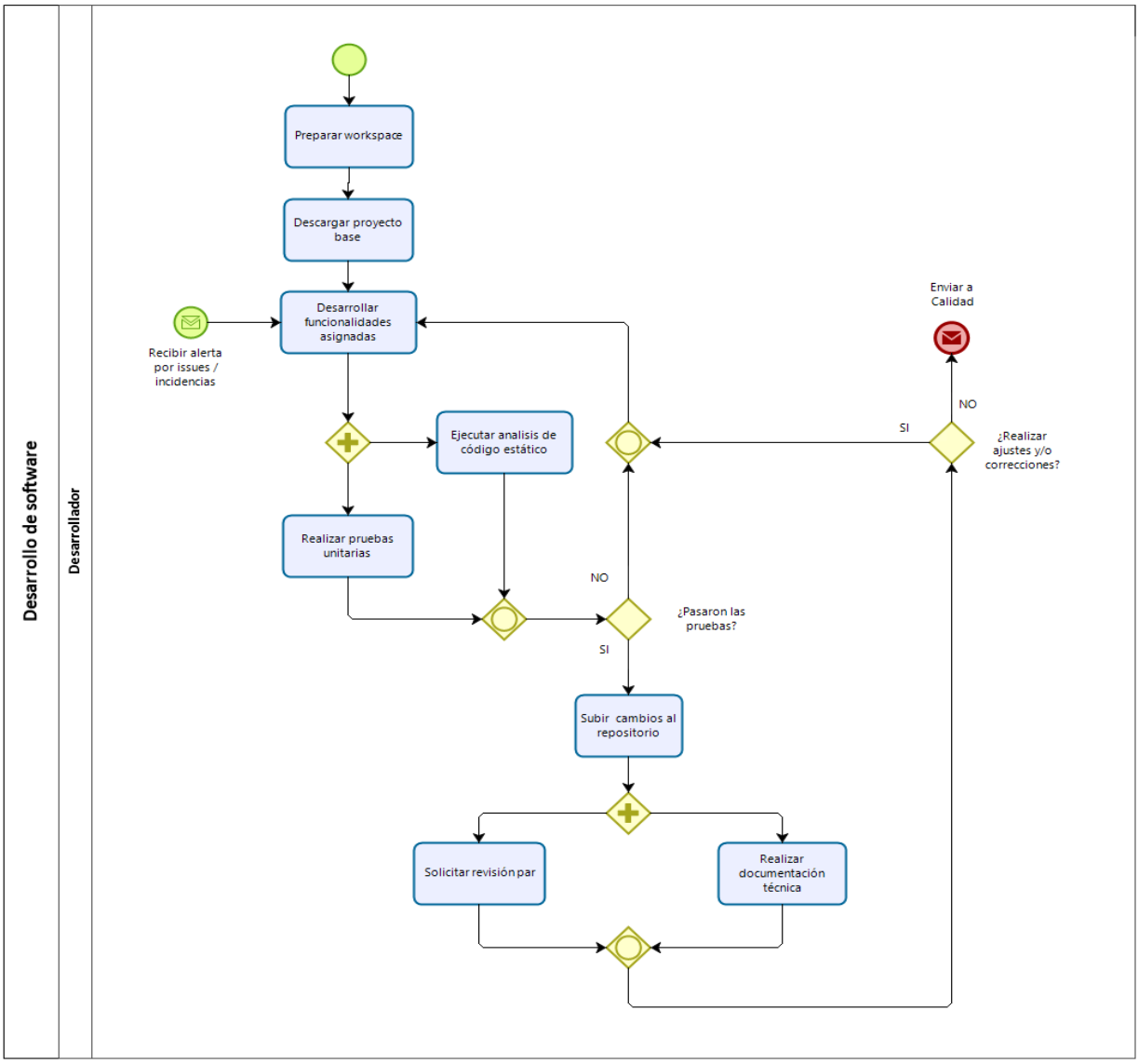
12. ANEXOS

12.1. Ciclo del proceso de desarrollo.



Anexo 1. Ciclo del proceso de desarrollo

12.2. Desarrollo de Software.



Anexo 2. Desarrollo de software

12.3. Formato encuesta de valoracion.

FORMATO 001.

ENCUESTA DE SONDEO PARA CATEGORIZACIÓN DE LA EMPRESA

INFORMACIÓN DE LA EMPRESA

NOMBRE / RAZÓN SOCIAL

DIRECCIÓN TELEFONO

CIUDAD DEPARTAMENTO

LIDER / RESPONSABLE DEVOPS

TELEFONO.

FACTIBILIDAD / DISPONIBILIDAD DE INFRAESTRUCTURA

1. ¿La empresa cuenta con centro de datos propio?
   SI NO

2. ¿La empresa cuenta con centro de datos en la nube?
   SI NO

3. ¿La empresa cuenta con servidor de repositorios de código?
   No posee servidor de repositorio.
   - Git
   - SVN
   - Mercurial
   - Otro. _____

4. ¿La empresa cuenta con algún software de gestión y construcción de proyectos?
   No cuenta con servicio de software de gestión
   - Maven
   - Gradle
   - Ant
   - Otro. _____

5. ¿La empresa cuenta con algun analizador de código estático?
   SI NO

En caso de ser afirmativa la respuesta anterior, responda las preguntas 6 y 7.

6. ¿Tienen métricas establecidas para la revisión de código estático?
   SI NO

7. ¿Las métricas están alineadas con los objetivos de calidad de software?
   SI NO

RECURSOS HUMANOS

8. ¿Cuántos empleados tiene la empresa?
   Menos de 20 empleados
   Entre 20 y 50 empleados
   Entre 50 y 75 empleados
   Entre 75 y 100 empleados
   Más de 100 empleados

9. ¿Cuántos empleados conforman el área de fábrica (Desarrollo, Calidad, Infraestructura)?
   Menos de 10 empleados
   Entre 10 y 20 empleados
   Entre 20 y 40 Empleados
   Más de 40 empleados

10. ¿Los equipos de trabajo con cuentan con un líder técnico y un líder de proyecto?
    SI NO

11. ¿La empresa cuenta con profesional(es) certificado(s) en Agile o scrum?
    SI NO

12. ¿La empresa tiene diseñado planes de capacitación?
    SI NO

13. ¿Las capacitaciones están orientadas a temas de desarrollo e ingeniería de software?

Nunca
Ocasionalmente
Casi siempre
Siempre

14. ¿Con que frecuencia son ejecutados los planes de capacitación?
    Mensualmente
    Trimestralmente
    Semestralmente
    Anualmente

15. ¿Considera que el personal de la empresa tiene la disposición suficiente para adoptar nuevas metodologías y/o esquemas de trabajo?
    SI
    Eventualmente
    NO

INGENIERIA DE SOFTWARE

16. ¿Cuantos proyectos en producción tiene la empresa?
    Menos de 5
    Entre 5 y 10 proyectos
    Entre 10 y 20 proyectos
    Más de 20 proyectos

17. ¿Cuántos proyectos en desarrollo tiene la empresa?
    Menos de 5
    Entre 5 y 10 proyectos
    Entre 10 y 20 proyectos
    Más de 20 proyectos

18. ¿Qué tipo de metodología aplican a sus proyectos de software?
    Ninguna
    Tradicional (Cascada, V)
    Iterativo / Incremental
    XP – Xtreme Programming

SCRUM
Otro._____

En caso de haber seleccionado una respuesta diferente de ninguna, responda:

19. ¿Qué tanto aplican la metodología en los proyectos?
    Menos del 30%
    Entre el 30% y 50%
    Entre el 50% y 75%
    Al 100%

20. ¿Emplean patrones de arquitectura en sus proyectos?
    SI NO

21. ¿Emplean patrones de diseño en sus proyectos?
    SI NO

22. ¿Cuentan con un sistema activo de repositorio de código?
    SI NO

23. ¿Realizan revisión par del código fuente de los desarrolladores?
    SI NO

24. ¿Tienen algún plan o sistema de control de versiones para el código fuente?
    SI NO

25. ¿Tienen claramente definidos los roles y permisos para la administración de los repositorios de código fuente?
    SI NO

26. ¿Cuánto tiempo requieren para realizar el despliegue de un producto en producción, por primera vez?
    Un día
    Entre 2 y 4 días
    Entre 5 y 8 días
    Más de 8 días

27. ¿Cuánto tiempo requieren para realizar el despliegue de actualizaciones de un producto en producción?

Entre 1 y 8 horas
Un día
Entre 1 y 3 días
Más de 3 días

28. ¿Cuándo deben realizar un despliegue a producción; quien lo hace?
    El personal de TI
    Un área diferente a TI
    Los desarrolladores
    Cualquier persona realiza despliegues en producción
29. ¿Con que ambientes cuenta la empresa?
    Solo desarrollo
    Solo Pruebas
    Solo Producción
    Desarrollo y Producción
    Desarrollo, pruebas y Producción
    Desarrollo, pruebas, pre-producción y producción

30. En caso de presentarse fallos en un despliegue realizado en producción, ¿cuán complicado es revertir los cambios?
    Muy complicado
    Medianamente complicado
    Medianamente fácil
    Muy fácil

31. ¿Cuánto tiempo les toma restaurar un despliegue estable en producción?
    Menos de una hora
    Entre 1 y 3 horas
    Entre 4 y 8 horas
    Más de 8 horas

32. ¿Realizan seguimiento o monitoreo de los productos que se encuentran en producción?
    SI NO

CERTIFICACIONES

33. ¿Está la empresa certificada en ISO 9001?
    SI NO

34. ¿Está la empresa certificada en ITIL/COBIT?
    Si NO

35. ¿La empresa tiene certificación ISO / EIC 27001?
    SI NO

36. ¿La empresa cuenta con certificación TOGAF?
    SI NO

37. ¿La empresa cuenta con certificación CMMI?
    SI NO

FACTIBILIDAD / DISPONIBILIDAD ECONÓMICA

38. ¿Cuál es la disponibilidad presupuestal con la que contaría el proyecto DevOps?
    Ninguna
    Menor a $10.000.000
    Entre $10.000.001 y $50.000.000
    Entre $50.000.001 y $100.000.000
    Mayor a $100.000.000

39. En caso de ser requerido, ¿Está la empresa en capacidad de contratar personal especializado para la implementación, administración y ejecución de DevOps?
    SI NO

40. En caso de ser requerido, ¿Está la empresa en capacidad de adquirir productos de infraestructura y/o servicios en la nube para el montaje de servicios DevOps?
    SI NO

Anexo 3. Formato encuesta de valoración

12.4. Asignación de pesos por preguntas y categorias.

Asignación de Pesos por preguntas y Categorias

Peso por Pregunta			*Puntaje Parcial por Categoria*	*Observaciones*
Infraestructura	1	20	*100*	2 Opciones: 20, 0
	2	10		2 Opciones: 10, 0
	3	15		5 Opciones: 0, 5, 5, 5, 0
	4	15		5 Opciones: 0, 5, 5, 5, 0
	5	20		2 Opciones: 20, 0
	6	10		2 Opciones: 10, 0
	7	10		2 Opciones: 10, 0
RRHH	8	10	*100*	5 Opciones. Excluyentes
	9	10		4 Opciones. 1,2,3,4
	10	15		2 Opciones: 15, 0
	11	15		2 Opciones: 15, 0
	12	15		2 Opciones: 15, 0
	13	10		4 Opciones: 0, 2, 3, 5
	14	10		5 Opciones: 4,3,2,1, 0
	15	15		3 Opciones: 10, 5, 0
ingeniera de Software	16	15	*300*	4 Opciones: 2,3,4,6
	17	15		4 Opciones: 2,3,4,6
	18	20		5 Opciones: 0, 3, 5, 5, 7
	19	20		4 Opciones: 2, 5, 6, 7
	20	15		2 Opciones: 15, 0
	21	15		2 Opciones: 15, 0
	22	15		2 Opciones: 15, 0
	23	15		2 Opciones: 15, 0
	24	15		2 Opciones: 15, 0
	25	15		2 Opciones: 15, 0
	26	30		4 Opciones: 15, 10, 5, 0
	27	15		4 Opciones: 7, 5 3, 0
	28	20		4 Opciones: 10, 7, 3, 0
	29	20		4 Opciones: 5 cada una
	30	20		4 Opciones: 0, 3, 7, 10
	31	20		4 Opciones: 10, 7, 3, 0
	32	15		2 Opciones: 15, 0
Certificaciones	33	20	*100*	2 Opciones: 0, 20
	34	20		2 Opciones: 0, 20
	35	20		2 Opciones: 0, 20
	36	20		2 Opciones: 0, 20
	37	20		2 Opciones: 0, 20
Economica	38	40	*100*	5 Opciones. 0, 5, 8, 12, 15
	39	30		2 Opciones: 0, 50
	40	30		2 Opciones: 0, 50
Puntaje Total		*700*		

Anexo 4. Asignación de pesos por preguntas y categorías

12.5. Resultados encuesta estudio de caso.

	Dimensión														
	Infraestructura			RRHH			Ingeniería de Software			Certificaciones			Financiera		
Preguntas, respuestas y valores	*P*	*R*	*V*	*P*	*R*	*V*	*P*	*R*	*V*	*P*	*R*	*V*	*P*	*R*	*V*
	1	SI	10	**8**	+100	2	**16**	10 a 20	4	**33**	SI	20	**38**	10 A 50	12
	2	NO	0	**9**	+40	4	**17**	10 a 20	4	**34**	NO	0	**39**	SI	30
	3	Git	5	**10**	SI	15	**18**	Iter/incr	5	**35**	SI	20	**40**	SI	0
	4	Maven	5	**11**	NO		**19**	50 al 75%	6	**36**	NO	0			
	5	NO	20	**12**	SI	15	**20**	SI	15	**37**	SI	20			
	6	NO	0	**13**	casi siempre	3	**21**	SI	15						
	7	NO	0	**14**	Mensual	4	**22**	SI	15						
				15	Eventual mente	10	**23**	NO							
							24	SI	15						
							25	SI	15						
							26	10 a 15	5						
							27	1 dia	5						
							28	Otra area	7						
							29	+ pruebas +	15						
							30	mediana	3						
							31	1 a 3	7						
							32	SI	15						
OBTENIDO	40			43			151			60			42		
MAXIMO	80			70			203			100			70		
% PONDERADO	50%			61%			74%			60%			60%		

Convenciones

P: Pregunta R: Respuesta V: Valor (Peso)

Anexo 5. Resultados encuesta Estudio de caso

Regresar al análisis del estudio de caso