UNIVERSIDAD NACIONAL TECNOLÓGICA DE LIMA SUR

FACULTAD DE INGENIERÍA Y GESTIÓN

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

"ELABORACIÓN DE LA MATRIZ DE MADUREZ PARA EL AREA DE DESARROLLO DE SOFTWARE EN UNA EMPRESA DE TELEFONIA"

TRABAJO DE SUFICIENCIA PROFESIONAL

Para optar el Título Profesional de

INGENIERO DE SISTEMAS

PRESENTADO POR EL BACHILLER

ORMACHEA HURTADO, GUILLERMO ALBERTO

Villa El Salvador

2019

DEDICATORIA

Dedico este trabajo de suficiencia profesional
a Dios porque sin él nada se podría lograr,
mis padres que me apoyan en todo momento,
a mis maestros que me apoyaron desde que
ingresé a la universidad.

AGRADECIMIENTO

Agradezco a Dios por permitirme estar aquí,
mis padres porque me impulsaron a seguir
adelante a pesar de los problemas, a mis
profesores que no dudaron de mis capacidades.

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCIÓN

El desarrollo de software es, actualmente, uno de los requerimientos más buscados en las empresas y lo que más se les impulsa hoy en día a los niños a aprender. El mundo de hoy se comunica por redes sociales, anuncios en diversos sitios web y no existe empresa que no quiera alinearse a lo que está de moda, la digitalización. Pero qué sucede si luego de contratar a un equipo de desarrolladores, no se sabe realizar un buen análisis de la situación actual del proyecto en sí y mucho menos un plan que ayude a fijar metas de corto, mediano y largo plazo. Lo más probable es que se tenga un equipo que haga un desarrollo "funcional" pero muy desordenado, que quizás en los primeros momentos no acarree muchos problemas pero luego de un tiempo y a medida que el equipo vaya rotando se vuelva un gran dolor de cabeza e imposible de manejar.

La implementación de diversos modelos de desarrollo de software ayudará al equipo de desarrollo pero la definición de una matriz de madurez permitirá ver el estado actual de todo el proyecto y también dará pie al desarrollo a un "camino" que deben seguir todos los involucrados para que poco a poco vayan desarrollando habilidades de desarrollo ordenado, seguro, sostenible en el tiempo y sobre todo ágil.

DevOps nace como una propuesta de unión y colaboración entre los diversos equipos involucrados. Este marco de trabajo permitirá definir una serie de pasos que deberán ser seguidos por los colaboradores para poder cumplir los objetivos propuestos, en el alcance definido y en el tiempo estimado. Para DevOps no existe la cultura de fallo y expulsión sino que incentiva a sus integrantes a cometer errores para que puedan aprender de ellos y con ello ganar la experiencia necesaria para no volver a cometer el mismo error, a esto se le conoce como mejora continua.

En los siguientes capítulos se explicará sobre lo que es DevOps, el cómo podríamos analizar la situación actual de un proyecto tomando como base una matriz de madurez que contempla las etapas de que debería tener un proyecto de desarrollo de software. Además se planteará sobre los pasos que deben ser seguidos por el equipo de desarrollo para que el proyecto que llevan pueda ser

desarrollado de la mejor manera, con software de calidad, con seguridad y que sea mantenible en el tiempo. Analizaremos sobre los tiempos estimados que podrían llevar esta lista de actividades y un posible costo de la implementación del DevOps en el equipo.

Finalmente tendremos una demostración que incluyan las etapas más elevadas de la matriz de madurez para mostrar que es posible fabricar software de calidad y que pueda ser desplegado rápidamente.

CAPITULO I

PLANTEAMIENTO DEL PROBLEMA

1.1 Descripción de la realidad problemática

Las buenas prácticas propuestas por DevOps han buscado mejorar el rendimiento de los equipos de desarrollo y reducir posibles riesgos desde los ambientes donde se hacen las pruebas hasta los ambientes productivos, sin embargo por factores como: presupuesto, tiempo, desconocimiento entre otros, se ha hecho caso omiso a nuevas propuestas que pretenden mejorar lo que ya había y que con el tiempo se ha ido modificando, además de que siempre hace falta un experto que pueda guiar a su equipo durante el proceso de mejora y renovación.

Las tareas que marcan el camino de un equipo de desarrollo para alcanzar la madurez con DevOps son mal definidas y muchas veces ni son definidas por lo que el equipo que adopta las practicas lo hace muy vagamente y sin conciencia ni motivación de querer mejorar su forma de trabajo.

1.2 Justificación del Problema

Esta propuesta se realizará porque es importante que las personas que deseen aplicar las buenas prácticas de DevOps tengan un modelo de cómo pueden hacer que su equipo alcance una madurez en los diferentes estados que tenga su proyecto.

1.3 Delimitación del Proyecto

1.3.1 Teórico:

Devops.-

DevOps es la unión de Desarrollo y Operaciones que en el pasado eran dos frentes en un mismo equipo que estaban separados. DevOps se encarga de unir ambos equipos (y algunos otros más) y, junto con la metodología ágil, plantear nuevas formas de hacer despliegues más rápidos, con mejor calidad, más seguros y siempre a tiempo. DevOps pasa de ser más que una automatización del proceso de despliegue ya que apoya desde la etapa del desarrollo hasta la etapa en la que se puede ver el producto.

Matriz de madurez.-

Es un modelo que se plantea en las organizaciones y ayuda a plantear los diferentes estados que se podrían tomar en un proceso. Fue desarrollado en sus inicios para los procesos de desarrollo de Software de la Universidad Carnegie-Mellon (Pensilvania, USA). Este modelo tiene como objetivo determinar cuál es el estado de desarrollo de los Procesos de una organización.

Desarrollo ágil de Software.-

Es un marco de trabajo conceptual de la ingeniería de software que promueve iteraciones en el desarrollo a lo largo de todo el ciclo de vida del proyecto. Existen muchos métodos de desarrollo ágil; la mayoría minimiza riesgos desarrollando software en cortos lapsos de tiempo. (Kent Beck, 2018)

1.3.2 Temporal:

El proyecto durará desde Octubre de 2019 hasta Diciembre de 2019.

1.3.3 Espacial:

Se usará en el equipo de desarrollo de una empresa de telefonía.

1.4 FORMULACIÓN DEL PROBLEMA

1.4.1 Problema General

El área de desarrollo de la empresa de telefonía no trabaja adecuadamente dentro de las prácticas ágiles y no existe una evaluación del desempeño de dicho equipo.

1.4.2 Problemas específicos

- No se ha realizado una evaluación del status actual de la forma de trabajar del equipo de desarrollo.
- Cómo realizar una matriz de madurez en base a la información recolectada del equipo en cuestión.

- Cómo demostrar al equipo de que lo que quiere lograr es posible y que no es un modelo puramente teórico.

1.5 OBJETIVOS

1.5.1 Objetivo General

Generar un modelo para evaluar la madurez del área de desarrollo de software mediante el uso de buenas prácticas de DevOps en una empresa de telefonía

1.5.2 Objetivos Específicos

- Realizar el diagnóstico de situación actual del equipo de desarrollo.
- Generar una matriz de madurez que permita visualizar el modelo del cómo debería ser y a donde se debe llegar.
- Elaborar un modelo demostrativo que ejemplifique el nivel más alto de madurez definido.

CAPITULO II

MARCO TEORICO

2.1 Antecedentes

Un primer trabajo corresponde a Rumiche y Pereira (2017) que realizaron la investigación: *Migración de los servicios de pagos en línea para soportar transacciones en el aplicativo móvil de una empresa de telecomunicaciones* en la escuela de pregrado de la Universidad de San Martín de Porres, Lima.

Su investigación estuvo basada en la migración de una aplicación a una arquitectura basada en microservicios que permitiera a la empresa de telefonía optimizar el tiempo de respuesta de unos servicios brindados por ellos. Los autores usaron la metodología SCRUM y se apoyaron en las buenas prácticas de DevOps para que el trabajo fuera más eficiente, más seguro y de un despliegue mucho más veloz.

Su investigación llegó a concluir que al implementar la arquitectura basada en microservicios se logró soportar el constante incremento de la demanda de transacciones realizadas mediante el aplicativo móvil.

Esta investigación aporta a mi trabajo de suficiencia ya que ha usado las buenas prácticas de DevOps en el proceso y esto ayudó a que el trabajo entre los equipos de desarrollo y operaciones sea mucho más colaborativo.

Un segundo trabajo corresponde a Belalcázar (2017) que realizó la investigación: *Arquitectura de un Data Center con herramientas DevOps* que fue presentada en la Universidad Nacional de la Plata, Argentina.

Esta tesis realiza el análisis y propone un modelo orientado a la administración eficiente de un Data Center que resulta a partir de un alineamiento estratégico entre el factor Negocio y Tecnología Informática. En esta investigación de implementan las herramientas al ciclo de vida de DevOps basándose en las buenas prácticas de ITIL de disponibilidad, monitoreo, integración continua y versionamiento de aplicaciones.

Belalcázar (2017) concluye que su pudo proponer un modelo de incorporación de herramientas al ciclo de vida de DevOps en base a ciertos criterios como disponibilidad, integración continua y versionamiento de aplicaciones.

Esta investigación aporta a mi trabajo de suficiencia pues aplica las buenas prácticas de DevOps y propone la inclusión de herramientas al ciclo de vida de DevOps respetando los criterios de Integración y entrega continua.

Un tercer trabajo corresponde a Pérez (2018) que realizó la investigación: *DevOps: IT Development in the Era of Digitalization(DevOps: Desarrollo TI en la Era de la Digitalización)* que fue presentada en la Universidad de Valladolid, España.

El propósito de Pérez (2018) fue transmitir una visión clara de la forma de cómo funciona DevOps en una organización. Identifica los beneficios que DevOps trae a una empresa y también los retos que la misma debe afrontar para poder adoptar DevOps de manera satisfactoria. Pérez (2018) se basa en textos como método para identificar las prácticas que se deben seguir para adoptar DevOps y con eso llegar a una conclusión.

Pérez (2018) concluye que DevOps es un fenómeno que ha ganado mucha popularidad en los últimos años. Muchas empresas con Facebook, Yahoo y Netflix han adoptado estas buenas prácticas de DevOps y les ha permitido mejorar su nivel de desarrollo.

Esta investigación aporta a mi trabajo de suficiencia pues desarrolla la base teórica y práctica analizando en base a textos el flujo que se debe seguir para poder adaptar el modelo de desarrollo en una empresa al flujo de DevOps basándose en los pilares fundamentales.

Un cuarto trabajo corresponde a Farías (2017) que realizó la investigación: *Definición de un ambiente de construcción de aplicaciones empresariales a través*

de Devops, Microservicios y Contenedores que fue presentada en la Universidad Técnica Particular de Loja, Ecuador.

Esta investigación se enfoca en la aplicación de las prácticas de las metodologías ágiles y su integración con DevOps. Farías (2017) presenta a DevOps como cultura de automatización de procesos en las organizaciones. La autora de esta investigación muestra también las herramientas que ha usado para cada fase del ciclo de vida de DevOps y las que ayudan a la incorporación de las buenas prácticas de DevOps.

Farías (2017) concluye que DevOps es una cultura de colaboración y comunicación entre el área de desarrollo y las otras áreas automatizando el proceso de entrega de software y cambios en la infraestructura. Además, agrega que la adopción de DevOps en una organización la obligará a adquirir nuevas habilidades técnicas y culturales.

Esta investigación aporta a mi trabajo de suficiencia realiza la aplicación de la teoría usando para sus depliegues la tecnología de microservicios usando contenedores. Hace uso de DevOps y sus buenas prácticas como marco de trabajo para automatizar los procesos dentro del flujo de vida de DevOps.

2.2 Bases teóricas

2.2.1 Transformación digital

En la actualidad la Transformación digital ha pasado de ser una opción a convertirse en una necesidad. Sin embargo se debe tomar en cuenta que aplicar este tipo de transformación en una empresa no es nada fácil ya que no solo implica un cambio del uso de las herramientas e implementación de nuevas sino que también es un cambio de cultura, pensamiento y hábitos.

Definición

Según Duro (2017) señala que: Es el proceso por el cual las organizaciones o empresas reorganizan estrategias y métodos de trabajo

para poder alinearse a la era digital y así obtener más beneficios. La transformación Digital busca aprovechar oportunidades que antes no se podían tomar por lo que suele ser complejo dependiendo del estado en el que se encuentre la organización en el momento de tomar la decisión.

Requisitos

Lo primero es ser honestos y sincerar toda la información que se tiene acerca de la empresa u organización para plantear un modelo de situación actual.

Luego es tener muy en cuenta que este proceso es un proceso disruptivo pero que su introducción debe ser adecuada al ritmo de la empresa para que su implantación impacte lo menos posible en la empresa.

Se necesitará de constante capacitación de el/los equipos que ha venido trabajando de la manera anterior. Ellos necesitaran empezar a conocer sobre herramientas y su modo de uso para sacarle el mayor provecho posible.

Beneficios

Según Duro (2017) se pueden listar los siguientes beneficios:

- Mejora de la comunicación con los clientes.
- Acceso a nuevas oportunidades de Negocio.
- Alarga la vida de la empresa.
- Reduce Costos.
- Vuelve a los empleados felices
- Facilita la captación de personal

Duro Limia, Sonia (2017). *Transformación Digital: Jose Facchin,* Recuperado de: https://josefacchin.com/transformacion-digital/

2.2.2 El manifiesto Agil

El manifiesto ágil es un documento que fue redactado en el año 2001 por 17 expertos en programación, este documento supuso un cambio radical en la forma en la que se desarrollaba software. El manifiesto dicta:

> Estamos descubriendo formas mejores de desarrollar
>
> software tanto por nuestra propia experiencia como
>
> ayudando a terceros. A través de este trabajo hemos
>
> aprendido a valorar:
>
> Individuos e interacciones sobre procesos y herramientas
>
> Software funcionando sobre documentación extensiva
>
> Colaboración con el cliente sobre negociación contractual
>
> Respuesta ante el cambio sobre seguir un plan
>
> Esto es, aunque valoramos los elementos de la derecha,
>
> valoramos más los de la izquierda. (Beck, et al., 2001)

Según el manifiesto citado la comunicación entre todos los miembros del proceso en general debe ser superior a los puestos que cada uno ejerce. Destaca la entrega de software rápido y deja de lado mucha documentación que antes solía solicitarse, adopta al cliente como parte del desarrollo y elimina la barrera que existía.

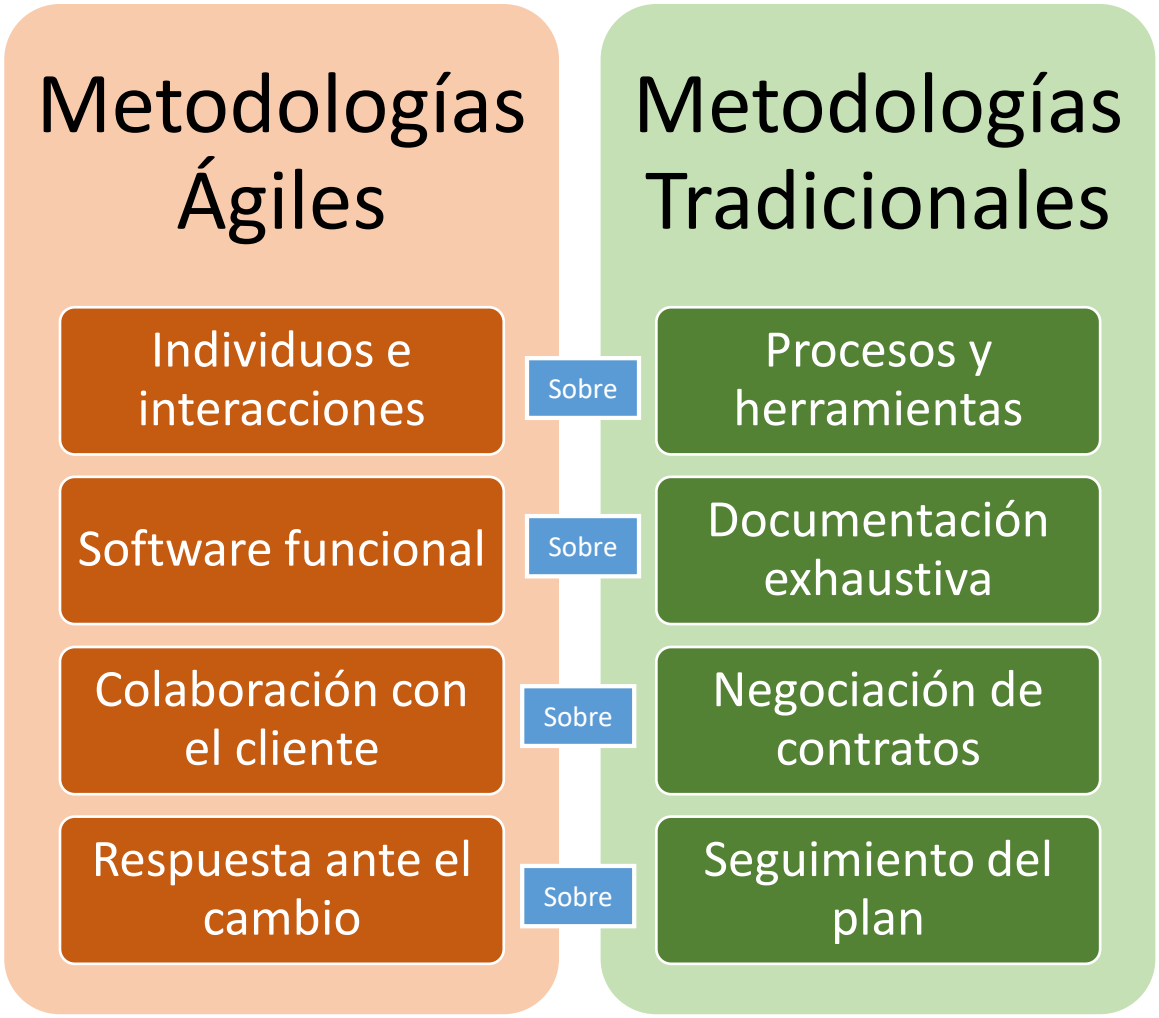

Figura 1: Comparación que presenta el Manifiesto Agile acerca de las metodologías Ágiles y las tradicionales. Fuente: Elaboración propia.

2.2.3 Metodologías Agiles

Son aquellas que permiten adaptar la forma del trabajo a las condiciones en las que se encuentra el proyecto lo que le da una flexibilidad mayor y es mucho más rápida la reacción frente a situaciones no planificadas.

Al aplicar las metodologías ágiles, las empresas consiguen gestionar los proyectos de forma flexible, eficaz y autónoma lo que incrementa la productividad.

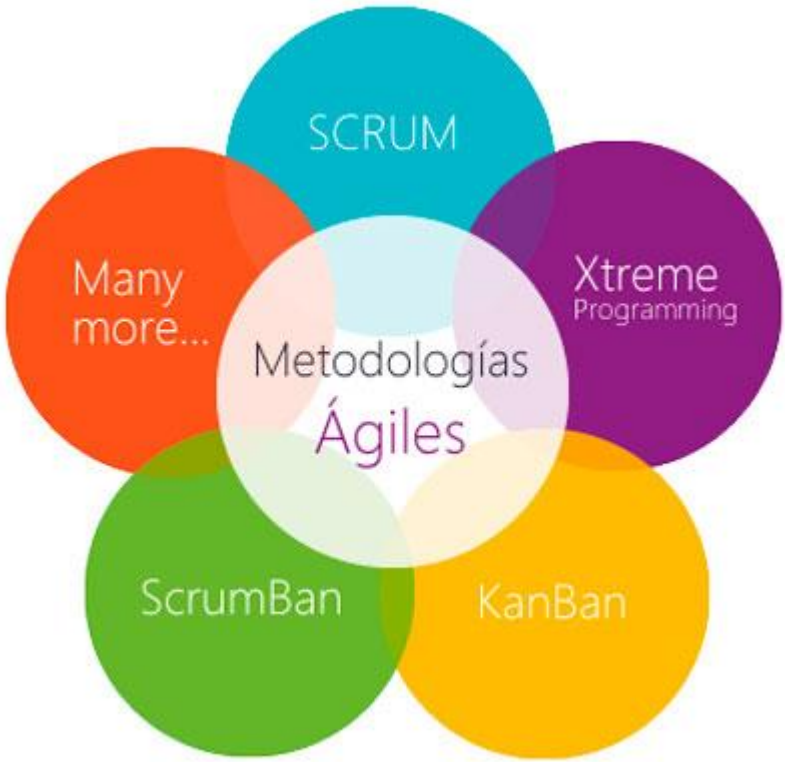

Figura 2: Algunas Metodologías ágiles:Las metodologías ágiles:'Qué beneficios aportan al desarrollo de Software?. Recuperado de: https://www.bravent.net/metodologias-agiles-que-beneficios-aportan-al-desarrollo-del-software

2.2.4 SCRUM

Definición

SCRUM es un proceso en el que se aplican un conjunto de buenas prácticas para que el trabajo se realice de manera colaborativa y obtener el mejor resultado del proyecto.

SCRUM se centra en realizar entregas parciales del producto final cada cierto periodo de tiempo, siempre priorizando por el beneficio que le traiga al proyecto. Esta metodología está indicada para proyectos donde se tienen entornos complejos, donde se necesitan tener resultados pronto, donde la incertidumbre de lo que pueda suceder en el proceso es alta con requisitos cambiantes y donde la innovación, flexibilidad y competitividad son fundamentales.

El proceso SCRUM

Para SCRUM, en un proyecto deben fijarse periodos de entrega que sean cortos y fijos que por lo general son de 2 semanas pero que pueden llegar a ser de hasta 4 semanas como máximo. A estos periodos SCRUM los conoce como Sprints.

En cada Sprint se realizan diferentes reuniones conocidas como ceremonias, tenemos las siguientes:

- Planificación del Backlog

  Trigas (2012) señala que en esta reunión se define un documento con los requisitos del sistema con sus respectivas prioridades.

  De esta reunión se debe obtener el Sprint Backlog que es la lista de tareas que se realizarán en el Sprint. (Trigas, 2012, Metodología SCRUM, p. 36)
- Seguimiento del Sprint(Dayli)

  Son las revisiones diarias que tiene el equipo acerca de cómo va el avance de las tareas asignadas. Por lo general se realiza al empezar el día y cada miembro tiene que responder 3 preguntas: ¿Qué hiciste el día anterior?, ¿Qué harás el día de hoy?, ¿Tienes algún bloqueante que no te permita continuar?
- Revisión del Sprint

  Es una reunión que se da al finalizar el Sprint donde se revisa el trabajo realizado y en caso no se hayan podido completar tareas, se dan a conocer al equipo para que puedan tenerlo en cuenta.

Figura 3: Reuniones a lo largo del Sprint, Fuente: Elaboración propia

2.2.5 DevOps

Definición

No existe una definición concreta de DevOps, pero según PMC Group (2018) se pueden rescatar algunos principios que pueden ayudar a entender el porqué de DevOps.

a) Acción centrada al cliente

   DevOps siempre tomará en cuenta la opinión del cliente, por lo que necesitará su Feedback cada que sea necesario. De esto dependerá si el rumbo trazado seguirá su curso o tendrá que variar.

b) Crear con el resultado en mente

   DevOps apoya que se deben realizar entregas pequeñas pero de valor. DevOps confía en que al entregar una pequeña funcionalidad al cliente y recibir su Feedback podrá tener mejores resultados en el tiempo.

c) Responsabilidad de inicio a fin

   El software desarrollado ya no pasa de las manos del desarrollador hacia el equipo de operaciones (Intentando así deslindar responsabilidad) sino que todos los equipos siguen el estado y son responsables del producto de principio a fin.

d) Equipos interfuncionales autónomos

   Los equipos que participan en flujo de desarrollo y despliegue deben ser colaborativos unos con otros y a la vez se autónomos de manera que no sean detenidos en caso falten algunas personas.

e) Mejora continua

   Este principio se basa en Lean que se enfoca en la adaptabilidad y aprendizaje. Es importante experimentar para poder obtener nuevos y mejores resultados y no tenerle miedo a los fracasos ya que se aprende de los mismos.

f) Automatizar todo lo posible

   La automatización, ayudará a reducir los desperdicios, a reducir las tareas manuales (y por consiguiente los fallos humanos), ayudará a reducir los tiempos de entrega y reducir las tareas redundantes.

(PMC Group, 2018, Devops: Un vistazo al Futuro, p. 16-19)

Objetivos de DevOps

DevOps se implementa en proyecto persiguiendo los siguientes objetivos:

- Entregas más constantes

  Siempre hay una versión lista para ser desplegada. Se da una mayor cantidad de revisiones antes de hacer un despliegue. Se tienen mayor cantidad de actualizaciones para el cliente.

- Entregas más rápidas

  Menor tiempo en la entrega a tienda (Despliegue de aplicaciones). Al automatizar el proceso se reduce el error humano. Reduce los costos y los esfuerzos.

- Entregas con mayor calidad

  Los errores se identifican mucho más rápido. Los errores en producción son menores o nulos. Las aplicaciones son más seguras y estables.

2.2.6 Ciclo de vida de DevOps

Figura 4: Ciclo de vida de DevOps: Recuperado de: https://tech.tribalyte.eu/blog-introduccion-devops

a) Desarrollo

   Es la etapa en la que el desarrollador codifica la aplicación antes de ser lanzada.

b) Prueba

   Se realizan las pruebas de código, pruebas unitarias, de integración, funcionales, no funcionales.

c) Integración

   Se compila el código desarrollado y se verifica el funcionamiento y su interacción con las demás herramientas usadas.

d) Despliegue

   Se despliega en los servidores dependiendo del estado al que se esté pasando. (Desarrollo, QA, Producción)

e) Monitoreo

   Según DevOps se deben tener herramientas que ayude a monitorear lo que ha sido desplegado en todo momento.

2.2.7 Cobit 5

Definición

COBIT (Control Objectives for Information and related Technology) es un framework que presenta buenas practicas con respecto a la supervisión y control en el área de TI.

Esta guía es desarrollada y mantenida por ISACA (Information Systems Audit and Control Association) y IT Governance Institute. Actualmente van por la versión 5 que fue presentada en el año de 2012.

Cobit 5 proporciona una visión de tipo empresarial del Gobierno de TI que tiene como creadores de valor para la empresa a la información y la tecnología.

COBIT Maturity Model

Según Pederiva (2003) el modelo de madurez presentado por ISACA y la IT GI, es una herramienta que ayuda a medir cuan bien desarrollados están los procesos administrativos con respecto a controles internos. El modelo de madurez califica desde el 0 (Inexistente) hasta el nivel 5 (Optimizado). Este Process Capability Model está basado en la ISO/IEC 15504 y es muy exigente con respecto a que los pasos para subir de nivel deben ser cumplidos a cabalidad.

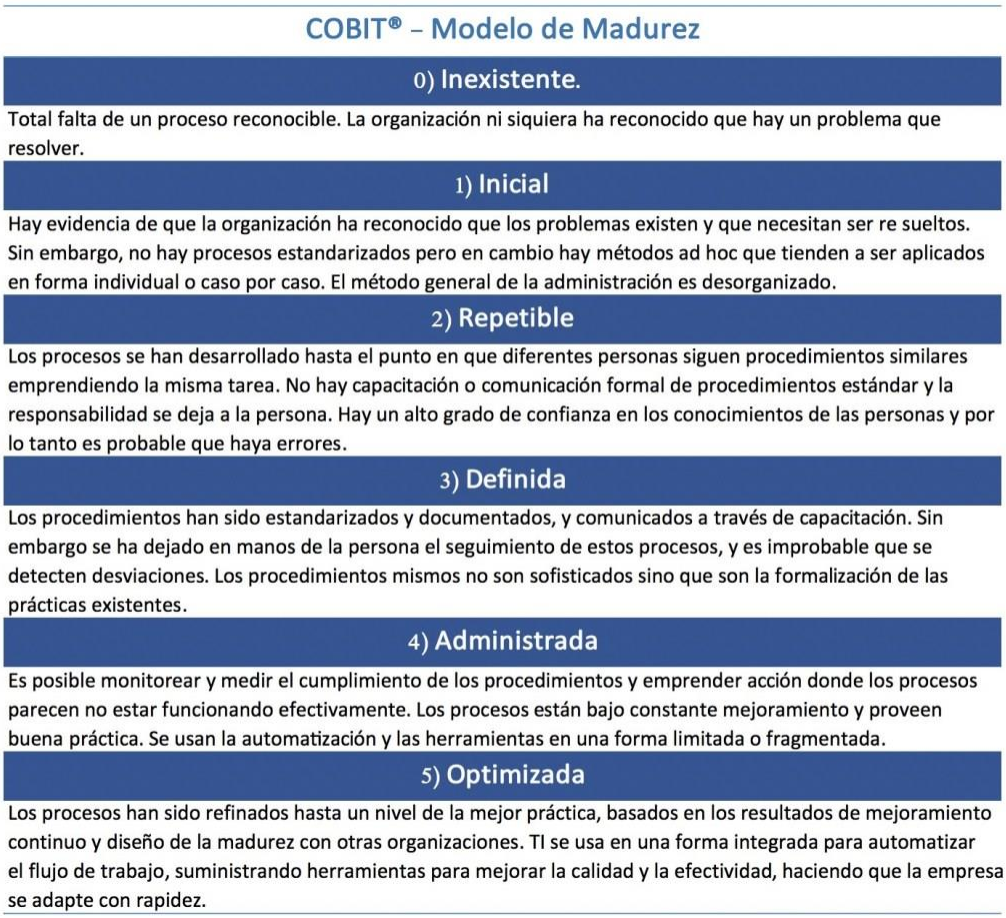

Figura 5: Niveles de madurez planteados por COBIT 5. Rescatado de: https://i0.wp.com/ipmoguide.com/wp-content/uploads/2018/06/COBIT%C2%AE-Modelo-de-Madurez-01.jpeg?resize=1024%2C956&ssl=1

2.2.8 ISO 38500: Gobierno de TI

La ISO 38500 fue publicada en el año de 2008 tomando como base la norma australiana AS8015:2005. Es la primera de una serie de normas dedicadas al Gobierno de TI.

Tiene por objetivo proporcionar una serie de principios que sean usados por el equipo de dirección que se encarga de evaluar, dirigir y monitorizar el uso de las Tecnologías de Información y comunicaciones

Esta norma también cuenta con un modelo de madures de 6 niveles al igual que Cobit 5, los niveles son:

- Nivel 0: La empresa u organización no conoce la norma ISO 38500 o no sabe que lo necesita.
- Nivel 1: El principio dado por la ISO 38500 se establece pero el área de gobierno de TI está desorganizado.
- Nivel 2: El principio de la ISO 38500 es inmaduro y se siguen procesos regulares.
- Nivel 3: El principio de la ISO 38500 empieza a madurar y los procesos de TI se documentan y comunican.
- Nivel 4: El principio de la ISO 38500 es maduro y los procesos se monitorean y miden.
- Nivel 5: El principio de ISO 38500 es maduro, el gobierno de TI usa las mejores prácticas.

2.2.9 CMMI (Capability Maturity Model Integration)

Definición

El CMMI (Modelo de Madurez de Capacidad Integrado) es una expansión del CMM (Modelo de Madurez). Es una herramienta que ayuda a mejorar procesos llevándolos a un estado de optimización alto. Este modelo fomenta la cultura de productividad y eficiencia que conlleva a reducir los riesgos en el desarrollo de Software.

El enfoque del CMMI puede variar, lo que nos pueden afirmar los libros es que nos ayuda a evaluar la madurez en procesos de una

organización y así proporcionar una orientación sobre el cómo se pueden llevar a cabo mejoras.

Beneficios

- Permite optimizar procesos.
- Une la parte de la gestión y la ingeniería en una organización y permite reconocer los objetivos puntuales.
- Incorpora la experiencia de muchos otros proyectos.
- Permite aplicar prácticas de muy alto nivel.
- Propone un estándar que ayuda a disolver discrepancias.

2.2.10 Matriz de Madurez

La matriz de madurez es una herramienta que nos ayudará identificar el estado de los procesos de desarrollo de Software. Se puede tomar como referencia el CMM – SW donde se clasifican los estados en: Inicial, repetible, Definido, Gestionado y Optimizado. Esta matriz define los lineamientos del funcionamiento de los procesos de desarrollo de software y permite a las empresas alinearse al mismo.

Según el CMMI los 5 niveles pueden representarse de la siguiente manera:

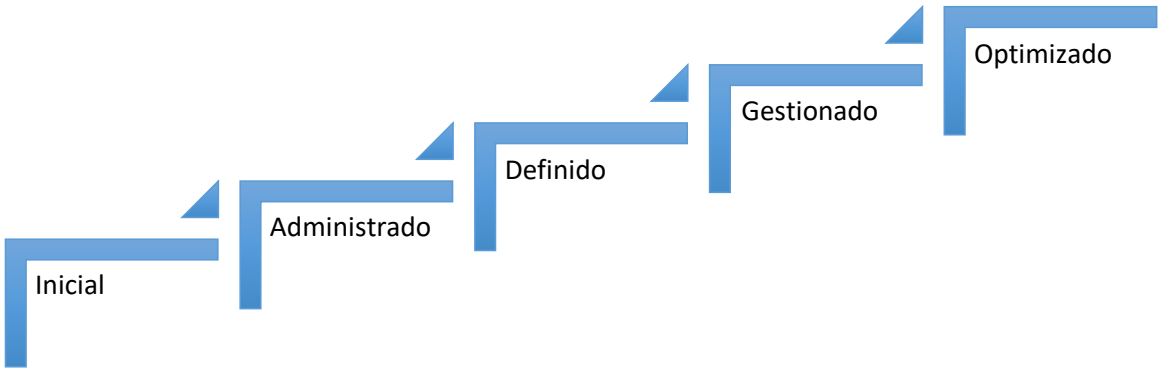

Figura 6: Niveles de madurez de los procesos según el CMMI, Fuente:Elaboración propia

Nivel	Característica
Inicial	El proceso es predecible, se tiene poco control.
Administrado	El proceso es reactivo, aplicado a proyectos.

Definido	El proceso de vuelve proactivo, a nivel de organización.
Gestionado	El proceso ya es medido y controlado.
Optimizado	El proceso se enfoca en mejora continua.

Tabla 1: Caracteristicas de los procesos por nivel según el CMMI, Fuente: Elaboración propia

2.3 Definición de términos básicos

- DevOps:
  DevOps es un conjunto de prácticas que tienen por objetivo reducir el tiempo entre realizar un cambio en un Sistema y euq el cambio se vea reflejado en los ambientes productivos, siempre asegurando una alta calidad en el código. (IEEE, 2016)
- Madurez en Tecnología:
  La madurez en tecnología se evalúa por el tipo de procesos que se tienen en la organización, el tiempo que toman y la forma en la que el equipo en general está involucrado.
- Agilidad:
  Palabra que está teniendo un gran auge, se define como la entrega de productos funcionales en corto tiempo, colaboración de equipo, inclusión del cliente en los procesos.
- Equipo de Desarrollo:
  Equipo encargado de codificar la/las aplicaciones para una organización, se encarga también de probarlas en primera instancia.
- Equipo de Operaciones:
  Equipo que lleva lo codificado por el equipo de desarrollo hasta los ambientes donde serán usados por los clientes y usuarios finales.
- Practicas Agiles:
  Las nuevas prácticas ágiles se caracterizan por ser mucho más organizadas, con menos desperdicio de recursos y optimizada.
- Desempeño:
  El desempeño del equipo se mide por la forma en la que ellos trabajan, su interacción con su entorno y su productividad.
- Deuda técnica:

Son generadas por las malas prácticas de desarrollo que adoptamos y que luego se tienen que corregir, lo que implica esfuerzo y costo humano.

- As – Is:
  Es el estado actual de una organización, ayuda a la identificación de posibles fallas actuales y puntos a mejorar.
- To – Be:
  Es el objetivo al que se quiere llegar mediantes un proceso o una guía acertada.
- Jenkins:
  Es un servidor de automatización que permite integrar muchas herramientas y hacer despliegues desde el desarrollo hasta llevarlo a ambientes productivos.
- SonarQube:
  Herramienta de análisis de código estático que devuelve un informe que es fácilmente analizado por los desarrolladores.
- Docker:
  Herramienta que permite levantar, generar y configurar contenedores para despliegues, pruebas y programas.
- Pipeline:
  Es un recetario de instrucciones que usa Jenkins para identificar las tareas que se tienen que realizar. Este puede estar dividido en nodos(en caso se trabaje con esclavos), en stages(etapas) y steps(pasos) que se deben seguir para completar la tarea.

CAPÍTULO III

DESARROLLO DEL TRABAJO DE SUFICIENCIA PROFESIONAL

3.1 Modelo de solución propuesto

Para el desarrollo de este trabajo, se optó por usar como guía los niveles del CMMI (5 niveles) ya que Cobit 5 tiene un nivel 0 que considera que no se tiene en consideración ninguna buena práctica y es inexistente lo que no se alinea a la realidad. La ISO 38500 también plantea un modelo de madurez específico para el gobierno de TI. Si bien el último nivel hace referencia al uso de las mejores prácticas, el modelo de madurez del CMMI representa mejor la situación actual.

DevOps reúne una gran variedad de prácticas en las diferentes etapas del ciclo de vida del desarrollo de software y en base a esa información se realiza una matriz para dar por sentado el nivel de madurez con respecto a prácticas de DevOps del equipo de desarrollo.

Cada uno de estos niveles puede ser identificado por la forma en la que realizan las tareas desde ser muy manuales y desorganizadas hasta ser automatizadas y regidas por un estándar que sea cumplido (en la medida de lo posible) por la organización.

Para poder tener una visión clara de la situación en la que se encuentra la empresa de telefonía se tuvo que observar su forma de trabajo consultando si existe algún modelo ya descrito que les diga el cómo deben realizar sus labores con respecto a desarrollo, ejecución de pruebas y despliegue a servidores.

3.1.1 El As – Is (Estado actual)

Se hizo un reconocimiento rápido en el área de desarrollo con respecto a los siguientes aspectos: Control de versiones, Compilación, Pruebas unitarias y los Despliegues a servidores.

Para cada frente existieron preguntas clave que ayudaron a identificar de manera rápida el estado actual del área de desarrollo en cuestión.

Frente	**Control de versiones**
Preguntas Clave	- ¿Se realiza un control de versiones del código? - ¿Utilizan alguna herramienta de paga o gratuita? - ¿Existe algún lineamiento de la forma en la que se debe versionar el código? - Si se tiene la herramienta de versionamiento, ¿Está enlazado con las herramientas de DevOps?

Tabla 2: Preguntas clave de control de versiones. Fuente: Elaboración propia.

Frente	**Compilación**
Preguntas clave	- ¿De qué manera se compilan los servicios desarrollados? - ¿Se compila en el mismo ordenador o en el servidor? - ¿Se tiene algún lineamiento de cómo realizar la compilación? - ¿La compilación se realiza con herramientas DevOps?

Tabla 3: Preguntas clave compilación. Fuente: Elaboración propia.

Frente	**Pruebas Unitarias**
Preguntas clave	- ¿Se realizan pruebas unitarias sobre el código? - ¿Utilizan alguna herramienta de análisis de código estático? - ¿Existe algún lineamiento sobre estándares de cobertura de código y pruebas unitarias? - Si la herramienta existe, ¿Está enlazada con las herramientas de DevOps?

Tabla 4: Preguntas clave Pruebas Unitarias. Fuente: Elaboración propia.

Frente	**Despliegue a servidores**
Preguntas clave	- ¿Quién o quiénes son los encargados de los despliegues a servidores? - ¿Los despliegues se realizan de manera manual o automatizada? - ¿Existe algún lineamiento que indique el proceso para los despliegues a servidor?

Tabla 5: Preguntas clave Despliegue a servidores. Fuente: Elaboración propia.

Luego de responder la lista de preguntas, se obtuvieron los siguientes resultados.

Frentes	**Control de versiones**	**Compilación**	**Pruebas unitarias**	**Despliegues a servidores**
Resultados	El código fuente se almacenaba en un repositorio central pero no se usa de manera adecuada	La compilación se realizaba de manera local ejecutando scripts.	Se habían desarrollado pruebas unitarias pero no se tenía una estimación en porcentaje y no existía una herramienta de análisis de código estático	Existía un área especial (PAP) que se encargaba de realizar los pases a producción con scripts ejecutados manualmente.

Tabla 6: Resultados de estado actual. Fuente: Elaboración propia.

Se puede evidenciar que por los resultados obtenidos, el área de desarrollo de software de la empresa de telefonía no tiene una gran madurez con respecto a prácticas de DevOps.

3.1.2 El To – Be (Como debería de ser)

Habiendo ya evidenciado el estado actual del área de desarrollo, se pasa a desarrollar la matriz de madurez que representa los estados de cada frente analizado en los que el área de desarrollo se puede clasificar. El To – Be muestra el objetivo al que se quiere llegar en un plazo determinado de tiempo y con una serie de pasos definidos.

Tomando en consideración los niveles que propone el CMMI (Inicial, Administrado, Definido, Gestionado y Optimizado) se hicieron unas pequeñas modificaciones a los nombres de los estados para que el modelo pueda ser descrito de manera correcta

NIVEL DE MADUREZ		CONTROL DE VERSIONES	COMPILACIÓN	PRUEBAS UNITARIAS	DESPLIEGUES A SERVIDORES
DevOps	**5**	El código fuente se sube a un repositorio central siguiendo los lineamientos del área de Gobierno	La compilación de la aplicación cumple con los lineamientos del área de gobierno para su integración con el pipeline de Dev Ops	La aplicación tiene pruebas unitarias y cumple con un porcentaje de cobertura > 80%	El despliegue de la aplicación cumple con los lineamientos del área de gobierno
Desarrollo Agile	**4**	El código fuente se almacena en un repositorio central utilizando versiones con cada cambio	Existe un proceso automatizado con herramientas para la compilación de la aplicación según la documentación	La aplicación tiene pruebas unitarias y cumple con un porcentaje de cobertura entre el 30% y el 80%	La aplicación se despliega de forma automatizada con herramientas
Automatización	**3**	El código fuente se almacena en un repositorio central sin lineamientos	Existe un proceso automatizado con scripts para la compilación de la aplicación según la documentación	La aplicación tiene pruebas unitarias y cumple con un porcentaje de cobertura < 30%	La aplicación se despliega de forma automática con scripts
Desarrollo Tradicional	**2**	El código fuente se respalda en un storage compartido por los desarrolladores	La aplicación se compila según un documento de procesos	La aplicación tiene pruebas unitarias pero no se cuenta con informe de cobertura	La aplicación se despliega de forma manual según un manual de procesos
Desarrollo ineficiente	**1**	El código fuente se encuentra en el equipo del desarrollador	La aplicación se compila en el equipo del desarrollador. No hay documentación	La aplicación tiene desarrolladas algunas pruebas unitarias	La aplicación es desplegada de forma manual por el desarrollador

Tabla 7: Matriz de madurez propuesta. Fuente: Elaboración propia.

Habiendo definido la matriz de madures propuesta, se clasificará al área de desarrollo de Software de la empresa de telefonía.

Frentes	**Control de versiones**	**Compilación**	**Pruebas unitarias**	**Despliegues a servidores**
Estado	Nivel 3: El código fuente se almacena en un repositorio central sin lineamientos	Nivel 3: Existe un proceso automatizado con scripts para la compilación de la aplicación según la documentación	Nivel 2: La aplicación tiene pruebas unitarias pero no se cuenta con informe de cobertura	Nivel 2: La aplicación se despliega de forma manual según un manual de procesos

Tabla 8: Clasificación del estado de madurez. Fuente: Elaboración propia

La matriz de madurez generada y la clasificación de las prácticas de desarrollo de Software nos ayudarán a generar además una guía del cómo llegar hacia el nivel más alto donde los aspectos analizados serán realizados de manera automática y siempre tomando en cuenta un lineamiento propuesto por el área de gobierno de TI.

Para cada frente analizado, se generarán una serie de pasos que nos permitan ir escalando los niveles hasta llegar al objetivo tomando como base el nivel actual.

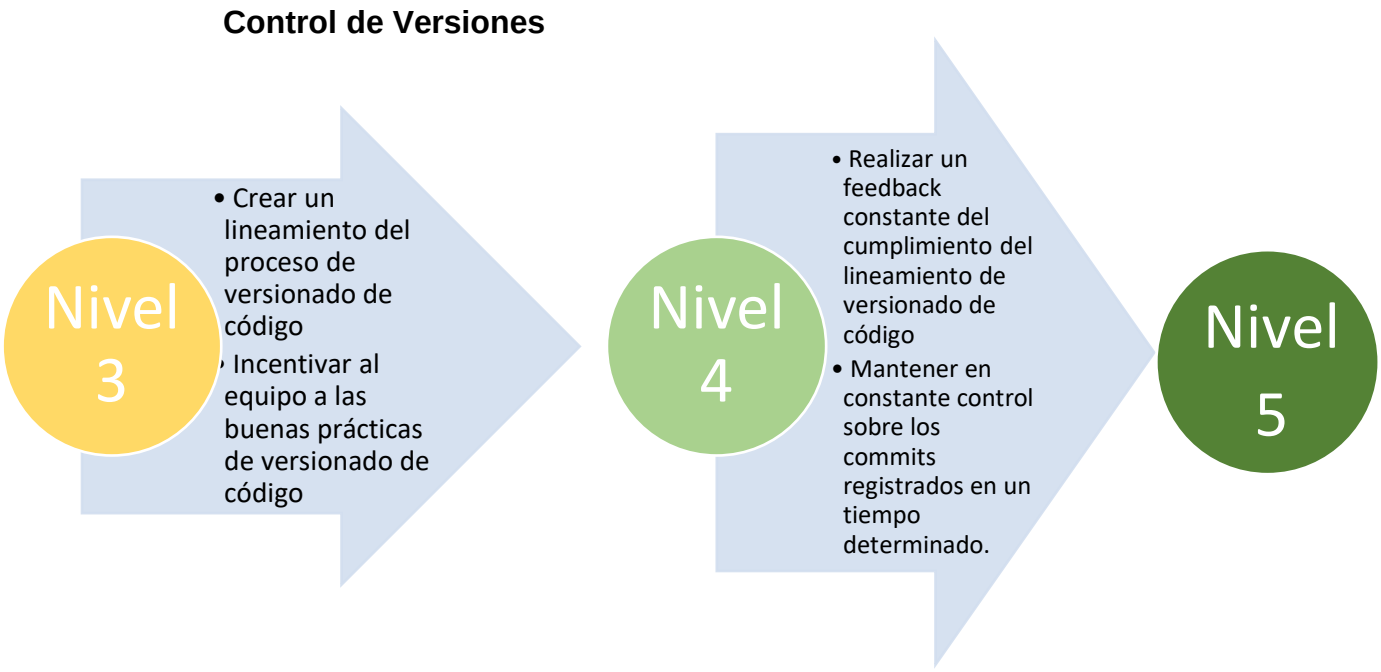

Figura 7: Pasos para conseguir los objetivos. Control de versiones: Fuente: Elaboración propia

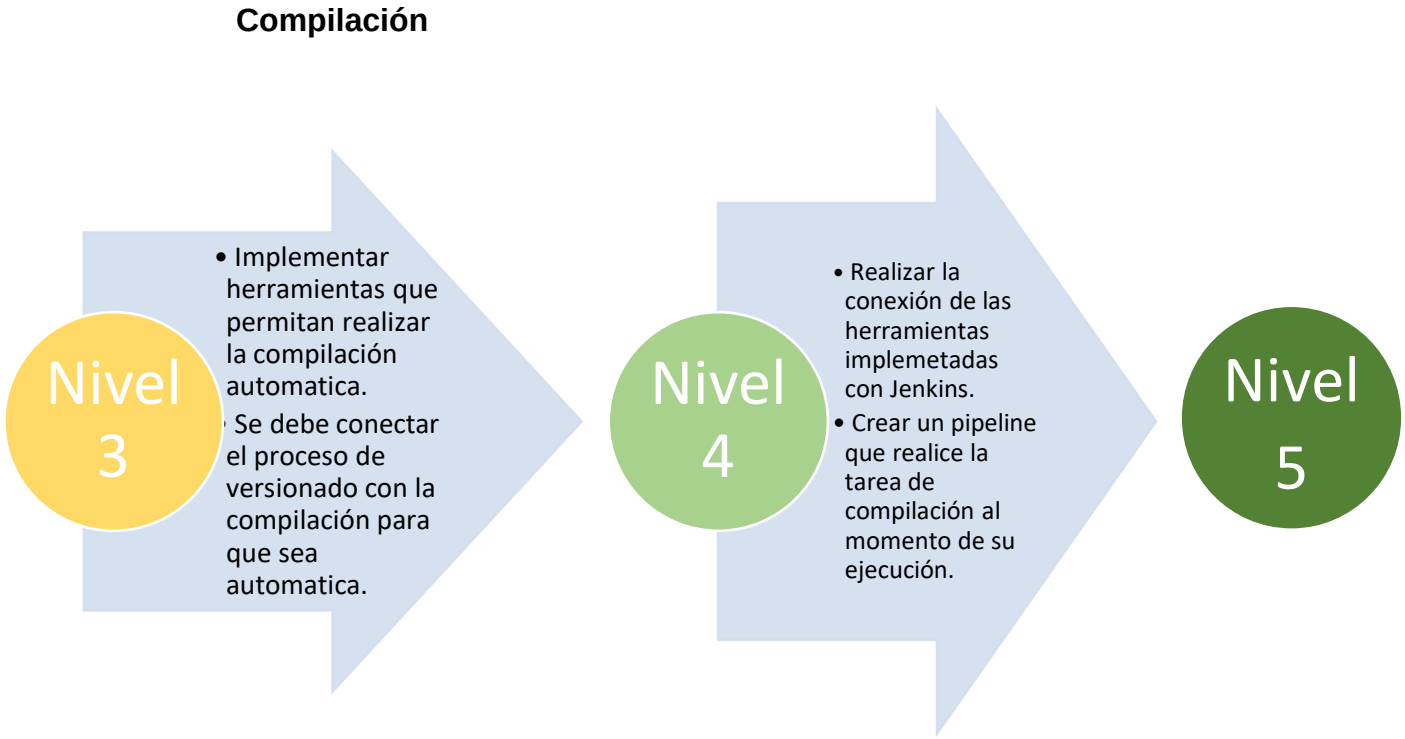

Figura 8: Pasos para conseguir los objetivos. Compilación. Fuente: Elaboración propia

Pruebas Unitarias

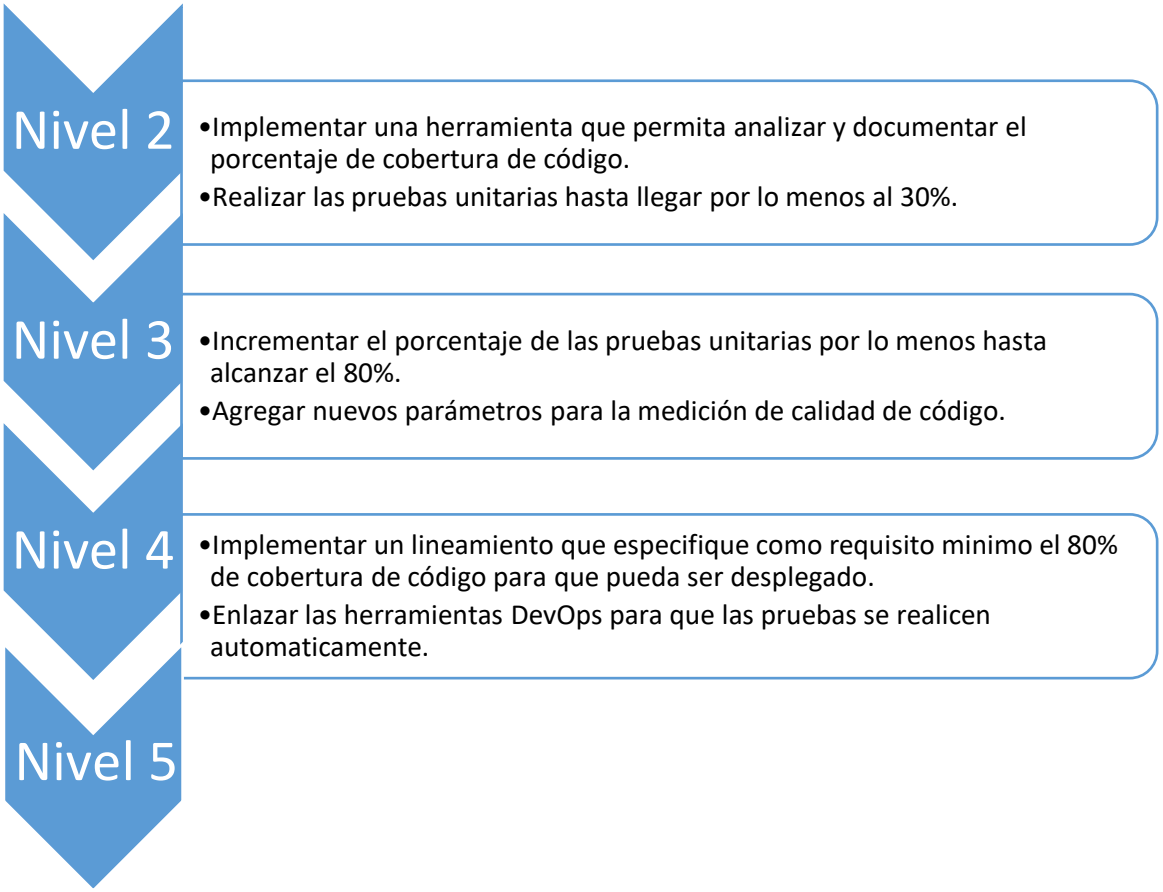

Figura 9: Pasos para conseguir los objetivos. Pruebas unitarias. Fuente: Elaboración propia.

Despliegues a servidores

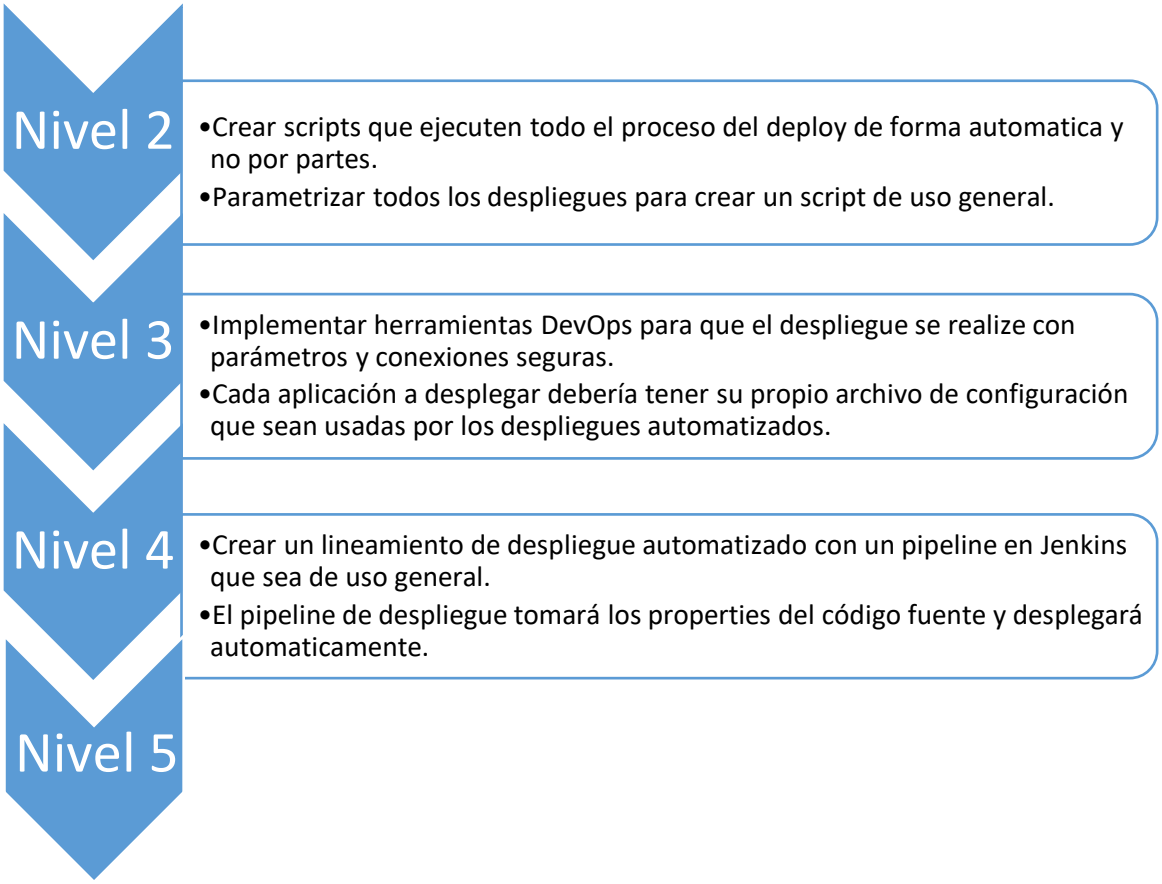

Figura 10: Pasos para conseguir los objetivos. Despliegue a servidores. Fuente: Elaboración propia.

3.2 Resultados

Luego del análisis efectuado al área de desarrollo de software de una empresa de telefonía, se generó una matriz de madurez y se pudo determinar el estado en el que se encuentra dicha área con respecto a buenas prácticas de DevOps.

Esta matriz ayudará a que los managers, líderes de equipo y desarrolladores tengan un rumbo definido con el objetivo de alcanzar las buenas prácticas de DevOps.

Se pudo rescatar el estado actual del área de desarrollo y se definieron los pasos que deberían seguir desde donde están hasta alcanzar el máximo nivel. Se debe tener en cuenta que este tipo de cambios y mejoras no es solo a nivel técnico sino que también se debe de llevar siempre presente la cultura DevOps.

Como resultado final se desarrolló un microservicio que servirá de ejemplo y demostración de que los anteriores estados de la matriz de madurez son alcanzables.

Se usó Github, Jenkins, SonarQube y docker para poder realizar esta demo.

Con respecto al versionamiento de código, todo el desarrollo fue subido a Github y cada cambio que se hizo sobre el código tuvo una pequeña descripción que señalaba que es lo que se hizo:

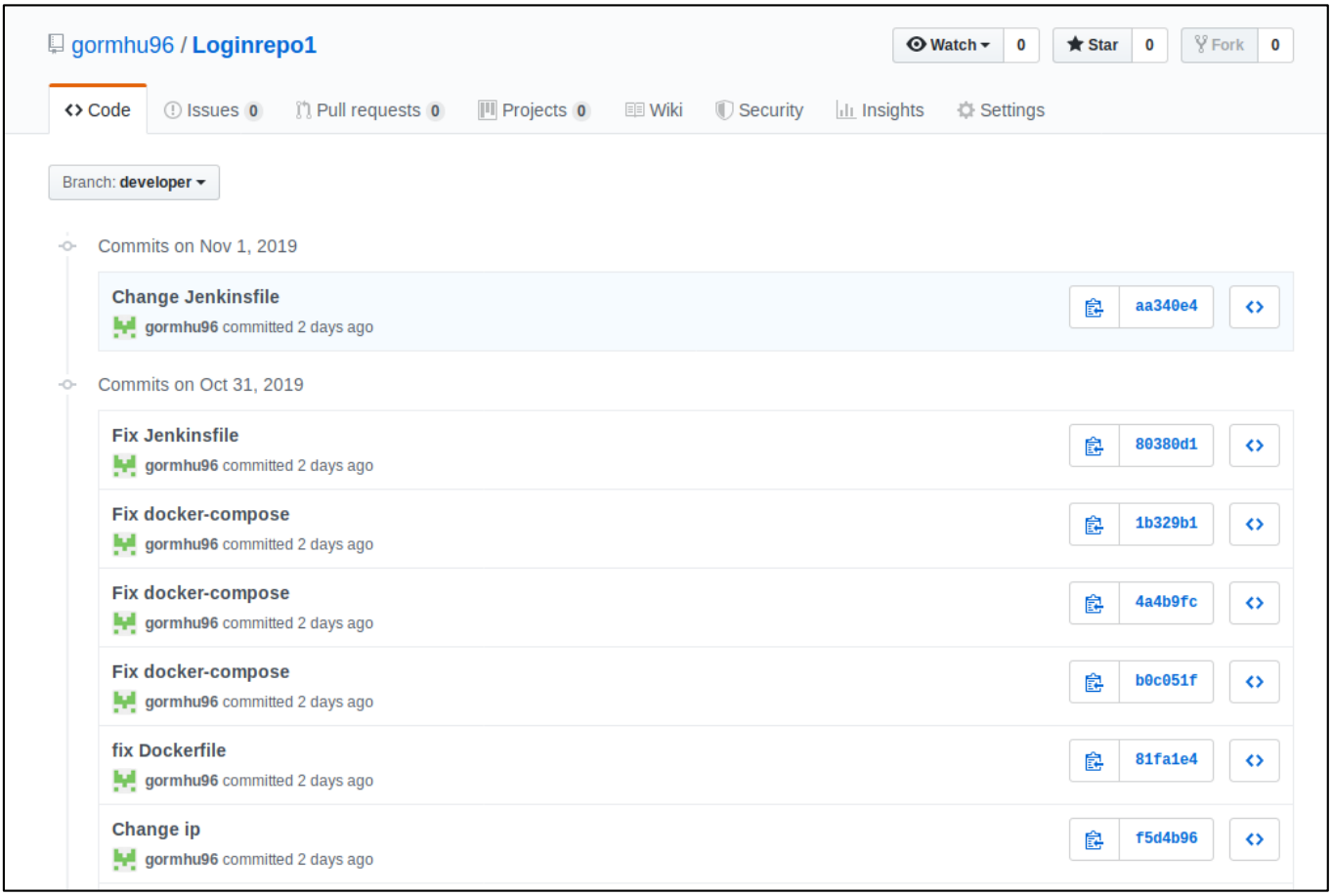

Figura 11. Versionado de código en Github. Fuente: Elaboración propia

Con respecto a la compilación, se usó Jenkins como orquestador de despliegues que incluye la compilación del código hecho en java.

En este caso es un microservicio de tipo java-maven que se conecta a una base de datos y se le lanza una petición por Curl enviándole un usuario y contraseña, si el usuario y contraseña están registrados en la base de datos, devolverá el mensaje “Usuario logueado” y sino devolverá un mensaje de error.

```
[Pipeline] sh
+ mvn clean install
[[1;34mINFO[m] Scanning for projects...
[[1;33mWARNING[m]
[[1;33mWARNING[m] Some problems were encountered while building the effective model for com.login.ws.rest:ApiLogin:jar:0.0.1-SNAPSHOT
[[1;33mWARNING[m] 'build.plugins.plugin.(groupId:artifactId)' must be unique but found duplicate declaration of plugin
org.sonarsource.scanner.maven:sonar-maven-plugin @ line 122, column 15
[[1;33mWARNING[m]
[[1;33mWARNING[m] It is highly recommended to fix these problems because they threaten the stability of your build.
[[1;33mWARNING[m]
[[1;33mWARNING[m] For this reason, future Maven versions might no longer support building such malformed projects.
[[1;33mWARNING[m]
[[1;34mINFO[m]
[[1;34mINFO[m] [1m----------------------< [0;36mcom.login.ws.rest:ApiLogin[0;1m >----------------------[m
[[1;34mINFO[m] [1mBuilding ApiLogin 0.0.1-SNAPSHOT[m
[[1;34mINFO[m] [1m--------------------------------[ jar ]---------------------------------[m
[[1;34mINFO[m]
[[1;34mINFO[m] [1m--- [0;32mmaven-clean-plugin:3.1.0:clean[m [1m(default-clean)[m @ [36mApiLogin[0;1m ---[m
[[1;34mINFO[m]
[[1;34mINFO[m] [1m--- [0;32mjacoco-maven-plugin:0.7.9:prepare-agent[m [1m(pre-unit-tests)[m @ [36mApiLogin[0;1m ---[m
[[1;34mINFO[m] argLine set to -javaagent:/var/lib/jenkins/.m2/repository/org/jacoco/org.jacoco.agent/0.7.9/org.jacoco.agent-0.7.9-
runtime.jar=destfile=/var/lib/jenkins/workspace/test-proyecto/${project.testresult.directory}/coverage/jacoco/jacoco.exec
[[1;34mINFO[m]
[[1;34mINFO[m] [1m--- [0;32mmaven-resources-plugin:3.1.0:resources[m [1m(default-resources)[m @ [36mApiLogin[0;1m ---[m
[[1;34mINFO[m] Using 'UTF-8' encoding to copy filtered resources.
[[1;34mINFO[m] Copying 1 resource
[[1;34mINFO[m] Copying 0 resource
[[1;34mINFO[m]
[[1;34mINFO[m] [1m--- [0;32mmaven-compiler-plugin:3.8.0:compile[m [1m(default-compile)[m @ [36mApiLogin[0;1m ---[m
[[1;34mINFO[m] Changes detected - recompiling the module!
[[1;34mINFO[m] Compiling 7 source files to /var/lib/jenkins/workspace/test-proyecto/target/classes
[[1;34mINFO[m]
[[1;34mINFO[m] [1m--- [0;32mmaven-resources-plugin:3.1.0:testResources[m [1m(default-testResources)[m @ [36mApiLogin[0;1m ---[m
[[1;34mINFO[m] Using 'UTF-8' encoding to copy filtered resources.
[[1;34mINFO[m] skip non existing resourceDirectory /var/lib/jenkins/workspace/test-proyecto/src/test/resources
[[1;34mINFO[m]
[[1;34mINFO[m] [1m--- [0;32mmaven-compiler-plugin:3.8.0:testCompile[m [1m(default-testCompile)[m @ [36mApiLogin[0;1m ---[m
[[1;34mINFO[m] Changes detected - recompiling the module!
[[1;34mINFO[m] Compiling 2 source files to /var/lib/jenkins/workspace/test-proyecto/target/test-classes
[[1;34mINFO[m]
[[1;34mINFO[m] [1m--- [0;32mmaven-surefire-plugin:2.22.1:test[m [1m(default-test)[m @ [36mApiLogin[0;1m ---[m
[[1;34mINFO[m]
[[1;34mINFO[m] -------------------------------------------------------
[[1;34mINFO[m]  T E S T S
[[1;34mINFO[m] -------------------------------------------------------
[[1;34mINFO[m] Running com.login.ws.rest.[1mApiLoginApplicationTests[m
00:37:36.289 [main] DEBUG org.springframework.test.context.junit4.SpringJUnit4ClassRunner - SpringJUnit4ClassRunner constructor called with
```

Figura 12: Compilación java-maven en Jenkins. Fuente: Elaboración propia.

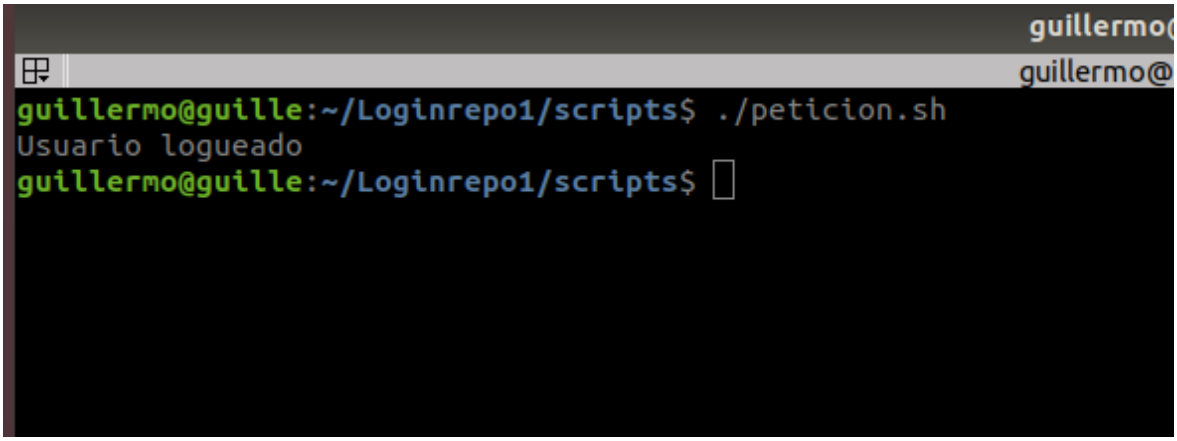

Figura 13: Enviando petición satisfactoria al microservicio de login. Fuente: Elaboración propia

Por el frente de pruebas unitarias y calidad de código se usó la herramienta SonarQube integrada a Jenkins, esta herramienta escanea el código estático y evalúa las pruebas unitarias dando como resultado un porcentaje de cobertura de pruebas unitarias. Además también indica con letras la calidad del código y si existen repeticiones de código.

```
[[1;34mINFO[m]  T E S T S
[[1;34mINFO[m] -------------------------------------------------------
[[1;34mINFO[m] Running com.login.ws.rest.[1mApiLoginApplicationTests[m
00:37:36.289 [main] DEBUG org.springframework.test.context.junit4.SpringJUnit4ClassRunner - SpringJUnit4ClassRunner constructor called with
[class com.login.ws.rest.ApiLoginApplicationTests]
00:37:36.310 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating CacheAwareContextLoaderDelegate from class
[org.springframework.test.context.cache.DefaultCacheAwareContextLoaderDelegate]
00:37:36.335 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating BootstrapContext using constructor [public
org.springframework.test.context.support.DefaultBootstrapContext(java.lang.Class,org.springframework.test.context.CacheAwareContextLoaderDelega
te)]
00:37:36.394 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating TestContextBootstrapper for test class
[com.login.ws.rest.ApiLoginApplicationTests] from class [org.springframework.boot.test.context.SpringBootTestContextBootstrapper]
00:37:36.430 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Neither @ContextConfiguration nor
@ContextHierarchy found for test class [com.login.ws.rest.ApiLoginApplicationTests], using SpringBootContextLoader
00:37:36.437 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default resource location for test
class [com.login.ws.rest.ApiLoginApplicationTests]: class path resource [com/login/ws/rest/ApiLoginApplicationTests-context.xml] does not exist
00:37:36.438 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default resource location for test
class [com.login.ws.rest.ApiLoginApplicationTests]: class path resource [com/login/ws/rest/ApiLoginApplicationTestsContext.groovy] does not
exist
00:37:36.439 [main] INFO org.springframework.test.context.support.AbstractContextLoader - Could not detect default resource locations for test
class [com.login.ws.rest.ApiLoginApplicationTests]: no resource found for suffixes {-context.xml, Context.groovy}.
00:37:36.441 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils - Could not detect default configuration
classes for test class [com.login.ws.rest.ApiLoginApplicationTests]: ApiLoginApplicationTests does not declare any static, non-private, non-
final, nested classes annotated with @Configuration.
00:37:36.546 [main] DEBUG org.springframework.test.context.support.ActiveProfilesUtils - Could not find an 'annotation declaring class' for
annotation type [org.springframework.test.context.ActiveProfiles] and class [com.login.ws.rest.ApiLoginApplicationTests]
00:37:36.715 [main] DEBUG org.springframework.context.annotation.ClassPathScanningCandidateComponentProvider - Identified candidate component
class: file [/var/lib/jenkins/workspace/test-proyecto/target/classes/com/login/ws/rest/ApiLoginApplication.class]
00:37:36.717 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Found @SpringBootConfiguration
com.login.ws.rest.ApiLoginApplication for test class com.login.ws.rest.ApiLoginApplicationTests
00:37:37.025 [main] DEBUG org.springframework.boot.test.context.SpringBootTestContextBootstrapper - @TestExecutionListeners is not present for
class [com.login.ws.rest.ApiLoginApplicationTests]: using defaults.
00:37:37.027 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Loaded default TestExecutionListener class
names from location [META-INF/spring.factories]: [org.springframework.boot.test.mock.mockito.MockitoTestExecutionListener,
org.springframework.boot.test.mock.mockito.ResetMocksTestExecutionListener,
org.springframework.boot.test.autoconfigure.restdocs.RestDocsTestExecutionListener,
org.springframework.boot.test.autoconfigure.web.client.MockRestServiceServerResetTestExecutionListener,
org.springframework.boot.test.autoconfigure.web.servlet.MockMvcPrintOnlyOnFailureTestExecutionListener,
org.springframework.boot.test.autoconfigure.web.servlet.WebDriverTestExecutionListener,
org.springframework.test.context.web.ServletTestExecutionListener,
org.springframework.test.context.support.DirtiesContextBeforeModesTestExecutionListener,
org.springframework.test.context.support.DependencyInjectionTestExecutionListener,
org.springframework.test.context.support.DirtiesContextTestExecutionListener,
org.springframework.test.context.transaction.TransactionalTestExecutionListener,
org.springframework.test.context.jdbc.SqlScriptsTestExecutionListener]
00:37:37.089 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Using TestExecutionListeners:
[org.springframework.test.context.web.ServletTestExecutionListener@68e5c7ae,
```

Figura 14: Ejecución del escaneo de SonarQube en Jenkins. Fuente: Elaboración propia

Por el lado de despliegue a servidores, el despliegue se realizó usando contenedores de docker, estos contenedores tienen todos los requisitos necesarios para lanzar las aplicaciones según se necesiten, en este caso usamos un contenedor MySQL y un contenedor para java que permita ejecutar el microservicio. Se hizo uso además de Dockerfiles para la creación de imágenes con las características y personalizaciones que necesitemos y también se usó docker-compose que permite gestionar los contenedores que se necesiten y levantar toda la aplicación con una línea de comandos. Todas las ejecuciones de despliegue se hicieron mediante Jenkins.

```
[Pipeline] stage
[Pipeline] { (Deploy Service)
[Pipeline] sh
+ docker-compose -f docker/docker-compose.yml up --build -d
Creating network "docker_default" with the default driver
Building db
Step 1/2 : FROM mysql
latest: Pulling from library/mysql
Digest: sha256:7345ce4ce6f0c1771d01fa333b8edb2c606ca59d385f69575f8e3e2ec6695eee
Status: Downloaded newer image for mysql:latest
 ---> c8ee894bd2bd
Step 2/2 : COPY conf/my.cnf /etc/mysql/my.cnf
 ---> Using cache
 ---> bdcb87150dee
Successfully built bdcb87150dee
Successfully tagged mysql-login:latest
Building api
Step 1/6 : FROM openjdk:8
8: Pulling from library/openjdk
Digest: sha256:08bf396d2e7e82b12d9c78d7e75137c1159c07f18f203391aa599adcb3643097
Status: Downloaded newer image for openjdk:8
 ---> 57c2c2d2643d
Step 2/6 : RUN apt-get update -y &&     apt-get -y install mysql-client &&     mkdir -p /data
 ---> Using cache
 ---> b701042b0339
Step 3/6 : COPY target/ApiLogin-0.0.1-SNAPSHOT.jar /data/ApiLogin-0.0.1-SNAPSHOT.jar
 ---> e6ed95c10a7d
Step 4/6 : COPY user-data/data.sql /data/data.sql
 ---> 0df2170b31c0
Step 5/6 : COPY docker/api/entrypoint.sh /data/entrypoint.sh
 ---> 79af1728901b
Step 6/6 : ENTRYPOINT ["/bin/bash","/data/entrypoint.sh"]
 ---> Running in 07b82f3fc821
Removing intermediate container 07b82f3fc821
 ---> 35da3616b6e8
Successfully built 35da3616b6e8
Successfully tagged docker_api:latest
Building proxy
Step 1/3 : FROM nginx
latest: Pulling from library/nginx
Digest: sha256:922c815aa4df050d4df476e92daed4231f466acc8ee90e0e774951b0fd7195a4
Status: Downloaded newer image for nginx:latest
 ---> 540a289bab6c
Step 2/3 : RUN mkdir -p /etc/ssl/certs && chmod 600 /etc/ssl/certs
 ---> Using cache
 ---> 44f2fa5eb7a2
Step 3/3 : COPY conf.d/api-login.conf /etc/nginx/conf.d/api-login.conf
```

Figura 15: Haciendo uso de docker-compose para despliegue y creación de imagenes docker. Fuente: Elaboración propia.

Todo ha sido ejecutado desde Jenkins, mediante un job que nos permite configurar el repositorio al que apuntamos, la rama, se pueden incluir parámetros externos, unificar otras herramientas como SonarQube y Git. Todo se separó por Stages o Etapas que nos permite segmentar nuestro despliegue y en caso ocurra un error podes hacer un debug más acertado y veloz.

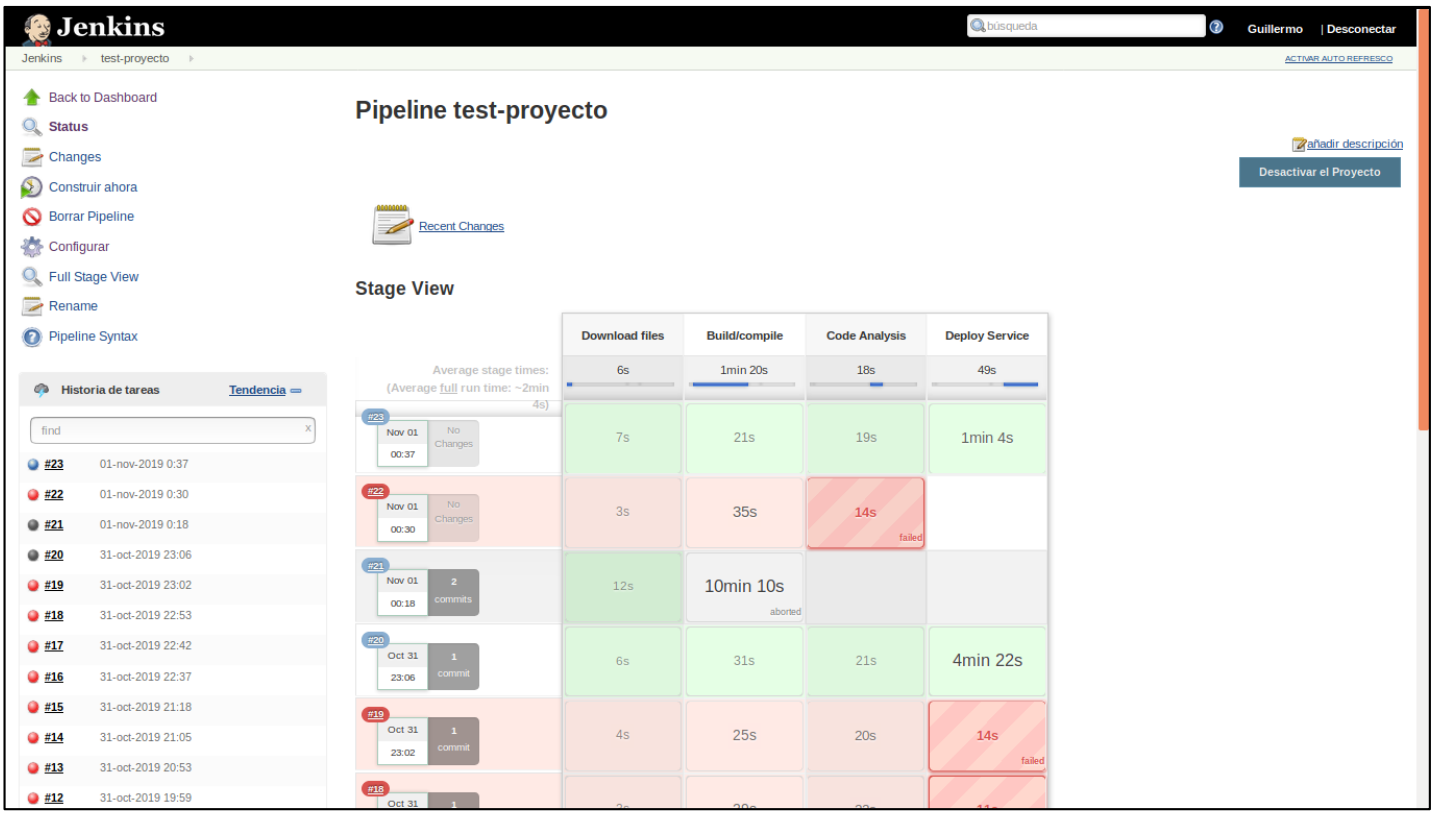

Figura 16: Ejecuciones realizadas en Jenkins. Fuente: Elaboración propia.

CONCLUSIONES

El objetivo general de este trabajo era generar un modelo para evaluar la madurez del área de desarrollo de software mediante el uso de buenas prácticas de DevOps para lo que se tuvo que recolectar información de los procesos actuales y luego se generó la matriz de madurez. Posteriormente se hizo la evaluación del nivel en el que se encontraban las prácticas tocadas en este trabajo.

Se realizó el análisis de los estados de las buenas prácticas en el área de desarrollo de la empresa de telefonía, esto nos permitió reconocer que las prácticas que ellos tenían no eran las mejores y que se podían implementar muchas mejoras paulatinamente.

Se pudo crear una matriz de madurez basándose en los niveles planteados por el CMMI, en esta matriz se pudo observar los diferentes niveles en los que se podía calificar al área de desarrollo de dicha empresa. Además se creó un tipo de guía de lo que tienen que hacer las personas si desean pasar al siguiente nivel.

Se elaboró un pipeline de despliegue usando Jenkins, SonarQube, docker y docker-compose que nos ayudó a ejemplificar de forma práctica que los estados más altos de la matriz de madurez son posibles de alcanzar.

Se concluye que si bien el área de desarrollo de la empresa de telefonía cuenta con prácticas de DevOps activas, no se realizan de la mejor manera posible por lo que se tendrá que seguir un plan para poder optimizar lo que existente.

RECOMENDACIONES

En el planteamiento de las soluciones para pasar de nivel se podría incluir algunas mejoras como la automatización de procesos repetitivos, procesos que van a ejecutarse una y otra vez y que no necesitan de la intervención de una persona.

A fin de asegurar que el código que vaya a ser desplegado en los servidores no sea interceptado en el trayecto, se podrían usar llaves SSH que garanticen una conexión segura entre el Jenkins y los servidores.

Puede crearse una relación entre el Jenkins y el Github que haga que cuando alguna persona miembro del equipo haga un push al repositorio, automáticamente se gatille el job y empiece a llevar el cambio realizado hasta los ambientes productivos, siempre pasando por los estados de calidad de código.

Para mantener un control del código fuente y las modificaciones que se le pueden realizar al mismo, se recomienda el uso de una estrategia de ramas como GitBranch que extrae de la rama principal (master) una rama de desarrollo (develop) y de esta última extrae las ramas de contribución llamadas ramas feature, todo esto con el fin de que no se trabaje directamente con la rama principal que debería ser modificada solo en los pases a producción.

Para tener trazabilidad sobre las versiones que se han llevado hasta los ambientes productivos, se puede usar un versionador de artefactos que pueda guardar los artefactos (compilados o empaquetados) de las versiones que se trabajan. Los versionadores como Artifactory o Nexus tienen un costo promedio pero son de mucha utilidad.

REFERENCIAS

Pereira & Rumiche (2017). Migración de los servicios de pagos en línea para soportar transacciones en el aplicativo móvil de una empresa de telecomunicaciones. Tesis de pregrado.

Belalcázar, A. (2017). Arquitectura de un data center con herramientas DevOps. Tesis de doctorado.

Pérez, L. (2018). DevOps: IT Development in the Era of Digitalization. Tesis de pregrado.

Farías, A. (2017). Definición de un ambiente de construcción de aplicaciones empresariales a través de Devops, Microservicios y Contenedores. Tesis de pregrado.

Beck et. Al (2001). Manifesto for Agile Software Development.

Duro, L. (2017). ¿Qué es la Transformación Digital y cuáles son las fases de la digitalización de una empresa?

Gallego, M. T. (2012). Metodologia scrum. Universitat Oberta de Catalunya

PMC Group (2018), Devops: Un vistazo al Futuro, p. 16-19.

Zhu, L., Bass, L., & Champlin-Scharff, G. (2016). DevOps and its practices. IEEE Software, 33(3), 32-34.

Pederiva, A. (2003). The COBIT® maturity model in a vendor evaluation case. Information Systems Control Journal, 3, 26-29.

ANEXOS

1. Modelo de Implementación del Flujo de DevOps desde Desarrollo hasta Despliegue.

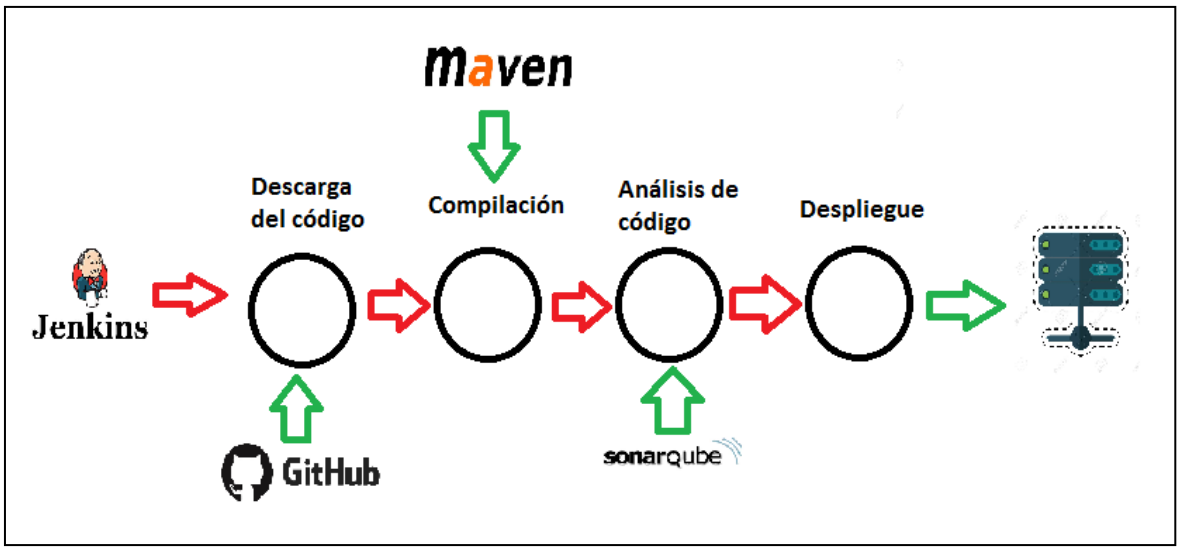

Figura 17: Flujo de despliegue con jenkins. Fuente: Elaboración propia

2. Jenkinsfile que realiza todas las etapas desde la descarga de código hasta despliegue.

```
 GNU nano 2.9.3                                Jenkinsfile

node
{
    cleanWs()
    def url_base = 'https://github.com/gormhu96/Loginrepo1';
    def sonar_url = 'http://localhost:9000';
    def sonar_token = 'bfd0736aef7524cbb3c80ef750f9525e4fe9a44c';
    def command_sonar = "mvn sonar:sonar -Dsonar.host.url=$sonar_url -Dsonar.login=$sonar_token";
    def command_compile = 'mvn clean install';
    def nexus_url = 'http://localhost:8081/repository/Loginrepo1/';
    def command_compose = "docker-compose -f docker/docker-compose.yml up --build -d";

        stage("Download files"){
        checkout scm
        }
        stage("Build/compile"){
        sh command_compile
        }
        stage("Code Analysis")
        {
        sh command_sonar
        }
        stage("Deploy Service"){
        sh command_compose
        }
}

                              [ 25 líneas leídas ]
^G Ver ayuda   ^O Guardar     ^W Buscar      ^K Cortar Texto^J Justificar  ^C Posición    M-U Deshacer   M-A Marcar text M-] A llave
^X Salir       ^R Leer fich.  ^\ Reemplazar  ^U Pegar txt   ^T Ortografía  ^_ Ir a línea  M-E Rehacer    M-6 Copiar txt  M-W Siguiente
```

Figura 18: Jenkinsfile usado para complicación, QA y despliegue. Fuente: Elaboración propia.

3. Docker-compose que realiza la creación y configuración de las imágenes de docker.

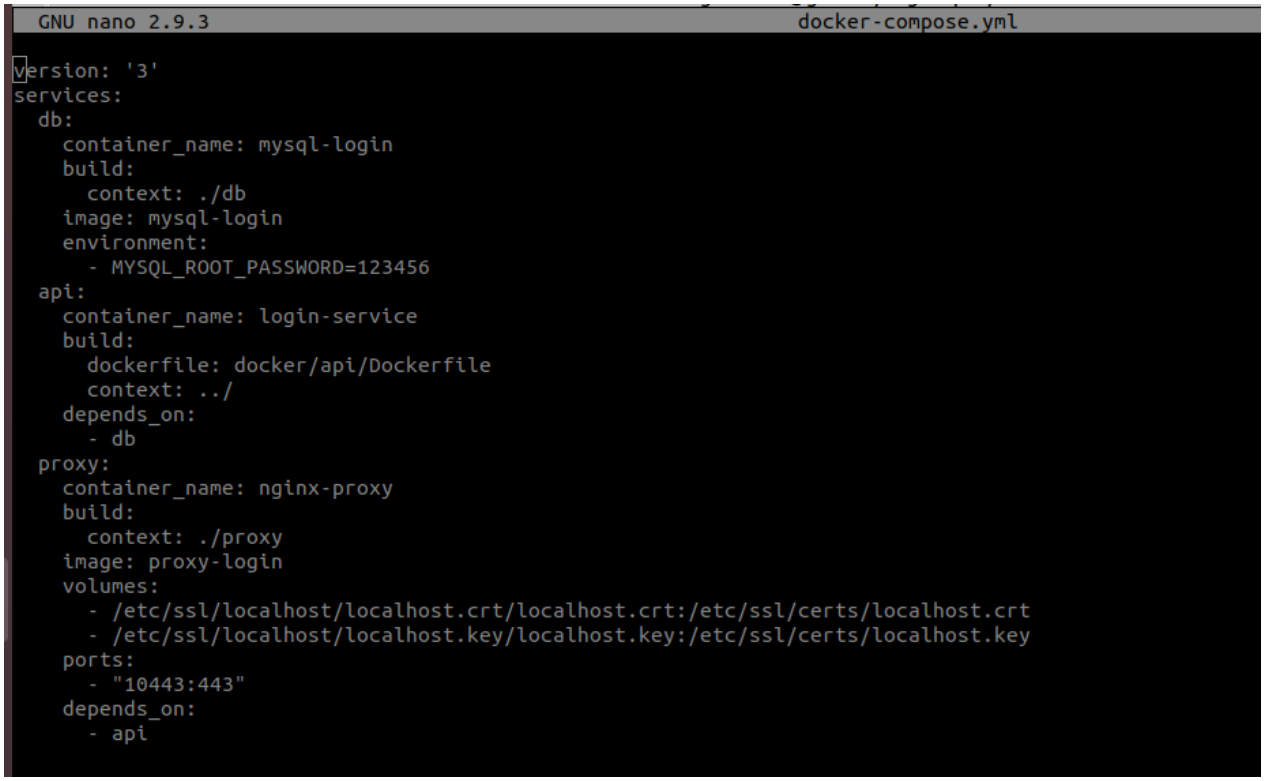

Figura 19. Docker compose usado para generación de imagenes de docker. Fuente: Elaboración propia

4. Petición con Curl usada para obtener respuesta de la API.

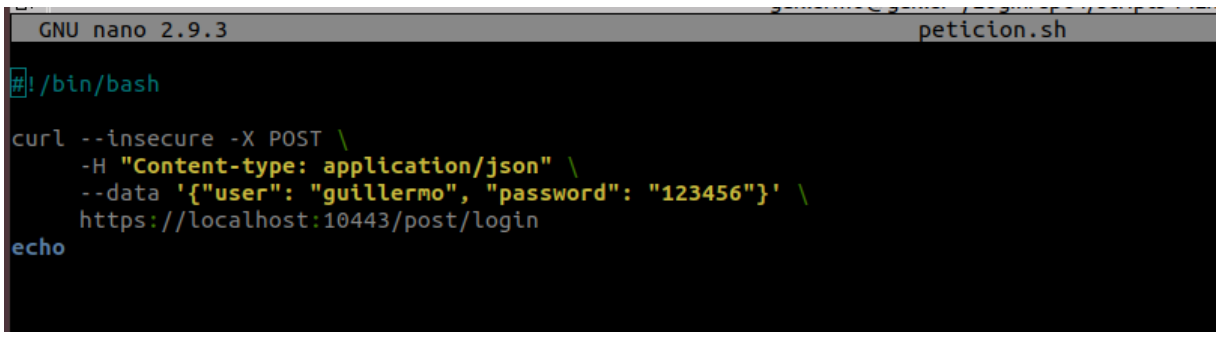

Figura 20: Petición en script usada para pegarle a la API. Fuente. Elaboración propia