Universidad Nacional de Loja

Facultad de la Energía, las Industrias y los de Recursos Naturales No Renovables

Maestría en Ingeniería en Software

Propuesta de un marco de trabajo metodológico de desarrollo de software para la Dirección de Tecnologías de Información de la Universidad Nacional de Loja

Trabajo de Titulación previo a la obtención del título de Magíster en Ingeniería en Software

AUTORES:

Danny Emanuel Muñoz Flores
Máximo Andrés Álvarez Pacheco

DIRECTOR:
Ing. Edison Leonardo Coronel Romero, Mg.Sc.

Loja - Ecuador
2023

Certificación

Loja, 30 de abril de 2023

Ing. Edison Leonardo Coronel Romero, Mg.Sc.
DIRECTOR DEL TRABAJO DE TITULACIÓN

CERTIFICO:

Que he revisado y orientado todo proceso de la elaboración del Trabajo de Titulación denominado: **Propuesta de un marco de trabajo metodológico de desarrollo de software para la Dirección de Tecnologías de Información de la Universidad Nacional de Loja**, previo a la obtención del título de **Magíster en Ingeniería en Software**, de autoría de los estudiantes **Danny Emanuel Muñoz Flores**, con **cédula de identidad Nro. 1104285604,** del estudiante **Máximo Andrés Álvarez Pacheco**, con **cédula de identidad Nro**. **1104128887**, una vez que el trabajo cumple con todos los requisitos exigidos por la Universidad Nacional de Loja para el efecto, autorizo la presentación para la respectiva sustentación y defensa.

Firmado electrónicamente por:
EDISON LEONARDO
CORONEL ROMERO

Ing. Edison Leonardo Coronel Romero, Mg.Sc.
DIRECTOR DEL TRABAJO DE TITULACIÓN

Autoría

Nosotros, **Danny Emanuel Muñoz Flores** y **Máximo Andrés Álvarez Pacheco**, declaramos ser los autores del Trabajo de Titulación y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos de posibles reclamos y acciones legales, por el contenido del mismo. Adicionalmente aceptamos y autorizamos a la Universidad Nacional de Loja la publicación del Trabajo de Titulación en el Repositorio Digital Institucional – Biblioteca Virtual.

Firma: 

Cédula de Identidad: 1104285604

Fecha: 03/05/2023

Correo electrónico: dmunoz@unl.edu.ec

Teléfono: 0994964490

Firma: 

Cédula de Identidad: 1104128887

Fecha: 03/05/2023

Correo electrónico: maalvarezp@unl.edu.ec

Teléfono: 0968413887

Carta de autorización por parte de los autores, para la consulta, reproducción parcial o total y/o publicación electrónica del texto completo, del Trabajo de Titulación.

Nosotros, **Danny Emanuel Muñoz Flores** y **Máximo Andrés Álvarez Pacheco**, declaramos ser autores del Trabajo de Titulación denominado: **Propuesta de un marco de trabajo metodológico de desarrollo de software para la Dirección de Tecnologías de Información de la Universidad Nacional de Loja** como requisito para optar el título de **Magíster en Ingeniería en Software**, autorizamos al sistema Bibliotecario de la Universidad Nacional de Loja para que con fines académicos muestre la producción intelectual de la Universidad, a través de la visibilidad de su contenido de la siguiente manera en el Repositorio Institucional.

Los usuarios pueden consultar el contenido de este trabajo en el Repositorio Institucional, en las redes de información del país y del exterior con las cuales tenga convenio la Universidad. La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia del Trabajo de Titulación que realice un tercero.

Para constancia de esta autorización, en la ciudad de Loja, a los tres días del mes de mayo de dos mil veintitrés.

Firma: Firmado electrónicamente por: DANNY EMANUEL MUNOZ FLORES
Autor: Danny Emanuel Muñoz Flores
Cédula de identidad: 1104285604
Dirección: Ecuador / Loja / Loja / Fleming y Hegel
Correo electrónico: dmunoz@unl.edu.ec
Teléfono: 0994964490

Firma: Firmado electrónicamente por: MAXIMO ANDRES ALVAREZ PACHECO
Autor: Máximo Andrés Álvarez Pacheco
Cédula de identidad: 1104285604
Dirección: Ecuador / Loja / Loja / Av. Chuquiribamba, Zalapa bajo
Correo electrónico: maalvarezp@unl.edu.ec
Teléfono: 0968413887

DATOS COMPLEMENTARIOS:

Director del Trabajo de Titulación: Ing. Edison Leonardo Coronel Romero, Mg.Sc.

Dedicatoria

El presente trabajo lo dedico a mi esposa e hijas por brindarme el apoyo y comprensión necesaria para afrontar cada uno de los retos y lograr con este objetivo; as mis padres, por ser los iniciadores de mi formación y alentarme a continuar superándome; a mis compañeros de trabajo, docentes, familiares y amigos que compartieron sus conocimientos, orientación y apoyo incondicional durante todo el proceso.

Danny Muñoz

A mi familia, por su amor incondicional, apoyo y motivación en cada etapa. A mi director del Trabajo de Titulación por su guía, paciencia y sabiduría en el desarrollo de esta investigación. A la institución, por brindarme la oportunidad de adquirir conocimientos y habilidades que me permitieron alcanzar este objetivo profesional. Sin su ayuda, este logro no habría sido posible.

Máximo Álvarez

Agradecimiento

De manera especial agradecemos a los docentes del programa de posgrado y tutores del Trabajo de Titulación que nos brindaron sus conocimientos y orientación durante todo el proceso de formación y ejecución de la investigación.

Así mismo, expresamos nuestro agradecimiento a la Dirección de Tecnologías de Información, por darnos la oportunidad de ejecutar la investigación y entregar la información necesaria de una manera profesional y oportuna.

Danny Muñoz

Máximo Álvarez

Índice de contenidos

Portada ..... **i**
Certificación ..... **ii**
Autoría ..... **iii**
Carta de autorización ..... **iv**
Dedicatoria ..... **v**
Agradecimiento ..... **vi**
Índice de contenidos ..... **vii**
Índice de tablas ..... **ix**
Índice de figuras ..... **x**
Índice de anexos ..... **xii**
1. Título **1**
2. Resumen **2**
2.1. Abstract 3
3. Introducción **4**
4. Marco teórico **6**
Ingeniería de Software 6
Origen de la Ingeniería de Software 6
Proceso de desarrollo de software 8
Priorización de requerimientos mediante el método MoSCoW 9
Herramientas para el proceso de desarrollo de software 10
Metodologías de Desarrollo de Software 15
Metodologías tradicionales 15
Metodologías ágiles 16
5. Metodología **27**
Área de estudio 27
Procedimiento 28
Procesamiento y análisis de datos 30
6. Resultados **32**
6.1. Análisis de la situación actual de la DTI 32
6.1.1. Observación directa 32
6.1.2. Encuestas aplicadas 35
6.1.3. Análisis de la normativa vigente 36

6.2. Análisis de metodologías ágiles. 44
6.3. Propuesta del marco de trabajo metodológico de desarrollo de software. 48
6.3.1. CONSIDERACIONES 48
6.3.2. ROLES 49
6.3.3. CICLO DE VIDA 50
6.3.4. MEJORES PRÁCTICAS 58
6.3.5. ARTEFACTOS 58
6.3.6. HERRAMIENTAS Y PIPE LINE DEVOPS 60
7. Discusión 63
8. Conclusiones 66
9. Recomendaciones 67
10. Bibliografía 68
11. Anexos 72

Índice de tablas:

TABLA I. VALORES Y PRINCIPIOS DE XP....18
TABLA II. OBSERVACIÓN DIRECTA – PROCESOS FUNDAMENTALES....32
TABLA III .OBSERVACIÓN DIRECTA - METODOLOGÍAS....34
TABLA IV .OBSERVACIÓN DIRECTA - ARTEFACTOS....34
TABLA V. OBSERVACIÓN DIRECTA - ROLES....35
TABLA VI. PERSONAL ENCUESTADO....35
TABLA VII. ANÁLISIS DE NORMATIVA VIGENTE....36
TABLA VIII. CARACTERÍSTICAS COMUNES DE XP, SCRUM, KANBAN, LEAN Y DEVOPS....45
TABLA IX. COMPARATIVA DE ROLES XP, SCRUM, KANBAN, LEAN Y DEVOPS.45
TABLA X. COMPARATIVA DE PROCESOS XP, SCRUM, KANBAN, LEAN Y DEVOPS....46
TABLA XI. MEJORES PRÁCTICAS XP, SCRUM, KANBAN, LEAN Y DEVOPS....46
TABLA XII. ARTEFACTOS XP, SCRUM, KANBAN, LEAN Y DEVOPS....47
TABLA XIII. PROPUESTA DE ROLES....49
TABLA XIV. PROPUESTA DE CICLO DE VIDA....50
T**ABLA XV.** PROPUESTA DE MEJORES PRÁCTICAS....58
TABLA XVI. PROPUESTA DE ARTEFACTOS....58
TABLA XVII .PROPUESTA DE HERRAMIENTAS....61

Índice de figuras:

Fig. 1. Ciclo de vida para la gestión de proyectos 51
Fig. 2. Fase Inicio - Ciclo de vida para la gestión de proyectos 52
Fig. 3. Fase Exploración - Ciclo de vida para la gestión de proyectos 53
Fig. 4. Fase Implementación - Ciclo de vida para la gestión de proyectos 54
Fig. 5. Fase Revisión y retrospectiva - Ciclo de vida para la gestión de proyectos 55
Fig. 6. Fase Lanzamiento - Ciclo de vida para la gestión de proyectos 56
Fig. 7. Fase Mantenimiento - Ciclo de vida para la gestión de proyectos 57
Fig. 8. Propuesta de pipe line DevOps (Autoría propia) 62
Fig. 9. Resultados encuesta - pruebas unitarias 81
Fig. 10. Resultados encuesta - herramientas analizar calidad código 81
Fig. 11. Resultados encuesta - formatos de control de calidad 82
Fig. 12. Resultados encuesta – otros formatos de control calidad 82
Fig. 13. Resultados encuesta - herramientas gestión actividades 83
Fig. 14. Resultados encuesta - DevOps 83
Fig. 15. Resultados encuesta - nivel de conocimiento DevOps 84
Fig. 16. Resultados encuesta - lenguajes de programación 84
Fig. 17. Resultados encuesta - nivel conocimiento lenguaje de programación 85
Fig. 18. Resultados encuesta – conocimiento de la metodología XP 86
Fig. 19. Resultados encuesta - nivel de conocimiento de la metodología XP 86
Fig. 20. Resultados encuesta - conocimiento de SCRUM 87
Fig. 21. Resultados encuesta - nivel de conocimiento de SCRUM 87
Fig. 22. Resultados encuesta - conocimiento de KANBAN 88
Fig. 23. Resultados encuesta - nivel de conocimiento de KANBAN 88
Fig. 24. Resultados encuesta - conocimiento de DEVOPS 89
Fig. 25. Resultados encuesta - nivel de conocimiento de DEVOPS 89
Fig. 26. Resultados encuesta - uso de IDE principal 90
Fig. 27. Resultados encuesta - uso de IDE secundario 90
Fig. 28. Resultados encuesta - conocimiento Guía Fundamentos para la Dirección de Proyectos 91
Fig. 29. Resultados encuesta - nivel de conocimiento terminal Linux 91
Fig. 30. Resultados encuesta - herramientas organizar tareas/actividades 92
Fig. 31. Resultados encuesta - frecuencia de uso de contenedores 92
Fig. 32. Resultados encuesta - lenguaje de programación 93
Fig. 33. Resultados encuesta – primer proyecto de mayor dedicación 93
Fig. 34. Resultados encuesta - segundo proyecto de mayor dedicación 94
Fig. 35. Resultados encuesta - tercer proyecto de mayor dedicación 94
Fig. 36. Resultados encuesta - herramientas automatizadas para paso a producción 95
Fig. 37. Resultados encuesta - herramientas automatizadas de paso a producción 95
Fig. 38. Resultados encuesta - pruebas unitarias 96
Fig. 39. Resultados encuesta - herramientas calidad de código 96
Fig. 40. Resultados encuesta - formatos desarrollo de software 97
Fig. 41. Resultados encuesta - otros formatos desarrollo de software 97

Fig. 42. Resultados encuesta - nivel de experiencia en seguridad de aplicaciones web.....98
Fig. 43. Resultados encuesta - herramientas empleadas en pruebas de seguridad 98
Fig. 44. Resultados encuesta - forma de elaborar pruebas de aplicaciones web 99
Fig. 45. Resultados encuesta - desafíos al evaluar la seguridad de aplicaciones web 100
Fig. 46. Resultados encuesta - métodos para mantener la seguridad de aplicaciones web ...100
Fig. 47. Resultados encuesta – manejo de problemas de seguridad de aplicaciones web.....101
Fig. 48. Resultados encuesta - aseguramiento compatibilidad con los estándares de seguridad 101

Índice de anexos:

Anexo 1. Encuesta - Perfil Control de Calidad....72
Anexo 2. Encuesta – Perfil Desarrollador Senior/Junior....74
Anexo 3. Encuesta – Perfil Seguridad de la Información....79
Anexo 4. Análisis de Resultados de Encuesta – Perfil Control de Calidad....81
Anexo 5. Análisis de Resultados de Encuesta – Perfil Desarrollador Senior / Junior....86
Anexo 6. Análisis de Resultados de Encuesta – Perfil Seguridad de la Información 98
Anexo 7. Formato 001 - Requerimiento de Software....103
Anexo 8. Formato 002 - Caso de Negocio 104
Anexo 9. Formato 003 – Estudio de Factibilidad....105
Anexo 10. Formato 004 - Lista de Stakeholders 105
Anexo 11. Formato 005 - Acta de Constitución del Proyecto....106
Anexo 12. Formato 006 – Épicas, Historias de Usuario, Backlog del producto, Backlog del Sprint....106
Anexo 13. Formato 007 - Plan de Lanzamiento....107
Anexo 14. Formato 008 - Diccionario de Datos....108
Anexo 15. Formato 009 - Plan de Pruebas 108
Anexo 16. Formato 010 - Casos de Prueba 109
Anexo 17. Formato 011 - Lista de Incidentes....109
Anexo 18. Formato 012 – Informe de Pruebas y Certificación de Control de Calidad....110
Anexo 19. Formato 013 – Acta de Reunión 111
Anexo 20. Formato 014 – Acta de Paso a Producción 112
Anexo 21. Guía de desarrollo de software....113
Anexo 22. Propuesta presentada a la DTI 114
Anexo 23. Certificado de traducción del resumen....115

1. Título

Propuesta de un marco de trabajo metodológico de desarrollo de software para la Dirección de Tecnologías de Información de la Universidad Nacional de Loja.

2. Resumen

El presente Trabajo de Titulación (TT), consiste en proponer un marco de trabajo metodológico de desarrollo de software en la Dirección de Tecnologías de Información de la Universidad Nacional de Loja DTI-UNL, se inicia el análisis en base al proceso de adaptación de las metodologías ágiles de desarrollo de software, marcos de trabajo y enfoques más relevantes en la actualidad, pues para un desarrollar software es fundamental que el equipo de trabajo seleccione la metodología o adapte una o varias metodologías a la realidad organizacional, en dicho proceso se debe considerar el tamaño del proyecto, la organización, el tiempo y demás recursos.

Por ello, el presente TT realiza un análisis de la situación actual, es decir: las etapas, metodologías, artefactos y herramientas; que son empleadas por los involucrados en el proceso de desarrollar software en la DTI, seguidamente se analiza las metodologías ágiles de desarrollo de software y marcos de trabajo, que se enfocan en la entrega de valor al cliente en el menor tiempo posible, con el objetivo de identificar los roles, procesos, mejores prácticas y entregables que sean comunes entre sí y que se puedan emplear en la propuesta, una vez recolectados estos datos se presenta la propuesta metodológica, con aquellos componentes que incrementen la calidad y productividad del equipo que interviene en el proceso de desarrollo de software de la DTI.

En el desarrollo de este TT se usó la metodología denominada "investigación cualitativa", debido a que es un método científico de observación, su conjunto de técnicas permitió obtener una visión general del problema, generar ideas y/o suposiciones y finalmente identificar y plantear las soluciones específicas.

El principal resultado al seguir esta metodología es la propuesta del marco de trabajo metodológico, en el que se identifica roles, ciclo de vida, prácticas, artefactos y herramientas acorde a la realidad organizacional de la DTI, de tal manera que la aplicación de la propuesta permita estandarizar y agilizar el proceso de desarrollo de software.

Palabras clave: *Gestión de proyectos de software, KANBAN, LEAN, Metodologías ágiles, SCRUM, XP.*

2.1. Abstract

The present Degree Work (DW) consists of proposing a methodological framework of software development in the Information Technologies Direction of the Universidad Nacional de Loja ITD-UNL, the analysis is started based on the adaptation process of agile methodologies of software development, framework and approaches more relevant currently, since to develop software is fundamental that work team selects the methodology or adjusts to one or several methodologies to the organizational reality, in this process it must be considered the size of the project, the organization, time and other resources.

Thus, the present DW carries out an analysis of the current situation, that is to say: the stages, methodologies, devices and tools, which are used by those involved in the process of developing software in the ITD, after the agile methodologies of software development are analyzed and framework that focuses on delivering value to the customer in the shortest possible time, with the aim of identifying the roles, process, deliverable and best practices that will be common to each other and they could be used in the proposal, once the data is collected the methodological proposal is showed, with those components that increase the quality and productivity of team that participates on software development process of the ITD.

The methodology called "qualitative research" was used in the development of this DW, because it is a scientific method of observation, its set of techniques allowed to get an overview of the problem, to generate ideas and/or assumptions and finally to identify and to pose the specific solutions.

The main result by following this methodology is the proposal of the methodological framework, in which roles, lifecycles, practices, devices and tools are identified according to the organizational reality of the ITD, in such a way that the application of the proposal allows to standardize and to speed up the process of software development.

Keywords: *Agile methodologies, KANBAN, LEAN, SCRUM, Software project management, XP.*

3. Introducción

Una metodología de desarrollo de software es un conjunto de etapas o fases que se emplean para diseñar soluciones de software. Su principal objetivo es el de estandarizar u organizar a los equipos de desarrollo de software y sus resultados, para lograr así entregar productos con mayor calidad [1].

Las metodologías de desarrollo de software, en la actualidad son una base indispensable y necesaria para la elaboración de un proyecto de software que desee ofrecer un producto profesional. Contar con este conocimiento mejorará la calidad de las etapas que involucran el desarrollo de software (análisis, diseño, desarrollo e implantación), por ello se dedica gran cantidad de tiempo para identificar la metodología a emplearse en un nuevo proyecto con la finalidad de lograr un mejor resultado [2].

Podemos dividir a las metodologías de desarrollo de software entre tradicionales y ágiles, donde las tradicionales "buscan imponer disciplina al proceso de desarrollo de software y de esa forma volverlo predecible y eficiente" [3] , mientras que por otro lado las metodologías ágiles no son rígidas sino adaptativas y orientadas a las personas, no a los procesos [3].

El presente Trabajo de Titulación (TT) pretende presentar una alternativa de metodología y herramientas para emplearse en el desarrollo de software, acoplado a la realidad de la Dirección de Tecnologías de Información (DTI) de la Universidad Nacional de Loja, contribuyendo de esta manera a que los productos se gestionen y se lancen a producción con un nivel de calidad más alto al que ya existe en dicha dependencia.

Para guiar el presente TT se definieron tres objetivos específicos: el primero es "Analizar la situación actual de la Dirección de Tecnologías de Información, a través una revisión sistemática y trabajo de campo, para conocer cómo se está ejecutando el proceso de desarrollo de software.", el segundo es "Analizar las metodologías de desarrollo software, marcos de trabajo y enfoques más relevantes: XP, SCRUM, KANBAN, LEAN y DevOps; mediante una revisión bibliográfica; para identificar las mejores prácticas, procesos y entregables. " y el tercero es "Realizar la propuesta del marco metodológico de desarrollo de software, acorde a la normativa vigente y a las mejores prácticas de las metodologías, marcos de trabajo y enfoques analizados; en base a los resultados del análisis elaborado; para estandarizar los procesos de desarrollo de software"; estos permitieron cumplir con el objetivo general, el mismo que es "Proponer un marco de trabajo metodológico de desarrollo de software para la

Dirección de Tecnologías de Información de la Universidad Nacional de Loja", cuyo propósito y meta es de dar respuesta a la interrogante: "¿Con la propuesta de un marco de trabajo metodológico de desarrollo de software, se permitirá ejecutar, estandarizar y documentar de manera adecuada el proceso de desarrollo de software?". Como parte de este trabajo de titulación se tuvo la colaboración del equipo de la Dirección de Tecnologías de Información, y que mediante el empleo de encuestas se pudo contar con su retroalimentación para elaborar una propuesta eficaz y sobre todo acoplada al trabajo que realiza todo el equipo involucrado en la construcción de software (programadores, control de calidad y responsable de seguridad de la información).

El presente TT se lo ha delimitado a la elaboración y presentación de una propuesta metodológica adaptada al equipo de desarrollo de software para la Dirección de Tecnologías de Información, es por esta razón y que para poder dar cumplimiento al primer objetivo específico se aplicó un conjunto de encuestas al equipo que interviene durante el proceso de desarrollo de software (programadores, control de calidad y responsable de seguridad de la información), lo que permitió conocer los procesos, métodos y artefactos que son utilizados al desarrollar software en la dependencia. Continuando con el segundo objetivo específico, nos permitió recopilar fuentes bibliográficas, procesos, artefactos y buenas prácticas de las diferentes metodologías de desarrollo de software que se aplican actualmente en las diferentes etapas del desarrollo de software, así mismo realizar una comparativa entre las metodologías ágiles para establecer las mejores alternativas que se puedan aplicar al equipo donde se aplicaría nuestra propuesta metodológica. Finalmente, para abordar el tercer objetivo y así cumplir el objetivo general se relacionó los resultados obtenidos del primer y segundo objetivos y así poder armar la propuesta metodológica que incluye las siguientes secciones: Introducción, Consideraciones, Roles, Ciclo de vida, Mejores prácticas, Artefactos y Herramientas y Pipe Line DevOps.

4. Marco teórico

En esta sección se realizó una recopilación de fuentes bibliográficas, que nos permitieron indicar una base teórica de la ingeniería de software, sus orígenes, etapas que se dan en un proyecto de software, herramientas que facilitan este proceso, un listado de las metodologías tradicionales y un desglose de las metodologías ágiles, sus roles, procesos y artefactos, que en su conjunto permitieron fundamentar el presente trabajo de titulación.

Ingeniería de Software

La ingeniería del software es una disciplina que integra procesos, métodos, herramientas y técnicas para generar un producto de calidad, para lo cual se debe ejecutar un conjunto de fases generales como: análisis, diseño, codificación, pruebas y mantenimiento; de acuerdo como se ejecuten dichas actividades existen diferentes modelos: cascada, incremental y evolutivo [4]. Para la producción de software, surge la necesidad de lograr enfoques disciplinados, nuevos métodos y herramientas para desarrollar, desplegar y evaluar los sistemas; así mismo se deben considerar las particularidades del nuevo medio, el contexto, los escenarios operativos y principalmente la diversidad de perfiles de usuarios, siendo desafíos adicionales al desarrollo de sistemas [5].

Origen de la Ingeniería de Software

El término usado por primera vez se sugiere en varias ocasiones, una de ellas menciona a la edición de junio de 1965 de la revista 'Computers and Automation' en un anuncio de trabajo buscando un "ingeniero de sistemas de software", otra sugerencia menciona a Anthony Oettinger, usando el término en 1966 para hacer referencia a la distinción entre ciencias de la computación y el desarrollo de sistemas intensivos de software, también se sugiere a Friedrich Bauer en la conferencia de ingeniería de software de la OTAN en 1968, sin embargo, Grady Booch sostiene que Margaret Hamilton usó por primera vez el término 'ingeniería de software' para distinguir su trabajo de la ingeniería de hardware mientras trabajaba en un programa espacial, el famoso proyecto Apollo y el Skylab, entre 1963 y 1964 [6].

La primera metodología de la computación se desarrolla en 1940 por el astrónomo estadounidense Wallace Eckert, en los años sesenta y ochenta se introduce la programación modular junto con conceptos de acoplamiento y cohesión como mecanismos de composición de algoritmos, Edsger Dijkstra dio la idea de la programación estructurada, tiempo después Ole

Dahl y Kristen Nygaard, crearían Simula, el primer lenguaje de programación orientada a objetos. Las ideas de Barbara Liskov sobre tipos Abstractos de Datos y el modelado de entidad-relación por Peter Chen y muchos más conceptos, aportarían en esta época, y con ello se generaron las primeras metodologías de ingeniería de software, por ejemplo, Winston Royce desarrollaría el famoso proceso en cascada [6].

En los años ochenta y más adelante, en la denominada edad de oro, debido a los crecientes problemas de calidad del software, el auge de los sistemas intensivos de software, la globalización del software y el desarrollo de sistemas distribuidos, nuevos enfoques fueron necesarios. Es aquí cuando las ideas de Ole Dahl y Kristen sobre programación orientado a objetos (OO) dio lugar a lenguajes de programación como Smalltalk, C with Classes (posteriormente renombrado a C++), Ada, entre otros [6].

Los métodos de análisis y diseño estructurado se adaptaron al enfoque OO, además de nuevas ideas de diseñar OO, como los de Rebecca Wirfs-Brock sobre Responsability Driven Design, Stephen Mellor sobre Modelos de Dominio Conceptuales, y las notaciones para modelar objetos como el de Booch, Objectory, OMT, estos últimos se unirían para formar UML [6].

En este contexto nacerían entre otras, el modelo de vistas 4+1 por Philippe Kruchten, modelo espiral del desarrollo de software y muchos aportes a la economía de desarrollar software por Barry Boehm, Métricas de software por Capers Jones, modelo formal clean-room por Harlan Mills, la programación literaria por Donald Knuth, modelo de madurez de capacidades (CMM) por Watts Humphrey, modelos de component-based engineering [6].

Los años noventa y el boom de Internet había llegado al mundo de los negocios, todo pasó a ser distribuido, acceso de los sistemas desde cualquier parte del mundo y conexión externa con otros sistemas, dando paso a la Integración Continua y el desarrollo, otro cambio fue la llegada de los dispositivos móviles, los servicios web, microservicios e infraestructuras web, apareciendo a su vez gigantes como Amazon, Google, Microsoft, IBM, Facebook, y claro nuevos lenguajes Javascript, Python, Swift, Rust. En esta época aparecen también los métodos ágiles, aunque en 1986, Hirotaka Takeuchi y Ikujiro Nonaka acuñaran el término “Scrum” como un enfoque de desarrollo de productos, Ken Schwaber y Jeff Sutherland lo adaptarían al mundo del software, Kent Beck, casi en la misma época lanzaría eXtreme Programming y en sí muchas nuevos métodos que se unirían bajo un mismo término paraguas "Agile" en Febrero del 2001 [6].

Una nueva época dorada en la Ingeniería de software apareció, nacieron Git y GitHub, Joel Spolsky creo Stack Overflow; Andrew Shafer y Patrick Debois el término DevOps; el mundo del Internet de las Cosas sigue emergiendo. Y las plataformas de e-learning se expandieron [6].

Actualmente la IA que ha existido décadas, ha sido posible su uso por la aparición del Big Data. Y ya hemos visto grandes hitos, como industria recién estamos viendo como impactará la IA a este campo, agregando a estos problemas, los cambios que traerán las nuevas tecnologías como la computación cuántica, la Realidad Aumentada, el Internet de las Cosas.

La ingeniería de software ha crecido y cambiado mucho, sin embargo, los fundamentos permanecen: desarrollar abstracciones (de Lenguaje máquina a alto nivel, de subrutinas a componentes y de componentes a servicios); separar intereses; distribuir responsabilidades (sea en Sistemas Operativos, en la gestión de proyectos o al diseñar software); buscar simplicidad [6].

Proceso de desarrollo de software

Como parte de todas las metodologías de desarrollo de software existen, ciertas etapas o fases que se ejecutan para poder cumplir con la meta del proyecto, de forma general podemos clasificar estas de la siguiente forma [7]:

- Análisis: Etapa en la que nace y formular un proyecto, en esta etapa se define los requisitos y el alcance del proyecto.
- Diseño: En esta etapa se prioriza el prototipado del proyecto, donde se pretende plasmas en pantallas las funcionalidades e interoperabilidad de este.
- Codificación: En esta etapa interviene el equipo de programación, quién se encarga de escribir el código que permita dar cumplimiento a los requisitos definidos en la primera etapa.
- Pruebas: Como parte de todo proyecto de software, se pasa por una etapa de validación, donde se verifica el cumplimiento, las buenas prácticas y seguridad de la solución presentada como producto final.
- Mantenimiento: Posterior a su entrega se considera que exista una etapa de mantenimiento en la que se pueda solventar posibles fallos o también perfeccionar funcionalidades implementadas.

Priorización de requerimientos mediante el método MoSCoW

Las reglas de MoSCoW son un método para la priorización de requerimientos, este acrónimo fue definido por D. Clegg and R. Baker[8], quienes en 1994 clasificaron a los requisitos en cuatro categorías [9]:

- **M**ust Have (Debe tener): Se incluyen en esta categoría aquellos requisitos mínimos que requieren el proyecto para garantizar su entrega o que imprescindibles.
- **S**hould Have (Debería tener): Son aquellos requisitos importantes, pero no vitales, que son relevantes para el proyecto pero que sin ellos el proyecto aún sigue siendo viable. Una forma de diferenciar estos de los Must Have, es revisando el grande de afectación que puede causar el no cumplimiento del requisito, midiendo esto en costo comercial o por número de personas afectadas.
- **C**ould Have (Podría tener): Son aquellos requisitos que se desean tener pero que son menos importantes, dentro del proyecto estos requisitos actúan como un plan de contingencia, si el tiempo de entrega del proyecto está en riesgo, los requisitos de esta categoría proporcionan las primeras opciones para eliminar requisitos y mitigar este riesgo.
- **W**on't Have this time (No tendrá en este momento): Estos requisitos son aquellos que el equipo del proyecto ha acordado que no se cumplirán, en esta primera iteración, y que permitirán aclarar el alcance del proyecto, reduciendo así el riesgo de que estos sean introducidos durante la ejecución del proyecto.

Las primeras 3 categorías se asigna un porcentaje del presupuesto de desarrollo, 60, 20 y 20 respectivamente, y se asignan los requisitos de acuerdo con la visión del Product Owner. La formulación actual de estas reglas de priorización se encuentra documentada en el DSDM Agile Project Framework [10].

Herramientas para el proceso de desarrollo de software

A continuación, se enlistan aquellas herramientas que comúnmente se utilizan durante el proceso de desarrollo de software, y que se acoplaron a la propuesta presentada:

Taiga

Es una herramienta de código libre, que facilita la gestión de proyectos ejecutados mediante Kanban y scrum. Nace en el 2013 con la finalidad de crear una herramienta intuitiva que facilitara el manejo de metodologías de desarrollo ágil, esta herramienta cuenta con las siguientes funcionalidades [11]:

- Tablero Kanban: para la gestión de tareas, flujos de trabajo del equipo de desarrollo.
- SCRUM: permite la planificación de backlog y sprints, así como la asignación de los responsables de las actividades.
- Peticiones: permite el registro, priorización y seguimiento de errores encontrados.
- Informes: permite contar con un timeline de seguimiento, notificaciones e informes personalizados y en tiempo real.
- Integración: permite la integración con herramientas como GitHub, Trello, entre otras.

Google Drive

Servicio de Google que permite acoplarse a organizaciones de todo tipo y tamaño, cuenta con servicios que amplían las características otorgadas, mediante el registro por suscripción. Para instituciones educativas cuenta con un paquete que permite acceso a sus herramientas con limitaciones que para instituciones pequeñas o medianas no causan algún inconveniente, entre sus principales funcionalidades tenemos [12]:

- Drive: almacenamiento seguro en la nube.
- Documentos, hojas de cálculo, presentaciones, formularios: alternativas online que permiten crear, editar, colaborar y compartir documentos entre colaboradores.
- Gmail: correo electrónico personalizado.
- Meet: para videoconferencias y reuniones.

GitLab

Esta herramienta permite gestionar el desarrollo de software desde el inicio hasta que sea puesta en producción, ofrece con detalle los cambios que se construyen en un proyecto, así como informes detallados de los mismos, ayuda a mejorar el tiempo de desarrollo y reducir los costos en sus diferentes etapas [13]:

- Planeación: permite la planificación y gestión del portafolio de proyecto mediante épica e hitos para organizar actividades y realizar un seguimiento del progreso del proyecto.
- Creación: creación y administración de código mediante el uso de herramientas poderosas como un editor de código en línea.
- Verificación: pruebas e informes automatizados.
- Empaquetado: gestión de paquetes integrado
- Seguridad: capacidad de integrar en su ciclo de vida de desarrollo, las pruebas de seguridad SAST, DAST, escaneo de contenedores y escaneo de dependencias.
- Despliegue: automatiza el lanzamiento de código, agilizando los procesos manuales y acelerando la velocidad del equipo.
- Monitoreo: obtenga comentarios y las herramientas que permitirán reducir la gravedad y la frecuencia de los incidentes.
- Gobierno: permite a los usuarios a administrar las vulnerabilidades de seguridad, las políticas y el cumplimiento en toda la organización.

Jetbrains

Jetbrains es una compañía cuyo objetivo es el de crear herramientas para los profesionales de desarrollo de software, planificación y mucho más, cuenta con 2 versiones o licenciamientos, la primera la versión de pago que amplía las características de las herramientas, mientras que la segunda es la versión comunitaria que, aunque limita sus funcionalidades permite construir aplicación de pequeña o gran escala [14]. Entre sus principales herramientas tenemos ah aquellas que benefician a los equipos de desarrollo de software, tales como:

- PyCharm: entorno de desarrollo para el lenguaje de programación Python y sus frameworks como DJango.

- WebStorm: entorno de desarrollo para frameworks y librearías de javascript como React y Angular.
- PhpStorm: entorno de desarrollo para el lenguaje de programación Php.
- DatGrip: entorno para poder trabajar con conexiones a base de datos, ejecutar consultar y gestionar diferentes motores de base de datos como postgresql o mysql.

Adicionalmente estas herramientas pueden ampliar sus funcionalidades con plugins para agilizar tareas o mejorar la calidad del código. Sus herramientas de desarrollo se integran con al trabajo con contenedores.

Visual Studio Code

Es un editor de código gratuito, distribuido por Microsoft, que permite construir código para cualquier lenguaje de programación en un mismo editor. Es extensible con una gran cantidad de plugins que incrementan sus características. Se integra a su entorno de desarrollo con contenedores, bases de datos, api's, entre otros [15].

GitHub Copilot

Con el apoyo de la Inteligencia Artificial (IA), GitHub Copilot, se comporta como un asistente en tiempo real para agilizar las tareas de desarrollo de software, desde generar código a partir de lenguaje natural, hasta predecir el código a escribir. Permite su integración a los diferentes ID's de desarrollo como los distribuido por Jetbrains o VisualStudioCode. Funciona con un modelo de IA preentrenado generativo, creado por OpenAI, como fuente de conocimiento emplea el código que existe en los repositorios públicos de GitHub. Cuenta con una capa gratuita si se realiza el registro desde una cuenta educativa [16].

Diagrams.net

Herramienta web para la creación de diagramas, integra la posibilidad de construir diagramas para los diferentes estándares en los ámbitos profesionales (UML, BPMN), entre sus principales características tenemos [17]:

- Colaboración en tiempo real para la creación de diagramas.
- Permite alojar los diagramas en las soluciones de nube propias del usuario final.
- Herramientas avanzadas de diseño

Camunda

Camunda se encuentra construido sobre la base de estándares abiertos, este proyecto colaboró a definir las especificaciones de BPMN y DMN [18]. Se puede hacer uso del mismo gracias a su versión comunitaria, entre sus características se pueden destacar las siguientes:

- Automatización de los procesos.
- Estricto con los estándares, minimiza la creación de diagramas que no estén acorde a los estándares.

Sonar Lint

Es una herramienta de análisis que identifica, notifica y facilita la corrección de errores comunes, complicados y críticos corrige. Cuenta con más de 5000 reglas de la detección de errores o vulnerabilidades. Se integra con facilidad a los ID's desarrollo más populares como Visual Studio Code, Visual Studio, IntelliJ, Eclipse, entre otras [19].

Sonar Qube

Sonar Qube es una herramienta que se integra a con los entornos de desarrollo para permitir mejorar la calidad del código mediante el análisis de más de 30 lenguajes de programación y frameworks, entre sus beneficios tenemos [20]:

- Integración a las plataformas de DevOps.
- Alta operabilidad
- Análisis rápido
- Análisis de reglas de seguridad
- Compartir y unificar configuraciones

OWASP ZAP

Es un escáner de aplicaciones web, gratuito y de código abierto, bifurcación de la variante de código abierto Paros Proxy, que permite realizar pruebas de penetración para identificar vulnerabilidades de aplicaciones web, es utilizado por expertos y desarrolladores que desean probar funcionalidades y la seguridad de aplicaciones web. Proporciona escáneres automatizados y herramientas que permite identificar vulnerabilidades de seguridad [21].

Slack

Slack es una Plataforma de comunicación y de gestión de equipos que permite su integración a otros flujos de trabajo como DevOps, permite facilitar la comunicación de una organización, cuando con versiones de pago como gratuitas (con sus limitaciones) entre sus funcionalidades tenemos [22]:

- Creación de canales de comunicación
- Reuniones
- Mensajería
- Integración

Discord

Discord fue creado para permitir comunicarse con todas las personas del mundo, crear comunidades, ampliamente utilizado por comunidades de videojuegos, grupos de estudio, artistas, es empleado por más de 200 universidades del mundo [23], permite su integración con herramientas como GitLab para acoplarse al flujo de trabajo de un equipo de desarrollo de software.

Docker

Docker introdujo lo que se convertiría en el estándar de la industria para contenedores. Los contenedores son una unidad de software que permite a los desarrolladores aislar su aplicación de su entorno. Docker es el estándar para crear y compartir aplicaciones en contenedores, desde el escritorio hasta la nube, sobre todo mejora los flujos de trabajo de los desarrolladores [24].

Jenkins

Jenkins es un servidor automatizado de código abierto, se integra fácilmente a las herramientas estándares de programación, como lo es gitlab, sonar qube, pytest, en otros; permite automatizar los procesos de desarrollo median la Integración Continua y Entrega Continua, aunque pueden trabajarse por separado [25]. Jenkins Pipeline es un conjunto de complementos que permiten automatizar el proceso de obtener software desde el control de versiones hasta la entrega pasar por las pruebas de calidad, de seguridad del código y la entrega de la solución a los usuarios finales. Mediante una visualización gráfica de la ejecución del pipeline, permite conocer el estado de cada una de las fases configuradas en el pipeline y con esto contar con un registro de las acciones y posibles errores que se presente en su ejecución [26].

Metodologías de Desarrollo de Software

Una metodología es un conjunto de técnicas y métodos que permiten realizar, gestionar y administrar un proyecto de software durante todo su ciclo de vida, logrando obtener una posibilidad muy alta de éxito del proyecto, se basan en una combinación de los modelos de proceso genéricos, definen artefactos, roles y actividades, junto con prácticas y técnicas recomendadas [2], [27].

Metodologías tradicionales

Estas metodologías surgieron a partir de los años 80, en donde crearon algunas propuestas para desarrollar software, naciendo como metodologías rigurosas y planificadas que se soportaban en herramientas CASE (Ingeniería de Software Asistido por computador), las mismas quitaban gran cantidad de tiempo al desarrollo del sistema, pues la gran parte de este se utilizaba en planificación, diseño y documentación [28].

Características de las Metodologías Tradicionales

Este tipo de metodologías de desarrollo de software se caracterizan por:

- Define total y rígidamente los requisitos al inicio del proyecto de ingeniería de software.
- Los ciclos de desarrollo son poco flexibles y no permiten realizar cambios.
- La organización del trabajo es lineal, no se puede iniciar una sin antes haber terminado la anterior, por lo que no se puede volver a una etapa pasada [1].

Principales Metodologías Tradicionales

A continuación, se enlistan las principales metodologías concebidas entre 1970 y 1990, en el presente trabajo de titulación no se profundiza en estas metodologías, puesto que de este grupo no se han seleccionado para la propuesta y por ende no contribuyen al tema de estudio.

- RUP (Rational Unified Process)
- RAD (Rapid Application Development)
- MSF (Microsoft Solution Framework)
- Win-Win Spiral Model
- Iconix

- JSD (Desarrollo de sistemas de Jackson)
- Ingeniería de la Información
- SSADM (Structured System Analysis and Design Method

Metodologías ágiles

Surgen a partir de 1990, su nombre se asocia a la necesidad de dejar las metodologías existentes, muy robustas y pesadas, por lo que se empiezan a crear metodologías que se centran más en el software y no en la documentación, permitiendo que se ajusten mejor a cambios, llegan a practicarse desde entonces variedad de metodologías ágiles, todas con procesos y fases distintas [28].

Características de las Metodologías Ágiles

Las metodologías agiles en su búsqueda de abandonar o dejar en el olvido las metodologías tradicionales, poseían las siguientes características para ser consideradas como ágiles.

- Poseen alta flexibilidad y agilidad.
- Los equipos de trabajo tienen total conocimiento de sus actividades y tiempos.
- Permiten adaptar el software a nuevas necesidades que vayan surgiendo.
- Se basan en la metodología incremental, en la que en cada ciclo permite agregar nuevas funcionalidades a la aplicación final.
- El cliente puede aportar con nuevos requerimientos o correcciones, ya que puede comprobar el avance del proyecto en tiempo real [1].

Principales Metodologías Ágiles

A continuación, se enlistan las principales metodologías ágiles, en este trabajo de titulación se toman las metodologías ágiles que se pueden acoplar al equipo de desarrollo de software de la DTI y pues son base en el tema en estudio.

- Adaptative Software Development
- Agile Modeling
- Agile Model Driven Development
- Agile Unified Process
- Crystal Methods

- Dynamic Systems development methods
- Evolutionary Project Management
- Extreme Programming
- Feature Driven Development
- Internet Speed Development
- Lean Development
- Mobile-D
- Open Unified Process
- Pragmatic programming
- Scrum
- Story cards driven development
- Test Driven Development
- Win Win Spiral
- X-Breed

Metodología XP

Es una metodología de desarrollo de software basada en las relaciones interpersonales, que se consideran la clave del éxito. Su principal objetivo es crear un buen ambiente de trabajo en equipo y que haya un feedback constante del cliente [28].

XP es la metodología ágil más conocida. Fue desarrollada por Kent Beck buscando guiar equipos de desarrollo de software pequeños o medianos, entre dos y diez desarrolladores, en ambientes de requerimientos imprecisos o cambiantes [3].

Los roles que componen esta metodología son [9] [29]:

- **Programador (Programmer). -** Se encarga de la programación y desarrollo del producto.
- **Cliente (Customer). -** Encargado de los requisitos del sistema.
- **Encargado de pruebas (Tester). -** Realiza las pruebas de funcionamiento y sirve como vínculo entre el cliente y el equipo.
- **Encargado de Seguimiento (Tracker). -** Su función es el seguimiento del desarrollo del trabajo.
- **Entrenador (Coach). -** Guía el proceso de desarrollo.

- **Consultor (Consultant). –** Miembro externo del equipo, experto en algún tema en específico.
- **Gestor (Big Boss). -** Sirve como vínculo entre el cliente y el equipo

En todo proceso de desarrollo de software se generan modelos de información, los artefactos son importantes para conocer cuál fue el proceso de desarrollo del software y lograr entender cómo está construido el sistema, así como la ruta a seguir para agregar funcionalidad al sistema.

- **Historias de Usuario (Story Card). -** Explicación de lo que el producto o sistemas debe realizar.
- **Pruebas de aceptación. -** Culminación y cumplimiento de requisitos.
- **Tarjetas de tareas para la descarga de documentos**
- **Código**
- **Pruebas unitarias**
- **Pruebas de integración**
- **Pruebas de aceptación**

Los valores de la Tabla I sirven como base para la definición de los principios de la metodología XP:

TABLA I
VALORES Y PRINCIPIOS DE XP

Valores XP	**Principios XP**
Simplicidad	Retroalimentación rápida
Comunicación	Asumir simplicidad
Retroalimentación	Cambio incremental
Respeto	Aceptación del cambio
Coraje	Trabajo de calidad.

Las prácticas de esta metodología se derivan de sus valores y principios y están enfocadas en darle solución a las actividades básicas de un proceso de desarrollo, esto es: escribir código, realizar pruebas, escuchar (planear) y diseñar. Las prácticas de XP incluyen:

- **Planning game. -** Define el alcance y la fecha de cumplimiento de una entrega funcional completa, la fecha de entrega de un release que puede ser puesto en funcionamiento y divide las responsabilidades entre el cliente y los desarrolladores. El cliente define, utilizando Historias de Usuario, una versión simplificada de los tradicionales Casos de Uso, los requerimientos de manera general, y precisa su

importancia; Con base en ellas, los desarrolladores estiman el costo de implementarlas y se definen las características de una entrega y el número de iteraciones que se necesitarán para terminarla. Para cada iteración el cliente define cuáles de las historias de usuario que componen la entrega funcional desea que se desarrollen. Se pueden crear o modificar historias de usuario en cualquier momento excepto cuando forman parte de una iteración en curso.

- **Pequeñas entregas. -** Se refiere al uso de ciclos cortos de desarrollo (iteraciones) que le muestran software terminado al cliente y obtienen retroalimentación de él. La definición de terminado está relacionada con las pruebas de aceptación.
- **Diseño simple. -** Indica que el sistema debe ser tan simple como sea posible, en un momento determinado, lo cual implica que los desarrolladores deben preocuparse únicamente por las historias de usuario planeadas para la iteración actual, sin importar cuanto pueden cambiar por funciones futuras.
- **Programación en pareja. -** Indica que todo el código debe ser desarrollado por dos programadores. Las parejas deben cambiar con cierta frecuencia para que el conocimiento de todo el sistema quede en todo el equipo, una práctica que fortalece los principios de diseño simple, calidad y propiedad colectiva del código. Las Pruebas son la guía del desarrollo del producto ya que los detalles de las historias de usuario, se obtienen en el desarrollo de la prueba de aceptación.
- **Pruebas. -** Las Pruebas son la guía del desarrollo del producto ya que los detalles de las historias de usuario (i.e., requerimientos específicos) se obtienen en el desarrollo de la prueba de aceptación. Las pruebas son lo primero que se desarrolla; con base en ellas, se desarrolla el código que las satisfaga. A este concepto se les denomina Desarrollo orientado a las pruebas.
- **Refactoring. -** Consiste en realizar cambios que mejoren la estructura del sistema sin afectar su funcionamiento. Para garantizar la no afectación, después de cada cambio se corre la prueba unitaria, con el fin de corroborar sus beneficios. Esta práctica ayuda a mantener el código simple.
- **Integración continua. -** Establece que cada tarea que se completa se integra al sistema, un proceso que puede darse, incluso, varias veces al día. Con cada tarea que se integra al sistema se corren las pruebas unitarias, las cuales validan si lo que ha sido agregado no perjudica las funcionalidades existentes. En ocasiones las pruebas unitarias fallan y es responsabilidad del desarrollador o de la pareja que integró el código, arreglarlo. Es

común encontrar una regla en los proyectos XP que evita que un desarrollador o pareja se enfoque en otra cosa distinta a arreglar los errores que surgieron de la integración de su código con el del sistema. Esta regla va acompañada de otra que evita que otras partes de código sean integradas en el sistema mientras no hayan superado exitosamente las pruebas. Esto pone un poco más de presión en aquellos desarrolladores cuyo código fue incapaz de superar las pruebas.

- **Propiedad común del código. -** Implica que no hay una persona propietaria del código que se está desarrollando o de una porción de este, por lo que cualquier desarrollador que esté participando en una iteración puede hacer cambios en el código, siempre que le agregue valor al sistema.
- **Paso sostenible. -** Hace referencia al ritmo de trabajo del equipo. Indica que debe ser adecuado para que sea factible terminar la iteración o la entrega sin trabajar horas extras. La metodología establece además que nunca se debe trabajar horas extras dos semanas consecutivas.
- **Cliente en sitio. -** Se refiere a la necesidad de incluir un representante del cliente trabajando a tiempo completo con el equipo de desarrollo. Su función es resolver oportunamente dudas en la implementación de las historias de usuario.
- **Metáfora. -** Establece que todos los implicados en el proyecto deben tener la misma visión del sistema. Para lograrlo deben hacer abstracciones que establezcan un lenguaje común entre el cliente y los desarrolladores. La metáfora es la guía global del desarrollo.
- **Estándares de código. -** Son las reglas que definen como se va a escribir la aplicación. La metodología establece que estos estándares deben estar definidos de tal forma que el código en sí mismo sirva como un documento. Este tema es de altísima relevancia porque tanto la programación en pareja, como la propiedad común del código, son imposibles sin estos estándares [3].

El ciclo de vida de esta metodología, por su parte, está compuesto por:

- **Exploración.** - Se hace una estimación con base en las historias de usuario requeridas para la primera entrega
- **Planeación. -** En la de Planeación, el cliente y los programadores definen las historias de usuario que se van a implementar y sus fechas

- **Iteraciones hacia la primera entrega. –** Se transforma en el calendario acordado con el cliente, expresado en iteraciones, donde cada una de ellas representa historias de usuario implementadas y probadas
- **Productionizing. -** Se afina el funcionamiento del programa y se despliega
- **Mantenimiento**. - Mantenimiento se continúan realizando mejoras y arreglos, e implementando nuevas funcionalidades [3].

En 2004 se publicó una versión revisada de XP en la que se modifican los principios y las prácticas. Los nuevos principios hacen más directa la asociación de los valores con las prácticas. Las nuevas prácticas se subdividen en dos categorías: primarias, las de mayor impacto y por lo tanto las primeras que deberían ser implementadas; y coralarias, las más difíciles de implementar y por ello las que deberían adoptarse después de haber adoptado las primarias. Esta versión, sin embargo, no ha tenido gran acogida y la mayoría de los autores siguen tomando como referencia la versión original [3].

Metodología SCRUM

Es una metodología incremental que divide los requisitos y tareas de forma similar a Kanban. Se itera sobre bloques de tiempos cortos y fijos (entre dos y cuatro semanas) para conseguir un resultado completo en cada iteración [30] [3].

Las etapas de SCRUM son:

- Planificación de la iteración (planning sprint)
- Ejecución (sprint)
- Reunión diaria (daily meeting)
- Demostración de resultados (sprint review).

Cada iteración por estas etapas se denomina también sprint [3].

Los roles que define SCRUM son:

- **Scrum máster**. - Es el dueño del producto y el equipo de desarrollo, su función es asegurar que el equipo está adoptando la metodología, sus prácticas, valores y normas; es el líder del equipo, pero no gestiona el desarrollo y es el vínculo entre el cliente y el equipo.

- **Dueño del producto (Product owner)**. - es una sola persona y representa a los interesados, responsable del seguimiento del desarrollo del trabajo orientado a maximizar el valor del producto y el trabajo del equipo de desarrollo; también se encarga de gestionar la lista ordenada de funcionalidades requeridas o Product Backlog.
- **Equipo de desarrollo**. – Está conformado de manera óptima entre tres a nueve personas, en donde no existen jerarquías, todos están al mismo nivel y cargo, su principal responsabilidad es convertir lo que el cliente quiere (Product Backlog) en iteraciones funcionales del producto o programa [3], [31].

Evento principal o Sprint. - Corresponde a una ventana de tiempo donde se crea una versión utilizable del producto (incremento). Cada Sprint, es considerado como un proyecto independiente. Su duración máxima es de un mes. Un Sprint se compone de los siguientes elementos [3]:

- **Reunión de planeación del Sprint. –** Aquí se define su plan de trabajo: qué se va a entregar y cómo se logrará, obteniendo el diseño del sistema y la estimación de cantidad de trabajo. Esta actividad dura ocho horas para un Sprint de un mes. Si el Sprint tiene una duración menor, se asigna el tiempo de manera proporcional.
- **Reunión Diaria. -** Es un evento del equipo de desarrollo de quince minutos, se realiza cada día con el fin de explicar lo que se ha alcanzado desde la última reunión; lo que se hará antes de la siguiente; y los obstáculos que se han presentado. Este evento se desarrolla mediante una reunión que normalmente es sostenida de pie con los participantes reunidos formando un círculo, esto, para evitar que la discusión se extienda.
- **Revisión del Sprint. -** La Revisión del Sprint ocurre al final del Sprint y su duración es de cuatro horas para un proyecto de un mes (o una proporción de ese tiempo si la duración es menor). En esta etapa el dueño del proyecto revisa lo que se hizo, identifica lo que no se hizo y discute acerca del Product Backlog; el equipo de desarrollo cuenta los problemas que encontró y la manera en que fueron resueltos, y muestra el producto y su funcionamiento. Esta reunión es de gran importancia para los siguientes Sprints.
- **Retrospectiva del Sprint. -** La Retrospectiva del Sprint es una reunión de tres horas del equipo Scrum en la que se analiza cómo fue la comunicación, el proceso y las herramientas; qué estuvo bien, qué no, y se crea un plan de mejoras para el siguiente

Sprint. El tiempo, se debe ajustar proporcionalmente en el caso de proyectos de duración menor a un mes [3]:

Existen también los denominados Artefactos de Scrum. Estos son subproductos de las actividades del marco de trabajo que le brindan dirección y transparencia al equipo. Los artefactos de Scrum son:

- **Product Backlog. -** Es una lista ordenada por valor, riesgo, prioridad y necesidad de los requerimientos que product owner define, actualiza y ordena. La lista tiene como característica particular que nunca está terminada, pues evoluciona durante el desarrollo del proyecto.
- **Sprint Backlog. -** Es un subconjunto de ítems del Product Backlog y el plan para realizar en el Incremento del producto. Debido a que el Product backlog está organizado por prioridad, el Sprint backlog es construido con los requerimientos más prioritarios del Product backlog y con aquellos que quedaron por resolver en el Sprint anterior. Una vez construido, el Sprint backlog debe ser aceptado por el equipo de desarrollo, pertenece a éste y solo puede ser modificado por él. Requerimientos adicionales deben ser incluidos en el Product backlog y desarrollados en el siguiente Sprint, si su prioridad así lo indica.
- **Monitoreo de Progreso e Incremento. -** El Monitoreo de Progreso consiste en la suma del trabajo que falta por realizar en el Sprint. Tiene como característica que se puede dar en cualquier momento, lo que le permite al dueño del producto evaluar el progreso del desarrollo. Para que esto sea posible, los integrantes del equipo actualizan constantemente el estado de los requerimientos que tienen asignados indicando cuánto consideran que les falta por terminar. El Incremento es la suma de todos los ítems terminados en el Sprint backlog. Si hay ítems incompletos deben ser devueltos al Product backlog con una prioridad alta para que sean incluidos en el siguiente Sprint. Se considera que un ítem está terminado si es funcional. La suma de ítems terminados es el producto por entregar [3].

El ciclo de vida de este marco de trabajo está compuesto de cuatro fases:

- **Planeación. -** Se establece la visión, se fijan las expectativas y se asegura el financiamiento.

- **Puesta en escena. -** Se identifican más requerimientos y se priorizan para la primera iteración.
- **Desarrollo. -** Se desarrolla el sistema
- **Entrega. -** Se hace el despliegue operativo [3].

Metodología KANBAN

Fue propuesta por la empresa de automóviles Toyota, consiste en dividir las tareas en porciones mínimas y organizarlas en un tablero de trabajo dividido en tareas pendientes, en curso y finalizadas. De esta forma, se crea un flujo de trabajo muy visual basado en tareas prioritarias e incrementando el valor del producto. Kanban es un mecanismo de control que facilita el seguimiento del trabajo, aumentar la efectividad de equipo y limitar el trabajo en curso, evitando así el cuello de botella o eliminarlo si existiera. Como medio para comunicación con el cliente es una poderosa herramienta ya que permite, de forma rápida, transmitir al cliente las actividades que se están desarrollando y el avance general del proyecto [31].

Metodología LEAN

Orientado a pequeños equipos de desarrollo muy capacitados para que elaboren cualquier tarea en poco tiempo. Los activos más importantes son las personas y su compromiso, relegando así a un segundo plano el tiempo y los costes. El aprendizaje, las reacciones rápidas y potenciar el equipo son fundamentales [32].

Lean, como paradigma, se refiere al conjunto de metodologías de desarrollo de software que toman como inspiración la manufactura Lean de Toyota. Además, Lean es un paradigma en el sentido de que no está conformado por prácticas específicas sino por ideas, filosofía aplicada a un proceso de desarrollo con el objetivo de mejorarlo [31], [32].

A gran escala, Lean se basa en la aplicación de los siguientes principios:

- **Eliminar la Basura:** Eliminar todo aquello que no entregue valor al cliente incluyendo: Trabajo parcial, procesos innecesarios, Features innecesarias, Multitasking, esperas, movimientos innecesarios, Defectos y Actividades de gestión.
- **Amplificar el Conocimiento**
- **Decidir lo más tarde posible:** Tomar las decisiones cuando exista la mayor cantidad de información disponible

- **Desarrollar lo más rápido posible:** Con el objetivo de entregar rápidamente valor al cliente y obtener su retroalimentación
- **Empoderar al equipo:** Permitir que el equipo tome sus propias decisiones con las herramientas adecuadas

DevOps

“El término "DevOps" es una combinación de las palabras "development" (desarrollo) y "operations" (operaciones), pero representa un conjunto de ideas y prácticas que van más allá de ambos conceptos, ya sea que estén juntos o separados. DevOps incluye sistemas de seguridad, maneras de trabajar en colaboración, análisis de datos, entre otras características.” [33].

DevOps es un término relativamente nuevo para muchos equipos pero que apareció por primera vez en 2009. Sin embargo, empresas como Etsy, Facebook, Amazon o Netflix son líderes en la implementación del nuevo paradigma [34].

Para Gartner, “DevOps representa un cambio en la cultura de Tecnologías de la Información (TI), que se centra en la entrega rápida de servicios de TI mediante la adopción de prácticas ágiles y ajustadas en el contexto de un enfoque orientado al sistema. DevOps hace hincapié en las personas (y la cultura) y busca mejorar la colaboración entre los equipos de operaciones y desarrollo. Las implementaciones de DevOps utilizan tecnología, especialmente herramientas de automatización, que pueden aprovechar una infraestructura cada vez más programable y dinámica desde la perspectiva del ciclo de vida" [35].

DevOps presenta algunas ventajas tales como:

- Colaboración y confianza
- Publicaciones más rápidas y una forma de trabajar más inteligente
- Acelerar el tiempo de resolución
- Mejor gestión del trabajo imprevisto

En 2018 en el informe de State of DevOps, el 50% de empresas encuestadas muestra que las empresas encuestadas han adoptado DevOps y están buscando mejoras y que el 32% de esta muestra planean adoptar DevOps en los próximos doce meses de realizada dicha encuesta [36].

Modelo ALM

“La gestión del ciclo de vida de las aplicaciones (ALM) involucra a las personas, las herramientas y los procesos que gestionan el ciclo de vida de una aplicación desde que se diseña hasta el final de su vida útil. La componen varias disciplinas que solían estar divididas por los procesos de desarrollo heredados, como el método de desarrollo en cascada. Entre ellas, se incluyen la gestión de proyectos y de los requisitos, el desarrollo de software, las pruebas y el control de calidad, la implementación y el mantenimiento. La ALM admite los enfoques de desarrollo ágiles y de DevOps gracias a que integra estas disciplinas y permite que la colaboración de los equipos sea más eficiente para la empresa.” [37].

Etapas:

- Control de aplicaciones
- Desarrollo de aplicaciones
- Prueba del software
- Operaciones y mantenimiento

Modelo CALMS

CALMS es una guía de los principios que se deben tener en cuenta en cualquier adopción de DevOps, así mismo “CALMS es un marco de trabajo que evalúa la capacidad de una compañía de adoptar los procesos de DevOps, además de una forma de medir el éxito mientras tiene lugar la transformación a este modelo. El acrónimo fue acuñado por Jez Humble, coautor de "The DevOps Handbook", y significa cultura (Culture), automatización (Automation), metodología lean (Lean), medición (Measurement) y compartir (Sharing).” [36], [38].

Principios CALMS:

- Cultura
- Automatización
- Lean
- Medición
- Compartir

5. Metodología

Para el desarrollo de la propuesta se utilizó la metodología denominada **"investigación cualitativa"**, debido a que es un método científico de observación y con su conjunto de técnicas permitió obtener una visión general del problema, generar ideas y/o suposiciones y finalmente identificar y plantear las soluciones específicas a la realidad organizacional. La investigación cualitativa enmarca las fases de [39]:

- ✓ **Planteamiento del problema**: Ejecutada en la propuesta del proyecto, de la cual surgió el siguiente problema de investigación "¿con la propuesta de un marco de trabajo metodológico de desarrollo de software, se permitirá ejecutar, estandarizar y documentar de manera adecuada el proceso de desarrollo de software?"
- ✓ **Revisión de la literatura**: La revisión de literatura permitió justificar el planteamiento del problema y la necesidad de estudio; así mismo se usó para recolectar información para el marco teórico y objetivo específico 2.
- ✓ **Recolección de datos**: Para la recolección de datos se plantearon los objetivos específicos 1 y 2.
- ✓ **Análisis de datos**: Para la recolección de datos se plantearon los objetivos específicos 1 y 2.
- ✓ **Reporte de resultados**: Esta fase comprende la presentación y difusión de resultados, para lo cual se plantó el objetivo específico 3.

Área de estudio

El análisis de la situación actual y propuesta se enfocó a la Dirección de Tecnologías de Información (DTI) de la Universidad Nacional de Loja (UNL). Según el Art. 37 del Reglamento Orgánico de Gestión Organizacional por Procesos de la Universidad Nacional de Loja, la misión de la Dirección de Tecnologías de Información es "***Administrar e implementar tecnologías de información (TI) en la Universidad Nacional de Loja para el desarrollo innovador en los procesos académicos y administrativos; que garanticen la confidencialidad, integridad y disponibilidad de la información, como un medio para contribuir a la excelencia académica.***"; así mismo en su portafolio de productos y servicios se encuentran: seguridad informática, provisión de servicios, sistemas de información e infraestructura tecnología.

Acorde al punto anterior, el producto o servicio "sistemas de información" es el encargado de la automatización de los procesos académicos, administrativos y financieros, es decir planifica y ejecuta los proyectos de desarrollo de software, para lo cual debe considerar las diversas metodologías de desarrollo de software, metodologías de gestión proyectos, enfoques y la realidad institucional.

Procedimiento

Apoyados en los métodos, técnicas de la investigación cualitativa, como: observación participativa, entrevista, investigación documental; y, a las fases de recolección de datos, análisis de datos y reporte de resultados, se detallan las actividades realizadas para lograr cada uno de los objetivos específicos:

1) **Analizar la situación actual de la Dirección de Tecnologías de Información, a través una revisión sistemática y trabajo de campo, para conocer cómo se está ejecutando el proceso de desarrollo de software.**

- ✓ Se realizó una observación directa del proceso de desarrollo de software en la DTI, con la finalidad de identificar procesos, artefactos y roles utilizados. (Ver sección: **6.1.1. Observación directa**).
- ✓ Luego se aplicó una encuesta al personal de la DTI involucrada en el proceso de desarrollo de software, para determinar metodologías usadas, prácticas, procesos, roles, etc. (Ver **Anexo 1, Anexo 2, Anexo 3, Anexo 4, Anexo 5, Anexo 6**).
- ✓ Posteriormente se analizó la normativa vigente y se la categorizó en una matriz, para determinar artículos relacionados a las metodologías o procesos del software, con la finalidad de observarlos y considerarlos al momento de realizar la propuesta. (Ver sección: **6.1.3. Análisis de la normativa vigente**).
- ✓ Finalmente se realizó la interpretación de los resultados de las encuestas aplicadas (Ver sección: **6.1.2. Encuestas aplicadas y Anexo 4, Anexo 5, Anexo 6**).

2) **Analizar las metodologías de desarrollo software, marcos de trabajo y enfoques más relevantes: XP, SCRUM, KANBAN, LEAN y DevOps; mediante una revisión bibliográfica; para identificar las mejores prácticas, procesos y entregables.**

Acorde a menciona la investigación de un diseño de un método ágil de desarrollo de software [28], se parte de las características que tienen en común las metodologías, como los procesos, roles y artefactos. A continuación, se detalla el proceso seguido:

- ✓ Se realizó una revisión bibliográfica de las metodologías a analizar.
- ✓ Luego se determinó las características comunes a analizar (Ver **Tabla VIII** de la sección **6.2. Análisis de metodologías ágiles.**)
- ✓ Se realizó una comparativa de los roles de las metodologías (Ver **Tabla IX** de la sección **6.2. Análisis de metodologías ágiles**.)
- ✓ Posteriormente se procedió a comparar los procesos y ciclo de vida de las metodologías (Ver **Tabla X** de la sección **6.2. Análisis de metodologías ágiles**.)
- ✓ Se identificó las prácticas, valores y principios de las metodologías (Ver **Tabla XI** de la sección **6.2. Análisis de metodologías ágiles**.)
- ✓ Finalmente se identificó los artefactos usados en cada una de las metodologías (Ver Tabla XII de la sección **6.2. Análisis de metodologías ágiles**.)

3) **Realizar la propuesta del marco metodológico de desarrollo de software, acorde a la normativa vigente y a las mejores prácticas de las metodologías, marcos de trabajo y enfoques analizados; en base a los resultados del análisis elaborado; para estandarizar los procesos de desarrollo de software.**

Con el proceso de recolección de datos y análisis de datos realizados en los objetivos anteriores se procedió a diseñar la propuesta para lo cual se realizó las siguientes actividades:

- ✓ De la revisión de la normativa vigente se planteó una lista de consideraciones a tomar en cuenta al abordar un proyecto (Ver sección **6.3.1. CONSIDERACIONES**)
- ✓ Tomando en cuenta los roles de la metodologías y cargos de la DTI, se realizó la propuesta de los roles a usar (Ver sección **6.3.2. ROLES**)

- ✓ Observando los procesos de las metodologías ágiles y de la DTI se determinó el ciclo de vida con sus fases y actividades principales (Ver sección **6.3.3. CICLO DE VIDA**)
- ✓ Se seleccionó las prácticas más relevantes, valores y principios de las metodologías ágiles (Ver sección **6.3.4. MEJORES PRÁCTICAS**)
- ✓ Se realizó la propuesta de artefactos tomando en cuenta las metodologías seleccionadas y procesos ejecutado por DTI (Ver sección **6.3.5. ARTEFACTOS**)
- ✓ Se elaboró el formato de los artefactos más críticos acorde a la realidad organizacional (Ver sección **6.3.5 ARTEFACTOS**)
- ✓ Se elaboró la propuesta de pipe line acorde a las herramientas usadas en la DTI y a las aplicables en DevOps (Ver sección **6.3.6 HERRAMIENTAS Y PIPE LINE DEVOPS**)
- ✓ Con la finalidad de presentar la propuesta se elaboró un documento editable con los elementos y artefactos propuestos (Ver sección **6.3. Propuesta del marco de trabajo metodológico de desarrollo de software.** y **Anexo 21**)
- ✓ Finalmente se presentó la propuesta ante la DTI para que sea considerada en la ejecución de sus proyectos.

Procesamiento y análisis de datos

La información obtenida mediante observación directa, encuesta e investigación documental, permitió conocer la situación actual de la Dirección de Tecnologías de Información y las buenas prácticas, artefactos, roles de las metodologías de desarrollo de software. A continuación, se describen y mencionan las técnicas utilizadas para el procesamiento y análisis de datos:

- ✓ **Observación participativa**: En vista de que los autores son parte del equipo de desarrollo de software, se aprovechó esta oportunidad para hacer un proceso de observación en calidad de participante, para lo cual se fueron tomando notas de los principales procesos y artefactos utilizados en los proyectos en ejecución (Ver sección **6.1.1. Observación directa**).
- ✓ **Encuesta**: La encuesta se diseñó con la finalidad determinar la situación actual de la DTI, para lo cual se usó la herramienta en línea Microsoft Forms y se aplicó a todo el personal involucrado en el desarrollo de software (Ver **Anexo 1, Anexo 2** y **Anexo 3**).
- ✓ **Investigación Documental**: Este método se usó con la finalidad de analizar la normativa vigente relacionada a la institución y al proceso de desarrollo de software.

Para este proceso se identificó las Políticas de Telecomunicaciones, Desarrollo de Software y Redes de la Universidad Nacional de Loja y las Normas de Control Interno de la Contraloría General del Estado (Ver sección **6.1.3. Análisis de la normativa vigente**).

- ✓ **Investigación Documental:** La aplicación de este método permitió revisar la información bibliográfica utilizando fuentes fiables ya existentes en internet, libros y revistas. Las actividades ejecutadas son:
- ✓ **Análisis de datos:** Se utilizó el **análisis exploratorio** con la finalidad de examinar la información recolectada con las diferentes técnicas (encuestas, investigación documental y observación directa) y encontrar relaciones entre ellas. Los pasos usados para este proceso se listan a continuación:
    - Preparar y organizar los datos: Se organiza la información obtenida para proceder a tabular.
    - Revisar y explorar los datos: Se buscan patrones o ideas repetidas con la finalidad de tabular la información.
    - Presentar los resultados: En la sección de procesamiento y análisis de datos se detalla de manera minuciosa la información.

6. Resultados

En la presente sección se muestra los resultados de la ejecución del proyecto por cada uno de los objetivos, los mismos que fueron planteados y ejecutados acorde a la metodología de investigación cualitativa.

6.1. Análisis de la situación actual de la DTI

En el Objetivo 1, se planteó: "Analizar la situación actual de la Dirección de Tecnologías de Información, a través una revisión sistemática y trabajo de campo, para conocer cómo se está ejecutando el proceso de desarrollo de software."; por lo que se procedió a realizar la encuesta digital al personal técnico de desarrollo de software de la Dirección de Tecnologías de Información de la Universidad Nacional de Loja.

6.1.1. Observación directa

En la Tabla II se documenta los procesos identificados por cada una de las fases en la observación directa realizada en la Dirección de Tecnologías de Información.

TABLA II
OBSERVACIÓN DIRECTA – PROCESOS FUNDAMENTALES

FASE	PROCESOS FUNDAMENTALES
Inicio	**1) El requerimiento de software**.- El requerimiento de software se hace a través de diferentes medios: • Petición a través del correo electrónico. • Petición mediante comunicación formal (Memorando, oficio). • Mediante disposición a través de las resoluciones del Rector u Órgano Colegiado Superior (OCS). • Por cambio de normativa nacional o institucional (Decreto, disposición). **2) Llenado de formato de requerimiento de software**.- A menudo en las peticiones a mediante correo electrónico y comunicaciones oficiales, no se indica de manera clara el requerimiento, por lo que con dicho formato se obtiene una idea inicial y justificada de la necesidad; así mismo el formato sirve para justificar la existencia de la necesidad ante entidades (Auditores internos y externos, Contraloría General del Estado, etc.). **3) Elaboración de caso de negocio y estudio de factibilidad**.- Se hace un estudio inicial en coordinación con la unidad requirente, con la finalidad de identificar los requerimientos de alto nivel, justificar adecuadamente la necesidad con los elementos de planificación institucional (PEDI, POA, PAC y normativa), estudiar alternativas, determinar la factibilidad, etc. **4) Aprobación del caso de negocio**.- El caso de negocio es presentado ante el Rector con la finalidad de que se determine si debe abordarse el proyecto, se priorice y autorice la ejecución del mismo. **5) Inclusión en la cartera de proyectos**.- En caso de haberse aprobado el caso de negocio, el proyecto se codifica y se incluye en la cartera de proyectos de desarrollo de software.

Análisis

1) **Recolección**. - Con la finalidad de determinar de una manera detallada los requerimientos se establece reuniones con los principales implicados de las unidades requirentes. Adicionalmente se revisa la normativa vigente a nivel nacional e interna. Este proceso se hace en varias ocasiones mientras se esté desarrollando el producto.
2) **Backlog del producto**. - A partir de la recolección de requerimientos, se elabora una matriz con las historias de usuario.
3) **Elaboración de especificación de requerimientos de software**. - Los requerimientos recogidos se documentan y se van actualizando durante todo el proceso de desarrollo de software. Se usa un formato propio basado en IEEE 830 y, en caso de requerirse se saca varias versiones del documento.

Diseño

A continuación, se detallan los artefactos usados:

- Diagrama de casos de uso
- Prototipado de pantallas
- Modelado de dominio
- Diagramas de proceso
- Diagrama de clases
- Diagramas de despliegue

Codificación

En esta fase se trabaja de la siguiente manera:

- **En pares**: Si el proyecto es mediado o grande se asigna dos o más programadores para que trabajen en pares; en caso de proyectos pequeños se asigna un programador para que trabaje directamente en la codificación y un programador que sea de guía.
- **Cliente**: En caso de que alguna historia de usuario no esté aclara se recurre a los clientes o los usuarios para su aclaración.
- **Herramientas**: Se usa herramientas open source y/o servicios gratuitos en la nube. A continuación, se menciona en orden de preferencia.
    - Lenguajes de programación: Python (DJango, Turbo Gears), Php (CodeIgniter), Java (SpringBoot, Seam)
    - Bases de datos: PostgreSql, Mysql, Redis
    - IDE: Pycharm, Visual Studio Code
    - Versionamiento y código: GitLab (Infraestructura propia), Sonar Link, Sonar Qube
    - Gestión de proyectos: Taiga (Infraestructura propia), Nube institucional (Google Workspace)
    - Comunicación: Slack, Telegram
    - Documentación: Nube institucional (Google Workspace)

Pruebas

En las pruebas se llevan a cabo con las siguientes acciones y artefactos:

- Petición de control de calidad
- Elaboración de plan de pruebas
- Ejecución de los casos de prueba
- Certificación de control de calidad
- Reuniones de demostración y validación
- Pilotajes
- Análisis y corrección de vulnerabilidades (informe)

Documentación y capacitación	Se considera: • Manuales de usuario (por roles de ser necesario) • Manual de configuración (técnico, programador) • Presentaciones para las capacitaciones. • Ejecución de la capacitación (presencial o virtual: zoom) • Entorno de pruebas (en los casos que se requiera) • Se cuenta con una guía de desarrollo de software propia.
Cierre	Comprende: • Acta de paso a producción (Una por cada iteración o una al final) • Checklist de paso a producción • Creación y/o actualización de entornos • Verificar los servicios
Mantenimiento	Asistencia técnica a través de: chat, zoom y/o acompañamiento.

En la Tabla III se identifica las metodologías usadas, es decir de las cuales se emplea al menos algún artefacto o proceso.

TABLA III
OBSERVACIÓN DIRECTA - METODOLOGÍAS

#	**METODOLOGÍA**
1	XP (Procesos y artefactos)
2	SCRUM (Procesos y artefactos)
3	KANBAN (Tablero)

En la Tabla IV se listan los principales artefactos que se emplean en el proceso de desarrollo de software de la DTI.

TABLA IV
OBSERVACIÓN DIRECTA - ARTEFACTOS

ARTEFACTOS
Caso de negocio
Épicas / hitos
Historia de Usuario
Especificación de Requisitos de Software
Backlog del producto
Backlog del Sprint
Tablero kanban
Burndown chart
Diagramas
Plan de pruebas
Casos de prueba
Certificación de control de calidad
Acta de paso a producción
Guía de desarrollo de software

En la Tabla V se muestran los roles desempeñados por el personal de la DTI, la unidad requirente los patrocinadores (Rectorados, funciones sustantivas).

TABLA V
OBSERVACIÓN DIRECTA - ROLES

CARGO / UNIDAD	ROL
Director de Tecnologías de Información	Product Owner
Especialista de Sistemas de Información	Scrum Master / Tracker
Analista de Sistemas de Información	Scrum Team: • Analista • Programador • Tester • Responsable Seguridad
Unidad académica - administrativa / usuarios	Stakeholders: • Cliente (Unidad requirente) • Usuarios (docentes, administrativos, trabajadores, estudiantes, externos) • Patrocinadores (Rectorado, funciones sustantivas)

6.1.2. Encuestas aplicadas

En la Tabla VI, se enlista el personal y la encuesta aplicada, de acuerdo con el perfil de cada miembro:

TABLA VI
PERSONAL ENCUESTADO

Rol	Cantidad	Encuesta
Control de calidad	2	Control de Calidad DTI UNL (Anexo 1. Encuesta – Perfil Control de Calidad)
Desarrollador Senior	3	Desarrollador Senior Junior DTI UNL (Anexo 2. Encuesta – Perfil Desarrollador Senior/Junior)
Desarrollador Junior	3	Desarrollador Senior Junior DTI UNL (Anexo 2. Encuesta – Perfil Desarrollador Senior/Junior)
Especialista de Seguridad Informática	1	Seguridad DTI UNL (Anexo 3. Encuesta – Perfil Seguridad de la Información)

A continuación, podemos encontrar los resultados obtenidos de las encuestas aplicadas:

a) Control de calidad

Como resultados pudimos obtener que en la actualidad las pruebas que se efectúan sobre los productos desarrollados por la DTI son de tipo funcional y manuales a través de la

comprobación de funcionalidades y no de manera automatizada, por ello se pudo identificar que el análisis de la calidad del código es de responsabilidad del equipo de desarrollo. Para profundizar en los resultados obtenidos referirse al Anexo 4. Análisis de Resultados de Encuesta – Perfil Control de Calidad.

b) Desarrollador Senior / Junior

De esta encuesta pudimos conocer que el equipo de desarrollo cuenta con conocimientos medios y avanzados en el manejo de las diferentes metodologías, procesos y artefactos que son parte de un proyecto de software, de la misma forma los entornos de desarrollo que se manejan son PyCharm Profesional y Visual Studio Code, y como lenguajes de programación se emplea el Python (Django) y JavaScript (React/Angular). Para profundizar en los resultados obtenidos referirse al Anexo 5. Análisis de Resultados de Encuesta – Perfil Desarrollador Senior / Junior.

c) Especialista de Seguridad Informática

Los resultados de esta encuesta nos permitieron conocer que en la actualidad se emplean herramientas automatizadas de escaneo de vulnerabilidades de las aplicaciones que se encuentran puestas en producción y en un entorno controlado (control de calidad), de la misma forma manifiesta aplicar las recomendaciones de OWASP y el Estándar de Pruebas de Penetración. Para profundizar en los resultados obtenidos referirse al Anexo 6. Análisis de Resultados de Encuesta – Especialista de Seguridad Informática.

6.1.3. Análisis de la normativa vigente

La Tabla VII detalla los artículos de la normativa vigente aplicable a la DTI, relevantes al proceso de desarrollo de software, los mismo que se han catalogado como proceso roles y artefactos.

TABLA VII
ANÁLISIS DE NORMATIVA VIGENTE

	NORMATIVA VIGENTE	ARTÍCULOS VINCULANTES	PROCESO / ROLES / ARTEFACTOS
1	Políticas de Telecomunicaciones, Desarrollo de Software y Redes.		
	2.2 Políticas de metodologías de desarrollo.	El desarrollo de sistemas se llevará a cabo mediante el uso de técnicas estructuradas de análisis y diseño de sistemas.	Metodologías / Métodos: estudio viabilidad, investigación

			situación actual, opciones de sistemas, requisitos, diseño lógico y físico
		El diseño de los sistemas de información se orientará a un ambiente cliente-servidor.	Tecnología
		La Universidad Nacional de Loja debe poseer un conjunto estructurado de normas y reglas que definan la interfaz que los sistemas presenten al usuario.	Guía de desarrollo de software
	2.3. Políticas de demandantes de sistemas.	A las diferentes áreas demandantes de sistemas les corresponderá una activa participación en el proceso de desarrollo de sistemas, comprometiendo tiempo efectivo de apoyo a las actividades de análisis, modelo de datos, diseño administrativo, y puesta en marcha de los sistemas.	Rol cliente
		Los actuales sistemas en explotación, desarrollados con lenguajes de última generación en un ambiente de Base de Datos Relacional, conforme al modelo general de la Universidad Nacional de Loja, se deben certificar por el área de desarrollo.	Certificación de control de calidad
	3.1. Políticas para la elaboración de sistemas.	Toda elaboración de sistemas, tanto interna como externa de carácter institucional, deberá estar avalada por un informe técnico de la Unidad de Telecomunicaciones e Información, este debe normar el uso y aprovechamiento de los recursos informáticos de acuerdo al reglamento interno de la institución.	Informe técnico - factibilidad
		Toda elaboración de sistemas institucionales deberá apegarse a los estándares en cuanto al uso de software. Cuando esto no sea posible, el área usuaria deberá solicitar un dictamen técnico a la dirección de informática de la institución o empresa, justificando plenamente el uso de las herramientas propuestas para el desarrollo.	Informe técnico - factibilidad / Caso de negocio
		Durante el análisis, desarrollo e implementación de cualquier sistema, el área solicitante deberá participar con la Unidad de Telecomunicaciones e Información.	Rol cliente
		Es responsabilidad de la Unidad de Telecomunicaciones e Información, el proporcionar la capacitación y asistencia técnica al personal operativo sobre el correcto uso del sistema.	Plan Capacitación
		En la elaboración y diseño de sistema informáticos internos la Unidad de Telecomunicaciones e Información será la encargada de establecer y mantener un sistema de calidad documentado para asegurar productos conforme a los requerimientos	Manual de procedimientos (procesos) / Manual de configuración /

		especificados por ella misma, además de alcanzar consistentemente los objetivos de calidad de la institución o empresa. Entre los documentos que se generarán por los desarrolladores, están los manuales de procedimientos, técnicos, de instalación, operativos y de usuario.	Manual de usuario
		La Unidad de Telecomunicaciones e Información será el encargado de establecer políticas de administración, calidad y control de calidad; así como la justificación y consistencia de éstas. (...)	Políticas de control de calidad
	3.2. Normas para la elaboración de sistemas.	2. La Unidad de Telecomunicaciones e Información y su sección de Desarrollo de Software deberá registrar, ordenar e inventariar: los programas fuentes y ejecutables, documentación técnica, manual de instalación y manual del usuario	Herramientas: GitLab, Nube institucional Matriz Inventario
		4. Todas las fases del desarrollo de sistemas deberán estar documentadas.	Identificar qué documentar por cada fase
		5. Para aquellos sistemas que se desarrollen con un software no estándar para la institución, será requisito indispensable que cuenten con un módulo de intercambio de información (importación/exportación) a través de múltiples protocolos o servicios (Webservice, LDAP). Por ninguna causa se deberá comenzar la etapa de programación del sistema en general, sin antes tener concluidas las etapas de análisis y diseño. Para el caso en que el sistema por su magnitud se haya dividido en módulos, será válido el comenzar la programación de cada uno de ellos si se cuenta con sus etapas de análisis y diseño concluidas, además de un análisis y diseño preliminar de carácter general del sistema.	Método / Metodología tradicional para software no estándar
	3.3. Normas para el análisis de sistemas.	1. Los desarrollos de sistemas deberán contar con un estudio de factibilidad tecnológica y económica que permita identificar y describir la necesidades del usuario con objeto de justifica la elaboración del sistema.	Estudio de factibilidad
		2. Se deberá establecer los grupos de trabajado encargados para las actividades de diseño de encuestas, entrevistas, recopilación de datos, etc.	Roles
		3. La fase de análisis de sistemas deberá apegarse a las metodologías de análisis orientados a objetos.	Método / Metodología.
	3.4. Normas para el diseño de sistemas.	2. La fase de diseño de sistemas deberá apegarse a las metodologías de análisis y diseño orientado a objetos.	Método / Metodología.

		3. Para los casos en los cuales se efectue un cambio en el diseño de un sistema, dicho cambio deberá ser documentado previa revisión y justificación, así como aprobación de los responsables para posterior control de la documentación, con el fin de que todas las áreas se enteren del cambio efectuado.	Control de cambios
	3.5. Normas para la programación y documentación de sistemas.	1. Todos los programas de integren cualquier sistema deberán estar documentados conforme al Manual de Procedimientos para el Desarrollo de Sistemas.	Manual de procedimientos.
		2. El área usuaria deberá aprobar el manual de usuario previo a la liberación de un sistema. La Unidad de Telecomunicaciones deberá revisar que el manual técnico se apegue a las especificaciones.	Manual de usuario, aprobado cliente Manual técnico, revisado UTI
		3. La Unidad de Telecomunicaciones mantendrá un control de la documentación de los sistemas.	Expediente del sistema
		4. La Unidad de Telecomunicaciones tendrá un estricto control de documentación de documentación, actualizando contenidos según el mejoramiento continuo de los sistemas.	Expediente del sistema
		5. Todos aquellos códigos que sean objeto de programación, ya sean módulo, programas, pantallas, etc. deberán contener información de quién efectuó la programación y en qué fecha; de ser posible en el mismo software, mediante comentarios y adicionalmente en la documentación por escrito.	Control de calidad / Checklist
		6. Después de concluida la programación de una parte del sistema, se deberá registrar en un documento que dicha parte del sistema ha sido concluida(...)	Acta de paso producción: parcial / total
	3.6. Normas para la implementación de sistemas y capacitación.	1. Antes de liberar un nuevo sistema, éste deberá ser sometido a pruebas de aceptación definidas por el área solicitante, utilizando para ello datos reales.(...)	Pruebas de aceptación
		2. La capacitación al personal técnico-operativo formará parte fundamental de la liberación de un sistema. (...)	Plan Capacitación
		3. El proceso de capacitación deberá ser posterior al aprobación de los manuales (...)	Método / Metodología.
	3.7. Normas para el mantenimiento de sistemas	Los usuarios deberán informar, solicitar caminos en el sistema a la Unidad de Telecomunicaciones e Información, siempre y cuando se identifiquen y justifiquen plenamente los ajustes y cambios basados en el reglamento (…)	Control de cambios / Solicitud de cambio
	3.8. Manual de procedimientos para el desarrollo de sistemas (MPDS).	El desarrollo de sistemas se hará según la metodología de desarrollo ágil de sistemas que será aceptado mediante la discusión del grupo de desarrollo para construir un software de calidad, este resultado será descrito en un acta	Acta de reunión - definir metodología.

		que servirá para todos el desarrollo de software de la Institución.	
		Para la etapa de programación será seleccionado el lenguaje de programación considerando que sean de Software Libre, el mismo que después de un consenso, de estabilidad, escalabilidad y tenga su plataforma una vigencia mínima de 10 años.	Acta de reunión - definir lenguaje de programación.
	4.7. Política de desarrollo informático.	Los proyectos de desarrollo informático nacerán como respuesta a la necesidad de cumplimiento de determinados objetivos de la Institución y estarán enmarcados dentro del Plan Estratégico y Plan de Desarrollo Institucional. Por lo tanto los proyectos tendrán siempre objetivos y finalidades específicas y hay que considerarlos como las herramientas para el logro de los objetivos institucionales.	PEDI, POA / Caso de negocio
		Todo proyecto de desarrollo informático debe iniciar con la etapa de planificación, que nos determina el alcance, etapas, tiempo y recursos necesarios para su ejecución.	Caso de negocio
2	Normas de Control Interno de la Contraloría General del Estado.		
	408 ADMINISTRACION DE PROYECTOS		
	408-01 Proyecto	1. En la primera etapa se llevarán a cabo todos los estudios necesarios para determinar la factibilidad de ejecutar el proyecto (...)	Estudio factibilidad
		2. En la etapa de operación, la obra entra en funcionamiento de acuerdo con lo planeado y programado previamente, al tiempo que, en forma simultánea, se implementan el plan y el programa de mantenimiento (...)	Evaluación de resultados previstos vs obtenidos
	408-02 Estudios de pre inversión de los proyectos	Todos los proyectos de obra pública deben estar respaldados por los estudios de preinversión, el procedimiento que se emplee para efectuarlos, el grado de profundidad y los criterios de evaluación que se utilizarán para seleccionar los más ventajosos (...)	Metodologías para estudio de viabilidad
	408-03 Diagnóstico e idea de un proyecto	Toda institución que desee desarrollar un proyecto debe elaborar un diagnóstico, donde se defina claramente el problema por solucionar, la necesidad por satisfacer, los bienes y servicios a ofrecer, quiénes se ven afectados, el impacto en el medio ambiente, y las alternativas de solución que se vislumbran	Diagnóstico de proyecto
	408-04 Perfil del proyecto	El perfil del proyecto abarcará el estudio de los antecedentes, las condiciones económicas, políticas, geográficas y sociales de la zona de influencia en la cual se enmarca; además, las políticas y objetivos de la institución, los aspectos legales y las políticas gubernamentales que afectan el sector al que pertenece el	Perfil del proyecto

		proyecto, todo con el fin de decidir la conveniencia de llevarlo a cabo.	
	408-05 Estudio de prefactibilidad	En esta fase, deben estudiarse los siguientes aspectos del proyecto: su marco legal; la tecnología por emplear y sus implicaciones, el estudio técnico y las normas técnicas; su impacto socio-económico; finalmente, tendrá que efectuarse un estudio del impacto del proyecto sobre el ambiente.	Estudio de impacto del proyecto
	408-06 Estudio de factibilidad	En esta fase se llevará a cabo el anteproyecto o diseño preliminar, así como la ingeniería preliminar del proyecto necesaria para efectuar el diseño definitivo (...)	Anteproyecto
	408-07 Evaluación financiera y socio-económica	Se determinará la rentabilidad utilizando indicadores privados y sociales tales como el VAN o Valor Actual Neto; la TIR o Tasa Interna de Retorno; la razón Beneficio/Costo (B/C), el Período de Retorno de la Inversión (...)	Evaluación financiera ex-ante
	408-08 Diseño definitivo	En esta etapa se elaborarán en detalle todos los documentos y planos constructivos necesarios para llevar a cabo la construcción o ejecución y puesta en operación del proyecto, de conformidad con lo establecido en el análisis técnico de la opción seleccionada en el estudio de factibilidad.	Diseño del proyecto
	408-09 Planos constructivos	Para evitar problemas técnicos o económicos en la construcción de la obra, los planos constructivos deberán tener toda la información necesaria para poder llevarla a cabo (...)	Planos del proyecto
	408-10 Condiciones generales y especificaciones técnicas	Estos documentos constituyen la base para que la administración y el contratista, definan el método de trabajo para cumplir con las condiciones estipuladas.	Especificaciones técnicas
	408-11 Presupuesto de la obra	El presupuesto detallado de la obra es un cálculo de su costo, a partir de los componentes del precio de cada uno de los rubros o de las unidades de obra que conforman el proceso de construcción.	Presupuesto del proyecto
	408-12 Programación de la obra	El método de programación por emplear en esta tarea será cualquier sistema de redes: CPM, PERT, diagrama de bloques, que ponga en relieve las actividades críticas. Además, con base en la red establecida, se elaborará el diagrama de barras correspondiente, diagrama de Gantt, indicando para cada actividad, su duración, los tiempos tempranos y tardíos de inicio y término, las holguras y el requerimiento de insumos: materiales, mano de obra, maquinaria y equipos.	Programa de avance del proyecto

	410 TECNOLOGIA DE LA INFORMACION		
	410-03 Plan informático estratégico de tecnología	La Unidad de Tecnología de la Información elaborará e implementará un plan informático estratégico para administrar y dirigir todos los recursos tecnológicos, el mismo que estará alineado con el plan estratégico institucional y éste con el Plan Nacional de Desarrollo y las políticas públicas de gobierno.	Cartera de proyectos de desarrollo de software
	410-04 Políticas y procedimientos	La Unidad de Tecnología de Información definirá, documentará y difundirá las políticas, estándares y procedimientos que regulen las actividades relacionadas con tecnología de información y comunicaciones en la organización, estos se actualizarán permanentemente e incluirán las tareas, los responsables de su ejecución, los procesos de excepción, el enfoque de cumplimiento y el control de los procesos que están normando, así como, las sanciones administrativas a que hubiere lugar si no se cumplieran.	Propuesta de marco de trabajo metodológico
	410-05 Modelo de información organizacional	El diseño del modelo de información que se defina deberá constar en un diccionario de datos corporativo que será actualizado y documentado de forma permanente, incluirá las reglas de validación y los controles de integridad y consistencia, con la identificación de los sistemas o módulos que lo conforman, sus relaciones y los objetivos estratégicos a los que apoyan a fin de facilitar la incorporación de las aplicaciones y procesos institucionales de manera transparente.	Diccionario de datos
	410-06 Administración de proyectos tecnológicos	La Unidad de Tecnología de Información definirá mecanismos que faciliten la administración de todos los proyectos informáticos que ejecuten las diferentes áreas que conformen dicha unidad. Los aspectos a considerar son:	
		1. Descripción de la naturaleza, objetivos y alcance del proyecto, su relación con otros proyectos institucionales, sobre la base del compromiso, participación y aceptación de los usuarios interesados.	Caso de negocio / Acta constitución proyecto / Documento de Visión
		2. Cronograma de actividades que facilite la ejecución y monitoreo del proyecto que incluirá el talento humano (responsables), tecnológicos y financieros además de los planes de pruebas y de capacitación correspondientes.	Cronograma, recursos, plan de pruebas, plan de capacitación
		3. La formulación de los proyectos considerará el Costo Total de Propiedad CTP; que incluya no sólo el costo de la compra, sino los costos directos e indirectos, los beneficios relacionados con la compra de equipos o programas informáticos, aspectos del uso y mantenimiento, formación para el personal de	Costos

		soporte y usuarios, así como el costo de operación y de los equipos o trabajos de consultoría necesarios	
		5. Se cubrirá, como mínimo las etapas de: inicio, planeación, ejecución, control, monitoreo y cierre de proyectos, así como los entregables, aprobaciones y compromisos formales mediante el uso de actas o documentos electrónicos legalizados.	Actas / documentos
		6. El inicio de las etapas importantes del proyecto será aprobado de manera formal y comunicado a todos los interesados	Acta de constitución de proyecto
		7. Se incorporará el análisis de riesgos. Los riesgos identificados serán permanentemente evaluados para retroalimentar el desarrollo del proyecto, además de ser registrados y considerados para la planificación de proyectos futuros.	Análisis de riesgos
		8. Se deberá monitorear y ejercer el control permanente de los avances del proyecto	Informe de avance de proyecto
		9. Se establecerá un plan de control de cambios y un plan de aseguramiento de calidad que será aprobado por las partes interesadas.	Plan de control de cambios / Plan de Aseguramiento de Calidad
		10. El proceso de cierre incluirá la aceptación formal y pruebas que certifiquen la calidad y el cumplimiento de los objetivos planteados junto con los beneficios obtenidos	Certificación de control de calidad / Acta de paso a producción
	410-07 Desarrollo y adquisición de software aplicativo	2. Adopción, mantenimiento y aplicación de políticas públicas y estándares internacionales para: codificación de software, nomenclaturas, interfaz de usuario, interoperabilidad, eficiencia de desempeño de sistemas, escalabilidad, validación contra requerimientos, planes de pruebas unitarias y de integración.	Método / Metodología
		3. Identificación, priorización, especificación y acuerdos de los requerimientos funcionales y técnicos institucionales con la participación y aprobación formal de las unidades usuarias. Esto incluye, tipos de usuarios, requerimientos de: entrada, definición de interfaces, archivo, procesamiento, salida, control, seguridad, plan de pruebas y trazabilidad o pistas de auditoría de las transacciones en donde aplique.	Especificación de requisitos de software
		4. Especificación de criterios de aceptación de los requerimientos que cubrirán la definición de las necesidades, su factibilidad tecnológica y económica, el análisis de riesgo y de costo-beneficio, la estrategia de desarrollo o compra del software de aplicación, así como el tratamiento que se dará a aquellos procesos de emergencia que pudieran presentarse.	Criterios de aceptación / Estudio de factibilidad / Gestión de riesgos.

		9. Los derechos de autor del software desarrollado a la medida pertenecerán a la entidad y serán registrados en el organismo competente. Para el caso de software adquirido se obtendrá las respectivas licencias de uso.	Registrar ante organismo competente
		10. Formalización con actas de aceptación por parte de los usuarios, del paso de los sistemas probados y aprobados desde el ambiente de desarrollo/prueba al de producción y su revisión en la post-implantación.	Actas de aceptación
		11. Elaboración de manuales técnicos, de instalación y configuración; así como de usuario, los cuales serán difundidos, publicados y actualizados de forma permanente.	Manuales técnico / instalación / configuración / usuario
	410-09 Mantenimiento y control de la infraestructura tecnológica	2. Los cambios que se realicen en procedimientos, procesos, sistemas y acuerdos de servicios serán registrados, evaluados y autorizados de forma previa a su implantación a fin de disminuir los riesgos de integridad del ambiente de producción. El detalle e información de estas modificaciones serán registrados en su correspondiente bitácora e informados a todos los actores y usuarios finales relacionados, adjuntando las respectivas evidencias.	Bitácora de cambios / Comunicación de cambios
		3. Control y registro de las versiones del software que ingresa a producción.	Herramientas versionamiento
		5. Se establecerán ambientes de desarrollo/pruebas y de producción independientes; se implementarán medidas y mecanismos lógicos y físicos de seguridad para proteger los recursos y garantizar su integridad y disponibilidad a fin de proporcionar una infraestructura de tecnología de información confiable y segura	Metodología
	410-10 Seguridad de tecnología de información	5. Implementación y administración de seguridades a nivel de software y hardware, que se realizará con monitoreo de seguridad, pruebas periódicas y acciones correctivas sobre las vulnerabilidades o incidentes de seguridad identificados.	Plan de pruebas / Requerimientos de seguridad / Gestión de riesgos

6.2. Análisis de metodologías ágiles.

Acorde a lo planteado en el Objetivo 2: “Analizar las metodologías de desarrollo software, marcos de trabajo y enfoques más relevantes: XP, SCRUM, KANBAN, LEAN y DevOps; mediante una revisión bibliográfica; para identificar las mejores prácticas, procesos y entregables.”, se procedió a realizar una revisión bibliográfica y acorde a investigaciones similares [28] se determinó las características comunes a analizar.

En la Tabla VIII, se muestra las características comunes que compartes de las diferentes metodologías, marcos de trabajo y enfoques, los mismos que se orientan ejecución de un proyecto. De esta manera para ejecutar un proyecto se requiere de personas que ejecuten diferentes actividades (procesos), enmarcadas en principios, valores o prácticas; para lo cual cada persona debe cumplir una o varias funciones (roles) para obtener un producto o servicio (entregables).

TABLA VIII
CARACTERÍSTICAS COMUNES DE XP, SCRUM, KANBAN, LEAN Y DEVOPS

CARACTERÍSTICA	DESCRIPCIÓN
Roles	Funciones que desempeña una persona en un proceso.
Procesos	Conjunto de fases o actividades para lograr un objetivo.
Mejores prácticas	Prácticas, valores, principios.
Entregables o artefactos	Es un resultado de haber ejecutado un proceso o actividad.

En la Tabla IX se realiza una comparación de los roles entre Xp, Scrum, Kanban, Lean y DevOps, para lo cual se ha tomado como base los roles de Scrum por ser más genéricos y adaptarse a cualquier tipo de proyecto; y, a partir de ellos, se agrupó acorde a las responsabilidades y funciones de cada rol.

TABLA IX
COMPARATIVA DE ROLES XP, SCRUM, KANBAN, LEAN Y DEVOPS

SCRUM	XP	DEVOPS	KANBAN	LEAN
Product Owner	Tracker (Encargado de seguimiento) Big boss (Gestor)	Product owner	Service Request Manager (Gestor de Solicitudes de Servicio)	Lean Team Members
Scrum Master	Coach (Entrenador)	Team lead (Jefe de equipo)	Service Delivery Manager (Gestor de Prestaciones de Servicio)	
Equipo Scrum	Programador Tester (Encargado de pruebas)	Cloud Architect Site Reliability Engineer (SRE) Administrador del Sistema DevOps	Organización, programa, equipo (no requiere roles predefinidos)	
Stakeholder(s): Cliente, usuarios, patrocinador (sponsor)	Cliente	--	--	Clientes, usuarios
Vendedores (individuos u organizaciones externas)	Consultant (Consultor)			
Guidance Body	--	--	--	--

En la Tabla X, se muestra la comparación realizada de los procesos de los diferentes marcos de trabajo, metodologías y enfoques.

TABLA X
COMPARATIVA DE PROCESOS XP, SCRUM, KANBAN, LEAN Y DEVOPS

XP	SCRUM	DEVOPS	KANBAN	LEAN
Exploración	Inicio	--	Por hacer	Petición Aprobación proyecto
Planificación	Planificación y estimación	Planificación	Planificado	Arquitectura preliminar
Iteraciones	Implementación	Desarrollo	En progreso	Seleccionar alternativas
	Revisión y retrospectiva	Pruebas	Pruebas	Revisión con el cliente
Producción	Lanzamiento	Integración Entrega Implementación	Desplegado	Despliegue
Mantenimiento	--	Monitoreo	--	
Cierre	--	--	Terminado	

En la Tabla XI, se enuncian las mejores prácticas, técnicas, principios de las diferentes metodologías águilas. Se puede apreciar que en algunas de ellas coinciden ciertas prácticas como incluir al cliente, retroalimentación, iteraciones, mejora continua, entrega continua, etc.

TABLA XI
MEJORES PRÁCTICAS XP, SCRUM, KANBAN, LEAN Y DEVOPS

METODOLOGÍA	MEJORES PRÁCTICAS / TÉCNICAS / PRINCIPIOS
XP	1. Test-driven development (Primero se construye las pruebas) 2. Planning game (Reunión entre equipo y clientes, para revisión y planificación de tareas) 3. On-site customer (Un representante del equipo del cliente forme parte del equipo de desarrollo) 4. Pair programming (2 personas que trabajan en conjunto, el primero escribe el código y el segundo lo verifica y propone mejoras) 5. Small releases (Los cambios se publican de forma rápida en entregas cortas, de tal forma que se obtenga feedback y apliquen correcciones lo antes posible) 6. Simple design (que permita entender a todos el proyecto) 7. Coding-standards (que se comparta la forma y formato de escribir el código) 8. Collective code ownership (que la responsabilidad, éxitos y errores, sean compartidos por todo el equipo) 9. System metaphor (Utilizar atributos, nombres de clases descriptivos para facilitar su comprensión)

SCRUM	1. Centrado en el cliente 2. Retroalimentación continúa (reuniones) 3. Mejora continua 4. Entrega continua de valor 5. Control del proceso empírico 6. Auto organización 7. Colaboración 8. Priorización basada en el valor, 9. Time boxing 10. Desarrollo iterativo.
KANBAN	1. Visualizar 2. Limitar el Trabajo en Curso 3. Gestionar el Flujo de Trabajo 4. Hacer Explícitas las Políticas 5. Implementar las Cadencias de Kanban (o ciclos de Feedback) 6. Mejorar en colaboración, evolucionar experimentalmente
LEAN	1. Definir el valor y hacerlo desde el prisma del cliente, que es quien necesita una solución. 2. Determinar la cadena de valor para poder mejorar, eliminando los desperdicios. 3. Crear un flujo dinámico en el que siempre se aporte valor. 4. Generar el tirón o pull del cliente, cuyo pilar sea la demanda real y no una perspectiva a largo plazo. 5. Mejora constante para conseguir la excelencia.
DEVOPS	1. Pruebas Automatizadas Continuas 2. Integración Continua 3. Despliegue Continuo 4. Monitoreo Continuo

En la Tabla XII, se aprecia los artefactos más relevantes en cada una de las metodologías, en las que se puede destacar: la visión del proyecto, historias de usuario, planes, tableros.

TABLA XII
ARTEFACTOS XP, SCRUM, KANBAN, LEAN Y DEVOPS

METODOLOGÍA	**ARTEFACTOS**
XP	1. Visión 2. Historias de Usuario 3. Pruebas de aceptación 4. Plan de lanzamiento 5. Sistema de nombres 6. Plan de iteración 7. Estándares de codificación 8. Pruebas Unitarias 9. Código
SCRUM	1. Declaración de la visión del proyecto 1. Historias de usuario 2. Backlog del producto 2. Backlog del sprint 3. Incremento 4. Panel de tareas
KANBAN	1. Tablero kanban

LEAN	1. Mapa de flujo de valor (VSM) 2. Kanban 3. Poka-yoke 4. Andon 5. Heijunka 6. Kaizen 7. Jidoka
DEVOPS	1. PipeLine (hoja de ruta de implementación de Devops) 2. Herramientas empleadas por cada etapa del pipeline propuesto

6.3. Propuesta del marco de trabajo metodológico de desarrollo de software.

Objetivo 3: Realizar la propuesta del marco metodológico de desarrollo de software, acorde a la normativa vigente y a las mejores prácticas de las metodologías, marcos de trabajo y enfoques analizados; en base a los resultados del análisis elaborado; para estandarizar los procesos de desarrollo de software.

6.3.1. CONSIDERACIONES

A continuación, se detallan algunas consideraciones que deben ser tomadas en cuenta, acorde a la normativa vigente:

1. Todo proyecto de desarrollo de software debe estar enmarcado en el Plan Estratégico de Desarrollo Institucional (PEDI). Se justifica en el requerimiento de software y caso de negocio.
2. La DTI, debe elaborar la cartera de proyectos de desarrollo de software (plan informático, plan de desarrollo de software) el mismo que debe estar alineado al PEDI.
3. Para el inicio de un proyecto nuevo, la unidad requirente, debe solicitarlo de manera formal y adjuntar el formato de requerimiento de software; así mismo, en coordinación con la DTI debe elaborar el caso de negocio en donde exponga claramente su necesidad.
4. Las unidades requirentes deben participar activamente el proceso de desarrollo de software. Acorde a la normativa vigente y metodologías ágiles.
5. La DTI, debe elaborar un informe técnico con el estudio de factibilidad.
6. Se debe considerar el análisis y diseño orientado a objetos acorde a la normativa vigente (Usar UML).
7. El inicio de las etapas importantes de un proyecto será aprobado de manera formal y comunicado a todos los interesados, para lo cual se propone el Acta de Constitución de Proyecto elaborado en la fase de exploración.

8. El proceso de cierre parcial o final de un entregable debe incluir la aceptación de software, para lo cual se lleva el Acta de Paso a Producción.
9. Para poner en producción un entregable se debe ejecutar el control de calidad y emitir el respectivo certificado.
10. Se debe elaborar manual de usuario (aprobado por el cliente), procedimientos, técnico (configuración).
11. Todas las fases del desarrollo de sistemas deben estar documentadas y deben constar en el expediente del proyecto y/o repositorio de código fuente.
12. Se debe documentar quién y cuándo realizó o actualizó el código fuente, arquitectura o artefacto; de no ser posible se debe dejar en la bitácora de cambios u otros instrumentos como actas.
13. La capacitación debe ser realizada luego de la elaboración y aprobación de los manuales.
14. Acorde a la normativa se debe llevar un diccionario de datos, el mismo que debe ser actualizado acorde a los cambios realizados.

6.3.2. ROLES

Lo roles de Scrum, son generales, lo que permite integrar al equipo scrum a especialistas de otras áreas, procesos o metodologías (DevOps, Kanban, Xp, Lean). Así mismo se integra el rol de cliente (XP) y rector, a los roles centrales de la propuesta listado en la Tabla XIII, debido que a que el desarrollo de software debe estar dirigido a él; así mismo se considera los cargos existentes en la DTI, para asignar de manera fácil las responsabilidades.

TABLA XIII
PROPUESTA DE ROLES

ROLES	ROL SCRUM	CARGO / UNIDAD / ESTAMENTO
Cliente	Cliente (Stakeholder)	
Rector	Patrocinador (Stakeholder)	Rector de la UNL
Director TI	Product Owner	Director de Tecnologías de Información
Especialista SI	Scrum Master	Especialista de Sistemas de Información

Equipo SI	Equipo Scrum	**# Desarrollo (Analistas, programadores, tester, seguridad)** Analista de Sistemas de Información 2 Analista de Sistemas de Información 1 Analista de Seguridad Informática **# Operaciones (Seguridad)** Especialista de Seguridad Informática Analista de Seguridad Informática **# Operaciones (Infraestructura)** Especialista de Infraestructura Tecnológica Analista de Infraestructura Tecnológica 2 Analista de Infraestructura Tecnológica 1 **# Operaciones (Capacitación, asistencia técnica)** Especialista de Provisión de Servicios Analistas de Provisión de Servicios Técnico de Provisión de Servicios
Stakeholder(s)	Stakeholder(s)	Usuarios: Docentes, estudiantes, administrativos, trabajadores, etc.

6.3.3. CICLO DE VIDA

Cada metodología utiliza su propio ciclo de vida, pero al haber hecho un análisis comparativo de las mismas, se concluye que existen similitudes; así mismo que se tratan de proyectos de desarrollo de software, la realidad organizacional y normativa vigente se propone el siguiente ciclo de vida acorde a la Tabla XIV:

TABLA XIV
PROPUESTA DE CICLO DE VIDA

#	FASES	PROCESOS FUNDAMENTALES
1	**Inicio** (Concepción)	Formulación del requerimiento Elaboración del caso de negocio Estudio de factibilidad Aprobación del proyecto Plan de desarrollo de software (inclusión, priorización)
2	**Exploración** (Planificación inicial)	Acta de constitución del proyecto. Desarrollar épicas Crear el backlog priorizado del producto Plan de lanzamiento
3	**Implementación** (Iteraciones)	# **Planificación del sprint** Refinar historias de usuario Estimar historias de usuario Comprometer historias de usuario Crear el backlog del sprint

		# **Diseño** Diseñar de la arquitectura del sistema Diseñar de interfaces de usuario Diseñar de componentes #**Codificación** Codificación y documentación Pruebas unitarias Pruebas de componente #**Pruebas** Elaborar plan de pruebas Elaborar casos de prueba Ejecutar pruebas
4	**Revisión y retrospectiva**	Demostrar y validar el sprint Documentar (Manuales) Capacitar a los usuarios
5	**Lanzamiento (Producción)**	Certificar el control de calidad Elaborar checklist de paso a producción Elaborar acta de paso a producción (parcial / final) Extraer respaldos de ser necesario Ejecutar paso a producción Validar la puesta en producción.
6	**Mantenimiento (xp, devops)**	Monitoreo Asistencia técnica Mejoras y corrección de errores

En la Fig. 1 se muestra un diagrama de procesos del ciclo de vida, en donde cada fase corresponde a un subproceso; en la fase de inicio se hace la concepción y aprobación del proyecto; en la fase de planificación inicial se crea el backlog priorizado del producto; en la fase de implementación se construye el producto, pudiéndose repetir las veces que sea necesario acorde a lo planificado (sprint); en la fase de revisión y retrospectiva se demuestra el producto y valida el producto; en la fase de lanzamiento se entrega el producto a los usuarios y en la fase de mantenimiento se monitorea el producto y se brinda asistencia técnica.

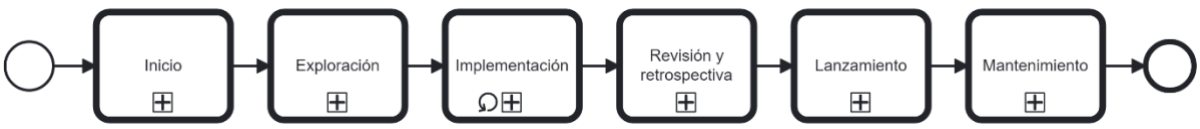


Fig. 1. Ciclo de vida para la gestión de proyectos

La Fig. 2, muestra las actividades fundamentales ejecutadas desde el planteamiento del requerimiento hasta la aprobación de este, en otras palabras, en esta fase se realiza la concepción del proyecto, aprobación e inclusión en la cartera de proyectos de desarrollo de software. Una vez aprobado el proyecto, acorde a la priorización realizada puede entrar a ejecución inmediata o puede quedar a la espera hasta que existan los recursos disponibles.

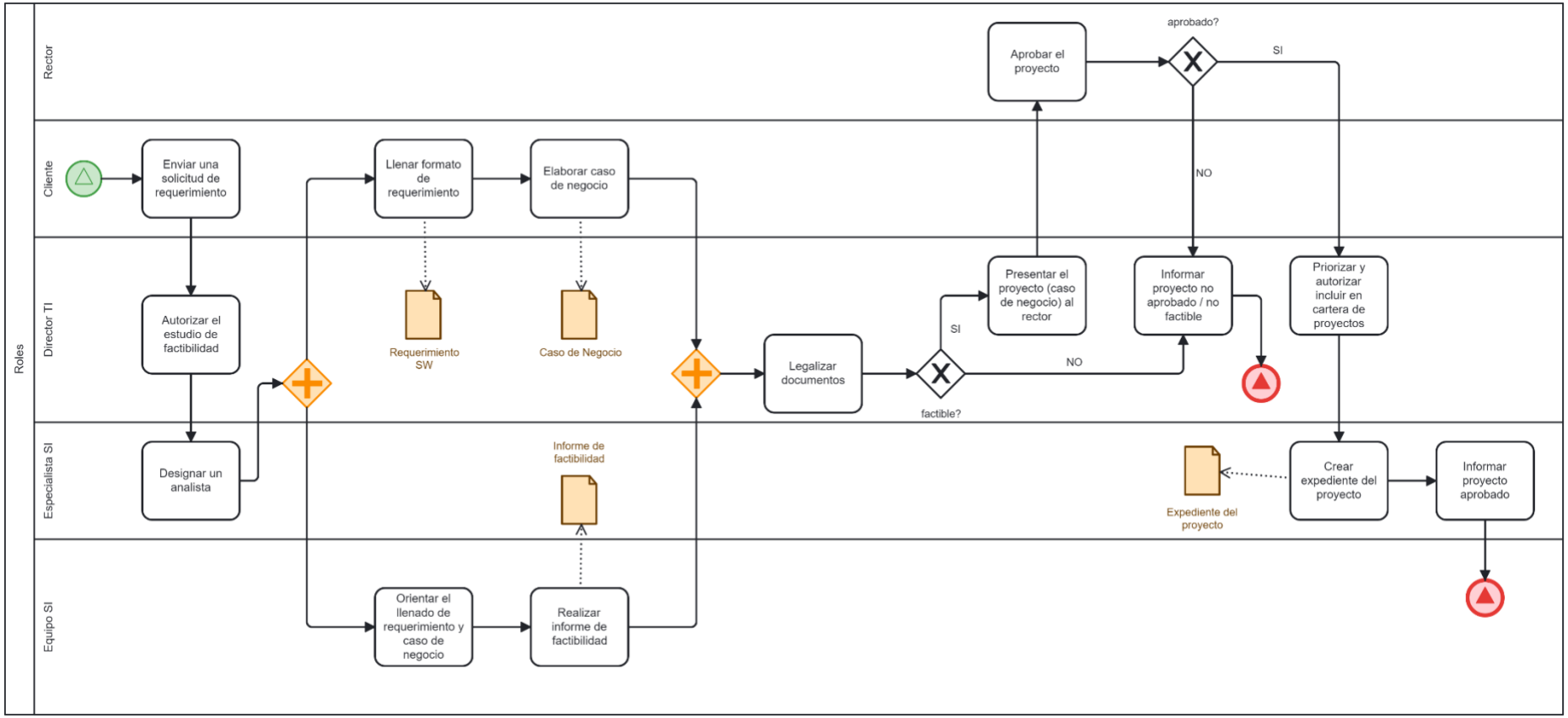


Fig. 2. Fase Inicio - Ciclo de vida para la gestión de proyectos

En la Fig. 3, se detalla las principales actividades de la fase de exploración, la mismas que consiste en formar el equipo scrum (Equipo SI), identificar los stakeholders, plantear épicas, crear la visión del producto, crear el backlog priorizado del producto y realizar un plan de lanzamiento.

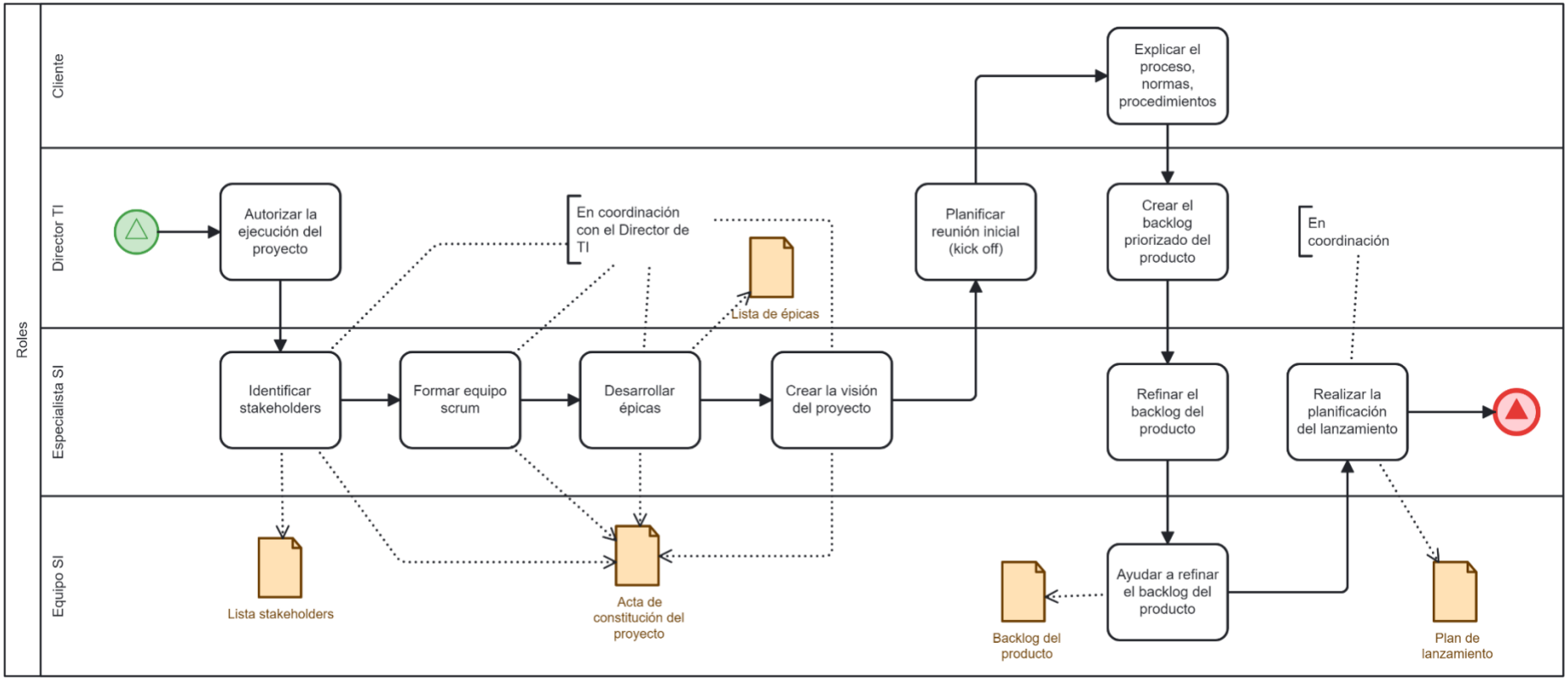


Fig. 3. Fase Exploración - Ciclo de vida para la gestión de proyectos

La Fig. 4, representa de manera generar la fase de implementación, en donde mediante una reunión de planificación se identifica las historias de usuario a ejecutar en el sprint, el equipo refina las historias de usuario con el cliente y a la par va realizando el diseño y codificación. Las pruebas también pueden irse planificando al mismo tiempo (plan de pruebas y casos de prueba) y se ejecutan conforme se vaya finalizando las historias de usuario.

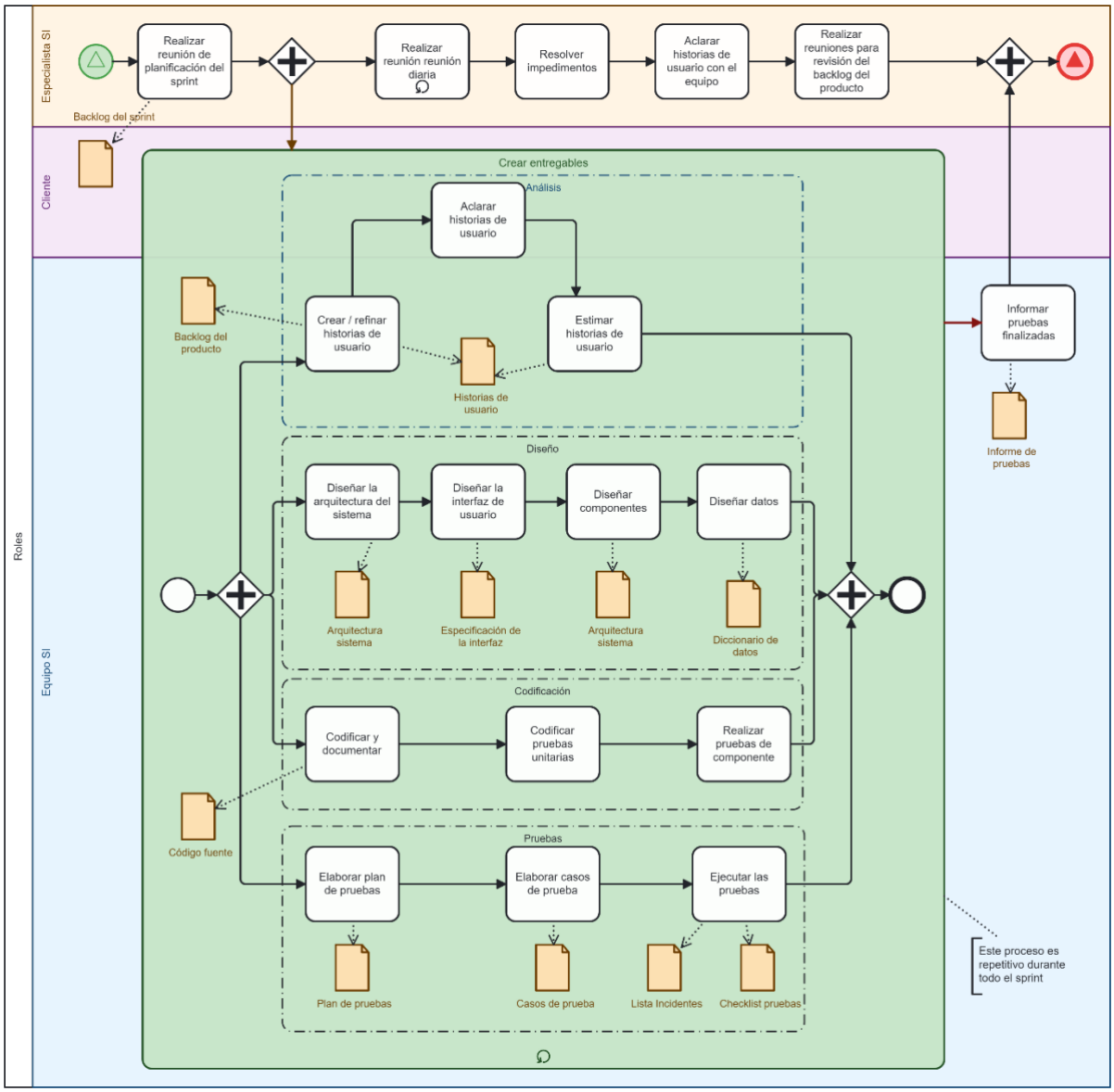


Fig. 4. Fase Implementación - Ciclo de vida para la gestión de proyectos

En la Fig. 5, se muestra la fase de revisión y retrospectiva, la misma que se orienta a presentar el entregable al cliente, validar que cumpla lo solicitado y obtener alguna retroalimentación en caso de requerirse alguna mejora o identificarse nuevas necesidades; en dicho caso se puede negociar los cambios, pudiendo llegar a acuerdos de cambiar la prioridad de algunas historias de usuario, dejar sin efecto alguna historia de usuario, poner en producción el entregable o poner en producción el entregable con el próximo sprint.

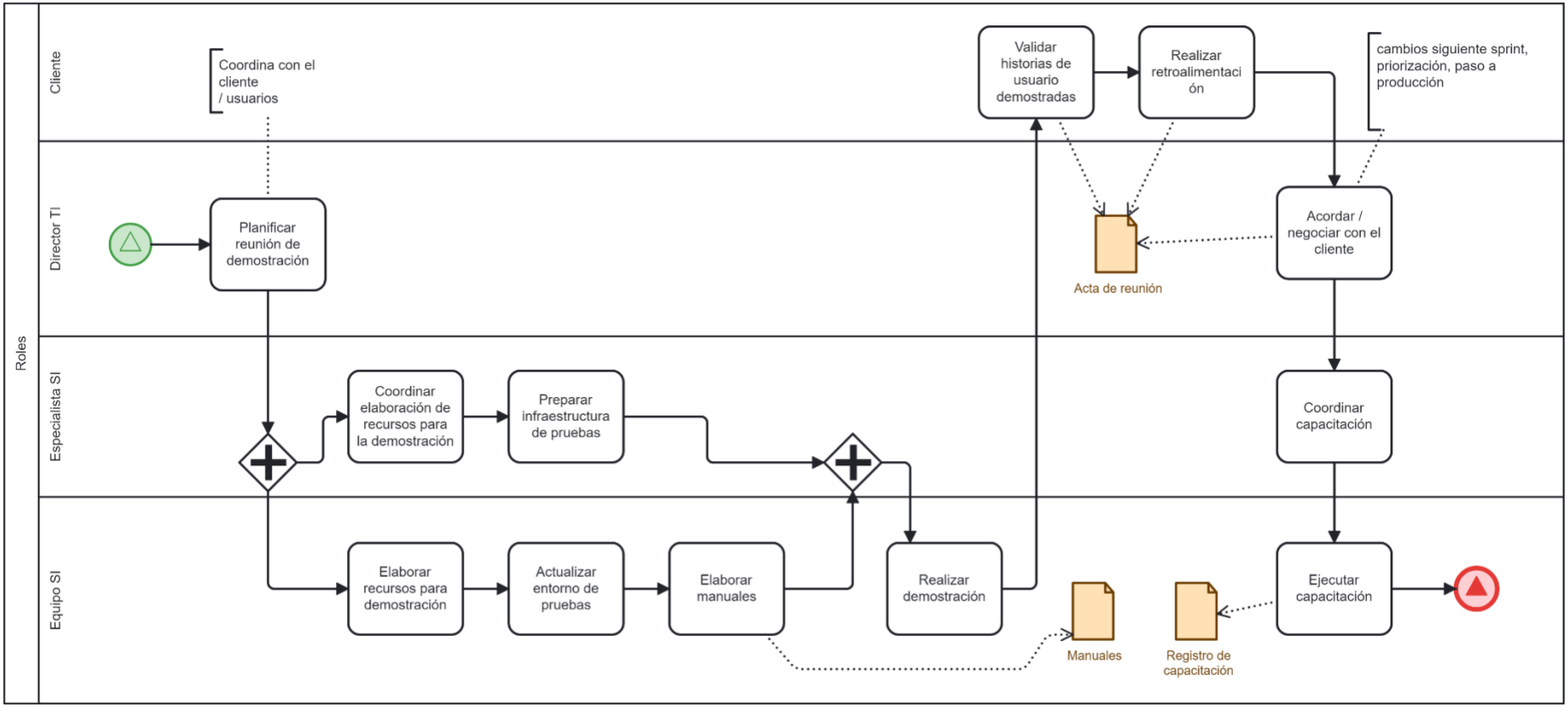


Fig. 5. Fase Revisión y retrospectiva - Ciclo de vida para la gestión de proyectos

La Fig. 6, detalla la fase de lanzamiento, en donde se puede ver que se emite la certificación de control de calidad, el acta de paso a producción, se configura el Pipe Line DevOps, se crea el reléase y finalmente se activan los cambios en producción a través del pipe line automatizado.

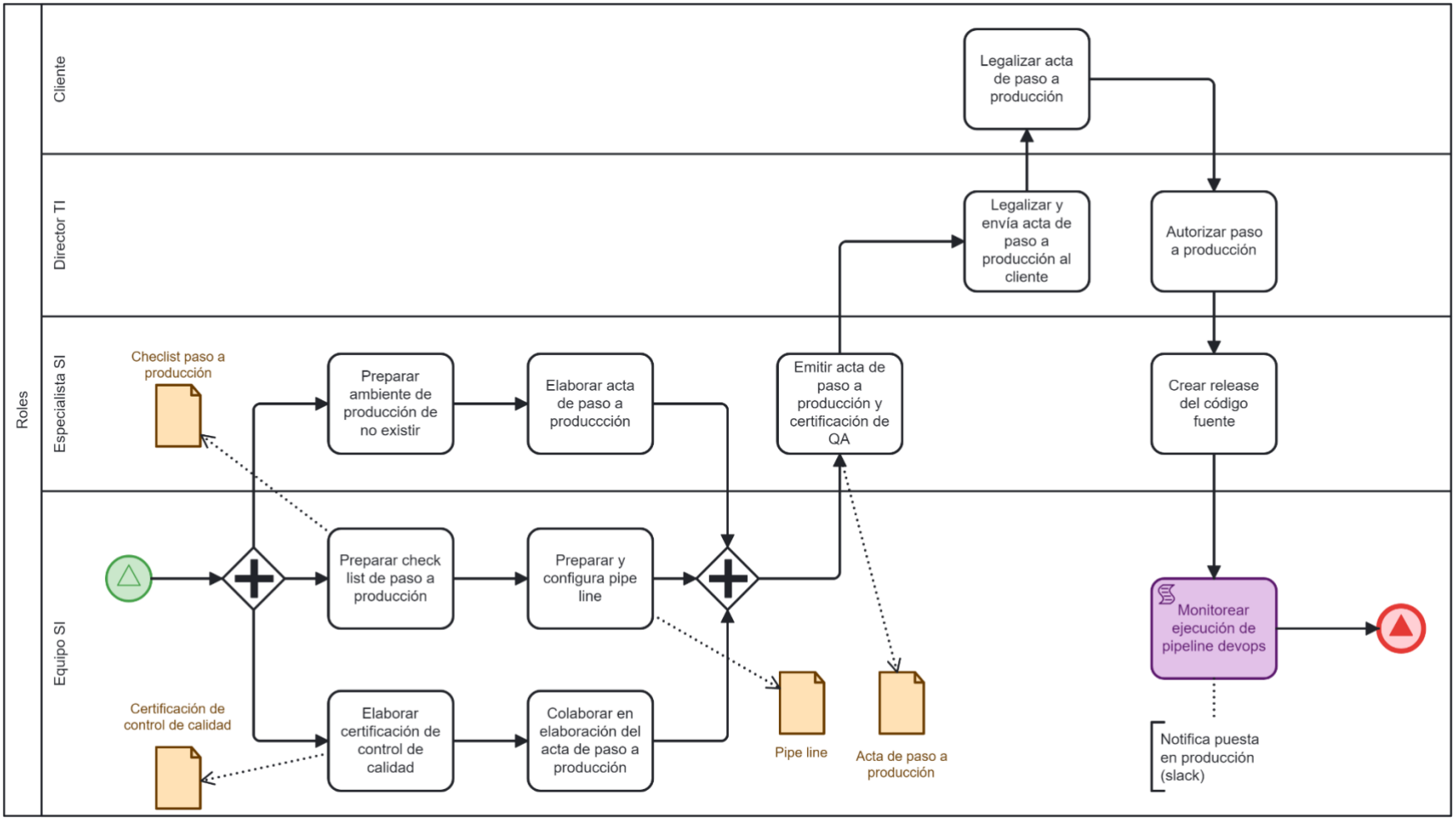


Fig. 6. Fase Lanzamiento - Ciclo de vida para la gestión de proyectos

En la Fig. 7, se aprecia la fase de mantenimiento, en donde especialmente después de haber puesto en producción un sprint se hace un monitoreo para garantizar el correcto funcionamiento e integración del sistema; también se brinda asistencia técnica y en caso de detectarse algún incidente (bug o mejora corta) se lo corrige de inmediato, de lo contrario se gestiona el cambio para ejecutarlo en el siguiente sprint; adicionalmente se generan métricas con la finalidad de que sirva de retroalimentación para nuevos sprints y proyectos.

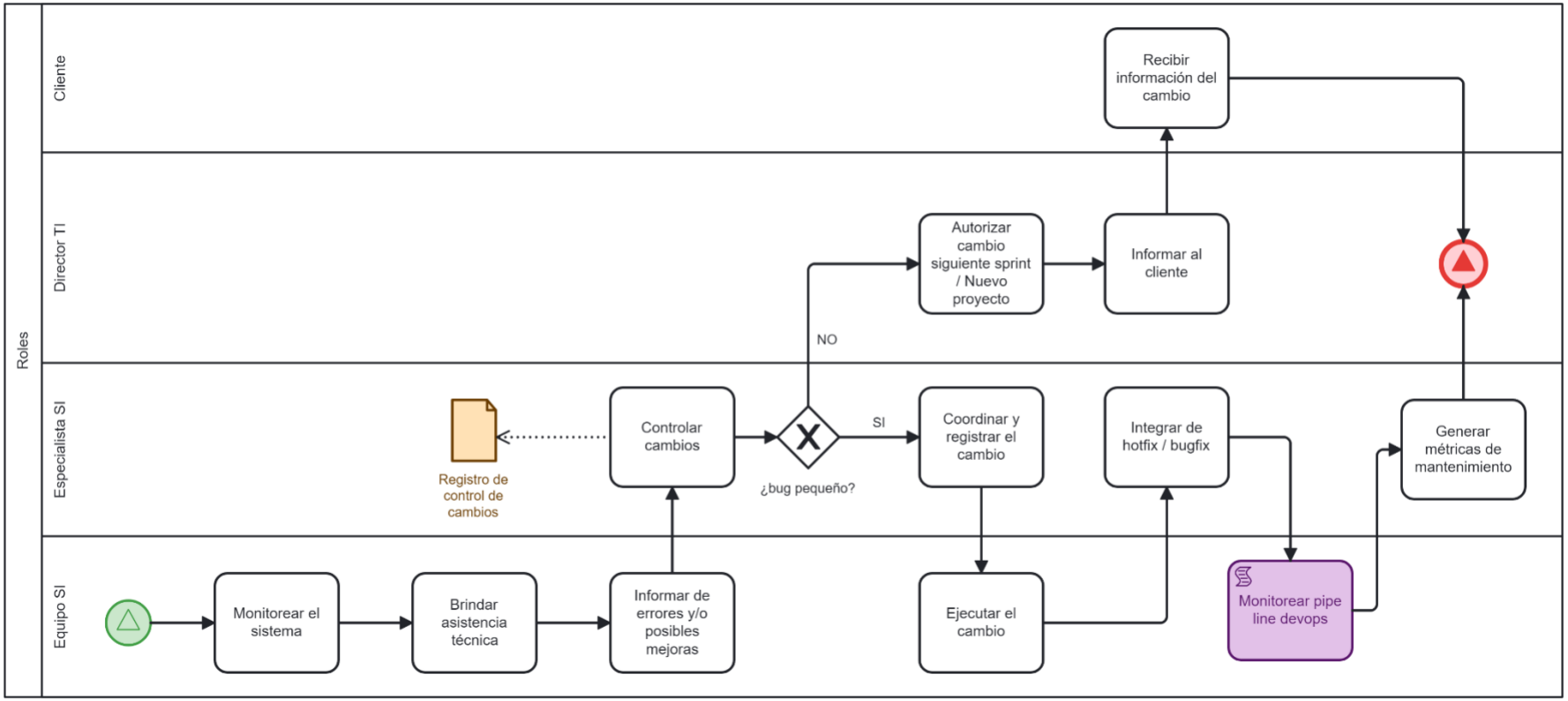


Fig. 7. Fase Mantenimiento - Ciclo de vida para la gestión de proyectos

6.3.4. MEJORES PRÁCTICAS

La Tabla XV, detalla las mejores prácticas consideradas de las diferentes metodologías, las mismas que se considera acorde a la experiencia en los proyectos ejecutados en la Dirección de Tecnologías de Información y a la mejora de procesos.

TABLA XV
PROPUESTA DE MEJORES PRÁCTICAS

#	PRÁCTICA	DESCRIPCIÓN
1	Transparencia	Todas las fases y/o actividades pueden ser observadas por el equipo y stakeholders.
2	Auto organización	El equipo se organiza por cuenta propia.
3	Comunicación	La comunicación debe ser efectiva no solo entre el equipo scrum, sino con el cliente con la finalidad de obtener retroalimentación en el menor tiempo posible.
4	Colaboración	El quipo trabaja en conjunto con los stakeholders crear, aclarar y validar los resultados del proyecto.
5	Priorización basada en el valor	Priorizar que es lo que debe hacerse primero y entregar valor al cliente lo antes posible.
6	Bloques de tiempo (Time boxing)	Usar los bloques de tiempo para: sprint (1 a 6 semanas), reunión diaria, reunión de planificación del sprint, reunión de revisión del sprint, reunión de retrospectiva del sprint.
7	Desarrollo iterativo	Entrega de valor al cliente en el menor tiempo posible, obtenido retroalimentación
8	Diseño simple	Aplicado en todos los ámbitos del proyecto (Análisis, diseño, arquitectura, pantallas, etc.)
9	Automatización de procesos	Automatizar todo lo que se pueda: pruebas, integración, despliegue, monitoreo, etc.

6.3.5. ARTEFACTOS

La Tabla XVI, lista los artefactos recomendados correspondientes al ciclo de vida de desarrollo de software en la DTI y a la normativa vigente. Se enuncia el nombre de artefacto (Con * los que se deben considerar obligatorios); el rol o responsable principal (marcado en *) y los roles de apoyo o con los que se debe coordinar; y, una descripción corta del artefacto.

TABLA XVI
PROPUESTA DE ARTEFACTOS

#	ARTEFACTO	ROL / RESPONSABLE	DESCRIPCIÓN
1	Requerimiento de Software	*Cliente Equipo SI	Es un documento que aclara de manera corta a alto nivel el requerimiento, justificando y delimitando un posible plazo desde el criterio del cliente (Ver **Anexo 7**).
2	Caso de Negocio	*Cliente Equipo SI	Expone de una manera más amplia el requerimiento, considerando alternativas, justificación, hitos y trata de vender la idea de proyecto. Se lo presenta al rector para su aprobación (Ver **Anexo 8**).

3	*Informe de Factibilidad	*Equipo SI Especialista SI	Informe técnico de factibilidad. Se lo presenta al Director TI y se lo considera para la presentación del Caso de Negocio al Rector (Ver **Anexo 9**).
4	*Lista Stakeholders	*Especialista SI Director TI Equipo SI	Lista de clientes, usuarios y personas de interés que pueden aportar con ideas o ayuden a aclarar los requerimientos (Ver **Anexo 10**).
5	*Acta de constitución del proyecto	*Especialista SI Director TI Equipo SI	Formalización del inicio del proyecto, contiene la visión inicial del proyecto con los requerimientos de alto nivel (Ver **Anexo 11**).
6	*Épicas	*Especialista SI Director TI Equipo SI	Descripción macro de un requerimiento de usuario (Ver **Anexo 12**).
7	*Historia de usuario	* Director TI * Especialista SI * Equipo SI * Cliente	Descripción pequeña de cada uno de los requerimientos de un cliente. Se usa el formato tabla por facilidad de manejo de la información (Ver **Anexo 12**).
8	*Backlog del producto	* Director TI * Especialista SI * Equipo SI * Cliente	Listado de requerimientos priorizados de todo el producto o servicio a desarrollar (Ver **Anexo 12**).
9	*Backlog del sprint	* Equipo TI Especialista SI Director TI	Listado de requerimientos a ser desarrollador un en sprint o iteración (Ver **Anexo 12**).
10	*Plan de lanzamiento	* Especialista SI Director TI	Cronograma del lanzamiento a alto nivel: épicas, hitos o sprints (Ver **Anexo 13**)
11	*Arquitectura del sistema	* Equipo TI Especialista SI	Diagramas y/o otros recursos relacionados al diseño del sistema. Se proponer usar el modelo 4+1 con los siguientes diagramas: clases, componentes, despliegue y procesos.
12	Especificación de la interfaz	* Equipo TI Especialista SI	Prototipado de pantallas o diseño de línea gráfica institucional.
13	*Diccionario de datos	* Equipo TI Especialista SI	Acorde a la normativa, debe realizar o actualizar el diccionario de datos (Ver **Anexo 14**).
14	*Código fuente	* Equipo TI Especialista SI	Código fuente y/o recursos relacionados a la construcción del proyecto.
15	*Plan de pruebas	* Equipo TI Especialista SI	Plan para ejecutar las pruebas acordes a los requerimientos del sprint y requerimientos de seguridad del producto (Ver **Anexo 19**).
16	*Casos de prueba	* Equipo TI Especialista SI	Define que es necesario validar para asegurar la calidad del producto (Ver **Anexo 20**).
17	Lista de incidentes	* Equipo TI Especialista SI	Lista de errores y mejoras determinadas al ejecutar las pruebas (Ver **Anexo 17**).
18	Informe de pruebas	* Equipo TI Especialista SI	Informe técnico de ejecución de las pruebas. Adjunta anexos de casos de prueba y lista de incidentes (Ver **Anexo 18**)
19	Acta de reunión	* Equipo TI Especialista SI Director TI	Acta de reunión (minuta), de reuniones relentes: planificación, validación de las pruebas, etc. (Ver **Anexo 19**).
20	*Registro de capacitación	* Equipo TI Especialista SI Director TI	Registro de las capacitaciones brindadas en físico y/o reporte de participantes de llevarse a cabo en laguna plataforma (Zoom, EVA, etc.)

21	*Manual de usuario	* Equipo TI Especialista SI Director TI	Manual de usuario aprobado por el cliente
22	*Manual técnico	* Equipo TI Especialista SI Director TI	Manual técnico y/o configuración revisada por la DTI
23	*Manual de procedimientos	* Equipo TI Especialista SI Director TI	Manual de procedimientos y/o diagramas de procesos.
24	*Expediente del proyecto	* Especialista SI * Equipo TI	Expediente con las actividades de gestión del proyecto y recursos técnicos de la ejecución del mismo.
25	Checklist de control de calidad	* Equipo TI Especialista SI	Checklist de verificación de actividades recurrentes o errores comunes: verifica recursos de análisis, diseño, codificación (documentación, autor fecha de cambios), pruebas (errores comunes), ect.
26	Checklist de paso a producción	* Equipo TI Especialista SI	Ante los cambios efectuados es necesario hacer un checklist para activarlos en producción: configuraciones, catálogos, parámetros, procesos, etc.
27	Certificado de control de calidad	* Especialista SI	Certificado emitido por la DTI, que indica que se ha pasado correctamente el control de calidad (Ver **Anexo 18**).
28	Pipe line	* Equipo TI Especialista SI	Pipe line gráfico actualizado acorde el proyecto o scripts del pipeline.
29	Acta de paso a producción	* Equipo TI * Especialista SI * Director TI * Cliente Rector	Indica que un entregable, módulo, sistema ha sido culminado y se entrega al usuario para su uso. El acta puede ser parcial o definitiva (Ver **Anexo 20**).
30	Registro de control de cambios	* Equipo TI Especialista SI	Bitácora de control de cambios.
31	Guía de desarrollo de software	* Especialista SI * Equipo TI	Guía de estándares de codificación, estilos, arquitectura, uso de herramientas, etc. En el **Anexo 21** se muestra una captura de pantalla del contenido debido a que este documento es de uso interno.

6.3.6. HERRAMIENTAS Y PIPE LINE DEVOPS

En la Fig. 48. (propuesta de pipe line DevOps) y Tabla XVII PROPUESTA DE HERRAMIENTAS, se muestran las herramientas elegidas para la presente propuesta metodológica, se puede profundizar sobre la definición y características de estas en el apartado de 4. Marco Teórico en su sección de Herramientas para el proceso de desarrollo de software.

TABLA XVII
PROPUESTA DE HERRAMIENTAS

#	ETAPA	HERRAMIENTA	VENTAJAS
1	Planificación	Taiga	- Software Libre - Se acopla con la metodología SCRUM, define sprints y se apoya en Kanban para la gestión de las tareas.
		Google Drive	- Capa gratuita para instituciones educativas. - Variedad de servicios que se acoplan a las necesidades de una organización.
2	Código	GitLab	- Software Libre - Permite su instalación en un servidor privado virtual, para contar con una instancia propia para la institución.
		PyCharm Professional	- Versión avanzada para el desarrollo de software con Python. - Permite obtener la licencia profesional con una cuenta institucional .edu. - Multiplataforma.
		Visual Studio Code	- Gratuito - Multiplataforma
		GitHub Copilot	- Generación de código. - Permite su uso mediante el uso de una cuenta institucional.
		Docker	- Software Libre - Permite estandarizar dependencias y distribuir de forma eficiente las aplicaciones, tanto entre el equipo de desarrollo como para desplegar en los ambientes de desarrollo, pruebas y producción.
3	Diseño	www.diagrams.net	- Facilitad para almacenar los diagramas creados en la nube de Google drive. - Variedad de tipo de diagramas soportados.
		Camunda Modeler	- Diagramado estricto a los estándares. - Multiplataforma
4	Pruebas y Despliegue	Jenkins	- Software Libre - Automatiza el proceso de pruebas mediante la Integración Continua - Se puede configurar la Entrega Continua.
		Sonar Lint	- Se incorpora al IDE de desarrollo - Apoya a la detección temprana de errores y vulnerabilidades del código.
		Sonar Qube	- Análisis de código - Integración a una arquitectura DevOps - Escáner de aplicaciones de web.

		OWASP ZAP	- Puede ser empleado por técnicos de seguridad o por los miembros del equipo de desarrollo.
		PyTest	- Permiten ejecutar pruebas a las funcionalidades y componentes críticos del sistema.
5	Documentación	Google Drive	- Capa gratuita para instituciones educativas. - Variedad de servicios que se acoplan a las necesidades de una organización.
6	Comunicación	Slack	- Herramienta para gestionar la comunicación en una organización. - Fácil integración con herramientas como GitLab.
		Discord	- Herramienta colaborativa - Permite su integración con GitLab - En la capa gratuita permite el almacenamiento ilimitado de archivos y mensajes (archivos de hasta 25MB). - Videollamadas grupales en la capa gratuita

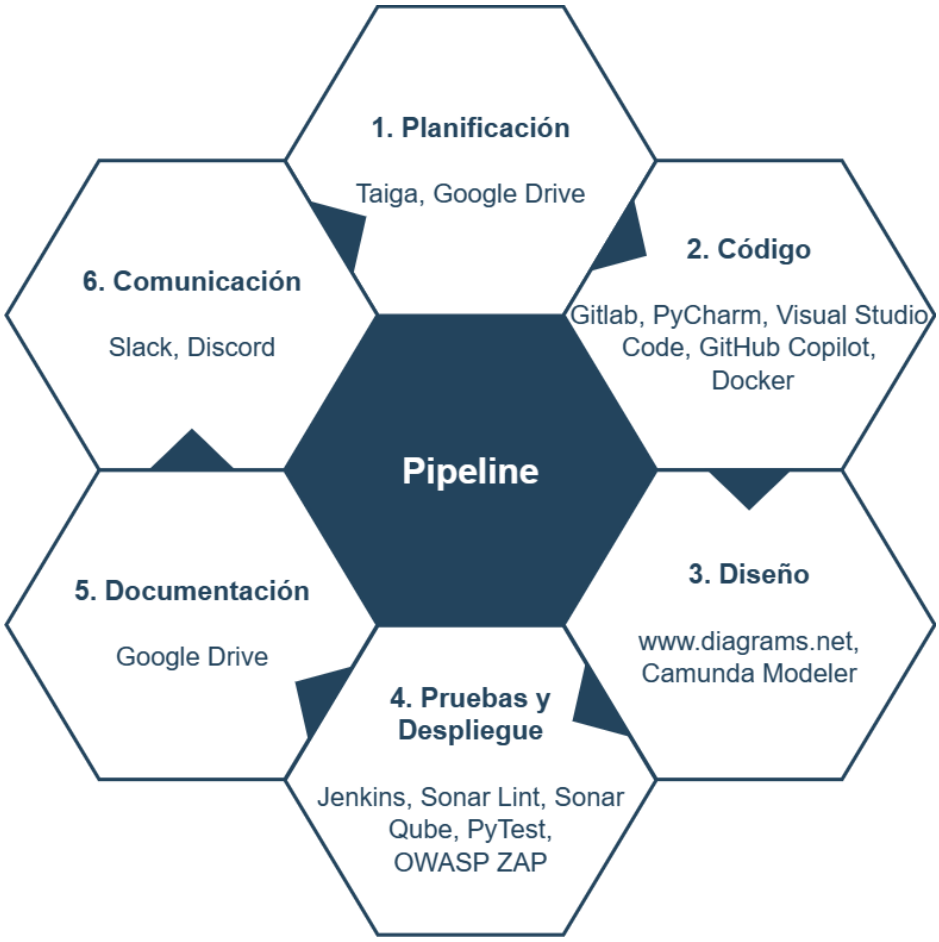


Fig. 8. Propuesta de pipe line DevOps (Autoría propia)

7. Discusión

Iniciar un proyecto de desarrollo de software implica como etapa fundamental para el equipo de trabajo, la selección de una metodología de desarrollo adecuada o correcta, basándose para ello en el tamaño del proyecto, la organización, los tiempos, recursos y demás. En el libro "Ingeniería de Software - Un enfoque práctico", Pressman [4] menciona que el software se construye con la aplicación de un proceso ágil y adaptable, de tal manera que dicho proceso no sea una prescripción rígida, sino que permita al equipo elegir el conjunto apropiado de acciones y tareas, con la finalidad de lograr un producto de calidad.

A pesar de que en la actualidad existen varios métodos, metodologías, estándares, marcos de trabajo y enfoques, que permiten el desarrollo de software de mejor calidad [33], muchos proyectos fallan debido a su bajo nivel de madurez, por lo que se necesita una metodología de gestión de proyectos bien definida y adaptada para la empresa [34], [32], [32], que además de facilitar la construcción del software permita la automatización de las tareas que se llevan a cabo en los diferentes ambientes en los que se publica el proyecto, automatización que se puede lograrse mediante la integración del marco de trabajo de DevOps.

Ahora bien, para seleccionar la metodología de desarrollo de software tendremos que enfrentarnos a la difícil tarea de identificar la más adecuada [40], puesto que se debe analizar el contexto de cada organización [33], tomando en cuenta la cantidad de recursos disponibles, las normas y los reglamentos internos. Debemos considerar, además, que nos enfrentamos a una gran variedad de opciones, y más aún, tenemos que ser conscientes de que, al escoger algún marco de trabajo o enfoque ágil, por lo general, están asociados a la gestión de proyectos [41], siendo necesario combinarlo con alguna metodología orientada al proceso de desarrollo de software [42]. Entre estos marcos de trabajo se puede considerar al marco de DevOps, que nos permitiría contar con un proceso de desarrollo lo más adaptado a la realidad de la institución.

Las metodologías ágiles se enfocan a la entrega de valor al cliente en el menor tiempo posible [43]; sin embargo, en proyectos de software, es necesaria cierta documentación mínima [44], con la finalidad de aplicar mejoras, integraciones o actualizaciones en la fase de mantenimiento. Así mismo, se han incorporado las metodologías híbridas [45], en donde se toman prácticas de las metodologías ágiles y tradicionales [46], [44], [44].

Acorde a lo mencionado anteriormente, se optó por una adaptación de las metodologías ágiles, y luego de haberse culminado el proyecto de manera exitosa, se concluye que se alcanzó los objetivos, la metodología de investigación cualitativa es válida y se respondió satisfactoriamente a la pregunta de investigación: **¿Con la propuesta de un marco de trabajo metodológico de desarrollo de software, se permitirá ejecutar, estandarizar y documentar de manera adecuada el proceso de desarrollo de software?**, con lo que se puede mencionar que ya no es necesario invertir tiempo en seleccionar un método, adoptar sus procesos, artefactos y prácticas; sino que, el marco propuesto, se lo aplica directamente, se sigue sus procesos y usa sus artefactos, gracias a que está estandarizado y elaborado acorde a la realidad organizacional. Para lograr este proceso se ejecutó los siguientes hitos:

Hito 1: Análisis de la situación actual de la DTI. – Con la finalidad de conocer la situación actual de la DTI y la normativa vigente relacionada al proceso de desarrollo de software se procedió a hacer un estudio de campo, una revisión sistemática y aplicar encuestas. A continuación, se detalla cada una de las actividades:

- ✓ Se realizó la observación directa del proceso de desarrollo de software en la DTI, para lo cual se registró en una matriz los procesos, artefactos y roles utilizados.
- ✓ Luego se aplicó una encuesta al personal de la DTI involucrada en el proceso de desarrollo de software acorde a las funciones ejecutadas, con la finalidad determinar metodologías usadas, prácticas, procesos, roles, etc.
- ✓ Posteriormente se analizó la normativa vigente y se la categorizó en una matriz, para determinar artículos relacionados a las metodologías o procesos del software, con la finalidad de observarlos y considerarlos al momento de realizar la propuesta.
- ✓ Finalmente se realizó la interpretación de los resultados de las encuestas aplicadas.

Hito 2: Análisis de metodologías ágiles. – Con la finalidad de conocer los diferentes procesos, artefactos, roles y prácticas se analizó las metodologías seleccionadas a través de una revisión bibliográfica. En el siguiente listado se muestran las actividades:

- ✓ Se realizó una revisión bibliográfica de las metodologías, buscando información de calidad en libros, revistas y artículos.

- ✓ Luego se determinó las características comunes a analizar para considerarlas en la propuesta.
- ✓ Se realizó una comparativa entre roles de las metodologías identificando similitudes entre los mismos acordes a las funciones realizadas.
- ✓ Posteriormente se procedió a comparar los procesos y ciclo de vida de las metodologías, relacionando sus fases y actividades.
- ✓ Se identificó las prácticas, valores y principios de las metodologías.
- ✓ Finalmente se identificó los artefactos más relevantes usados en cada una de las metodologías.

Hito 3: Propuesta del marco de trabajo metodológico de desarrollo de software. – Con la información obtenida en los hitos anteriores, se procede a elaborar la propuesta, para lo cual se realizó las siguientes actividades:

- ✓ En primer lugar, con la información analizada de la normativa vigente se planteó una lista de consideraciones a tomarse en cuenta al abordar un proyecto.
- ✓ Luego se elabora la propuesta de roles considerando las metodologías y cargos de la DTI, con la finalidad de que sean compatibles y no causa de confusión.
- ✓ Observando los procesos de las metodologías ágiles y de la DTI se determinó el ciclo de vida con sus fases y actividades.
- ✓ Pasamos a la selección de las prácticas más relevantes que pueden aplicarse en la DTI para mejorar los procesos.
- ✓ Seguidamente realizamos la propuesta de artefactos tomando en cuenta las metodologías seleccionadas y procesos ejecutado por DTI; así mismo se identificó quién es el responsable principal de su elaboración y los que deben ser obligatorios.
- ✓ Para aumentar la productividad, cuando se tenga que usar un artefacto, se elaboró el formato de aquellos artefactos que son más relevantes.
- ✓ Se elaboró la propuesta de pipe line acorde a las herramientas usadas en la DTI y a las aplicables en DevOps.
- ✓ Con la finalidad de presentar la propuesta se elaboró un documento editable con los elementos y artefactos propuestos.
- ✓ Finalmente se presentó la propuesta ante la DTI para que sea considerada en la ejecución de sus proyectos.

8. Conclusiones

- Realizar un análisis inicial de las actividades, métodos y artefactos que se emplean en la DTI, permitieron contar con un contexto del conocimiento y procesos que realiza el equipo de desarrollo, logrando así construir una propuesta que se pueda combinar y mejorar el trabajo realizado por el equipo.

- Analizar el contexto legal que se relaciona al departamento, permitió definir una necesidad y alcance en cuanto a los formatos (contenido y forma) que permiten recopilar y resguardar la información que se usan como evidencia para los procesos de fiscalización de entes de control.

- Estudiar y comparar las ventajas y similitudes entre las metodologías ágiles, tradicionales, marcos de trabajo y herramientas de desarrollo de software, permitieron obtener aquellos componentes (roles, procesos, artefactos) que incrementan la productividad del equipo de desarrollo.

- La adaptación de procesos, roles, artefactos y formatos de las metodologías ágiles seleccionadas, a la realidad organizacional, permitió estandarizar el proceso de desarrollo de software, asegurando que los resultados de los proyectos tengan un mayor nivel de calidad, tanto en su documentación como en las funcionalidades implementadas.

- Integrar herramientas automatizadas, en el proceso de desarrollo de software, tanto para la generación de código, elaboración y ejecución de pruebas, análisis de la calidad del código, análisis de vulnerabilidades, integración continua, despliegue continuo, permiten mejorar la calidad del producto, agilizar el proceso, mitigar errores y evitar problemas de seguridad.

9. Recomendaciones

- Integrar al flujo de trabajo los contenedores de desarrollo, que aceleran el inicio de nuevos proyectos o la integración de miembros permanentes o temporales (pasantes y tesistas) al equipo de desarrollo, con este tipo de contenedores los nuevos elementos tendrían una base sólida donde empezar su trabajo y así acoplarse, en un menor tiempo, al ritmo del equipo principal.

- Adaptar la propuesta de manera periódica para estar a la par con los procesos de transformación digital llevados a cabo en la universidad y a la constante evolución de la tecnológica a nivel mundial; de tal manera que podamos aprovechar al máximo nuevos métodos, enfoques y herramientas para ofrecer productos y servicios de calidad.

- Se recomienda realizar nuevos estudios para integrar otras metodologías de gestión de proyectos al proceso de desarrollo de software, como: OKR (Objetivos y Resultados Claves), Design Thinking (Pensamiento de Diseño), TDD (Desarrollo Guiado por Pruebas), etc., con la finalidad de que se tome las mejores bondades y se mejoren los procesos.

10. Bibliografía

[1] Santander Universidades, «Metodologías de desarrollo de software: ¿qué son?», *https://www.becas-santander.com/es/blog/metodologias-desarrollo-software.html*, 21 de diciembre de 2020.

[2] M. DE Desarrollo De Software y E. Gabriel Pacienzia, «FACULTAD DE QUÍMICA E INGENIERIA "FRAY ROGELIO BACON" PONTIFICIA UNIVERSIDAD CATÓLICA ARGENTINA SANTA MARIA DE LOS BUENOS AIRES Cátedra Seminario de Sistemas».

[3] A. N. Cadavid, J. Daniel Fernández Martínez, y J. Morales Vélez, «Revisión de metodologías ágiles para el desarrollo de software A review of agile methodologies for software development».

[4] R. S. Pressman, *Ingeniería del software. Un enfoque práctico*, 7.ª ed., n.º 9. Mexico: McGraw-Hill, 2010.

[5] S. I. Mariño y P. L. Alfonzo, «Evidencias de Accesibilidad Web en la generación de sitios. Propuesta de un método», *Revista Iberoamericana de Tecnología en Educación y Educación en Tecnología*, n.º 23, 2019, doi: 10.24215/18509959.23.e06.

[6] G. Booch, «The History of Software Engineering», *IEEE Softw*, vol. 35, n.º 5, 2018, doi: 10.1109/MS.2018.3571234.

[7] O. A. P. A., «Cuatro enfoques metodológicos para el desarrollo de Software RUP – MSF – XP - SCRUM», *INVENTUM*, vol. 6, n.º 10, pp. 64-78, feb. 2011, doi: 10.26620/UNIMINUTO.INVENTUM.6.10.2011.64-78.

[8] «Case Method Fast-Track | Guide books». https://dl.acm.org/doi/abs/10.5555/561543 (accedido 27 de abril de 2023).

[9] E. Miranda, «Moscow Rules: A Quantitative Exposé», *Lecture Notes in Business Information Processing*, vol. 445 LNBIP, pp. 19-34, 2022, doi: 10.1007/978-3-031-08169-9_2/TABLES/1.

[10] «Chapter 10: MoSCoW Prioritisation». https://www.agilebusiness.org/dsdm-project-framework/moscow-prioririsation.html (accedido 27 de abril de 2023).

[11] «Taiga: Tu herramienta de gestión de proyectos ágil y opensource: Kanban & Scrum». https://www.taiga.io/es (accedido 27 de abril de 2023).

[12] «Plataforma de almacenamiento personal en la nube y uso compartido de archivos: Google». https://www.google.com/intl/es-419/drive/#overview (accedido 27 de abril de 2023).

[13] «Platform | GitLab». https://about.gitlab.com/platform/?stage=plan (accedido 27 de abril de 2023).

[14] «JetBrains: Essential tools for software developers and teams». https://www.jetbrains.com/ (accedido 27 de abril de 2023).

[15] «Get Started with Visual Studio Code». https://code.visualstudio.com/learn (accedido 27 de abril de 2023).

[16] «GitHub Copilot · Your AI pair programmer». https://github.com/features/copilot (accedido 27 de abril de 2023).

[17] «Features of diagrams.net and draw.io». https://www.diagrams.net/features (accedido 27 de abril de 2023).
[18] «Camunda Platform 8 | Software for Process Orchestration». https://camunda.com/platform/ (accedido 27 de abril de 2023).
[19] «IDE Linter Tool & Real-Time Software for Code | Sonar | Sonar». https://www.sonarsource.com/products/sonarlint/ (accedido 27 de abril de 2023).
[20] «Self-managed | SonarQube | Sonar». https://www.sonarsource.com/products/sonarqube/ (accedido 27 de abril de 2023).
[21] «OWASP ZAP – The OWASP ZAP Desktop User Guide». https://www.zaproxy.org/docs/desktop/ (accedido 27 de abril de 2023).
[22] «Funciones | Slack». https://slack.com/intl/es-ec/features (accedido 27 de abril de 2023).
[23] «Discord para organizaciones y clubes universitarios». https://discord.com/college (accedido 27 de abril de 2023).
[24] «Why Docker | Docker». https://www.docker.com/why-docker/ (accedido 29 de abril de 2023).
[25] «Jenkins». https://www.jenkins.io/ (accedido 29 de abril de 2023).
[26] «Pipeline». https://www.jenkins.io/doc/book/pipeline/ (accedido 29 de abril de 2023).
[27] «Metodologías de desarrollo de software y su - ProQuest». https://www.proquest.com/docview/2648273778 (accedido 26 de abril de 2023).
[28] J. P. GAMBOA CARRASCAL, «DISEÑO DE UN METODO AGIL DE DESARROLLO DE SOFTWARE BASADO EN XP, SCRUM, OPENUP Y VALIDADO CON LA HERRAMIENTA DE ANALISIS 4-DAT PARA MEJORAR LA CALIDAD DE LOS PROYECTOS DESARROLLADOS POR LOS GRUPOS DE GESTION DE SOFTWARE DE LA UFPSO.», 2015. Accedido: 31 de marzo de 2023. [En línea]. Disponible en: http://repositorio.ufpso.edu.co/bitstream/123456789/1052/1/27780.pdf
[29] P. Letelier y M. Penadés, «Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).», *Técnica Administrativa*, abr. 2006.
[30] SCRUMstudy, *Guía de los Fundamentos de Scrum (Guía del SBOK®)*, 4.ª ed., vol. 1, n.º 69. VMEdu, Inc, 2022.
[31] A. Abdullah Albarq y R. Qureshi, «The Proposed L-Scrumban Methodology to Improve the Efficiency of Agile Software Development», *International Journal of Information Engineering and Electronic Business*, vol. 10, n.º 3, pp. 23-35, may 2018, doi: 10.5815/IJIEEB.2018.03.04.
[32] P. Patrocinante, Y. Farrán Leiva, y C. C. González Gonzalo Rojas D, «UNIVERSIDAD DE CONCEPCIÓN FACULTAD DE INGENIERÍA DEPARTAMENTO DE INGENIERÍA INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN Metodología Lean-PMI para desarrollo de proyectos de software».
[33] I. Faustino y J. Mejia, «Proposal for a software development framework based on the ISO/IEC 29110 standard: Public organizations», *Applications in Software*

Engineering - Proceedings of the 9th International Conference on Software Process Improvement, CIMPS 2020, pp. 132-140, oct. 2020, doi: 10.1109/CIMPS52057.2020.9390135.

[34] L. Caiza, D. Chicaiza, R. P. R. Ch., y F. Montaluisa, «MEAC: An experience to keep the production line active in the software development process», *KnE Engineering*, pp. 22-39, ene. 2020, doi: 10.18502/KEG.V5I1.5916.

[35] A. Rasnacis y S. Berzisa, «Method for Adaptation and Implementation of Agile Project Management Methodology», *Procedia Comput Sci*, vol. 104, pp. 43-50, ene. 2017, doi: 10.1016/J.PROCS.2017.01.055.

[36] D. A. Muñoz, H. Ordóñez, V. Bucheli, D. A. Muñoz, H. Ordóñez, y V. Bucheli, «Lineamientos para la implementación del modelo CALMS de DevOps en mipymes desarrolladoras de software en el contexto surcolombiano», *Revista Guillermo de Ockham*, vol. 18, n.º 1, pp. 81-91, oct. 2020, doi: 10.21500/22563202.4270.

[37] «¿Qué es la gestión del ciclo de vida de las aplicaciones (ALM)?» https://www.redhat.com/es/topics/devops/what-is-application-lifecycle-management-alm (accedido 26 de abril de 2023).

[38] «Marco CALMS | Atlassian». https://www.atlassian.com/es/devops/frameworks/calms-framework (accedido 26 de abril de 2023).

[39] N. D. P. Burgos, F. A. A. Marquez, y G. E. B. Baquerizo, «Métodos y técnicas en la investigación cualitativa. Algunas precisiones necesarias», *Revista Conrado*, vol. 15, n.º 70, oct. 2019, [En línea]. Disponible en: https://conrado.ucf.edu.cu/index.php/conrado/article/view/1162

[40] L. Simelane y T. Zuva, «Decision Support Framework for the Adoption of Software Development Methodologies», en *Proceedings - 2019 International Multidisciplinary Information Technology and Engineering Conference, IMITEC 2019*, 2019. doi: 10.1109/IMITEC45504.2019.9015859.

[41] H. Patilla, E. Gómez, J. Pulache, J. Lozano, E. Solórzano, y Y. Meneses, «Modelo de Gestión de Desarrollo de Software Ágil mediante Scrum y Kanban sobre la Programación Extrema», *Revista Ibérica de Sistemas e Tecnologías de Informação*, 2021.

[42] Janampa. Hubner Patilla, Karel. Sotomayor Peralta, y L. Miguel. Prado Vásquez, «Modelo de Desarrollo de Software de la Programación Extrema sobre Scrum para Gestión de Software Ágil Extreme Programming Software Development Model on Scrum for Agile Software Management», *Revista Ibérica de Sistemas e Tecnologias de Informação*, vol. E39, 2021.

[43] W. Behutiye, P. Rodriguez, y M. Oivo, «Quality Requirement Documentation Guidelines for Agile Software Development», *IEEE Access*, vol. 10, 2022, doi: 10.1109/ACCESS.2022.3187106.

[44] C. Tecnológica, S.- Romero, M. Antonio, ; Escudero-López, y N. Ezequiel, «Propuesta de Metodología Híbrida y Base de Documentación para el Desarrollo de Software Actual», *Conciencia Tecnológica ISSN*, 2020.

[45] M. Kuhrmann *et al.*, «What Makes Agile Software Development Agile?», *IEEE Transactions on Software Engineering*, vol. 48, n.º 9, 2022, doi: 10.1109/TSE.2021.3099532.

[46] J. R. Molina Ríos, M. P. Zea Ordóñez, F. F. Redrován Castillo, N. M. Loja Mora, M. R. Valarezo Pardo, y J. A. Honores Tapia, *SNAIL, Una metodología híbrida para el desarrollo de aplicaciones web*. 2018. doi: 10.17993/ingytec.2018.38.

11. Anexos

Anexo 1. Encuesta - Perfil Control de Calidad

Universidad Nacional de Loja

Maestría en Ingeniería de Software

Tema: Propuesta de un marco de trabajo metodológico de desarrollo de software para la Dirección de Tecnologías de Información de la Universidad Nacional de Loja.

Autores: Danny Muñoz y Máximo Álvarez

Objetivo: Analizar la situación actual de la Dirección de Tecnologías de Información, a través una revisión sistemática y trabajo de campo, para conocer cómo se está ejecutando el proceso de desarrollo de software.

Público objetivo: Control de Calidad

La presente entrevista tiene como objetivo conocer las herramientas y metodologías que se emplean en el control de calidad, en la Dirección de Tecnologías de Información de la UNL.

1. ¿Se ejecutan pruebas unitarias al código elaborado por los desarrolladores?
    - Si
    - No
2. ¿Cuál/es de las siguientes herramientas utiliza para analizar la calidad del código?
    - Sonar Qube (Servidor)
    - Sonar Lint (IDE)
    - Ninguna
    - Otra
3. ¿Cuál/es de los siguientes formatos usa al momento de realizar sus actividades de control de calidad?
    - Casos de prueba (Base de conocimientos)
    - Casos de prueba (Específico por proyecto)
    - Certificado de control de calidad
    - Ninguno
    - Otros
4. Si en la pregunta anterior eligió la opción "Otros", especifique cuales:
    - _____

5. ¿Cuál/es de las siguientes herramientas usa para organizar tareas/actividades?
    - Jira
    - Trello
    - Notion
    - Taiga
    - Monday
    - Slack
    - Otro
6. ¿Conoce la metodología/marco de trabajo DevOps?
    - Si
    - No
7. Indique su nivel de conocimiento en esta metodología

0	1	2	3	4	5	6	7	8	9	10

Muy Básico Avanzado

8. ¿Cuál de los siguientes lenguajes de programación conoce?
    - Python
    - Java
    - C#
    - C++
    - .Net
    - Php
    - JS(React/Node/Etc)
    - Otro
    - Ninguno
9. Indique su nivel de conocimiento en el lenguaje de programación

0	1	2	3	4	5	6	7	8	9	10

Muy Básico Avanzado

Anexo 2. Encuesta – Perfil Desarrollador Senior/Junior

Universidad Nacional de Loja

Maestría en Ingeniería de Software

Tema: Propuesta de un marco de trabajo metodológico de desarrollo de software para la Dirección de Tecnologías de Información de la Universidad Nacional de Loja.

Autores: Danny Muñoz y Máximo Álvarez

Objetivo: Analizar la situación actual de la Dirección de Tecnologías de Información, a través una revisión sistemática y trabajo de campo, para conocer cómo se está ejecutando el proceso de desarrollo de software.

Público objetivo: Desarrollador Senior / Desarrollador Junior

La presente entrevista tiene como objetivo conocer las herramientas y metodologías que se emplean por el equipo de desarrollo de software, tanto seniors como juniors, en la Dirección de Tecnologías de Información de la UNL.

1. ¿Conoce la metodología XP (Programación Extrema)?
    - Si
    - No
2. Indique su nivel de conocimiento en esta metodología

0	1	2	3	4	5	6	7	8	9	10

Muy Básico Avanzado

3. ¿Conoce la metodología/marco de trabajo SCRUM?
    - Si
    - No
4. Indique su nivel de conocimiento en esta metodología

0	1	2	3	4	5	6	7	8	9	10

Muy Básico Avanzado

5. ¿Conoce la metodología KANBAN?
    - Si
    - No

6. Indique su nivel de conocimiento en esta metodología

0	1	2	3	4	5	6	7	8	9	10

Muy Básico Avanzado

7. ¿Conoce la metodología/marco de trabajo DevOps?
    - Si
    - No
8. Indique su nivel de conocimiento en esta metodología

0	1	2	3	4	5	6	7	8	9	10

Muy Básico Avanzado

9. ¿Utiliza alguno de los siguientes IDE's de desarrollo para escribir el código fuente (IDE Principal)?
    - PyCharm Community
    - PyCharm Profesional
    - Visual Studio Code
    - Otros
10. ¿Utiliza alguno de los siguientes IDE's de desarrollo para escribir el código fuente (IDE Secundario)?
    - PyCharm Community
    - PyCharm Profesional
    - Visual Studio Code
    - Otros
11. ¿Conoce la Guía de los Fundamentos para la Dirección de Proyectos?
    - Si
    - No
12. En la siguiente escala marque el nivel de conocimiento que tiene de la terminal de linux
    - Ninguno
    - Básico
    - Medio
    - Avanzado

13. ¿Cuál/es de las siguientes herramientas usa para organizar tareas/actividades?
    - Jira
    - Trello
    - Notion
    - Taiga
    - Monday
    - Slack
    - Otro
14. ¿Qué tan frecuente utiliza contenedores (Docker)?
    - Diario
    - Algunas veces por semana
    - Algunas veces al mes
    - Nunca
15. ¿Qué lenguaje de programación utiliza principalmente?
    - Python
    - Java
    - C#
    - C++
    - .Net
    - Php
    - JS (React/Node/Etc)
    - Otro
16. Indique el primer proyecto al que dedica mayor cantidad de tiempo en desarrollo de funcionalidades / correciones de bug's
    - SIAAF
    - SIAAF- Recaudaciones
    - SGA (Docentes)
    - SGA (Estudiantes)
    - SGA(Administrativos)
    - SGA (Web Service)
    - Inscripciones
    - MGT

- Admisiones
- Otro

17. Indique el segundo proyecto al que dedica mayor cantidad de tiempo en desarrollo de funcionalidades / correciones de bug's
    - SIAAF
    - SIAAF- Recaudaciones
    - SGA (Docentes)
    - SGA (Estudiantes)
    - SGA(Administrativos)
    - SGA (Web Service)
    - Inscripciones
    - MGT
    - Admisiones
    - Otro
18. Indique el tercer proyecto al que dedica mayor cantidad de tiempo en desarrollo de funcionalidades / correcciones de bug's
    - SIAAF
    - SIAAF- Recaudaciones
    - SGA (Docentes)
    - SGA (Estudiantes)
    - SGA(Administrativos)
    - SGA (Web Service)
    - Inscripciones
    - MGT
    - Admisiones
    - Otro
19. ¿Utiliza alguna herramienta para la automatización del paso a producción de nuevas funcionalidades / corrección de errores?
    - Si
    - No
20. Si eligió SI en la pregunta anterior, indique cuál:
    - _____

21. ¿Se ejecutan pruebas al código construído previo a subirlo a producción (Junit / Pytest / Otras)?
    - Si
    - No
22. ¿Cuál/es de las siguientes herramientas utiliza para analizar la calidad del código?
    - Sonar Qube (Servidor)
    - Sonar Lint (IDE)
    - Ninguna
    - Otra
23. ¿Cuál/es de los siguientes formatos usa al momento de realizar sus actividades de desarrollo de software?
    - Casos de negocio
    - Especificación de requisitos de software
    - Requerimiento de Software
    - Ninguno
    - Otros
24. Si en la pregunta anterior eligió la opción Otras, especifique cuales:
    - _____

Anexo 3. Encuesta – Perfil Seguridad de la Información

Universidad Nacional de Loja

Maestría en Ingeniería de Software

Tema: Propuesta de un marco de trabajo metodológico de desarrollo de software para la Dirección de Tecnologías de Información de la Universidad Nacional de Loja.

Autores: Danny Muñoz y Máximo Álvarez

Objetivo: Analizar la situación actual de la Dirección de Tecnologías de Información, a través una revisión sistemática y trabajo de campo, para conocer cómo se está ejecutando el proceso de desarrollo de software.

Público objetivo: Especialista de Seguridad Informática

La presente entrevista tiene como objetivo conocer las herramientas y procesos que son empleadas en el análisis de la seguridad en las aplicaciones desarrolladas en la Dirección de Tecnologías de Información de la UNL.

1. ¿Como calificaría su nivel de experiencia en seguridad de aplicaciones web?
    - Principiante
    - Intermedio
    - Avanzado
2. De las siguientes herramientas: ¿Cuáles emplea para realizar pruebas de seguridad en las aplicaciones web?
    - Herramientas de escaneo de vulnerabilidades
    - Herramientas de análisis estático de código
    - Herramientas de análisis dinámico de código
3. ¿Qué herramientas de escaneo de vulnerabilidades conoce?
    - _____
4. ¿Qué herramientas de análisis estático de código conoce?
    - _____
5. ¿Qué herramientas de análisis dinámico de código conoce?
    - _____
6. ¿De qué forma evalúa las aplicaciones web? *
    - Pruebas manuales
    - Pruebas automatizadas

7. ¿Qué desafíos enfrenta al evaluar la seguridad de aplicaciones web?
    - Falta de recursos (herramientas)
    - Falta de conocimiento técnico
    - Falta de tiempo
8. ¿Qué tipo de métodos aplica para mantener la seguridad de las aplicaciones web?
    - Ejecutando evaluaciones de seguridad regulares
    - Implementando medida de seguridad adicionales
    - Otras (Especifique cuáles)
9. ¿Cómo maneja los problemas de seguridad en las aplicaciones web?
    - Corrigiendo el problema con las herramientas disponibles
    - Reportando los problemas a los técnicos responsables de construir la aplicación
10. ¿Cómo se asegura que las aplicaciones web sean compatibles con los estándares de seguridad?
    - Realizando pruebas regulares
    - Siguiendo las mejores prácticas recomendabas por OWASP
11. ¿Qué normativas/estándares aplica al realizar pruebas de seguridad a las aplicaciones web?
    - _____

Anexo 4. Análisis de Resultados de Encuesta – Perfil Control de Calidad

1. ¿Se ejecutan pruebas unitarias al código elaborado por los desarrolladores?

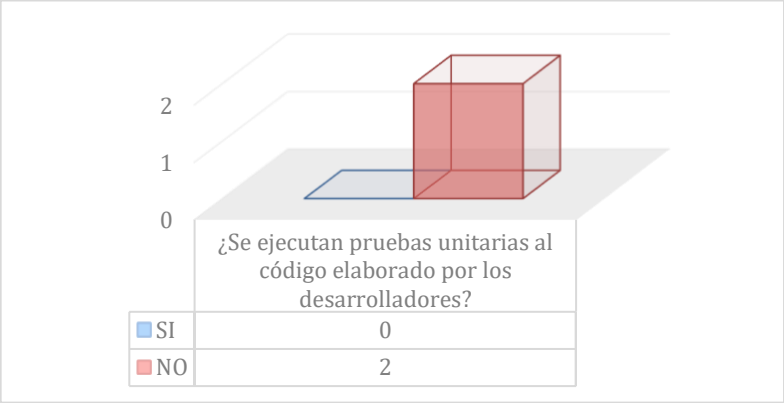


Fig. 9. Resultados encuesta - pruebas unitarias

El 100% de los encuestados manifiesta que no se realizan pruebas unitarias al código. Esto sugiere que al enviarse nuevos cambios/funcionalidades al ambiente de producción, sin el uso de pruebas unitarias, se incrementa el riesgo de que se puedan presentar errores o fallos, lo que puede generar inactividad en sistemas hasta que el error pueda ser identificado y solucionado.

2. ¿Cuál/es de las siguientes herramientas utiliza para analizar la calidad del código?

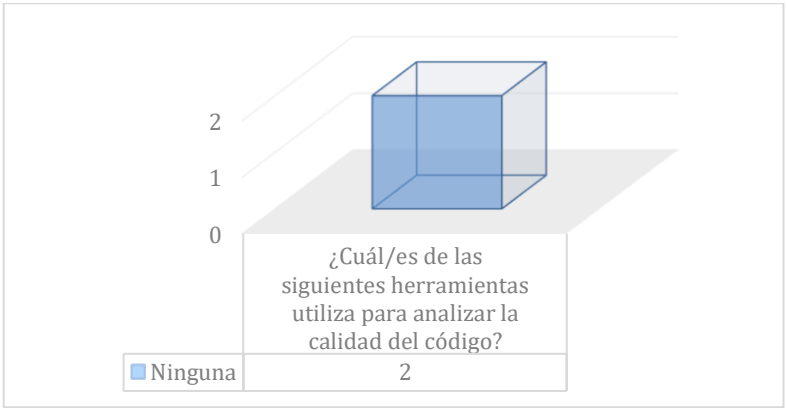


Fig. 10. Resultados encuesta - herramientas analizar calidad código

El 100% de los encuestados manifiesta que no se utiliza herramientas de análisis de calidad de código. Esto sugiere que el código, aunque funcional, puede contener demasiada complejidad cognitiva o código innecesario que en el futuro dificultará su mantenimiento.

3. ¿Cuál/es de los siguientes formatos usa al momento de realizar sus actividades de control de calidad?

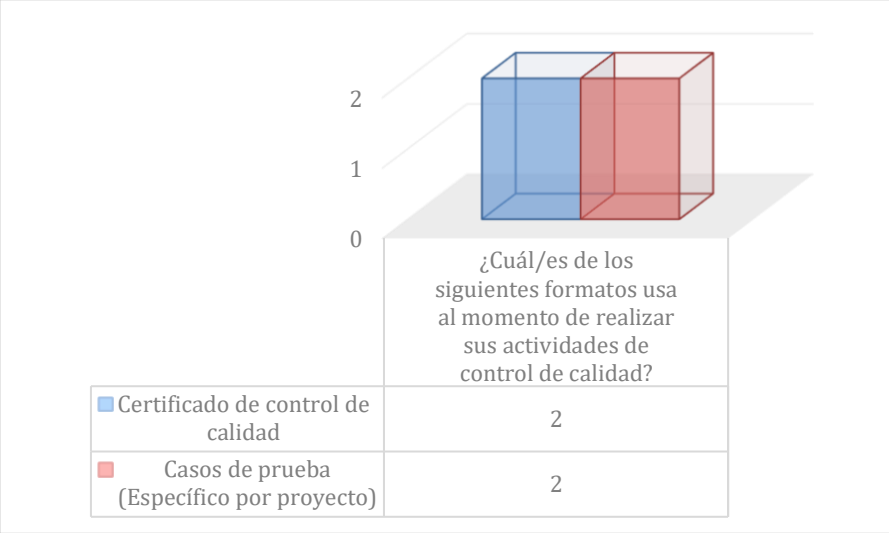


Fig. 11. Resultados encuesta - formatos de control de calidad

El 100% de los encuestados manifiesta emplear el Certificado de control de calidad y los casos de prueba. Esto nos permite conocer que se aplican pruebas de funcionalidad sobre las aplicaciones, previo a su puesta en producción.

4. Si en la pregunta anterior eligió la opción "Otros", especifique cuales:

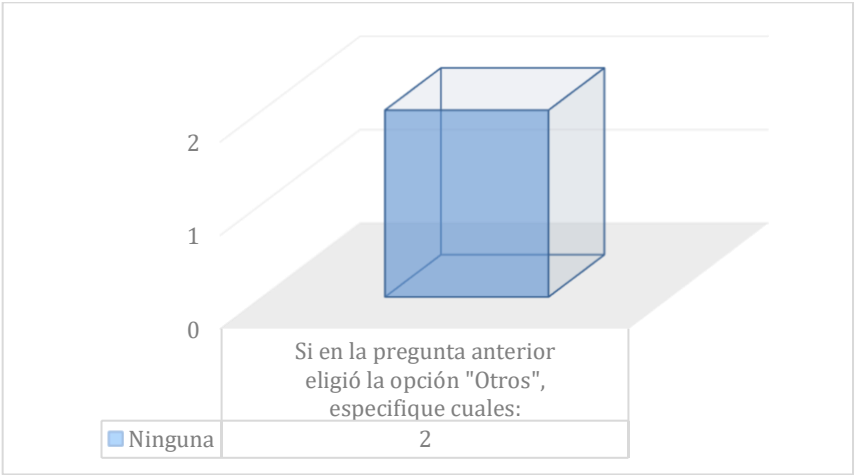


Fig. 12. Resultados encuesta – otros formatos de control calidad

El 100% de los encuestados manifiesta no emplear otros documentos para realizar las actividades de control de calidad.

5. ¿Cuál/es de las siguientes herramientas usa para organizar tareas/actividades?

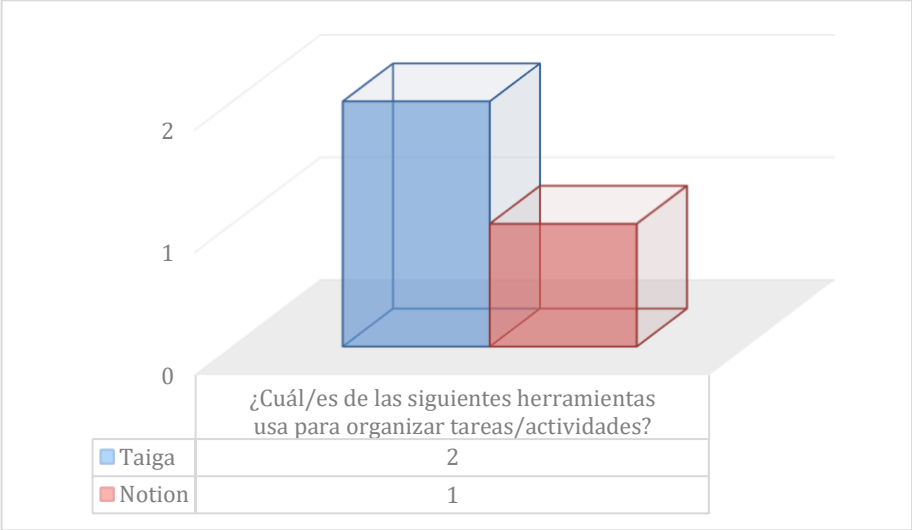


Fig. 13. Resultados encuesta - herramientas gestión actividades

El 100% de los encuestados manifiesta que emplea a taiga como herramienta para la gestión de tareas y el 50% también emplea Notion para esta actividad. Esto nos sugiere que se cuenta con herramientas para registrar y dar seguimientos a los incidentes identificados.

6. ¿Conoce la metodología/marco de trabajo DevOps?

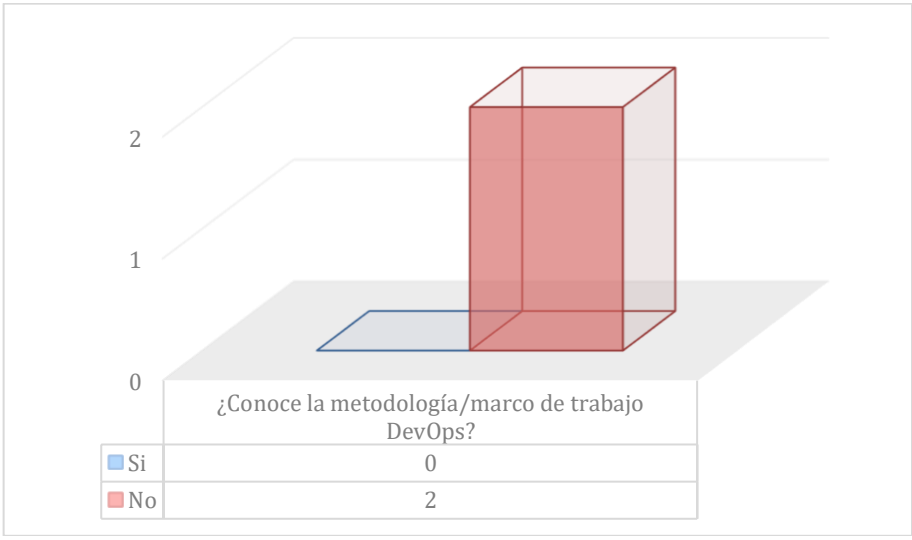


Fig. 14. Resultados encuesta - DevOps

El 100% de los encuestados manifiesta no conocer la metodología/marco de trabajo DevOps. Esto nos siguiere que no se integra un proceso automatizado en el proceso de control de calidad.

7. Indique su nivel de conocimiento en esta metodología

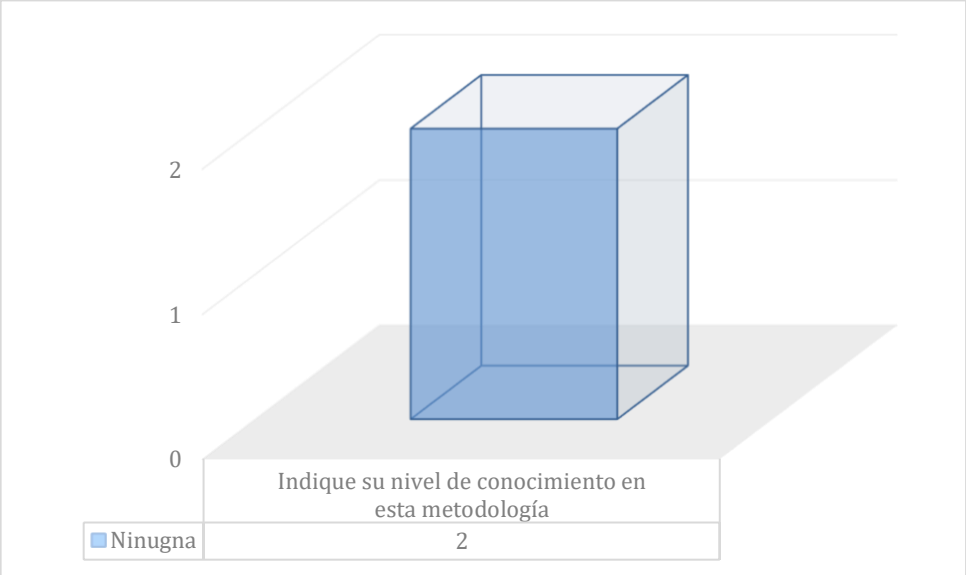


Fig. 15. Resultados encuesta - nivel de conocimiento DevOps

El 100% de los encuestados manifiesta no conocer la metodología/marco de trabajo DevOps.

8. ¿Cuál de los siguientes lenguajes de programación conoce?

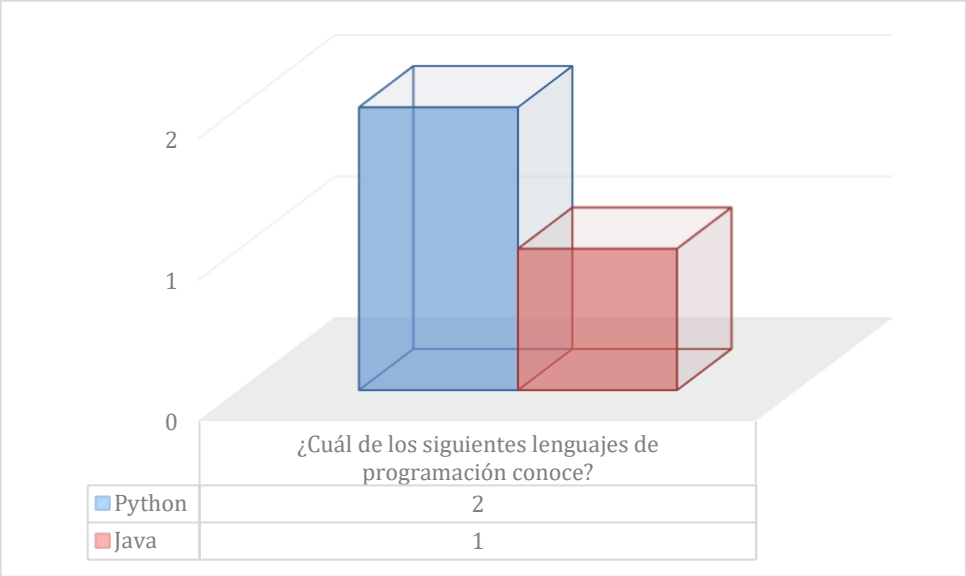


Fig. 16. Resultados encuesta - lenguajes de programación

El 100% de los encuestados manifiesta conocer el lenguaje de programación python y el 50% adicionalmente el lenguaje de programación java. Lo que nos permite identificar que son las tecnologías que en mayor frecuencia se utilizan por el equipo de desarrollo.

9. Indique su nivel de conocimiento en el lenguaje de programación

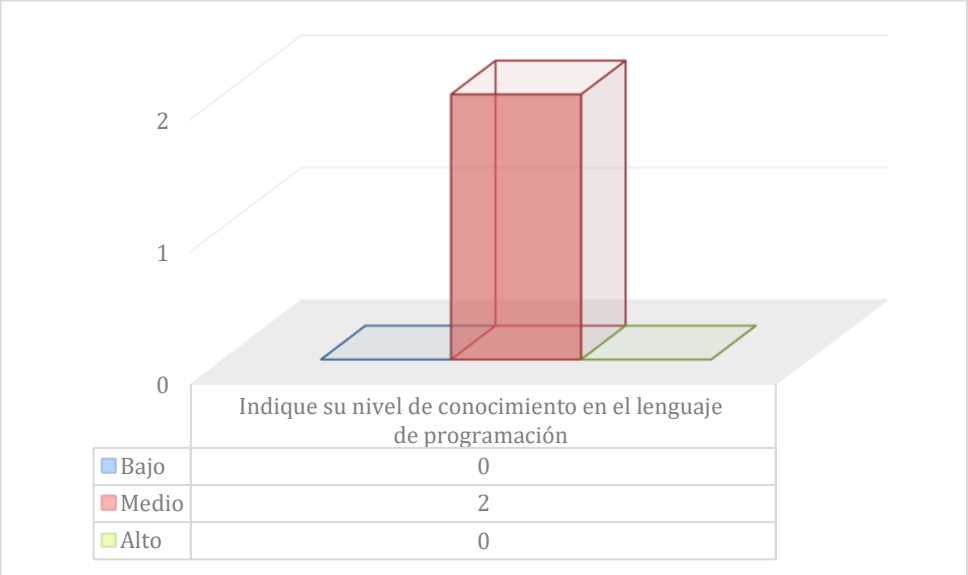


Fig. 17. Resultados encuesta - nivel conocimiento lenguaje de programación

El 100% de los encuestados manifiesta tener un conocimiento medio de los lenguajes de programación. Esto sugiere que el equipo de control de calidad tiene el conocimiento sobre los lenguajes de programación empleados por el equipo de desarrollo.

Anexo 5. Análisis de Resultados de Encuesta – Perfil Desarrollador Senior / Junior

1. ¿Conoce la metodología XP (Programación Extrema)?

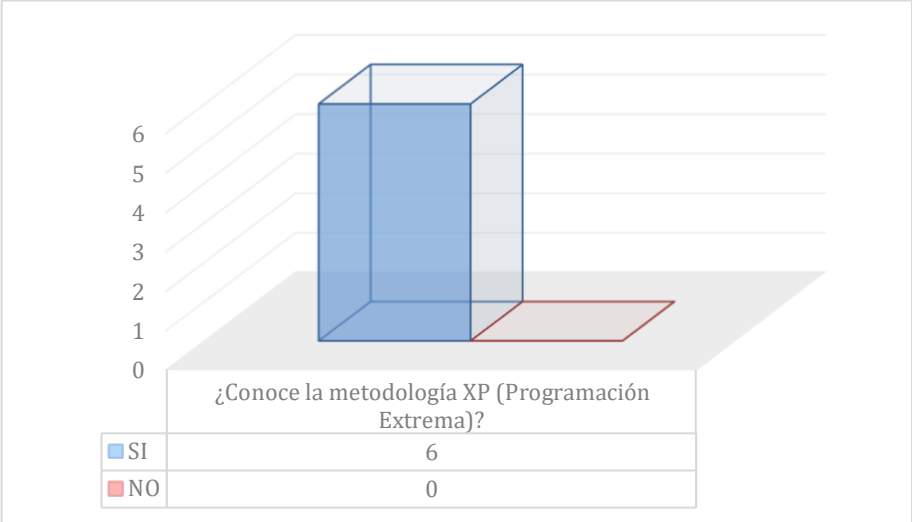


Fig. 18. Resultados encuesta – conocimiento de la metodología XP

El 100% de los encuestados manifiesta conocer la metodología XP. Esto nos sugiere que el equipo de desarrollo cuenta con los conocimientos necesarios para trabajar con metodologías de desarrollo ágiles.

2. Indique su nivel de conocimiento en esta metodología

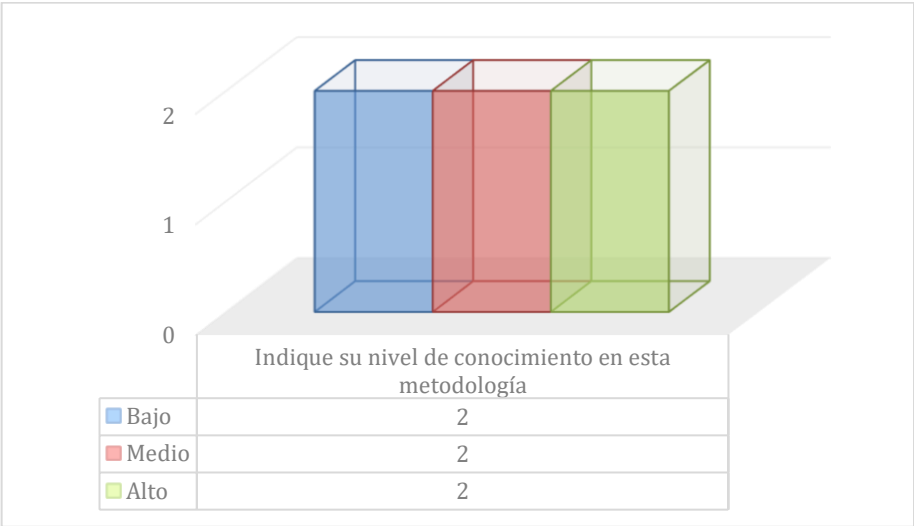


Fig. 19. Resultados encuesta - nivel de conocimiento de la metodología XP

El 66% de los encuestados cuenta con un nivel de experiencia medio y alto sobre esta metodología. Esto nos sugiere que el equipo de desarrollo cuenta con conocimientos en el manejo de esta metodología de desarrollo ágil.

3. ¿Conoce la metodología/marco de trabajo SCRUM?

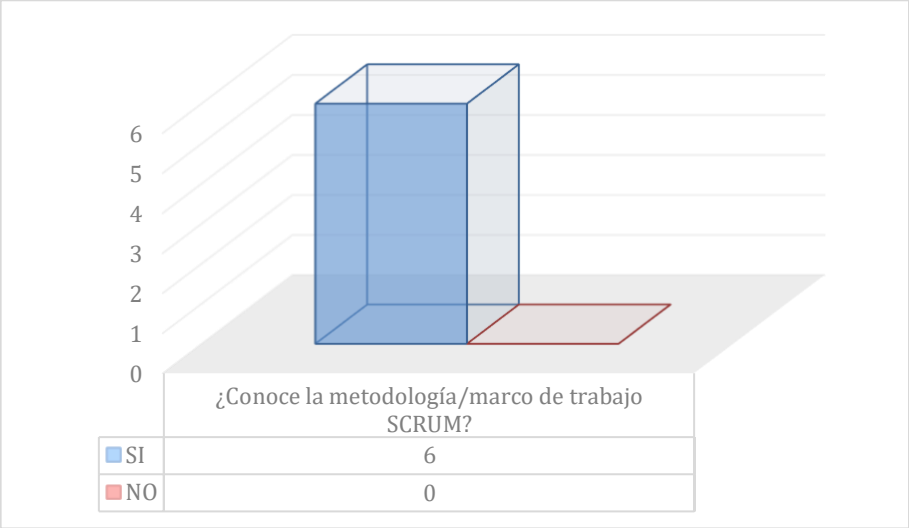


Fig. 20. Resultados encuesta - conocimiento de SCRUM

El 100% de los encuestados conoce la metodología/marco de trabajo SCRUM. Lo que nos sugiere que también se aplican los procesos que esta nos recomienda.

4. Indique su nivel de conocimiento en esta metodología

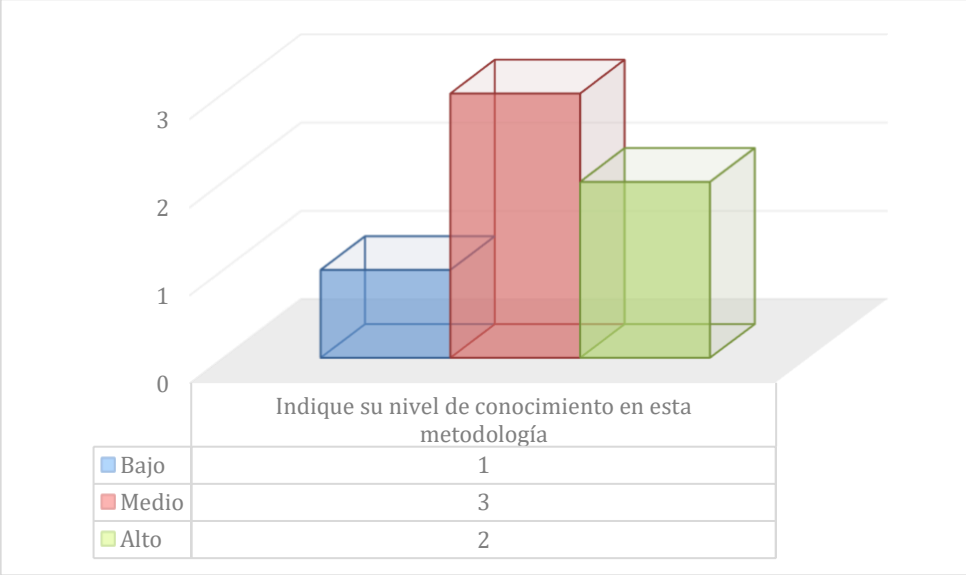


Fig. 21. Resultados encuesta - nivel de conocimiento de SCRUM

El 83% de los encuestados cuenta con un nivel de experiencia medio y alto sobre esta metodología. Esto nos sugiere que el equipo de desarrollo cuenta con conocimientos altos y maneja con frecuencia esta metodología de desarrollo ágil.

5. ¿Conoce la metodología KANBAN?

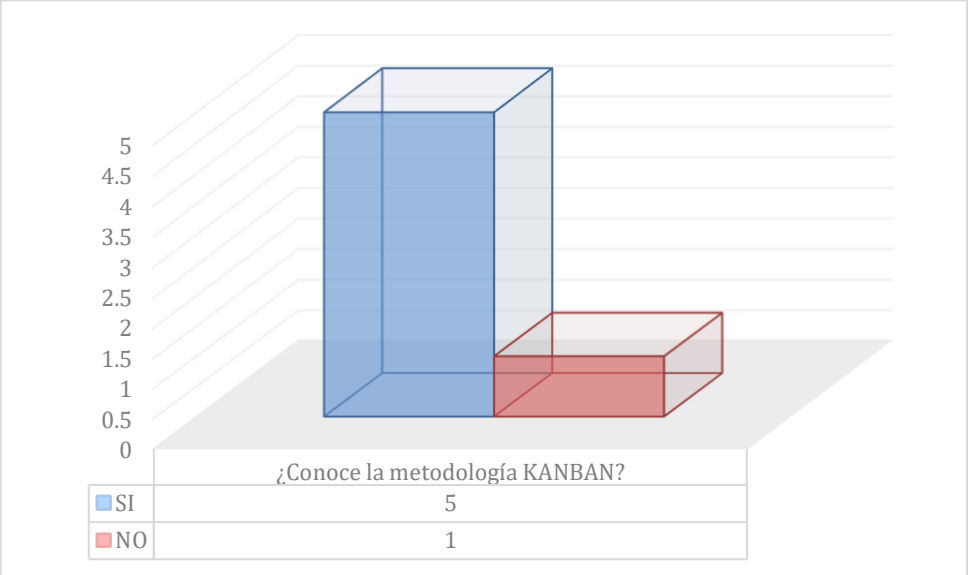


Fig. 22. Resultados encuesta - conocimiento de KANBAN

El 83% de los encuestados conoce la metodología/marco de trabajo KANBAN. Lo que nos sugiere que se aplican los procesos que esta metodología nos recomienda.

6. Indique su nivel de conocimiento en esta metodología

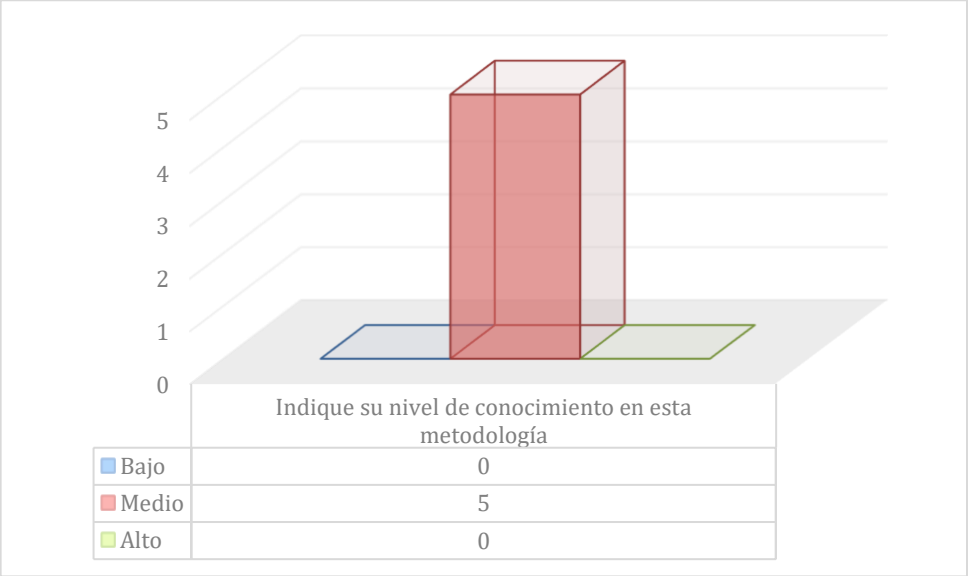


Fig. 23. Resultados encuesta - nivel de conocimiento de KANBAN

El 100% de los encuestados cuenta con un nivel medio de experiencia con el uso de KANBAN. Esto nos sugiere que todo el equipo emplea los roles y prácticas que esta metodología ágil recomienda.

7. ¿Conoce la metodología/marco de trabajo DevOps?

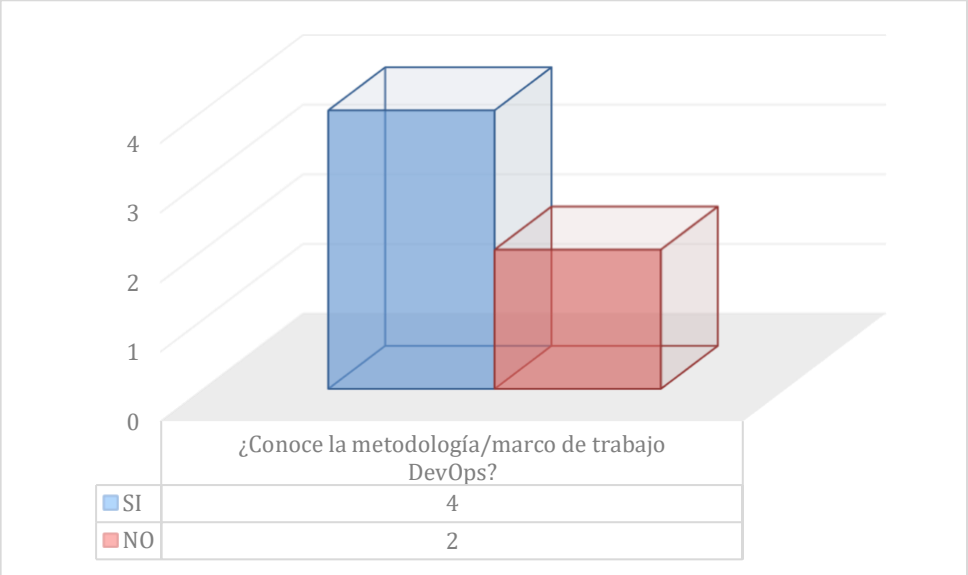


Fig. 24. Resultados encuesta - conocimiento de DEVOPS

El 66% de los encuestados manifiesta conocer la metodología/marco de trabajo DevOps. Esto nos sugiere que durante el proceso y puesta en producción de funcionalidades se aplican conceptos o herramientas de DevOps.

8. Indique su nivel de conocimiento en esta metodología

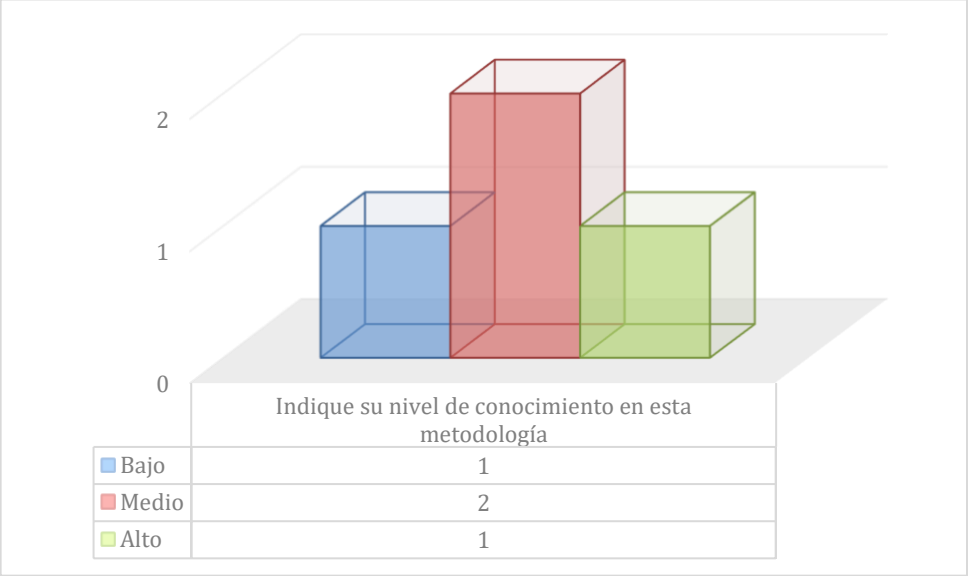


Fig. 25. Resultados encuesta - nivel de conocimiento de DEVOPS

Tomando como relación a todos los encuestados, el 50% tiene conocimientos medios sobre el uso de las diferentes herramientas o procesos de DevOps.

9. ¿Utiliza alguno de los siguientes IDE's de desarrollo para escribir el código fuente (IDE Principal)?

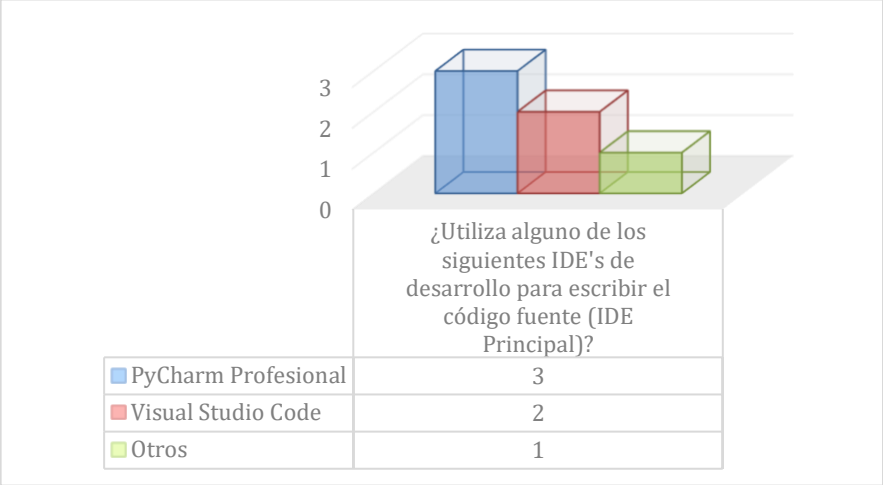


Fig. 26. Resultados encuesta - uso de IDE principal

El 50% de los encuestados utilizan como IDE principal a PyCharm Profesional para el desarrollo de software.

10. ¿Utiliza alguno de los siguientes IDE's de desarrollo para escribir el código fuente (IDE Secundario)?

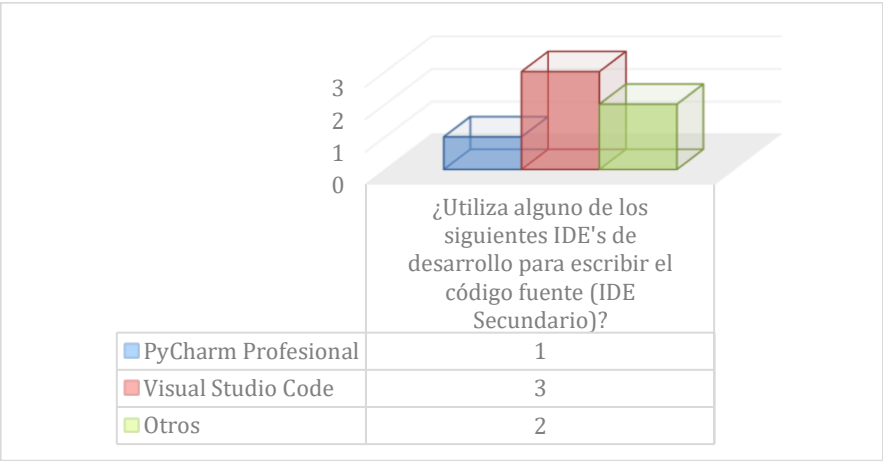


Fig. 27. Resultados encuesta - uso de IDE secundario

El 50% de los encuestados utilizan como editor de código a Visual Studio Code para el desarrollo de software.

Entre los resultados de la pregunta 9 y 10 nos sugiere que se emplean herramientas avanzadas para el desarrollo de software.

11. ¿Conoce la Guía de los Fundamentos para la Dirección de Proyectos?

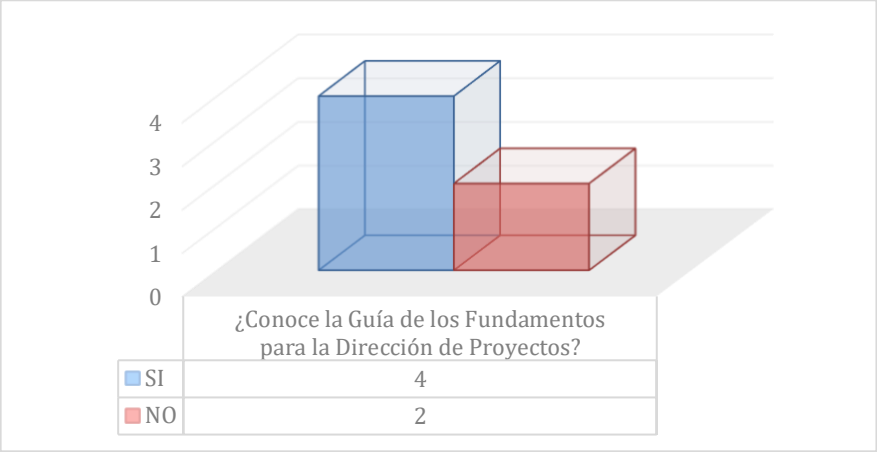


Fig. 28. Resultados encuesta - conocimiento Guía Fundamentos para la Dirección de Proyectos

El 66% de los encuestados manifiesta conocer la guía de fundamentos para la dirección de proyectos. Esto nos da a comprender que se utilizan la misma dentro del proceso de desarrollo de software.

12. En la siguiente escala marque el nivel de conocimiento que tiene de la terminal de Linux

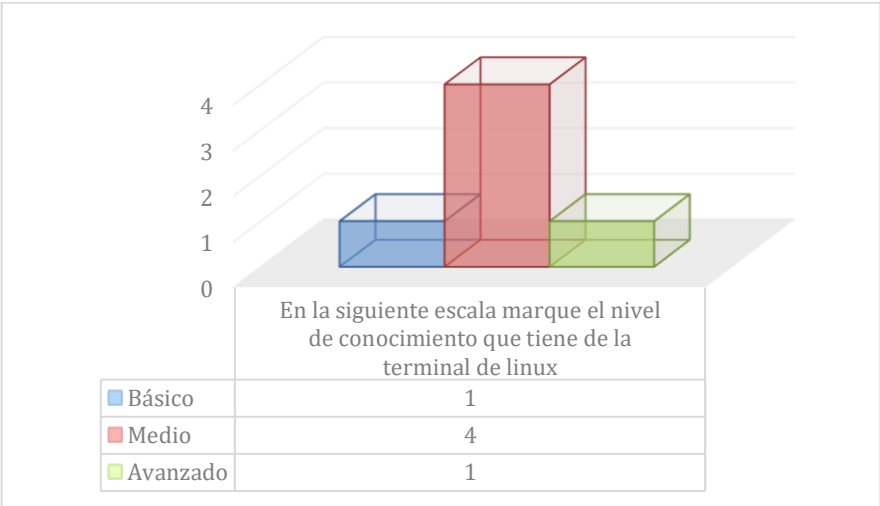


Fig. 29. Resultados encuesta - nivel de conocimiento terminal Linux

El 83% de los encuestados manifiesta tener un conocimiento medio y avanzado de la terminal de Linux. Lo que nos da a comprender que existe el conocimiento necesario en el uso de comandos en entonces o sistemas operativos de gnu/Linux.

13. ¿Cuál/es de las siguientes herramientas usa para organizar tareas/actividades?

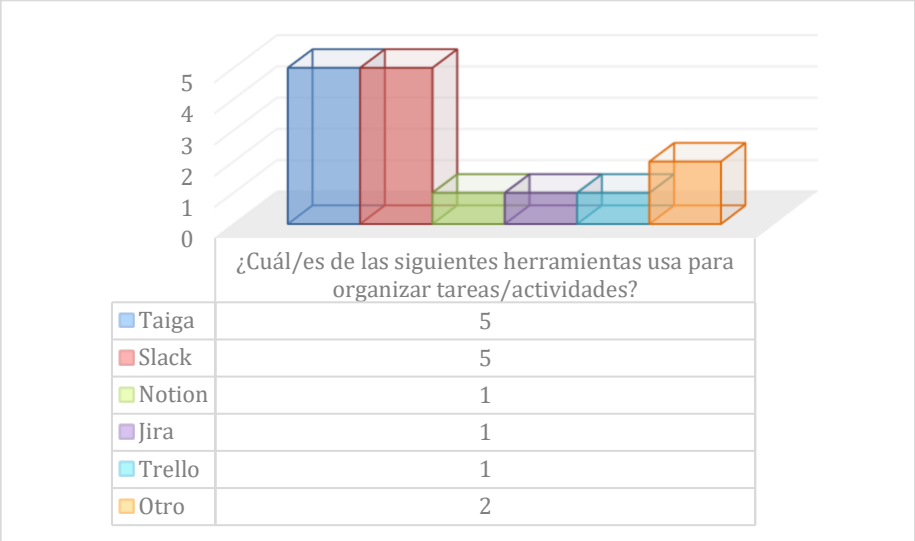


Fig. 30. Resultados encuesta - herramientas organizar tareas/actividades

El 83% de los encuestados utilizan Taiga y Slack como herramienta para organizar tareas/actividades. Lo que supone que cuentan con estas herramientas como medios para comunicación u organización de los proyectos ejecutados por el equipo de desarrollo.

14. ¿Qué tan frecuente utiliza contenedores (Docker)?

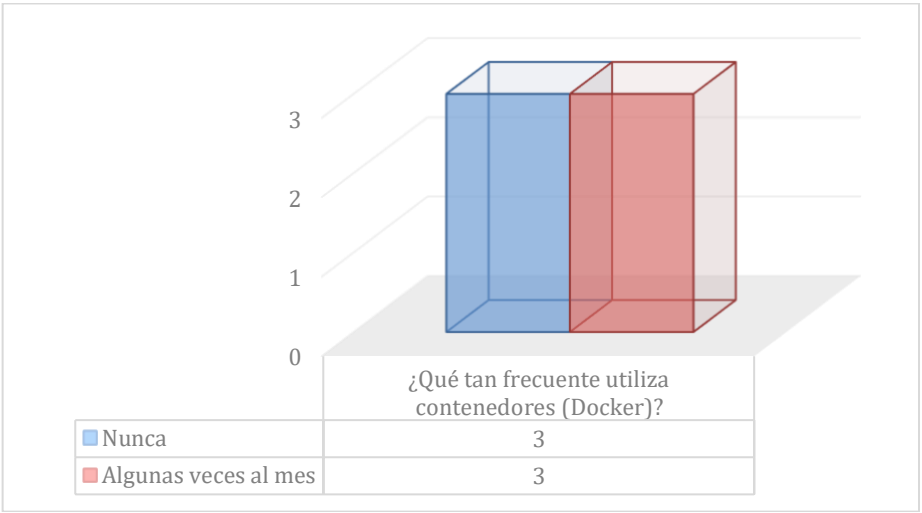


Fig. 31. Resultados encuesta - frecuencia de uso de contenedores

De los encuestados un 50% manifiesta utilizar esporádicamente Docker en sus actividades, mientras que el otro 50% no lo requiere para sus actividades. Esto nos sugiere que Docker no es utiliza con frecuencia dentro del proceso de desarrollo de software.

15. ¿Qué lenguaje de programación utiliza principalmente?

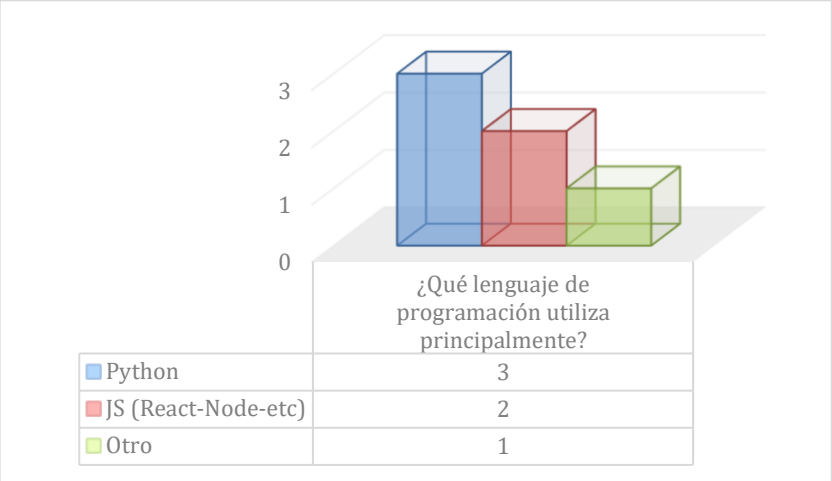


Fig. 32. Resultados encuesta - lenguaje de programación

El 50% de los encuestados utiliza el lenguaje de programación Python y así mismo un 33% utiliza JS (React-Node-etc). Esto nos da a comprender que los proyectos de desarrollo de software, en su mayoría, son construidos con Python y JavaScript (o haciendo uso de algún framework para agilizar su desarrollo).

16. Indique el primer proyecto al que dedica mayor cantidad de tiempo en desarrollo de funcionalidades / correcciones de bug's

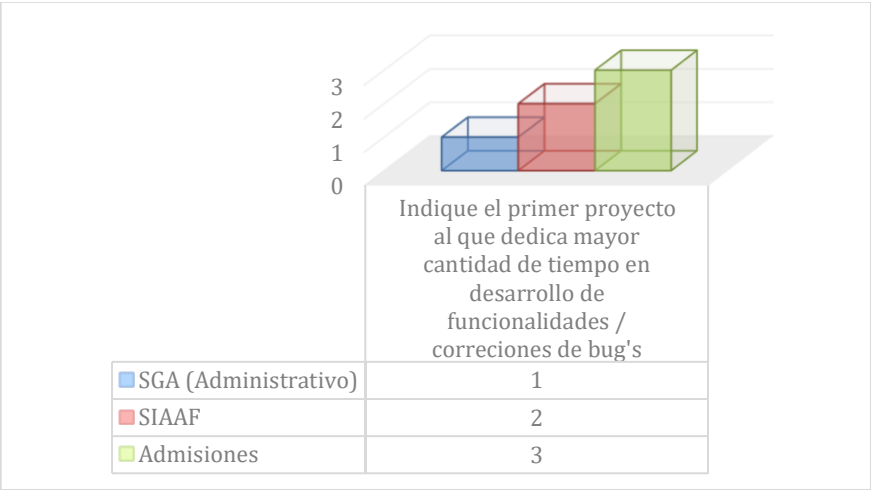


Fig. 33. Resultados encuesta – primer proyecto de mayor dedicación

El 50% del equipo de desarrollo emplea mayor cantidad de tiempo, como primera prioridad, a construir código en el proyecto de Admisiones y el 33% en desarrollar sobre el proyecto.

17. Indique el segundo proyecto al que dedica mayor cantidad de tiempo en desarrollo de funcionalidades / correcciones de bug's

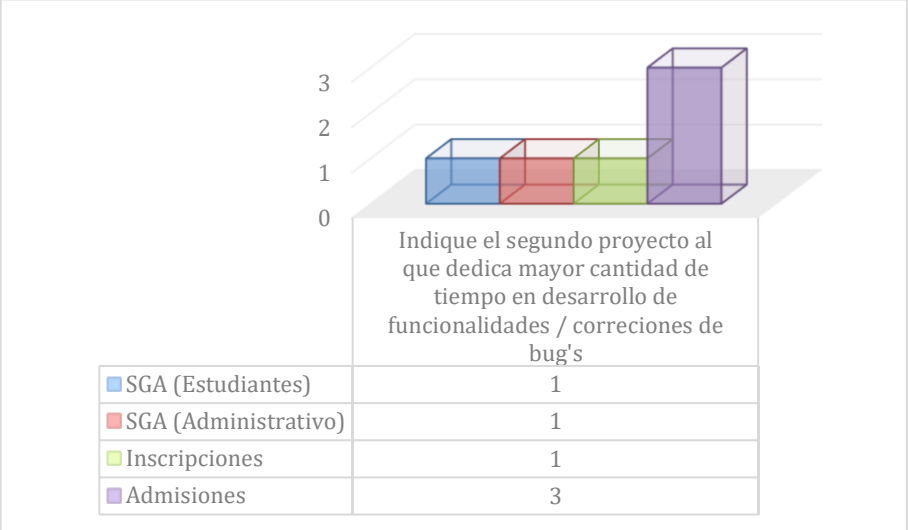


Fig. 34. Resultados encuesta - segundo proyecto de mayor dedicación

El 50% del equipo de desarrollo emplea mayor cantidad de tiempo, como segunda prioridad, a construir código en el proyecto Admisiones**.**

18. Indique el tercer proyecto al que dedica mayor cantidad de tiempo en desarrollo de funcionalidades / correcciones de bug's

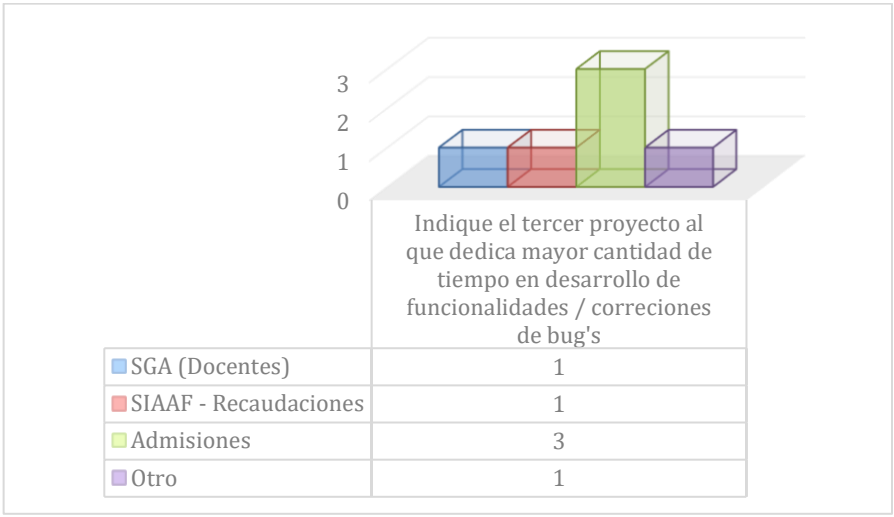


Fig. 35. Resultados encuesta - tercer proyecto de mayor dedicación

El 50% del equipo de desarrollo emplea mayor cantidad de tiempo, como tercera prioridad, a construir código en el proyecto Admisiones**.**

Con el resultado de las preguntas 16, 17 y 18 nos sugiere que los encuestados desarrollan mejoras o correcciones, al proyecto de Admisiones**,** y en menor medida al Sistema de gestión Académico y SIAAF.

19. ¿Utiliza alguna herramienta para la automatización del paso a producción de nuevas funcionalidades / corrección de errores?

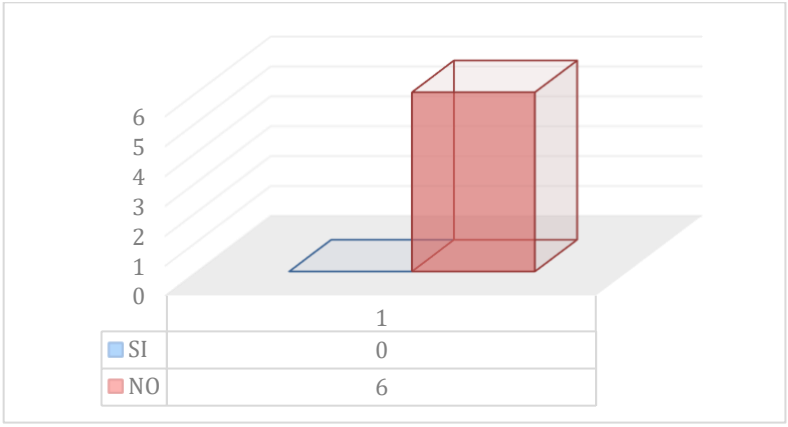


Fig. 36. Resultados encuesta - herramientas automatizadas para paso a producción

El 100% de los encuestados manifiesta que no se emplean herramientas automatizadas para el despliegue de nuevos cambios a los diferentes entornos de despliegue de aplicaciones.

20. Si eligió SI en la pregunta anterior, indique cuál:

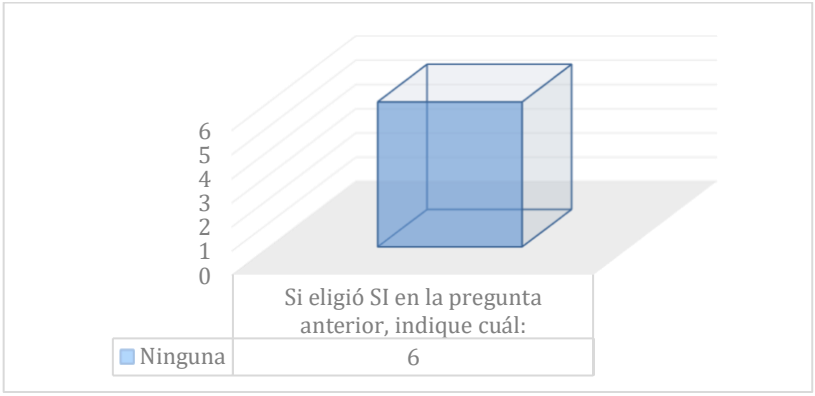


Fig. 37. Resultados encuesta - herramientas automatizadas de paso a producción

Con el resultado de la pregunta 19 y 20 se puede evidenciar que en la actualidad no se aplican herramientas que permitan de forma automatizada desplegar los cambios a los entornos de desarrollo, control de calidad y producción.

21. ¿Se ejecutan pruebas al código construido previo a subirlo a producción (Junit / Pytest / Otras)?

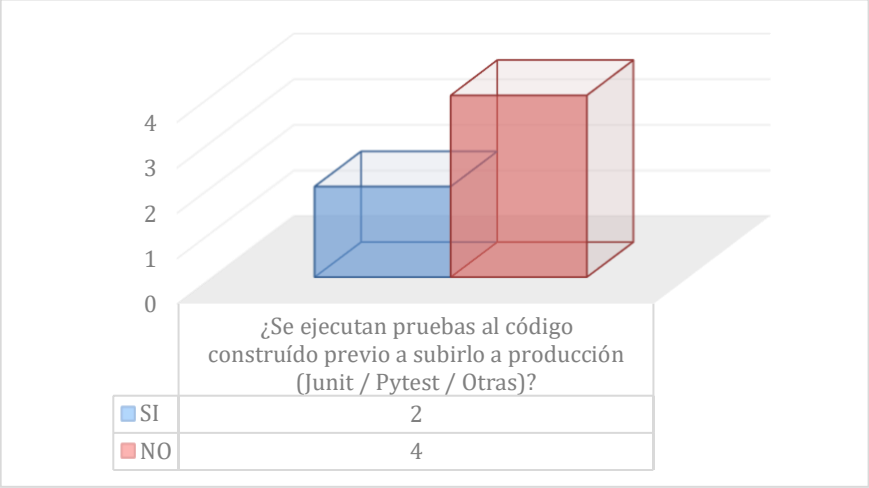


Fig. 38. Resultados encuesta - pruebas unitarias

El 83% de los encuestados manifiesta que no realiza pruebas unitarias al código. Lo que sugiere que las pruebas se realizan a nivel de validación de cumple/ no cumple la funcionalidad a través de pruebas por la interfaz gráfica.

22. ¿Cuál/es de las siguientes herramientas utiliza para analizar la calidad del código?

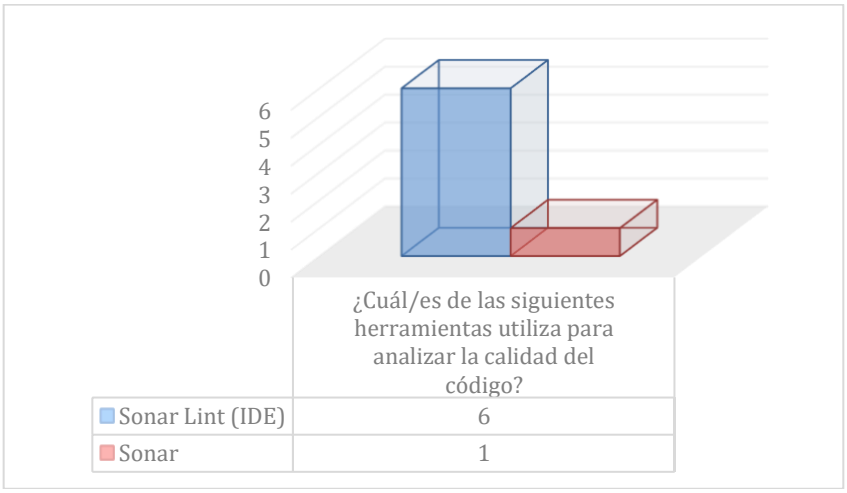


Fig. 39. Resultados encuesta - herramientas calidad de código

El 100% de los encuestados manifiesta utilizar herramientas automatizadas para analizar la calidad del código. Lo que sugiere que el desarrolladore recibe una retroalimentación de la herramienta para reducir así la complejidad cognitiva, duplicidad de código, entre otras; que el código pueda contener.

23. ¿Cuál/es de los siguientes formatos usa al momento de realizar sus actividades de desarrollo de software?

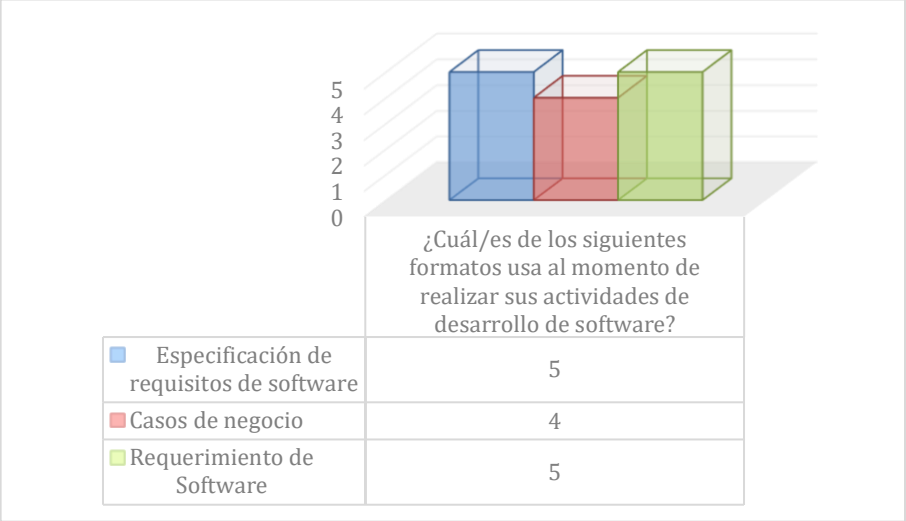


Fig. 40. Resultados encuesta - formatos desarrollo de software

El 100% de los encuestados manifiesta utilizar los documentos de Especificación de requisitos de software y de Requerimientos de Software, y también del total de encuestados el 88% emplea el documento de Caso de Negocio. Estos resultados implican que el equipo cuenta con el conocimiento y aplica los mismos en sus proyectos.

24. Si en la pregunta anterior eligió la opción Otras, especifique cuales:

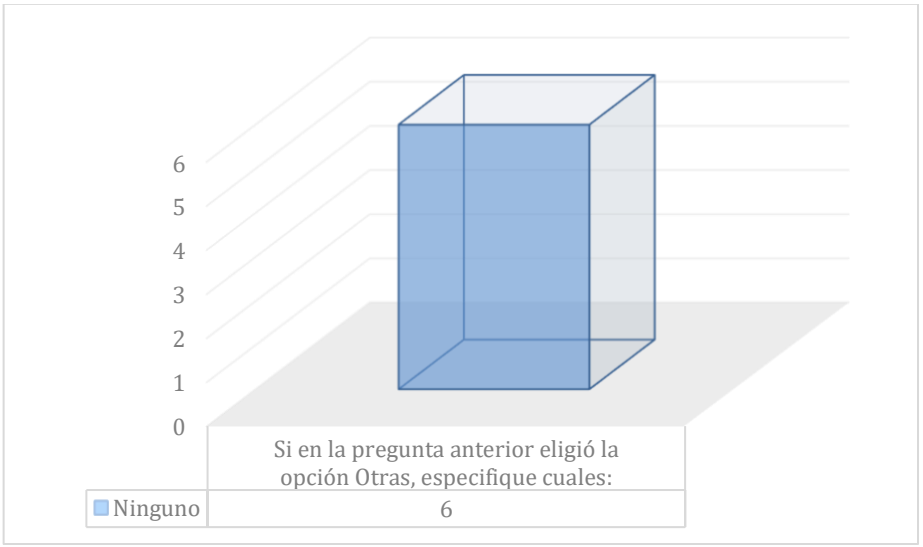


Fig. 41. Resultados encuesta - otros formatos desarrollo de software

El 100% de los encuestados manifiesta que no utiliza documentos adicionales a los elegidos en la pregunta 23.

Anexo 6. Análisis de Resultados de Encuesta – Perfil Seguridad de la Información

1. ¿Como calificaría su nivel de experiencia en seguridad de aplicaciones web?

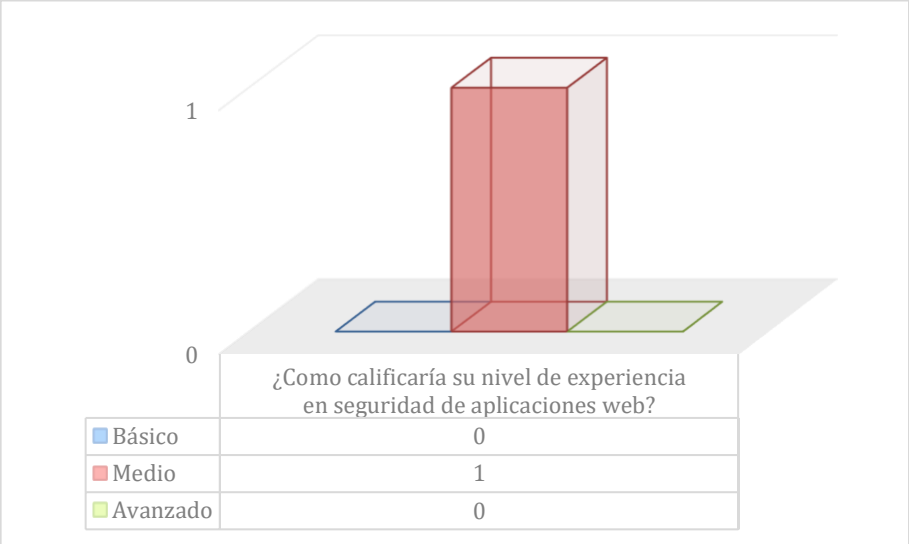


Fig. 42. Resultados encuesta - nivel de experiencia en seguridad de aplicaciones web

El encuestado manifiesta tener un conocimiento medio en el manejo de seguridad de aplicaciones web, lo que permite identificar que cuenta con la experiencia necesaria para evaluar las aplicaciones web.

2. De las siguientes herramientas: ¿Cuáles emplea para realizar pruebas de seguridad en las aplicaciones web?

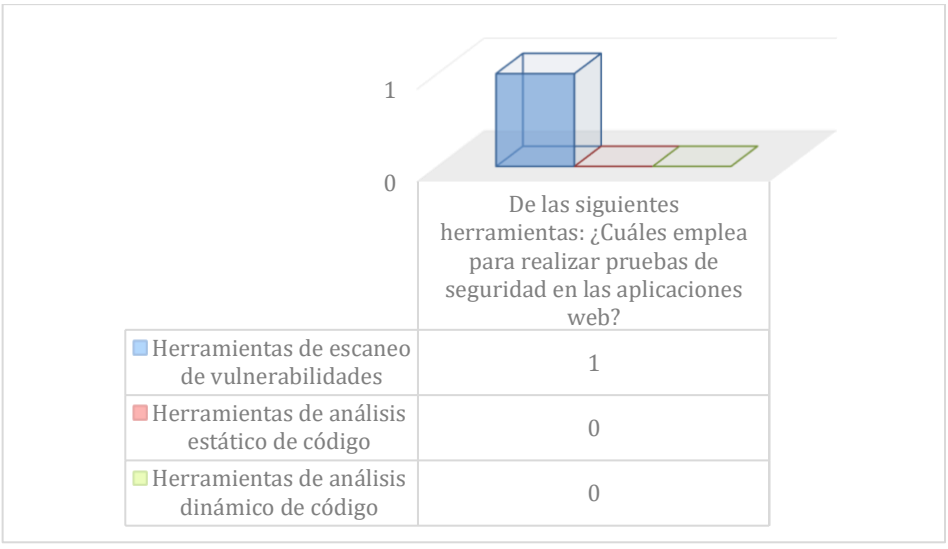


Fig. 43. Resultados encuesta - herramientas empleadas en pruebas de seguridad

El encuestado manifiesta que, de las herramientas listadas, emplea Herramientas de escaneo de vulnerabilidades para realizar las pruebas de seguridad en aplicaciones web.

3. ¿Qué herramientas de escaneo de vulnerabilidades conoce?

Respuesta: "*OWASP, NESSUS, MALTEGO, ARACHNI, NMAP*"
El encuestado manifiesta emplear las herramientas como **OWASP, NESSUS, MALTEGO, ARACHNI y NMAP,** para realizar las pruebas de seguridad en aplicaciones web.

4. ¿Qué herramientas de análisis estático de código conoce?

Respuesta: "*Actualmente, no se tiene acceso al código fuente de los proyectos, por lo que no se realiza este tipo de análisis (Review Board, Crucible, CodeScene)***"**

El encuestado manifiesta que, por el momento, no se cuenta con acceso al código fuente de los proyectos, motivo por el cual aún no es posible realizar este tipo de análisis.

5. ¿Qué herramientas de análisis dinámico de código conoce?

Respuesta: "*Actualmente, no se tiene acceso al código fuente de los proyectos, por lo que no se realiza este tipo de análisis (Review Board, Crucible, CodeScene)***"**

El encuestado manifiesta que, por el momento, no se cuenta con acceso al código fuente de los proyectos, motivo por el cual aún no es posible realizar este tipo de análisis.

6. ¿De qué forma evalúa las aplicaciones web?

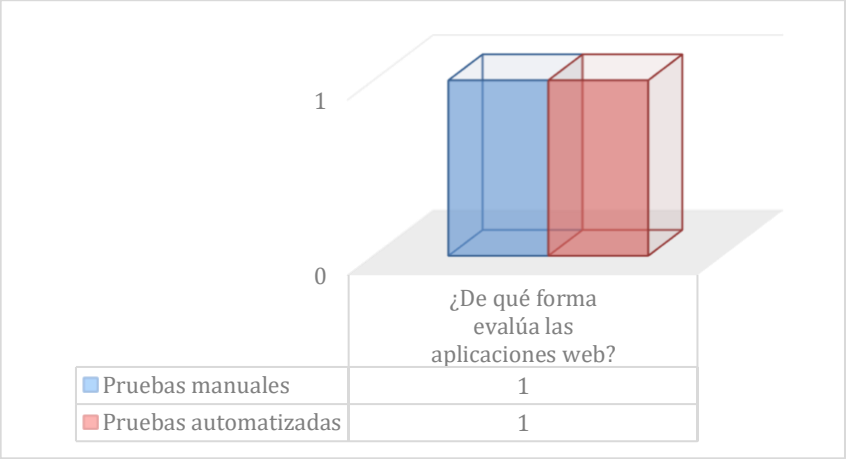


Fig. 44. Resultados encuesta - forma de elaborar pruebas de aplicaciones web

El encuestado manifiesta que actualmente utiliza pruebas manuales y automatizadas para evaluar las aplicaciones web.

7. ¿Qué desafíos enfrenta al evaluar la seguridad de aplicaciones web?

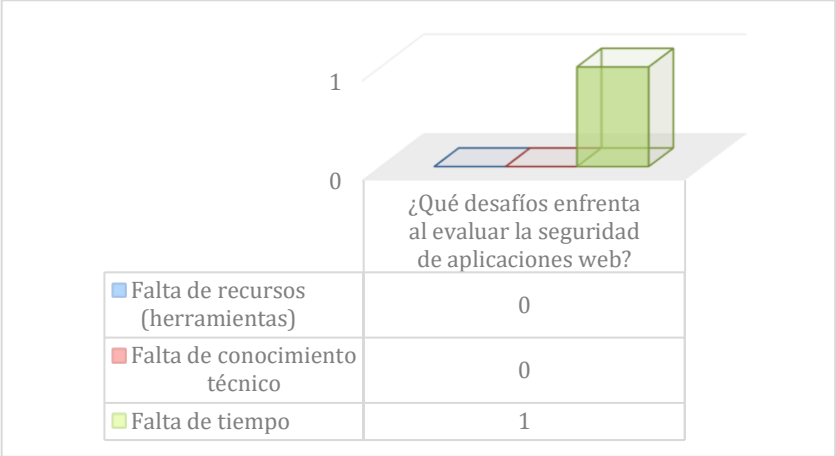


Fig. 45. Resultados encuesta - desafíos al evaluar la seguridad de aplicaciones web

El encuestado manifiesta que lo principales desafíos que tiene al momento de evaluar la seguridad de las aplicaciones web. Esto nos sugiere que al momento de realizar los análisis de aplicaciones web, por optimizar el tiempo, no se puede aplicar todos los análisis que se recomiendan.

8. ¿Qué tipo de métodos aplica para mantener la seguridad de las aplicaciones web?

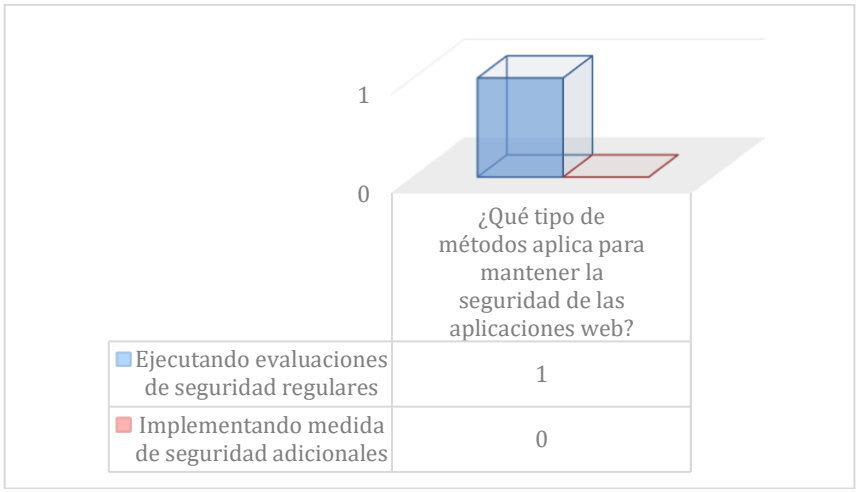


Fig. 46. Resultados encuesta - métodos para mantener la seguridad de aplicaciones web

El encuestado manifiesta que realiza evaluaciones de seguridad regulares de las aplicaciones. Esto nos sugiere que una vez que las aplicaciones llegan a un entorno de producción, son evaluadas para minimizar el riesgo de exponer o que se vuelva vulnerable los datos de las aplicaciones web.

9. ¿Cómo maneja los problemas de seguridad en las aplicaciones web?

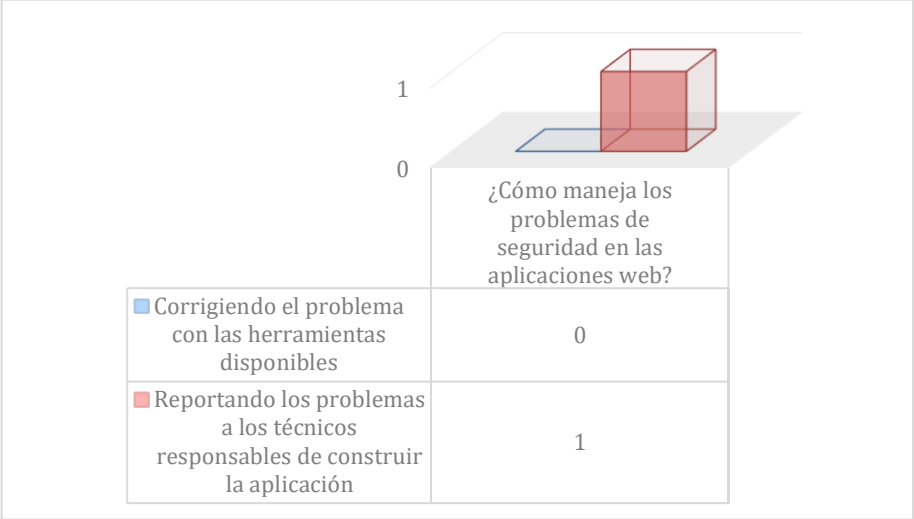


Fig. 47. Resultados encuesta – manejo de problemas de seguridad de aplicaciones web

El encuestado manifiesta que, una vez identificado los problemas de seguridad, estos se reportan directamente a los técnicos/desarrolladores responsables para su corrección.

10. ¿Cómo se asegura que las aplicaciones web sean compatibles con los estándares de seguridad?

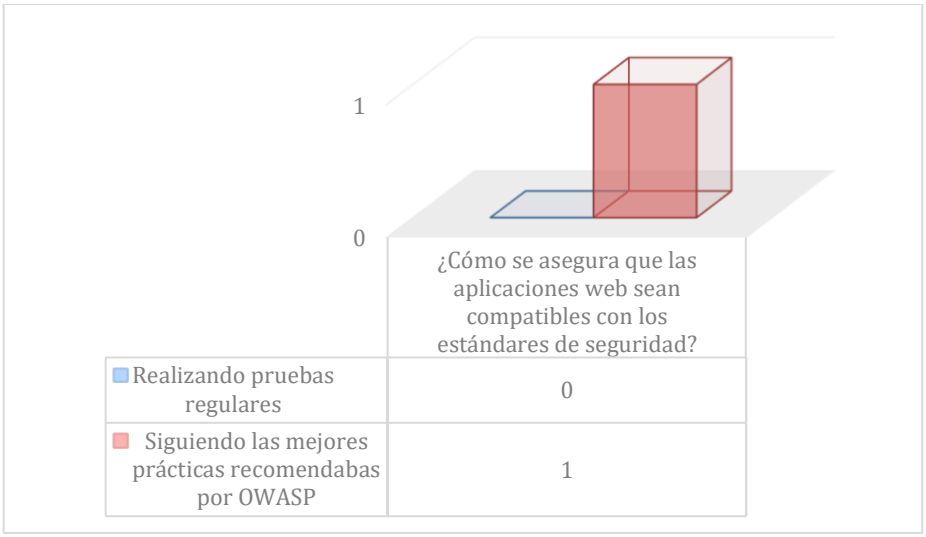


Fig. 48. Resultados encuesta - aseguramiento compatibilidad con los estándares de seguridad

El encuestado manifiesta que se para asegurar que las aplicaciones web sean compatibles con los estándares de seguridad, se siguen las prácticas recomendadas por OWASP.

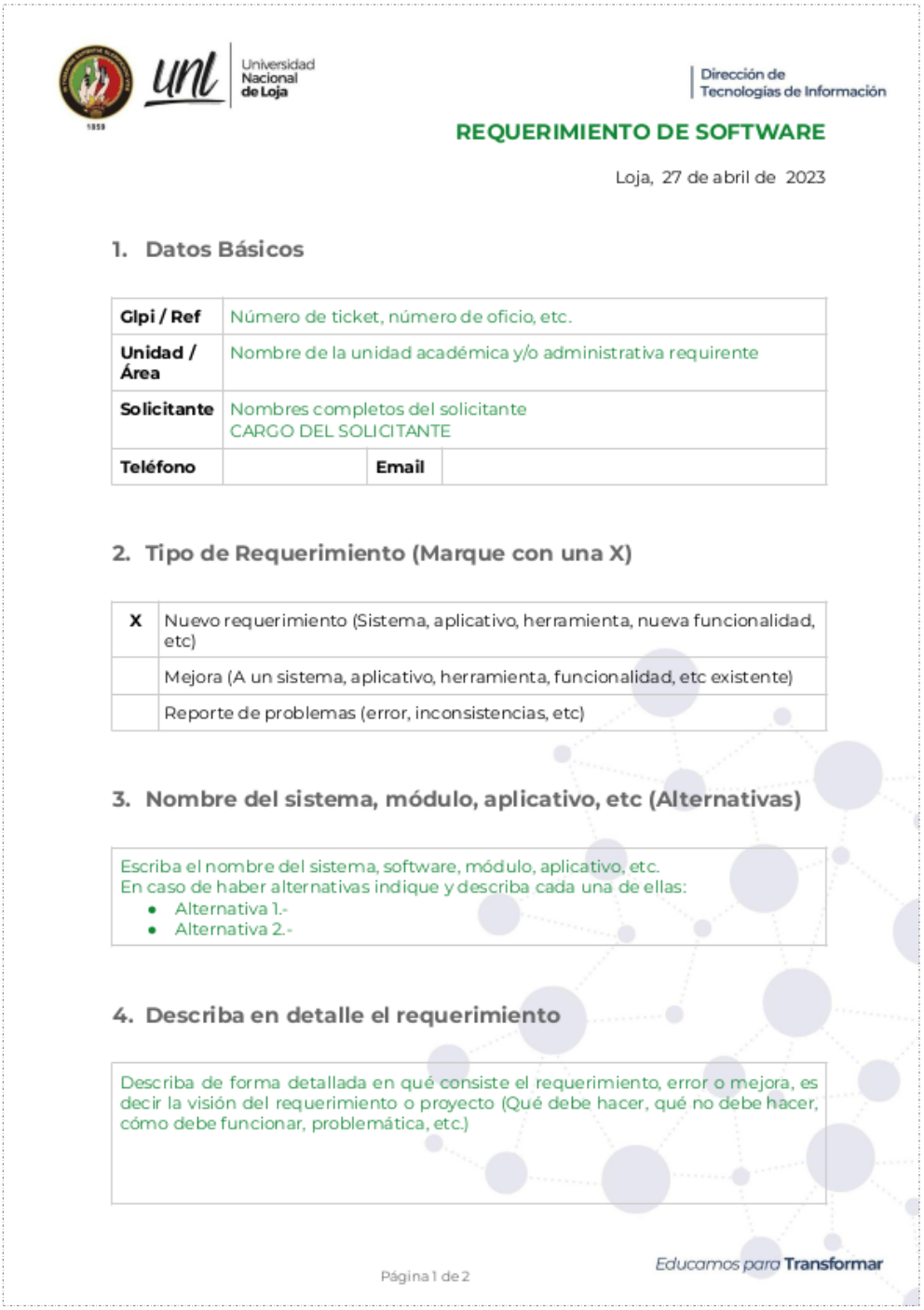
11. ¿Qué normativas/estándares aplica al realizar pruebas de seguridad a las aplicaciones web?

Respuesta: "*OWASP (Open Web Application Security Project) ISO/IEC 27001 y El Estándar de Pruebas de Penetración (PTES).*

*Cabe mencionar que estos marcos son tomados como referencia, más no se han implementado como política institucional.***"**

El encuestado manifiesta que se toman como referencia los marcos mencionados, aunque esto no implique que sean parte de la política institucional.

Anexo 7. Formato 001 - Requerimiento de Software



Universidad Nacional de Loja

Dirección de Tecnologías de Información

REQUERIMIENTO DE SOFTWARE

Loja, 27 de abril de 2023

1. Datos Básicos

Glpi / Ref	Número de ticket, número de oficio, etc.		
Unidad / Área	Nombre de la unidad académica y/o administrativa requirente		
Solicitante	Nombres completos del solicitante CARGO DEL SOLICITANTE		
Teléfono		**Email**	

2. Tipo de Requerimiento (Marque con una X)

X	Nuevo requerimiento (Sistema, aplicativo, herramienta, nueva funcionalidad, etc)
	Mejora (A un sistema, aplicativo, herramienta, funcionalidad, etc existente)
	Reporte de problemas (error, inconsistencias, etc)

3. Nombre del sistema, módulo, aplicativo, etc (Alternativas)

Escriba el nombre del sistema, software, módulo, aplicativo, etc.
En caso de haber alternativas indique y describa cada una de ellas:

- Alternativa 1.-
- Alternativa 2.-

4. Describa en detalle el requerimiento

Describa de forma detallada en qué consiste el requerimiento, error o mejora, es decir la visión del requerimiento o proyecto (Qué debe hacer, qué no debe hacer, cómo debe funcionar, problemática, etc.)

Página 1 de 2

Educamos para **Transformar**

Nota: Documento_F001_REQUERIMIENTO_SOFTWARE.docx, disponible en https://1drv.ms/w/s!Amh-r5SS12tKgkSy4-CSAePoojEj?e=B4MWwx.

Anexo 8. Formato 002 - Caso de Negocio

Universidad Nacional de Loja

Dirección de Tecnologías de Información

Caso de Negocio Nro.: UNL-DTI-2023-001

Loja, 27 de abril del 2023

Nombre del proyecto

Unidad requirente	Nombre unidad requirente
Cliente	Nombre Cliente **CARGO**
Referencias	Oficios, estudios, informes Ticket

Elaborado por:	Nombre Especialista SI **CARGO**
Revisado por:	Nombres **CARGO** Nombres **CARGO**
Aprobado por:	Nombres **CARGO**

CONTENIDOS

1. Resumen Ejecutivo 3
2. Definición del Problema 3
3. Visión General del Proyecto 3
3.1. Descripción 3
3.2. Requerimientos del alto nivel 3
3.3. Objetivos e indicadores de éxito 3
3.4. Premisas 4
3.5. Restricciones 4
4. Alineación del Proyecto con los Objetivos Estratégicos 4
5. Análisis de Alternativas 4
5.1. Alternativa seleccionada 4
6. Propuesta calendario / tiempos 5
7. Análisis Costo Beneficio 5
7.1. Costos 5
7.2. Beneficios 5
8. Conclusiones 5
9. Aprobaciones 6

Nota: Documento _F002_CASO_NEGOCIO.docx disponible en https://1drv.ms/w/s!Amh-r5SS12tKgkUc6pPTpVLMBW2N?e=fhzayd.

Anexo 9. Formato 003 – Estudio de Factibilidad

Universidad Nacional de Loja

Dirección de Tecnologías de Información

Informe Nro.: UNL-DTI-2023-000

Para:	Tit. Nombres Apellidos **DIRECTOR DE TECNOLOGÍAS DE INFORMACIÓN** email:
De:	Tit. Nombres Apellidos **CARGO**
Asunto:	Estudio de Factibilidad: [Nombre del proyecto]
Ticket:	Nro. Ticket / Oficio u otra referencia
Fecha:	Loja, 00 de mes del yyyy

RESUMEN EJECUTIVO

- Debe ser conciso y abarcar en resumen todo el estudio.
- Resalta los hallazgos claves de todo el estudio.
- Prepararlo al final (después de completar todo el resto de esta plantilla).

ANTECEDENTES DEL PROYECTO

En esta sección se presenta una introducción de la visión del proyecto y sus orígenes:

- Qué cosas dio origen a la necesidad y al proyecto.
- Factores impulsadores de la investigación de factibilidad.
- Quién inició el proyecto (Persona u organización).
- Los interesados clave (Stakeholders) involucrados en el inicio del proyecto, fue un departamento de la organización o quizás otra empresa, fundación o grupo de la comunidad trajo la idea.
- Los interesados clave según la visión del proyecto.
- Actividades de anteproyecto que se han realizado antes del estudio de factibilidad.

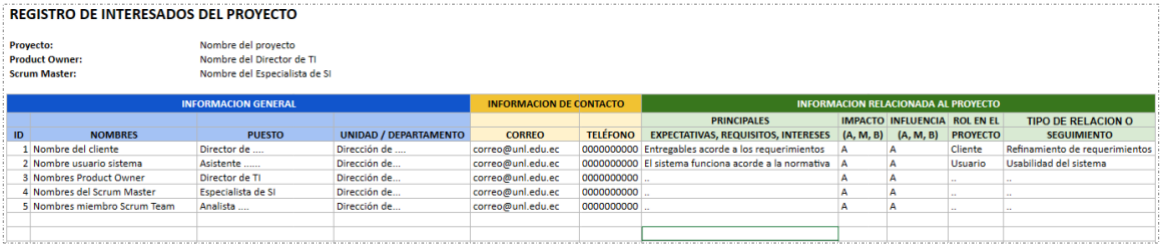
CONTEXTO

Descripción: Propósito principal del proyecto con sus principales entregables.

Objetivos: Definir objetivos y por cada uno que se va a hacer y qué beneficios entrega para solventar lo que pide el cliente.

Nota: Documento _F003_ESTUDIO_FACTIBILIDAD.docx disponible en https://1drv.ms/w/s!Amh-r5SS12tKgkZALge0PjYcFvlT?e=ad9lz8.

Anexo 10. Formato 004 - Lista de Stakeholders



REGISTRO DE INTERESADOS DEL PROYECTO

Proyecto: Nombre del proyecto
Product Owner: Nombre del Director de TI
Scrum Master: Nombre del Especialista de SI

INFORMACION GENERAL				INFORMACION DE CONTACTO		INFORMACION RELACIONADA AL PROYECTO				
ID	NOMBRES	PUESTO	UNIDAD / DEPARTAMENTO	CORREO	TELÉFONO	PRINCIPALES EXPECTATIVAS, REQUISITOS, INTERESES	IMPACTO (A, M, B)	INFLUENCIA (A, M, B)	ROL EN EL PROYECTO	TIPO DE RELACION O SEGUIMIENTO
1	Nombre del cliente	Director de	Dirección de	correo@unl.edu.ec	0000000000	Entregables acorde a los requerimientos	A	A	Cliente	Refinamiento de requerimientos
2	Nombre usuario sistema	Asistente	Dirección de...	correo@unl.edu.ec	0000000000	El sistema funciona acorde a la normativa	A	A	Usuario	Usabilidad del sistema
3	Nombres Product Owner	Director de TI	Dirección de...	correo@unl.edu.ec	0000000000	..	A	A	..	..
4	Nombres del Scrum Master	Especialista de SI	Dirección de...	correo@unl.edu.ec	0000000000	..	A	A	..	..
5	Nombres miembro Scrum Team	Analista	Dirección de...	correo@unl.edu.ec	0000000000	..	A	A	..	..

Nota: Documento _F004_LISTA_STAKEHOLDERS.xlsx disponible en https://1drv.ms/x/s!Amh-r5SS12tKgkdXB3QAHI2ki2Ln?e=rM94fn.

Anexo 11. Formato 005 - Acta de Constitución del Proyecto

Universidad Nacional de Loja

Dirección de Tecnologías de Información

Acta de Constitución del Proyecto Nro.: UNL-DTI-2023-001

Loja, 27 de abril del 2023

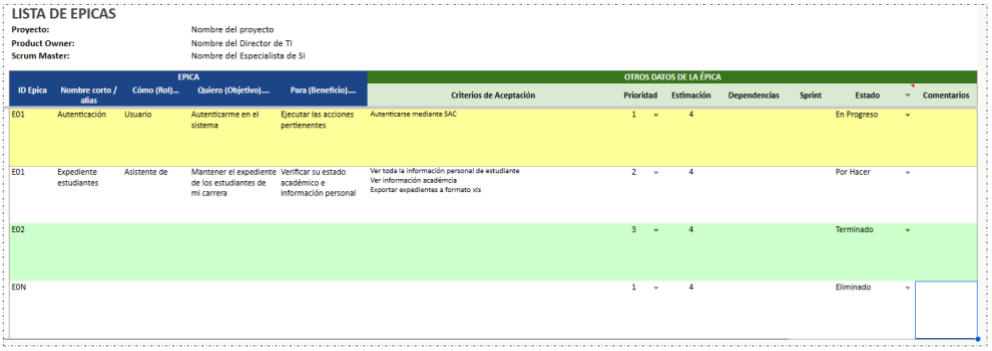
Nombre del proyecto

Unidad requirente	Nombre unidad requirente
Cliente	Nombre Cliente **CARGO**
Patrocinador principal	Nombre Rector / Otro patrocinador **CARGO**
Product Owner	Nombre Director TI **CARGO**
Scrum Master	Nombre Especialista SI **CARGO**

Elaborado por:	Nombre Especialista SI **CARGO**
Revisado por:	Nombres **CARGO** Nombres **CARGO**
Aprobado por:	Nombres **CARGO**

Nota: Documento _F005_ACTA_CONSTITUCIÓN_PROYECTO.docx disponible en https://1drv.ms/w/s!Amh-r5SS12tKgklductSs85Jzbmr?e=y5tauJ.

Anexo 12. Formato 006 – Épicas, Historias de Usuario, Backlog del producto, Backlog del Sprint



LISTA DE EPICAS

Proyecto: Nombre del proyecto
Product Owner: Nombre del Director de TI
Scrum Master: Nombre del Especialista de SI

EPICA					OTROS DATOS DE LA ÉPICA					
ID Épica	Nombre corto / alias	Cómo (Rol)...	Quiero (Objetivo)...	Para (Beneficio)...	Criterios de Aceptación	Prioridad	Estimación	Dependencias	Sprint	Estado	Comentarios
E01	Autenticación	Usuario	Autenticarme en el sistema	Ejecutar las acciones pertienentes	Autenticarse mediante SAC	1	4			En Progreso	
E01	Expediente estudiantes	Asistente de	Mantener el expediente de los estudiantes de mi carrera	Verificar su estado académico e información personal	Ver toda la información personal de estudiante Ver información académica Exportar expedientes a formato xls	2	4			Por Hacer	
E02						3	4			Terminado	
E0N						1	4			Eliminado	

BACKLOG DEL PRODUCTO

Proyecto: Nombre del proyecto
Product Owner: Nombre del Director de TI
Scrum Master: Nombre del Especialista de SI

EPICA		HISTORIA DE USUARIO				OTROS DATOS DE LA ÉPICA O HISTORIA DE USUARIO					
ID Épica	ID	Cómo (Rol)...	Quiero (Objetivo)....	Para (Beneficio)....	Criterios de Aceptación	Priorida	Estimac	Dependen	Sprint	Estado	Comentari
E01	H001	Director de ..	Registrar el periodo académico de admisión	Iniciar con el proceso de admisión	Se visualiza el periodo académico registrado.	1	4		1	Por Hacer	
E01	H002	Director de ..	Configurar el periodo académico de admisión	Establecer el cronograma que se llevará a cabo dentro del periodo académico de admisión	Se visualiza el periodo académico configurado.	1	4		1	En Progreso	
E02	H003	Director de ..	Registrar la oferta académica en el período académico de admisión	Agregar los programas de estudio que se van a ofertar con sus especificaciones	Se despliega la lista de programas de estudio ofertados en el periodo de admisión.	1	4		1	Terminado	
E0N	H004	Director de ..	Consumir la información de la oferta académica registrada en el período académico de admisión	Cargar esta información en la plataforma informática del SNNA	Se visualiza un botón para descargar el reporte de la configuración de la oferta académica de un periodo académico de admisión y sus especificaciones (número de cupos, modalidad de estudio y jornada).	1	4		2	Eliminado	

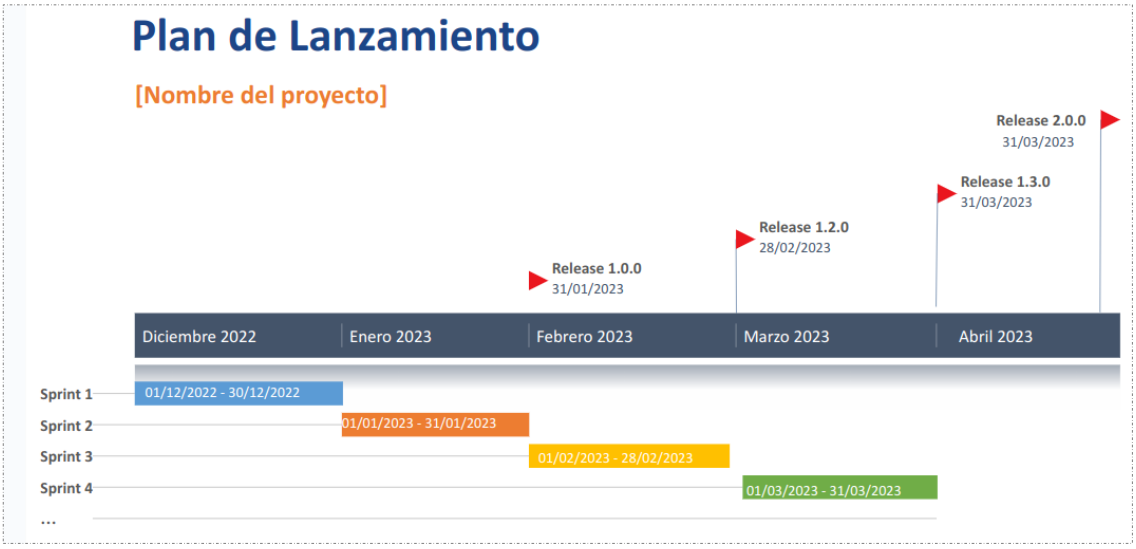
BACKLOG DEL SPRINT: N

Proyecto: Nombre del proyecto
Product Owner: Nombre del Director de TI
Scrum Master: Nombre del Especialista de SI
Esfuerzo estimado: 12 — Esfuerzo pendiente: 8
F. Inicio: 01/01/2023 — F. Fin: 31/01/2023

EPICA		HISTORIA DE USUARIO				OTROS DATOS DE LA HISTORIA DE USUARIO					
ID Épica	ID	Cómo (Rol)...	Quiero (Objetivo)....	Para (Beneficio)....	Criterios de Aceptación	Priorida	Estimac	Dependen	Sprint	Estado	Comentari
E01	H001	Director de ..	Registrar el periodo académico de admisión	Iniciar con el proceso de admisión	Se visualiza el periodo académico registrado.	1	4		1	Por Hacer	
E01	H002	Director de ..	Configurar el periodo académico de admisión	Establecer el cronograma que se llevará a cabo dentro del periodo académico de admisión	Se visualiza el periodo académico configurado.	1	4		1	En Progreso	
E02	H003	Director de ..	Registrar la oferta académica en el período académico de admisión	Agregar los programas de estudio que se van a ofertar con sus especificaciones	Se despliega la lista de programas de estudio ofertados en el periodo de admisión.	1	4		1	Terminado	

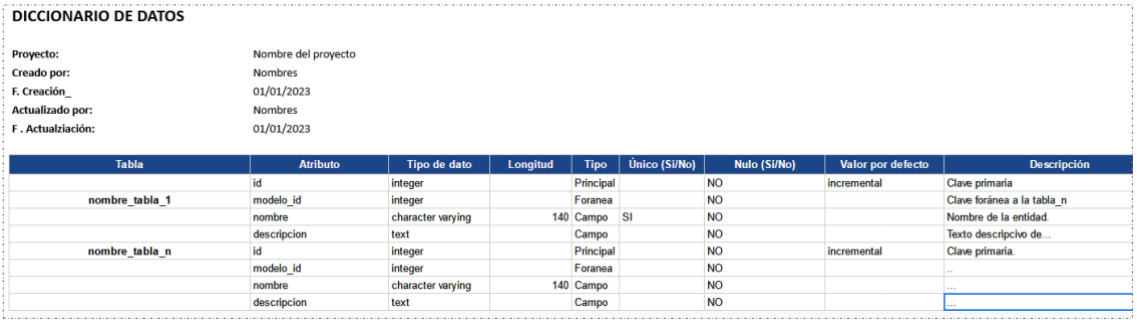
Nota: Documento _F006_BACKLOG.xlsx disponible en https://1drv.ms/x/s!Amh-r5SS12tKgkiDag5I_STu7LVU?e=oAhXPb.

Anexo 13. Formato 007 - Plan de Lanzamiento



Nota: Documento _F007_PLAN_LANZAMIENTO.pptx disponible en https://1drv.ms/p/s!Amh-r5SS12tKgkp-E565VYNDY_TP?e=nTDZkV.

Anexo 14. Formato 008 - Diccionario de Datos



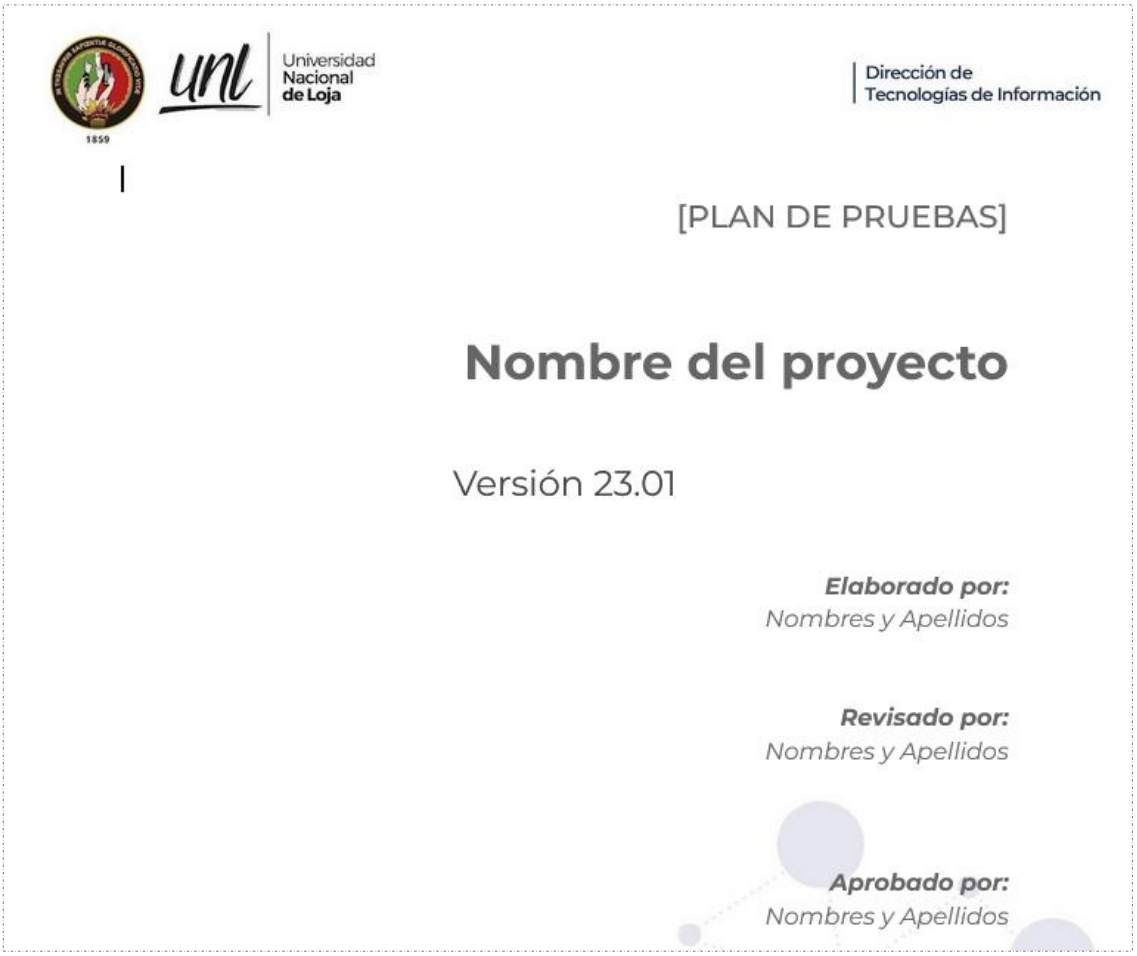
DICCIONARIO DE DATOS

Proyecto:	Nombre del proyecto
Creado por:	Nombres
F. Creación_	01/01/2023
Actualizado por:	Nombres
F . Actualziación:	01/01/2023

Tabla	Atributo	Tipo de dato	Longitud	Tipo	Único (Si/No)	Nulo (Si/No)	Valor por defecto	Descripción
	id	integer		Principal		NO	incremental	Clave primaria
nombre_tabla_1	modelo_id	integer		Foranea		NO		Clave foránea a la tabla_n
	nombre	character varying	140	Campo	SI	NO		Nombre de la entidad.
	descripcion	text		Campo		NO		Texto descripcivo de...
nombre_tabla_n	id	integer		Principal		NO	incremental	Clave primaria.
	modelo_id	integer		Foranea		NO		..
	nombre	character varying	140	Campo		NO		...
	descripcion	text		Campo		NO		...

Nota: Documento _F008_DICIONARIO_DATOS.xlsx disponible en https://1drv.ms/x/s!Amh-r5SS12tKgkwRn68jzbQ3S7w5?e=vW9zhZ.

Anexo 15. Formato 009 - Plan de Pruebas



Nota: Documento _F009_PLAN_PRUEBAS.docx disponible en https://1drv.ms/w/s!Amh-r5SS12tKgkuCB7KFyI8BDSlk?e=IbexOH.

Anexo 16. Formato 010 - Casos de Prueba

Plantilla de Casos de Prueba de Software

Proyecto:	**[Nombre del proyecto]**		**Versión:**
Sprint / Historia / Requisito:	Sprint 001 - Registrar la asistencia del docente al aula de clases.		**Último cambio:**
Analista de Pruebas:	Nombres Apellidps		**Descripción del cambio**
Tester:			

Diseño Casos de Prueba

ID	Fecha	Precondiciones	Descripción	Pasos
CASOS DE PRUEBA FUNCIONALES				
HU01-CP001	26/08/2022	* El docente debe haber sido registrado en el Active Directory. * El docente debe tener carga horaria.	Verificar que las credenciales del docente se encuentren vigentes. **(Ejecutar prueba de regresión: requisitos asociados)**	Ingresar credenciales del docente (usuario y clave) en la página de autenticación.
HU01-CP002	26/08/2022	* El docente debe tener carga horaria.	Verificar que al docente se le habilite la agenda del día. **(Ejecutar prueba de regresión: CU001)**	1. Ingresar al sistema 2. El sistema le muestra la agenda del día
HU01-CP003	26/08/2022	* El docente debe tener carga horaria.	Verificar que el botón de **"Iniciar clase"** se encuentre en cada asignatura de la agenda del docente.	1. Ingresar al sistema 2. El sistema le muestra la agenda del día
HU01-CP004	26/08/2022	* Estar en el campus universitario.	Verificar que al docente se le habiliten los botones de registro de asistencia únicamente en el campus universitario: * aula de clases, * laboratorios o * secretarías satélites	1. Ingresar al sistema desde el campus universitario. 2. El sistema le muestra la agenda del día

Nota: Documento _F010_CASOS_PRUEBA.xlsx disponible en https://1drv.ms/x/s!Amh-r5SS12tKgk5UMgY2z4iQ-RGK?e=2ezD2h.

Anexo 17. Formato 011 - Lista de Incidentes

LISTA DE INCIDENTES

PROYECTO:	**Nombre del proyecto**
SPRINT:	0
RESPONSABLE:	
DESCRIPCIÓN:	Control de Calidad de los Sistemas Institucionales
FECHA:	26/1/2023

Nro.	Caso de prueba	Descripción	Estado	Tipo	Tester	Té
1	CP001	Examenes de selección simple	DESARROLLO	MEJORA	Nombre Apellido	Nombr
2	CP002	Formato para la carga de evaluaciones	PENDIENTE	MEJORA	Nombre Apellido	Nombr
8			PENDIENTE			
4			PENDIENTE			
5			PENDIENTE			

Nota: Documento _F011_LISTA_INCIDENTES.xlsx disponible en https://1drv.ms/x/s!Amh-r5SS12tKgk2Vao_hQLecuffS?e=GQsrhb.

Anexo 18. Formato 012 – Informe de Pruebas y Certificación de Control de Calidad

Universidad Nacional de Loja

Dirección de Tecnologías de Información

Informe Nro.: UNL-DTI-2023-001

Para:	Nombres Apellidos **DIRECTOR DE TECNOLOGÍAS DE INFORMACIÓN** email: correoi@unl.edu.ec
De:	Nombres Apellidos **CARGO**
Asunto:	**Informe de Pruebas y Certificación de Control de Calidad**: Sprint 0 - Descripción corta del sprint (hitos, funcionalidades) **Nombre proyecto**
Referencia Tickets - Oficios No	0000 - Decreto Ejecutivo Nro. XYZ 0000 - Oficio Nro ....
Fecha:	Loja, 01 de enero del 2023

ANTECEDENTES

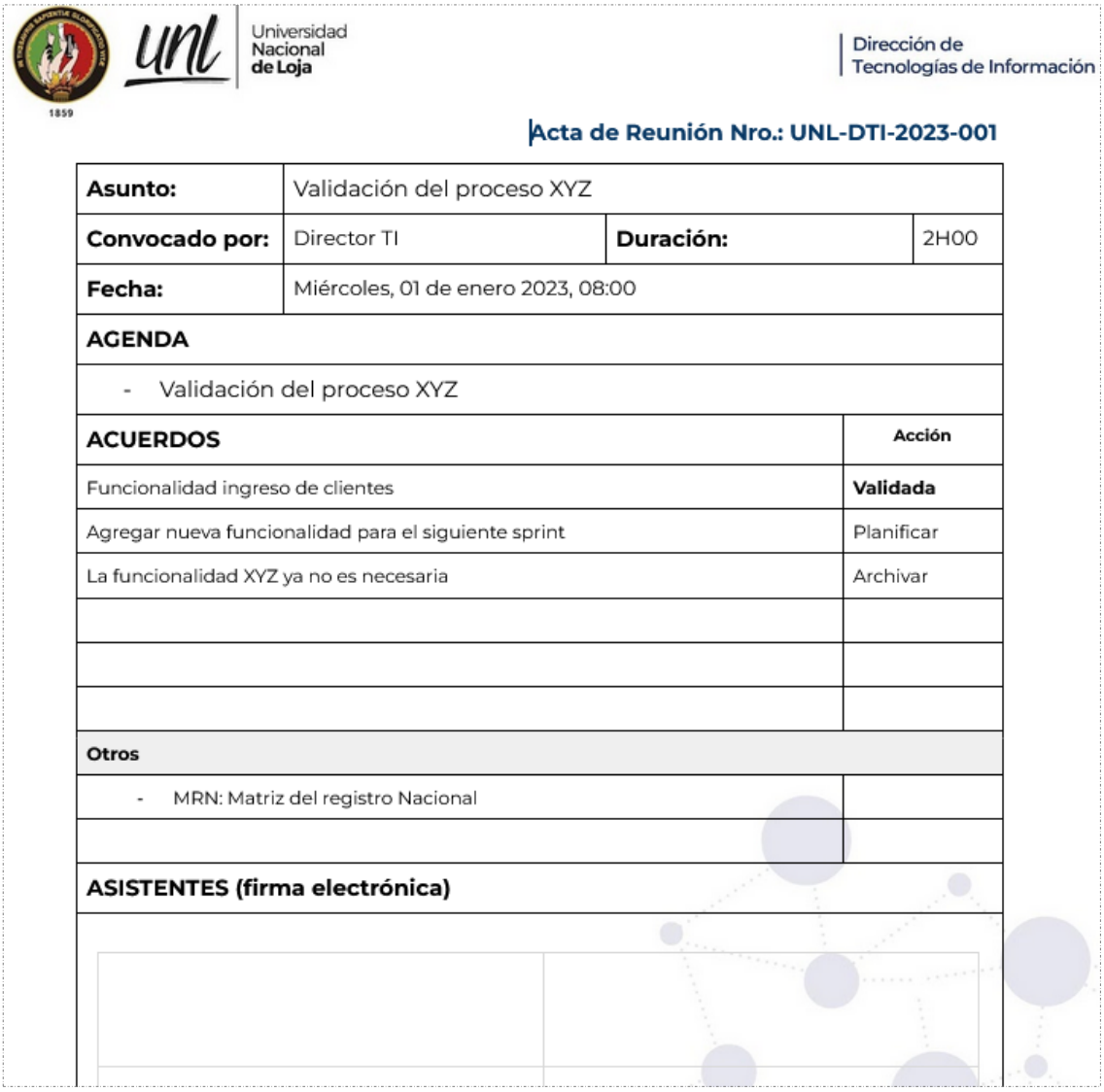
Las funcionalidades para el desarrollo e implementación en la automatización del proceso de XYZ para las XYZ el Sprint 4, se lleva a cabo considerando los antecedentes y normativas indicados a continuación:

CERTIFICACIÓN DE CONTROL DE CALIDAD

Luego de haber culminado exitosamente el control de calidad del **Sprint 0** - Nombre corto / descripción, con un total de **41 historias de usuario**. Con el **proceso de control de calidad**, se analizaron **50 casos de prueba**, en los que se encontraron **53 incidencias,** entre las que se describen **20** bug y **33** mejoras, clasificadas de acuerdo a su gravedad (1 baja, 47 media, 4 alta y 1 críticas). Todos los incidentes fueron analizados y reportados al Equipo SI, los mismos que fueron solventados, por lo que el producto / entregable se encuentra validado y listo para puesta en producción.

Nota: Documento _F012_INFORME_PRUEBAS.docx disponible en https://1drv.ms/w/s!Amh-r5SS12tKgk824LWlnbyAuRQJ?e=ZYYQhj.

Anexo 19. Formato 013 – Acta de Reunión



Universidad Nacional de Loja

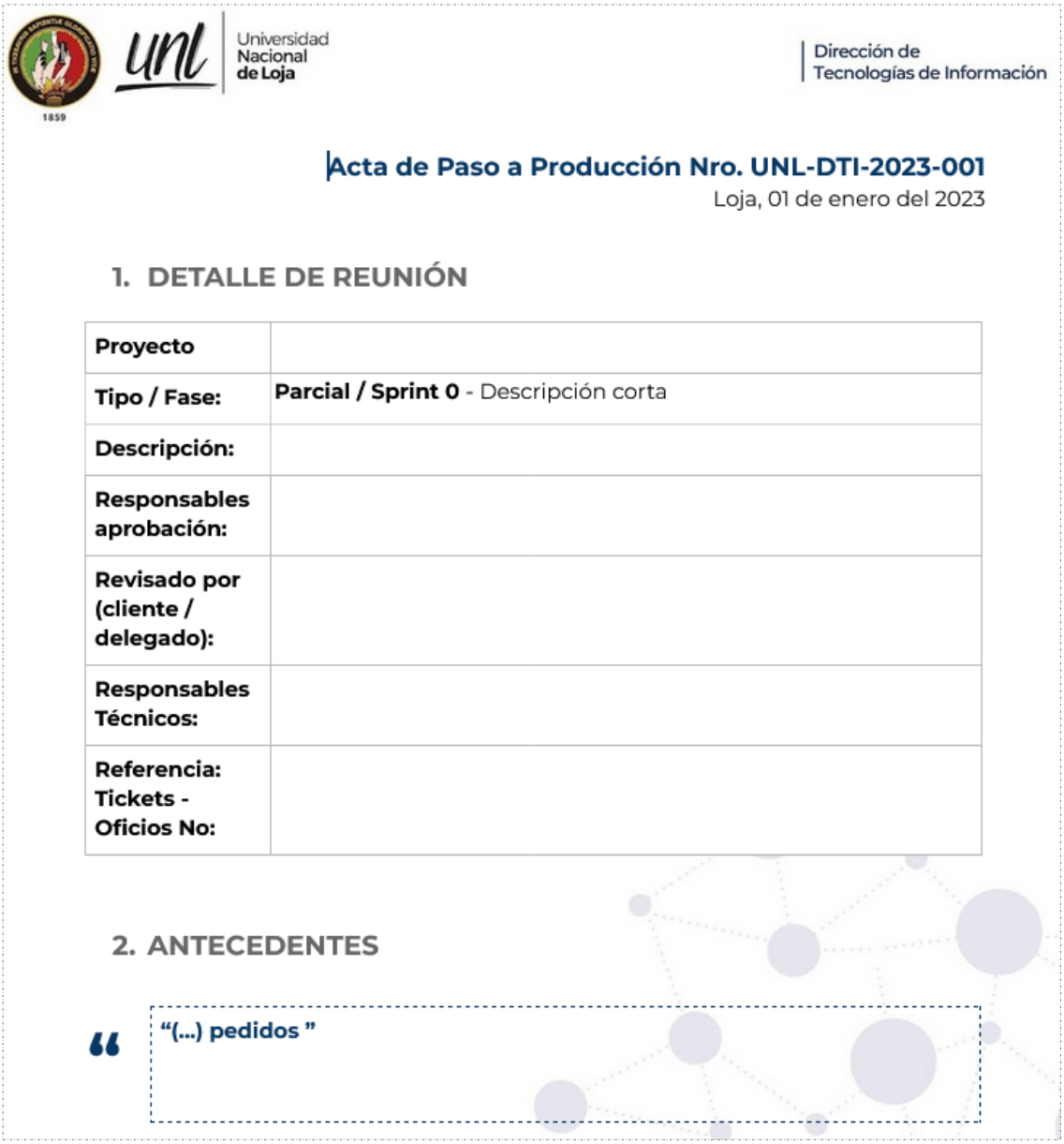
Dirección de Tecnologías de Información

Acta de Reunión Nro.: UNL-DTI-2023-001

Asunto:	Validación del proceso XYZ		
Convocado por:	Director TI	**Duración:**	2H00
Fecha:	Miércoles, 01 de enero 2023, 08:00		
AGENDA			
- Validación del proceso XYZ			
ACUERDOS			**Acción**
Funcionalidad ingreso de clientes			**Validada**
Agregar nueva funcionalidad para el siguiente sprint			Planificar
La funcionalidad XYZ ya no es necesaria			Archivar
Otros			
- MRN: Matriz del registro Nacional			
ASISTENTES (firma electrónica)			

Nota: Documento _F013_ACTA_REUNIÓN.docx disponible en https://1drv.ms/w/s!Amh-r5SS12tKglC1R7CZ4ocsmAJ8?e=MckX5v.

Anexo 20. Formato 014 – Acta de Paso a Producción



Universidad Nacional de Loja

Dirección de Tecnologías de Información

Acta de Paso a Producción Nro. UNL-DTI-2023-001

Loja, 01 de enero del 2023

1. DETALLE DE REUNIÓN

Proyecto	
Tipo / Fase:	**Parcial / Sprint 0** - Descripción corta
Descripción:	
Responsables aprobación:	
Revisado por (cliente / delegado):	
Responsables Técnicos:	
Referencia: Tickets - Oficios No:	

2. ANTECEDENTES

"(...) pedidos "

Nota: Documento _F014_ACTA_PASO_PRODUCCIÓN.docx disponible en https://1drv.ms/w/s!Amh-r5SS12tKglFXkF44EXLfvkHK?e=i7610L.

Anexo 21. Guía de desarrollo de software

CONTENIDOS

1. Introducción **5**

2. Repositorio de código fuente **5**

Proyectos 5
Ramas 5
Uso Gitlab 6
Clonar un proyecto 6
Branchs 6
Cambios 6
Rebase 7

3. Herramienta de Gestión de Tareas **8**

4. Documentación **8**

Documentos Esenciales 8
Análisis 9
Diseño 9
Qa 10
Paso a Producción 11
Documentos optativos 11

5. Arquitectura **12**

6. Codificación **14**

Documentación del código 14
Consideraciones de codificación 15
Modelos y clases 16
Html 17
Javascript 18
Css 18
Base de datos 18
Interfaz de usuario 18

7. QA **19**

Pruebas automáticas 20
Test Driven Development Django 20
Herramientas de validación de código (QA) 22

8. Proyectos con Django (Siaaf, Eventos, xxx) **23**

Estructura de proyecto 23
Parametrizaciones 24
Codificación forms, urls y views 25
Auditoría 27
Reportes 28

Anexo 22. Propuesta presentada a la DTI

Universidad Nacional de Loja

Dirección de Tecnologías de Información

[PROPUESTA]

<<Proceso: Sistemas de Información>>

Propuesta de un marco de trabajo metodológico de desarrollo de software

Unidad	Dirección de Tecnologías de Información
Tipo documento	Propuesta
Fecha	27 de abril de 2023
Versión	**1.0**

Elaborado por:	Danny Emanuel Muñoz Flores **TESISTA** Máximo Andrés Álvarez Pacheco **TESISTA**
Revisado por:	Ing. Edison Leonardo Coronel Mg.Sc. **DIRECTOR DEL PROYECTO DE TITULACIÓN**
Aprobado por:	Ing. Jhon Alexander Calderón Sanmartín **DIRECTOR DE TECNOLOGÍAS DE INFORMACIÓN**

Nota: Documento _PROPUESTA_MARCO_TRABAJO_DESAROLLO_SOFTWARE.docx disponible en https://1drv.ms/w/s!Amh-r5SS12tKgT2aZNBWBhcIPE5g?e=ZODmbn

Anexo 23. Certificado de traducción del resumen

CERTIFICADO DE TRADUCCIÓN

Lady Marvelia Rodríguez Paucar
LICENCIADA EN CIENCIAS DE LA EDUCACIÓN ESPECIALIDAD IDIOMA INGLÉS

CERTIFICA:

Haber realizado la traducción al Idioma Inglés correspondiente al resumen del trabajo investigativo titulado**: Propuesta de un marco de trabajo metodológico de desarrollo de software para la Dirección de Tecnologías de Información de la Universidad Nacional de Loja**, cuyos autores son: **DANNY EMANUEL MUÑOZ FLORES** con CI: **1104285604** y **MÁXIMO ANDRÉS ÁLVAREZ PACHECO** con CI **1104128887**.

Es todo cuánto puedo certificar en honor a la verdad y autorizo al portador de este documento para que pueda hacer uso del mismo en lo que estime conveniente.

Loja, 29 de abril del 2023

Lady Rodríguez Paucar
LICENCIADA EN IDIOMA INGLES
CI. 1103993596