



Universidad Nacional Mayor de San Marcos

Universidad del Perú. Decana de América

Facultad de Ingeniería de Sistemas e Informática

Escuela Profesional de Ingeniería de Software

**Implementación de prácticas DevOps en el proceso de
pruebas de una entidad financiera**

TRABAJO DE SUFICIENCIA PROFESIONAL

Para optar el Título Profesional de Ingeniero de Software

AUTOR

Katherine Luisa VELÁSQUEZ HINOSTROZA

ASESOR

Mg. Fany Yexenia SOBERO RODRÍGUEZ

Lima, Perú

2022



Reconocimiento - No Comercial - Compartir Igual - Sin restricciones adicionales

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

Usted puede distribuir, remezclar, retocar, y crear a partir del documento original de modo no comercial, siempre y cuando se dé crédito al autor del documento y se licencien las nuevas creaciones bajo las mismas condiciones. No se permite aplicar términos legales o medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier cosa que permita esta licencia.

Referencia bibliográfica

Velásquez, K. (2022). *Implementación de prácticas DevOps en el proceso de pruebas de una entidad financiera*. [Trabajo de suficiencia profesional de pregrado, Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería de Sistemas e Informática, Escuela Profesional de Ingeniería de Software]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	Katherine Luisa Velásquez Hinostroza
Tipo de documento de identidad	DNI
Número de documento de identidad	46601857
URL de ORCID	https://orcid.org/0000-0001-9867-8095
Datos de asesor	
Nombres y apellidos	Fany Yexenia Sobero Rodríguez
Tipo de documento de identidad	DNI
Número de documento de identidad	20120467
URL de ORCID	https://orcid.org/0000-0002-0323-6110
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	Nilo Eloy Carrasco Oré
Tipo de documento	DNI
Número de documento de identidad	09342780
Miembro del jurado 1	
Nombres y apellidos	Joel Fernando Machado Vicente
Tipo de documento	DNI
Número de documento de identidad	40476778
Miembro del jurado 2 (Asesor)	
Nombres y apellidos	Fany Yexenia Sobero Rodríguez
Tipo de documento	DNI
Número de documento de identidad	20120467
Datos de investigación	
Línea de investigación	No aplica

Grupo de investigación	No aplica
Agencia de financiamiento	Sin financiamiento
Ubicación geográfica de la investigación	País: Perú Departamento: Lima Provincia: Lima Distrito: Cercado de Lima Jr. Carlos Amezaga No. 375 Universidad Nacional Mayor de San Marcos Latitud: -12.0564232 Longitud: -77.0843327
Año o rango de años en que se realizó la investigación	2022
URL de disciplinas OCDE	2.02.04 -- Ingeniería de sistemas y comunicaciones https://purl.org/pe-repo/ocde/ford#2.02.04



UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
Escuela Profesional de Ingeniería de Software

Acta Virtual de Sustentación
del Trabajo de Suficiencia Profesional

Siendo las 21:00 horas del día 23 de agosto del año 2022, se reunieron virtualmente los docentes designados como Miembros de Jurado del Trabajo de Suficiencia Profesional, presidido por el Ing. Carrasco Ore Nilo Eloy (Presidente), Mg. Machado Vicente Joel Fernando (Miembro) y la Mg. Sobero Rodríguez Fany Yexenia (Miembro Asesor), usando la plataforma Meet (<https://meet.google.com/nvd-sypk-tbu>), para la sustentación virtual del Trabajo de Suficiencia Profesional intitulado: **“IMPLEMENTACIÓN DE PRÁCTICAS DEVOPS EN EL PROCESO DE PRUEBAS DE UNA ENTIDAD FINANCIERA”**, por la Bachiller **Velásquez Hinostroza Katherine Luisa**; para obtener el Título Profesional de Ingeniera de Software.

Acto seguido de la exposición del Trabajo de Suficiencia Profesional, el Presidente invitó a la Bachiller a dar las respuestas a las preguntas establecidas por los miembros del Jurado.

La Bachiller en el curso de sus intervenciones demostró pleno dominio del tema, al responder con acierto y fluidez a las observaciones y preguntas formuladas por los señores miembros del Jurado.

Finalmente habiéndose efectuado la calificación correspondiente por los miembros del Jurado, la Bachiller obtuvo la nota de **18 (DIECIOCHO)**.

A continuación el Presidente de Jurados el Ing. Carrasco Ore Nilo Eloy, declara a la Bachiller **Ingeniera de Software**.

Siendo las 21:55 horas, se levantó la sesión.

Presidente

Ing. Carrasco Ore Nilo Eloy

Miembro

Mg. Machado Vicente Joel Fernando

Miembro Asesor

Mg. Sobero Rodríguez Fany Yexenia



Universidad Nacional Mayor de San Marcos
Universidad del Perú. Decana de América
Facultad de Ingeniería de Sistemas e Informática
Escuela Profesional de Ingeniería Software

INFORME DE EVALUACIÓN DE ORIGINALIDAD
Nº 027-EPISW-FISI-2022

1. Autoridad Académica que emite el Informe de Originalidad:	Directora de la Escuela Profesional de Ingeniería de Software
2. Apellidos y Nombres de la autoridad académica:	Dra. Nora Bertha La Serna Palomino
3. Operador del programa informático de similitudes:	Dra. Nora Bertha La Serna Palomino
3. Documento evaluado:	Tesis para Pregrado Título: “Implementación de prácticas DevOps en el proceso de pruebas de una entidad financiera”
5. Autores del documento:	Velásquez Hinostroza, Katherine Luisa
6. Fecha de recepción de documento	Recepción: 10/09/2022
7. Fecha de aplicación del programa detector de similitudes:	Revisión: 10/09/2022
8. Software utilizado:	Turnitin
9. Configuración del programa detector de similitudes:	Excluye textos entrecomillados: Sí Excluye biografías: Sí Excluye cadenas menores a 40 palabras: Sí Otro criterio (especificar): No
10. Porcentaje de similitudes según programa detector de similitudes	Cinco por ciento (5%)
11. Fuentes originales de las similitudes encontradas	Se adjuntan en 01 (una) foja al presente informe
12. Observaciones:	Ninguna
13. Calificación de originalidad i. Documento cumple criterios de originalidad, sin observaciones ii. Documento cumple criterios de originalidad, con observaciones iii. Documento no cumple criterios de originalidad.	Documento cumple criterio de originalidad, sin observación
14. Fecha del Informe:	14/11/2022

Dra. Nora Bertha La Serna Palomino
Directora (e) de la EPISW

DEDICATORIA

Dedico este trabajo a mis padres, Luisa y Enrique, por su apoyo incondicional y ser siempre mi soporte en cada etapa, y a Guillermo y Bricia, por acompañarme en cada momento y brindarme aliento para perseguir mis objetivos.

AGRADECIMIENTOS

En primer lugar, agradezco a Dios, por todas las oportunidades que me brinda a diario.

A mi asesora, Mg. Fany Sobero Rodríguez, por su dedicación, sugerencias y recomendaciones para realizar el presente informe profesional.

A mis compañeros de trabajo, por ser grandes mentores y compartir sus conocimientos en cada organización en la cual he participado.

Y a mis grandes amigos, por siempre creer en mí y darme fuerzas en cada momento complicado

....

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
ESCUELA PROFESIONAL DE INGENIERÍA DE SOFTWARE

**Implementación prácticas DevOps en el proceso de pruebas de una entidad
financiera**

Autor: Velásquez Hinostroza, Katherine Luisa

Asesor: Sobrero Rodríguez, Fany Yexenia

Título: Trabajo de Suficiente Profesional para optar el Título Profesional de
Ingeniera de Software

Fecha: Junio de 2022

RESUMEN

Dentro de una entidad financiera, se identificó que la etapa de pruebas abarcaba el 30% de tiempo del ciclo de desarrollo de software, debido principalmente a diversas actividades manuales que se ejecutaban en esta fase. La entidad financiera ya había iniciado su camino hacia la adopción de prácticas DevOps, orientándose a la integración y entrega de valor continua, y automatizando los procesos de despliegue; sin embargo, estos conceptos no se estaban aplicando dentro del área de pruebas. Es por ello, que se decidió implementar un marco de trabajo de Testing Continuo para automatizar las pruebas, integrar este flujo con las herramientas DevOps que ya tenía la entidad financiera y adoptar prácticas que permitieran impulsar las pruebas hacía etapas tempranas del desarrollo de software, con el fin de mejorar la calidad de los entregables y reducir los tiempos de ejecución de las pruebas. Como parte de la implementación del proyecto se definieron nuevos procesos, roles y responsabilidades y se dispuso de nuevos artefactos, como un framework de automatización de pruebas, que fueron adoptados de forma incremental por los equipos de la entidad financiera. La adopción de Testing Continuo en estos equipos, dio como principales resultados una reducción en los tiempos de la etapa de pruebas y por consiguiente una reducción en los tiempos de entrega de valor a producción.

Palabras claves: Testing Continuo, DevOps, pruebas tempranas, automatización de pruebas, tiempo de entrega

MAJOR NATIONAL UNIVERSITY OF SAN MARCOS
FACULTY OF SYSTEMS AND INFORMATICS ENGINEERING
PROFESSIONAL SCHOOL OF SOFTWARE ENGINEERING

Implementation of DevOps practices in the testing process of a financial institution

Author: Velásquez Hinostrroza, Katherine Luisa

Advisor: Sobero Rodríguez, Fany Yexenia

Title: Professional Sufficiency Work to opt for the Professional Title of
Software Engineer

Date: June 2022

ABSTRACT

A financial entity identified that the testing stage covered 30% of the time of their software development life cycle, mainly due to different manual activities that were carried out in this phase. The entity had already begun the adoption of DevOps practices, focusing on continuous integration and delivery of value, and automating deployment processes; however, these concepts were not being applied within the tests area. For this reason, the entity decided to implement a Continuous Testing framework to automate tests, integrate this flow with the DevOps tools that they already had, and adopt practices that would allow the shift left testing; to improve the quality of the deliverables and reduce the execution times of the tests. To implement this project, new processes, roles, and responsibilities were defined and new artifacts were made available, such as a test automation framework, which were adopted incrementally by the teams of the institution. The mainly results, after the adoption of Continuous Testing in these teams, was a reduction in the times of the testing stage and therefore a reduction in the delivery times of the value to production.

Key words: Continuous Testing, DevOps, Shift Left, testing automatization, lead time

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS	VIII
ÍNDICE DE TABLAS.....	IX
INTRODUCCIÓN.....	1
CAPÍTULO I TRAYECTORIA PROFESIONAL	3
CAPÍTULO II CONTEXTO EN EL QUE SE DESARROLLA LA EXPERIENCIA	8
2.1 EMPRESA – ACTIVIDAD QUE REALIZA	8
2.2 VISIÓN	9
2.3 MISIÓN.....	9
2.4 ORGANIZACIÓN DE LA EMPRESA	9
2.5 ÁREA, CARGO Y FUNCIONES DESEMPEÑADAS	11
2.6 EXPERIENCIA PROFESIONAL REALIZADA EN LA ORGANIZACIÓN	12
CAPÍTULO III ACTIVIDADES DESARROLLADAS.....	13
3.1 SITUACIÓN PROBLEMÁTICA	13
3.1.1 <i>Definición del problema</i>	14
3.2 SOLUCIÓN	14
3.2.1 <i>Objetivos</i>	14
3.2.2 <i>Alcance</i>	15
3.2.3 <i>Etapas y metodologías</i>	16
3.2.4 <i>Fundamentos utilizados</i>	26
3.2.5 <i>Implementación de las áreas, procesos, sistemas y buenas prácticas</i>	34
CAPÍTULO IV REFLEXIÓN CRÍTICA DE LA EXPERIENCIA	68
CAPÍTULO V CONCLUSIONES Y RECOMENDACIONES.....	70
5.1 CONCLUSIONES	70
5.2 RECOMENDACIONES	71
5.3 FUENTES DE INFORMACIÓN.....	72

ÍNDICE DE FIGURAS

FIGURA 1: ORGANIGRAMA DE LA EMPRESA	10
FIGURA 2: ORGANIGRAMA DEL ÁREA DE DIVISIÓN DE TECNOLOGÍAS DE LA INFORMACIÓN.....	11
FIGURA 3: ROLES EN AGILIDAD DENTRO DE LA ENTIDAD FINANCIERA	18
FIGURA 4: FLUJO SCRUM DENTRO DE LA ENTIDAD FINANCIERA	19
FIGURA 5: FASES PARA LA ADOPCIÓN DE TESTING CONTINUO.....	20
FIGURA 6: LOS 4 PILARES DEL MARCO DE TRABAJO DE TESTING CONTINUO	21
FIGURA 7: NIVELES DE MADUREZ DE LOS APLICATIVOS.....	22
FIGURA 8: MATRIZ DE EVALUACIÓN RISK BASED TEST	23
FIGURA 9: REDACCIÓN EN GHERKIN.....	23
FIGURA 10: CREACIÓN DE UN CASO DE PRUEBA EN JIRA XRAY	24
FIGURA 11: FRAMEWORK DE AUTOMATIZACIÓN DE PRUEBAS FUNCIONALES.....	25
FIGURA 12: INTEGRACIÓN DE LAS PRUEBAS CON HERRAMIENTAS DEVOPS	26
FIGURA 13: MARCO DE TRABAJO SCRUM	27
FIGURA 14: ROLES EN SCRUM.....	28
FIGURA 15: ESTRATEGIA DE PRUEBAS TESTING CONTINUO.....	29
FIGURA 16: CONCEPTOS TESTING CONTINUO	30
FIGURA 17: ARQUITECTURA DEVOPS.....	31
FIGURA 18: END-TO-END DEVOPS TESTING	32
FIGURA 19: SHIFT LEFT TESTING	33
FIGURA 20: DIAGRAMA DE ARQUITECTURA JENKINS	45
FIGURA 21: DIAGRAMA DE REDES - HERRAMIENTAS DEVOPS	46
FIGURA 22: DIAGRAMA REDES – APLICATIVOS.....	48
FIGURA 23: SERVIDORES JENKINS CONFIGURADOS	51
FIGURA 24: PERFILADO JIRA XRAY.....	52
FIGURA 25: PERFILADO BITBUCKET	52
FIGURA 26: PERFILADO JENKINS	53
FIGURA 27: DISEÑO LIBRERÍA EJECUCIÓN PRUEBAS	54
FIGURA 28: LIBRERÍA EJECUCIÓN PRUEBAS AUTOMATIZADAS	55
FIGURA 29: CONFIGURACIÓN LIBRERÍA JENKINS	56
FIGURA 30: EJEMPLO USO LIBRERÍA TESTING	57
FIGURA 31: ESTRUCTURA PROYECTOS JIRA XRAY	58
FIGURA 32: ESTRUCTURA PROYECTOS BITBUCKET.....	59
FIGURA 33: FLUJO GIT FLOW.....	60
FIGURA 34: ESTRUCTURA CARPETAS JENKINS	61
FIGURA 35: PIPELINE DE TESTING	62
FIGURA 36: EJECUCIÓN PIPELINE TESTING DESDE JENKINS	63
FIGURA 37: REPORTE CUCUMBER JENKINS.....	64
FIGURA 38: RESULTADOS JIRA XRAY.....	64
FIGURA 39: OPTIMIZACIÓN DE LAS TAREAS EJECUTADAS POR LOS QE.....	65

ÍNDICE DE TABLAS

TABLA 1: INFORMACIÓN ACADÉMICA DE LA AUTORA	3
TABLA 2: CERTIFICACIONES DE LA AUTORA	4
TABLA 3: EXPERIENCIA PROFESIONAL DE LA AUTORA	5
TABLA 4: TIEMPOS ESTIMADOS DE EJECUCIÓN DE PRUEBAS MANUALES.....	13
TABLA 5: PRODUCT BACKLOG DEL PROYECTO PRIORIZADO POR EL PRODUCT OWNER.....	34
TABLA 6: HU: SQTC-1 - DEFINIR LA ARQUITECTURA	36
TABLA 7: HU: SQTC-2 - SOLICITAR APROVISIONAMIENTO DE LA ARQUITECTURA	36
TABLA 8: HU: SQTC-3 - HABILITAR COMUNICACIÓN CON HERRAMIENTAS DEVOPS.....	37
TABLA 9: HU: SQTC-4 - CONFIGURAR NODO LINUX.....	37
TABLA 10: HU: SQTC-5 - CONFIGURAR NODO WINDOWS	38
TABLA 11: HU: SQTC-6 - HABILITAR COMUNICACIÓN CON AMBIENTE PRUEBAS	38
TABLA 12: HU: SQTC-7 - DEFINIR PERFILADO	39
TABLA 13: HU: SQTC-8 - ANALIZAR Y DISEÑAR LIBRERÍA EJECUCIÓN PRUEBAS	39
TABLA 14: HU: SQTC-9 - IMPLEMENTAR IMÁGENES DOCKER	40
TABLA 15: HU: SQTC-10 - IMPLEMENTAR LIBRERÍA PARA EJECUTAR PRUEBAS	40
TABLA 16: HU: SQTC-11 - DESPLEGAR IMÁGENES DOCKER.....	41
TABLA 17: HU: SQTC-12 - DESPLEGAR LIBRERÍA PARA EJECUTAR PRUEBAS	41
TABLA 18: HU: SQTC-13 - ELABORAR MODELO OPERATIVO	42
TABLA 19: HU: SQTC-14 - DESPLEGAR MODELO OPERATIVO	42
TABLA 20: HU: SQTC-15 - REALIZAR PILOTOS DE ADOPCIÓN.....	43
TABLA 21: INFRAESTRUCTURA NUEVA SOLICITADA	49
TABLA 22: TIEMPO EJECUCIÓN PRUEBAS MANUALES	66
TABLA 23: TIEMPO EJECUCIÓN PRUEBAS AUTOMATIZADAS.....	66
TABLA 24: AHORRO TIEMPOS EJECUCIÓN PRUEBAS	67

INTRODUCCIÓN

El área de Tecnologías de la Información de la empresa ya contaba con unos años aplicando prácticas DevOps, lo cual permitió una reducción de hasta 40% del tiempo que les tomaba llevar nuevas soluciones a producción. Esta reducción del tiempo se dio gracias a la automatización de procesos de integración y despliegue continuo de las aplicaciones, sin embargo, se identificó que la etapa de pruebas no se vio influenciado por estas prácticas y demandaba un 30% del tiempo del ciclo de vida de desarrollo de software.

El principal problema se daba por la cantidad de tareas manuales y repetitivas que se ejecutaban en esta etapa al momento de realizar pruebas funcionales y de regresión, y que estaban expuestas a errores humanos o tenían dependencia con otros flujos, como la estabilidad de ambientes no productivos y generación de datos de pruebas. De esta manera, el ciclo de pruebas se extendía sin poder confirmar la fiabilidad de los entregables de desarrollo y así continuar con el despliegue en producción. Ante esta problemática, nació la necesidad de implementar prácticas de DevOps en las pruebas, Testing Continuo, con el fin de reducir los tiempos, mejorar la calidad y minimizar los riesgos de incidentes en producción.

El marco de trabajo de Testing Continuo permite realizar pruebas automatizadas como parte de la integración y entrega continua de los sistemas, permitiendo recibir feedback en etapas tempranas de desarrollo e identificar posibles riesgos asociados al producto. Se basa en una serie de buenas prácticas enfocadas al Shift Left y soportadas por herramientas DevOps que permiten su integración.

El presente informe se encuentra estructurado de la siguiente manera:

En el CAPITULO I se detalla cronológicamente la trayectoria profesional de la autora considerando sus cargos, funciones y actividades realizadas en diversas empresas, así como los grados y certificaciones obtenidos.

En el CAPITULO II se describe la historia e información de la empresa en donde se desarrolló el trabajo en mención, resaltando su estructura organizacional, misión y visión.

En el CAPITULO III se detalla el desarrollo del trabajo realizado que consiste en la implementación de prácticas DevOps en el proceso de Testing dentro del área de Tecnologías de la Información de la entidad financiera con la finalidad de mejorar la calidad y reducir los tiempos durante la etapa de pruebas de los aplicativos.

En el CAPITULO IV se describe una crítica de cómo se llevó a cabo la implementación del proyecto y las lecciones aprendidas.

En el CAPITULO V se detallan las conclusiones del proyecto y recomendaciones para las siguientes fases dentro de la entidad financiera.

CAPÍTULO I TRAYECTORIA PROFESIONAL

Presentación profesional

Soy una Bachiller en Ingeniería de Software con más de siete años de experiencia en las diferentes áreas del desarrollo de software tanto en empresas públicas como privadas. Durante estos años he llevado diversos cursos que han contribuido a mi desarrollo profesional en donde resalta mis conocimientos en soluciones móviles, web, cloud y metodologías de Integración Continua, Entrega Continua y Testing Continuo.

Me destaco por ser una persona proactiva, perseverante y autodidacta, con mucha curiosidad e ímpetu por aprender de manera continua.

Formación Académica

A continuación, se detallan los grados académicos obtenidos por la autora.

Tabla 1

Información académica de la autora

Educación Superior: Grado	Fechas
Diplomado en Gerencia de Proyectos en Tecnología de Información	2020-2021
Escuela de Posgrado	
Facultad de Ingeniería de Sistemas e Informática	
Universidad Nacional Mayor de San Marcos	
Bachiller en Ingeniería de Software	2009-2014
Escuela Académico Profesional de Ingeniería de Software	
Facultad de Ingeniería de Sistemas e Informática	
Universidad Nacional Mayor de San Marcos	

Nota. La tabla contiene los estudios realizados por la autora detallados en orden cronológico. Fuente: Elaboración propia.

Certificaciones

En la siguiente tabla se lista las certificaciones obtenidas por la autora durante su experiencia profesional.

Tabla 2

Certificaciones de la autora

Certificación / Curso	Fechas
DevOps Culture and Mindset Course Certificate Universidad de California, Davis, Coursera – ID de la credencial C377SR3WPWY5	2021
Continuous Testing Foundation (CTF) Certification DevOps Institute – ID de la credencial 21433299	2020
Microsoft Certified: Azure Fundamentals Microsoft – ID de la credencial H489-2685	2020
Scrum Master Certified SCRUMstudy – ID de la credencial 741611	2019
Scrum Developer Certified SCRUMstudy – ID de la credencial 741424	2019
Scrum Fundamentals Certified SCRUMstudy – ID de la credencial 740774	2019

Nota. La tabla contiene las certificaciones obtenidas por la autora durante su experiencia profesional y sus respectivas credenciales, detalladas en orden cronológico. Fuente: Elaboración propia.

Experiencia

A continuación, se describe la experiencia profesional de la autora especificando los diferentes cargos y funciones que ha desempeñado.

Tabla 3

Experiencia profesional de la autora

Cargo / Empresa	Fechas
Ingeniera DevOps – Banco de Crédito del Perú Responsable de soluciones de automatización e implementación de nuevas herramientas que contribuyen a los procesos de Integración Continua, Entrega Continua y Testing Continuo dentro del área de Tecnologías de la Información del banco.	Feb. 2020 – Actualidad
Analista Programador Java – Vector Software Factory Vector ITC es un grupo tecnológico que provee soluciones tecnológicas a diferentes clientes del sector bancario como Santander y BANBIF. Principales funciones: – Análisis y especificación de requerimientos. – Análisis y diseño de soluciones utilizando tecnologías como Spring Boot y Docker. – Análisis, diseño e implementación de los aplicativos del cliente SANTANDER. – Análisis, diseño e implementación de servicios REST y SOAP de la capa de integración del cliente BANBIF. – Control y seguimiento de las tareas del equipo de desarrollo. – Elaboración de la documentación de análisis y diseño. – Soporte durante los pases de producción.	Jun. 2018 – Ene. 2020
Arquitecto de Software – Sapia SAPIA es una empresa proveedora de soluciones tecnológicas para el cliente América Móvil Perú SAC, Claro. Principales funciones:	Jul. 2015 – May. 2018

-
- Análisis de requerimientos y definición de componentes bajo una arquitectura SOA.
 - Definición e implementación de un entorno de Integración Continua con herramientas Open Source como Jenkins, Jira, Git, Nexus y SonarQube.
 - Definición de estándares y herramientas para automatización de pruebas y control de calidad de código.
 - Estimación de tiempos y esfuerzo, y asignación de responsabilidades.
 - Levantamiento de la arquitectura AS IS de los procesos para los canales auto asistidos del cliente Claro, evaluación del impacto de la migración de los componentes a una nueva plataforma, definición de la arquitectura TO BE y servicios a desarrollar o modificar.
 - Seguimiento y gestión de la implementación de la arquitectura propuesta y estimación de tiempos y esfuerzo de proyecto.
 - Análisis y diseño de soluciones de integración (EAI) con arquitectura SOA y tecnología Java, Spring Framework 3.0 y Oracle sobre la plataforma WebLogic Server 10g, 11g, 12c/ Linux.
 - Desarrollo de servicios REST, componentes y servicios OSB (Oracle Service Bus) y BPEL.
 - Administración y soporte a servidores WebLogic 10g, 11g, 12c/ Linux.

Analista Programador Java – Infoparque Perú

Abr. 2015 –

Encargada de la documentación, desarrollo y mantenimiento de los sistemas de las empresas con las que trabaja Grupo Infoparque, en este caso Mapfre, utilizando tecnologías Java, Oracle, Hibernate y Primefaces.

Jun. 2015

Las principales funciones son:

- Implementación y mantenimiento de los módulos del sistema TronWeb, que es el sistema transaccional de la compañía de seguros Mapfre.
 - Desarrollo del sistema Web de reportes de siniestros.
-

-
- Apoyo en el análisis y diseño de requerimientos y elaboración de la documentación.

Implantador de Sistemas – RENIEC	Dic. 2013 –
Encargada de evaluar los requerimientos para el desarrollo de aplicaciones móviles y Web, y las soluciones a incidencias o mejoras que se puedan dar en el desarrollo.	Mar. 2015
Análisis e implementación de Rest Services, utilizando tecnologías Java, Spring Framework, Spring WEB Services, Ibatis 2.0, arquitectura MVC, Ajax, Oracle 11g, PL/SQL, Netbeans, Eclipse y Maven. Implementación de packages y procedures necesarios para el desarrollo de los diferentes aplicativos.	
Análisis e implementación de aplicaciones móviles para Android haciendo uso de servicios y acceso a base de datos, conexión con API's REST y JSON, mapas, geolocalización y GPS.	
Apoyo en la elaboración de la documentación de análisis y diseño, y ejecución de pruebas de los sistemas informáticos.	

Nota. La tabla contiene la experiencia laboral de la autora en orden descendente. Fuente: Elaboración propia.

CAPÍTULO II CONTEXTO EN EL QUE SE DESARROLLA LA EXPERIENCIA

2.1 EMPRESA – ACTIVIDAD QUE REALIZA

La empresa es una entidad del sistema financiero peruano que ofrece diversos productos y servicios, consolidándose como una de las entidades financieras líderes de la región y principal activo del holding al que pertenece. Su principal objetivo social es favorecer el desarrollo de los sectores comerciales y productivos del país con actividades bancarias comerciales y de ahorros. Cuenta con divisiones de Banca Empresa y Corporativa desde donde atiende a clientes corporativos y empresas medianas, y también con una Banca Minorista enfocada a las pequeñas empresas y clientes individuales con productos de alto valor agregado.

La historia de la empresa data del siglo XIX en donde se constituyó como sociedad anónima y la escritura pública se custodia en el Archivo General de la Nación. En 1942 la institución cambia de razón social y actualmente sus actividades se basan en la Ley 267025, cuyo objetivo principal es tender un sistema financiero y de seguros competitivo, sólido y confiable que contribuya al desarrollo nacional (Superintendencia de Banca, Seguros y AFP, 2020).

En los últimos años, la empresa ha fortalecido su área de Tecnologías de la Información (TI) invirtiendo en infraestructura, recursos y aliándose de los mejores profesionales con el fin de mejorar los procesos internos, los tiempos de despliegue de las soluciones tecnológicas a los usuarios finales y la estabilidad de sus sistemas.

Si bien la actividad principal de la empresa se enfoca en los productos financieros que ofrece a sus clientes, actualmente cuenta con una Gerencia de Transformación Digital que busca ser referente en soluciones tecnológicas y posicionarse en la región.

2.2 VISIÓN

La visión de la entidad financiera se divide en tres ejes fundamentales que se detallan a continuación:

- Ser una de las empresas peruanas que brinde la mejor experiencia a sus clientes.
- Ser referentes regionales en gestión empresarial, resaltando su liderazgo histórico en la industria financiera.
- Ser una de las comunidades laborales de preferencia en el Perú con los mejores profesionales.

2.3 MISIÓN

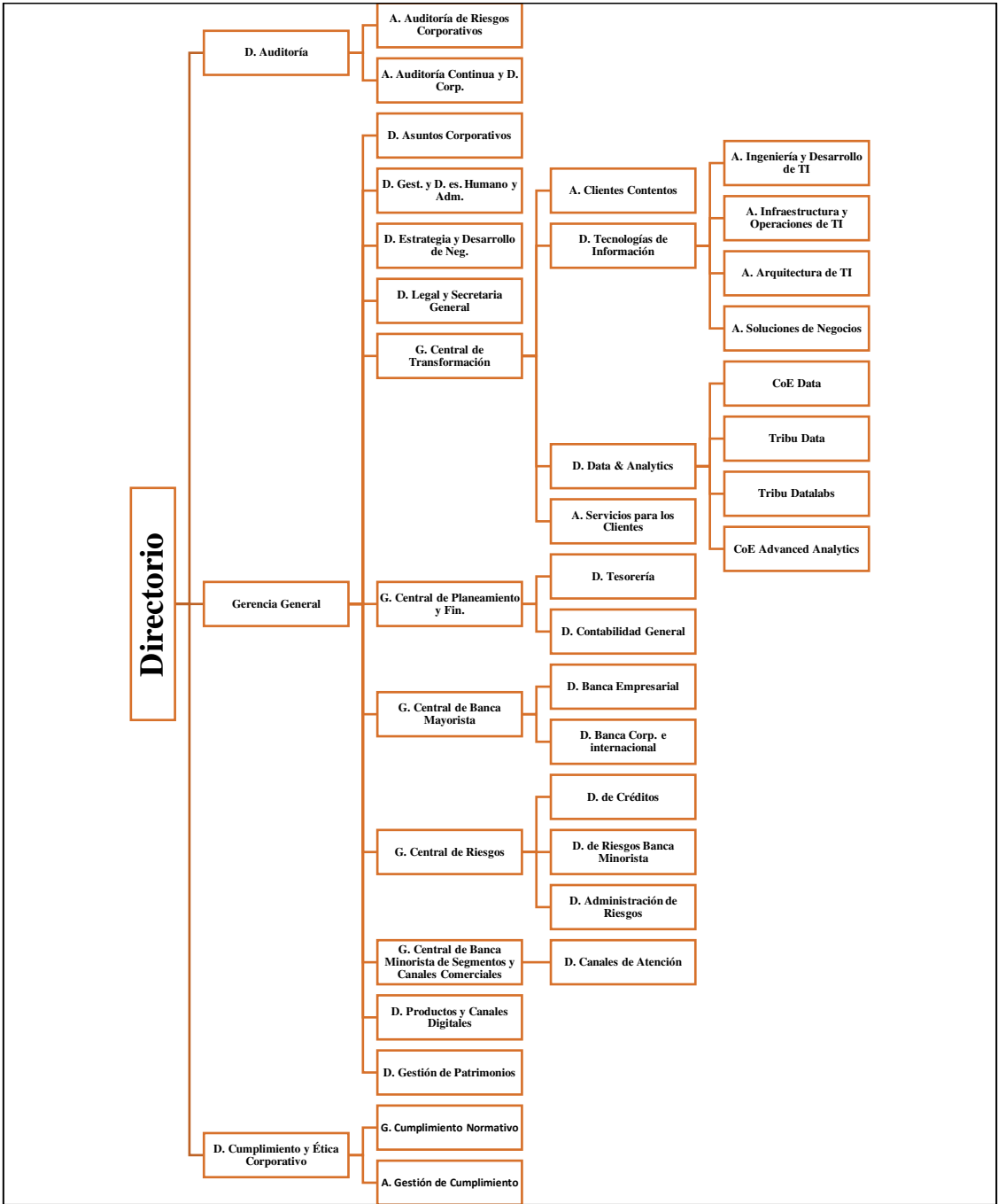
La principal misión de la empresa es la transformación de los proyectos y sueños de sus clientes en realidad y contribuir en la historia de desarrollo y superación del Perú.

2.4 ORGANIZACIÓN DE LA EMPRESA

La siguiente figura representa parte de la organización general de la empresa y sus áreas principales.

Figura 1

Organigrama de la empresa



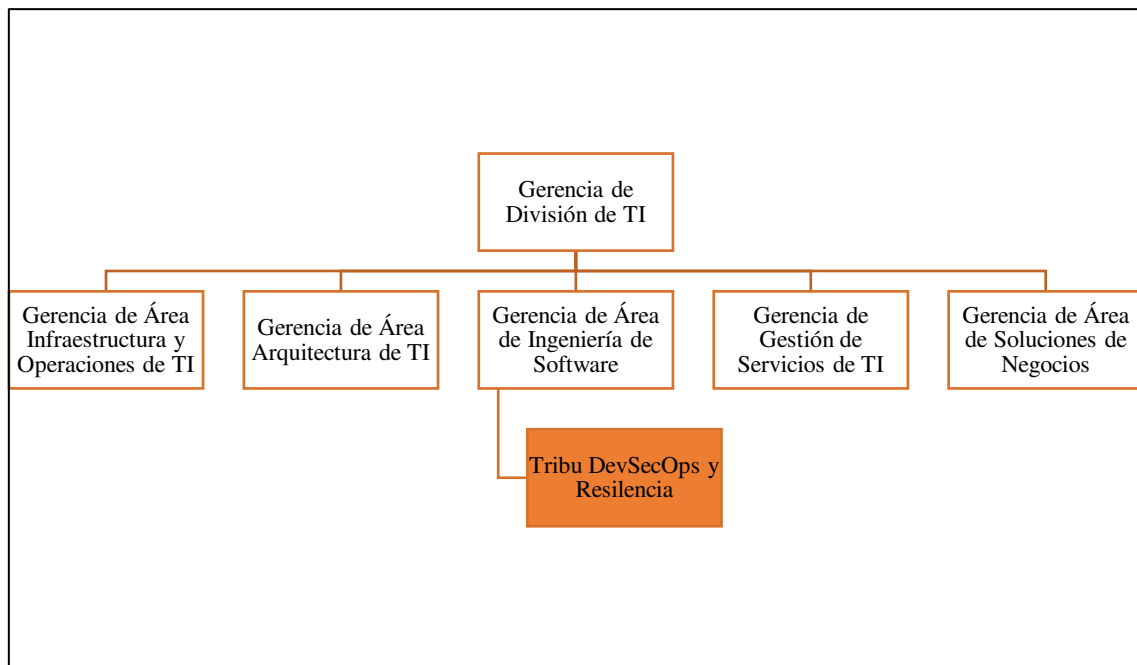
Nota. La figura representa la distribución de la empresa y cómo se encuentra organizada. Fuente: Elaboración propia.

Organización de la División de TI

El proyecto se desarrolló dentro del equipo de DevSecOps que actualmente pertenece a la Tribu DevSecOps y Resiliencia como se muestra en el gráfico a continuación.

Figura 2

Organigrama del Área de División de Tecnologías de la Información



Nota. La figura representa cómo se encuentra distribuida el área de la División de TI de la empresa. Fuente: Elaboración propia.

2.5 ÁREA, CARGO Y FUNCIONES DESEMPEÑADAS

Área: Squad Testing Continuo – Tribu DevSecOps y Resiliencia

Cargo: DevOps Engineer

Funciones desempeñadas:

- Análisis, diseño e implementación de automatizaciones, pipelines de los procesos manuales que se tienen en el área de Tecnologías de la Información.
- Evaluación técnica de nuevas herramientas que contribuyan a los procesos de Testing Continuo.
- Definición de la arquitectura de infraestructura y redes de nuevas herramientas.
- Implementación de nuevas herramientas.
- Definición del modelo operativo y lineamientos para el uso de herramientas de Testing Continuo.
- Soporte a las herramientas de Integración Continua, Entrega Continua y Testing Continuo.
- Automatización de los procedimientos de pases a producción del equipo.

2.6 EXPERIENCIA PROFESIONAL REALIZADA EN LA ORGANIZACIÓN

Entre las funciones que la autora ha realizado en la organización, se encuentran:

- Automatizar el proceso de generación de certificados digitales internos de la empresa para ser ejecutado desde un pipeline.
- Evaluar desde una perspectiva técnica la compra de nuevas herramientas para la virtualización de servicios y gestión de datos de pruebas.
- Implementar e instalar una herramienta para la virtualización de servicios.
- Implementar una herramienta para la gestión de datos de pruebas.
- Realizar seguimiento al estado de los proyectos.
- Desarrollar imágenes Docker para la ejecución de pruebas desde pipelines.
- Desarrollar una librería para la ejecución de pruebas automatizadas desde Jenkins.
- Aprovisionar y configurar recursos Azure.
- Realizar y gestionar los pases a producción.
- Brindar soporte 24x7 a las herramientas y aplicaciones de la tribu.

CAPÍTULO III ACTIVIDADES DESARROLLADAS

3.1 SITUACIÓN PROBLEMÁTICA

La empresa empezó hace algunos años a adoptar practicas DevOps dentro del área de TI para la automatización de procesos rutinarios obteniendo grandes beneficios en la reducción de tiempos al momento de entregar soluciones a producción y facilitando la entrega continua de nuevas soluciones y/o mejoras con mayor frecuencia. Sin embargo, luego de haber alcanzado un grado de madurez DevOps en un gran porcentaje de las aplicaciones, se identificó que la etapa de pruebas seguía manteniendo diversas actividades manuales que finalmente influían en los tiempos de despliegue a producción hasta que se cuente con la conformidad de pruebas exitosas.

Tabla 4

Tiempos estimados de ejecución de pruebas manuales

Aplicativo	Número de casos de pruebas manual	Creación de data de prueba (días)	Tiempo aprox. ejecución (días)
Aplicativo Web	280	3	14
Aplicativo Mainframe	390	-	15

Nota. La tabla muestra un estimado de la cantidad de días que demora realizar pruebas de una muestra para dos aplicativos principales de la entidad financiera en base a lo indicado por el Quality Engineer (QE) asignado a cada equipo. Fuente: Elaboración propia.

3.1.1 Definición del problema

Dado lo indicado en el punto previo, se identificó que la etapa de pruebas presentaba los siguientes problemas que influían en sus extensos tiempos:

- Las pruebas funcionales se realizaban de manera manual lo cual, en ocasiones, concluía en errores humanos o dependía de la agilidad de cada QE para terminar la tarea.
- Las evidencias de los resultados de las pruebas realizadas se tenían que importar de manera manual a la herramienta de gestión de proyectos para poder brindar el conforme de la entrega y continuar con el despliegue en producción.
- Aunque muchas de las pruebas que se realizaban eran repetitivas en los diferentes Sprint o durante las pruebas End to End y de regresión, se tenían que volver a diseñar en cada iteración del ciclo de desarrollo.
- Existía un desorden y desgobierno de los casos de pruebas.
- La generación de datos de pruebas dependía de procesos manuales y laboriosos que extendían los tiempos de las pruebas.

3.2 SOLUCIÓN

Implementación de un marco de trabajo de Testing Continuo para mejorar la calidad y reducir los tiempos durante la etapa de pruebas del ciclo de vida del desarrollo de software (SDLC) de las aplicaciones de la entidad financiera.

3.2.1 Objetivos

Objetivo general

Optimizar la etapa de pruebas por la cual atraviesan las aplicaciones a través de la implementación de prácticas de Testing Continuo dentro de la entidad financiera.

Objetivos específicos

- Implementar prácticas de Testing Continuo tanto en las aplicaciones de la entidad financiera que siguen un marco de trabajo ágil como en las aplicaciones que siguen un marco de trabajo cascada.
- Reducir los tiempos durante la etapa de pruebas de las aplicaciones de la entidad financiera que adopten el marco de trabajo.
- Automatizar las pruebas de las aplicaciones que utilizan alguna de las siguientes tecnologías: API, Web y/o Mainframe.
- Contar con un repositorio de casos de pruebas y lineamientos para la gestión del ciclo de pruebas.
- Incrementar la capacidad de ejecución de pruebas que actualmente tienen los QE mediante el uso de los artefactos del marco de trabajo.
- Reducir la cantidad de incidentes en producción provocados por falsos positivos o errores humanos durante la fase de pruebas.
- Reducir los tiempos y costos por despliegue de los aplicativos provocados por las demoras durante la etapa de pruebas.

3.2.2 Alcance

Alcance funcional

El alcance funcional del proyecto consistió en implementar prácticas de Testing Continuo para automatizar las pruebas de los flujos de las aplicaciones de la entidad financiera que utilizan alguna de las siguientes tecnologías: Web, API y/o Mainframe.

Alcance organizacional

El proyecto se desarrolló dentro de la División de TI de la entidad financiera y estuvo alineado al cumplimiento de los Objectives and Key Results (OKR) organizacionales de reducción de tiempos de despliegue y costos de la unidad.

3.2.3 Etapas y metodologías

Para llevar a cabo el proyecto de implementación de herramientas y lineamientos DevOps para la automatización de la etapa de pruebas, se siguió la metodología ágil SCRUM. Adicionalmente, se definió un marco de trabajo para aplicar Testing Continuo en la entidad financiera que fue adoptado por los diferentes aplicativos.

3.2.3.1 Marco de trabajo SCRUM.

El proyecto fue implementado siguiendo la metodología SCRUM y tuvo una duración de 24 Sprint de 2 semanas aproximadamente cada uno, que, a su vez fueron divididos en 4 trimestres.

Los roles que participaron en el proyecto fueron:

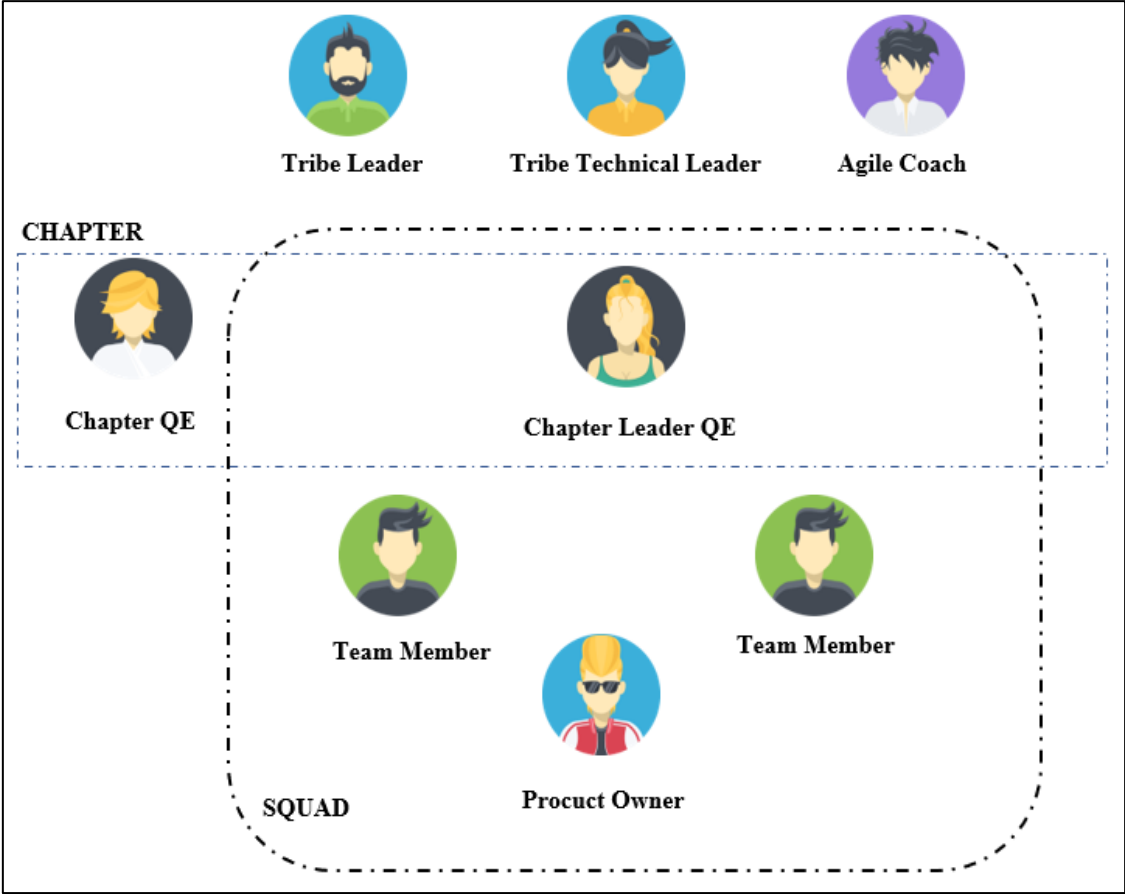
- **Product Owner:** Actuó como dueño del producto y fue responsable de definir los requerimientos y criterios de aceptación de cada historia de usuario para poder alcanzar el release requerido en cada entrega. También fue responsable de priorizar el Product Backlog y resolver dudas del equipo sobre el alcance del proyecto.
- **Scrum Master / Agile Coach:** Fue responsable de que el equipo siguiera la metodología y se comportó como un facilitador ante los bloqueos e inconvenientes que presentaba el equipo. También actuó como entrenador y elaboraba dinámicas para mejorar los indicadores ágiles del equipo.
- **Tech Lead:** Miembro del equipo con el rol de líder técnico, responsable de brindar soporte a nivel técnico a los otros miembros del equipo y también se encargó de las revisiones y conformidades de revisión de pares.
- **Team member:** Los perfiles de los miembros del equipo que participaron del proyecto fueron Ingeniero DevOps e Ingeniero de Calidad.

También se tuvieron los siguientes roles que participaron como especialistas y contribuyeron en las definiciones y aprobaciones:

- **Arquitecto de TI:** Participó en la definición y aprobación de la arquitectura de las herramientas DevOps, infraestructura y arquitectura de redes para la comunicación con los aplicativos de la entidad financiera.
- **Chapter Lead Quality Engineer (QE):** Participó como especialista de la división de pruebas, brindando detalle sobre los procesos de pruebas manuales que se llevaban hasta ese momento e intervino activamente en la definición de los nuevos lineamientos para pruebas automatizadas.
- **Analista de Seguridad:** Como especialista de Seguridad, participó en la revisión de la arquitectura de componentes, redes e infraestructura. También en la definición de los perfiles y permisos para las herramientas DevOps y en las revisiones del código de software desarrollado por el equipo, con el fin de asegurar que se sigan los lineamientos de Seguridad definidos dentro de la entidad financiera.

Figura 3

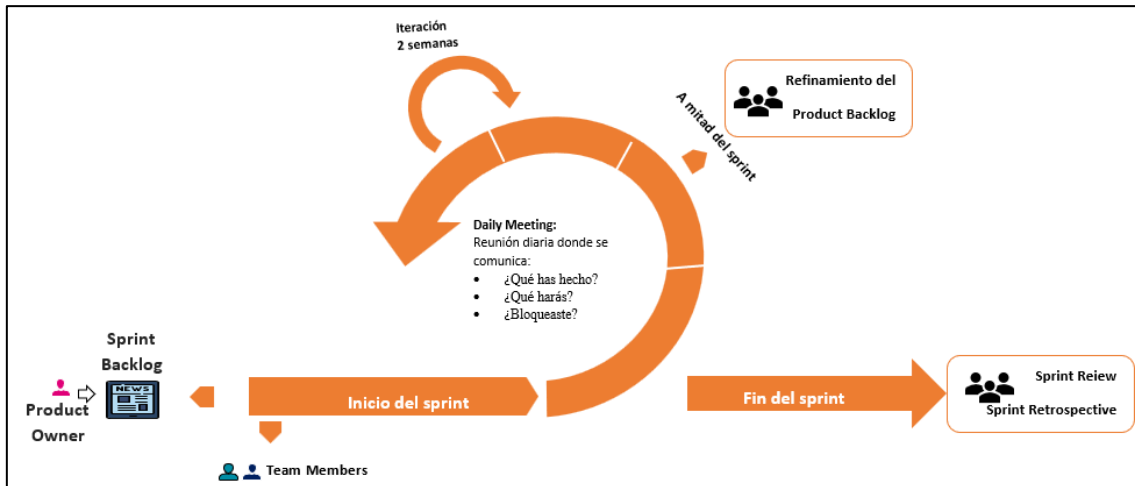
Roles en Agilidad dentro de la entidad financiera



Nota. La figura muestra la composición de los equipos de la División de TI según los roles definidos dentro del marco de agilidad dentro de la entidad financiera. Fuente: Elaboración propia.

Figura 4

Flujo SCRUM dentro de la entidad financiera



Nota. La figura muestra cómo se desarrolla SCRUM en el día a día de los equipos ágiles de la entidad financiera, incluyendo los artefactos, roles y ceremonias que se realizan.

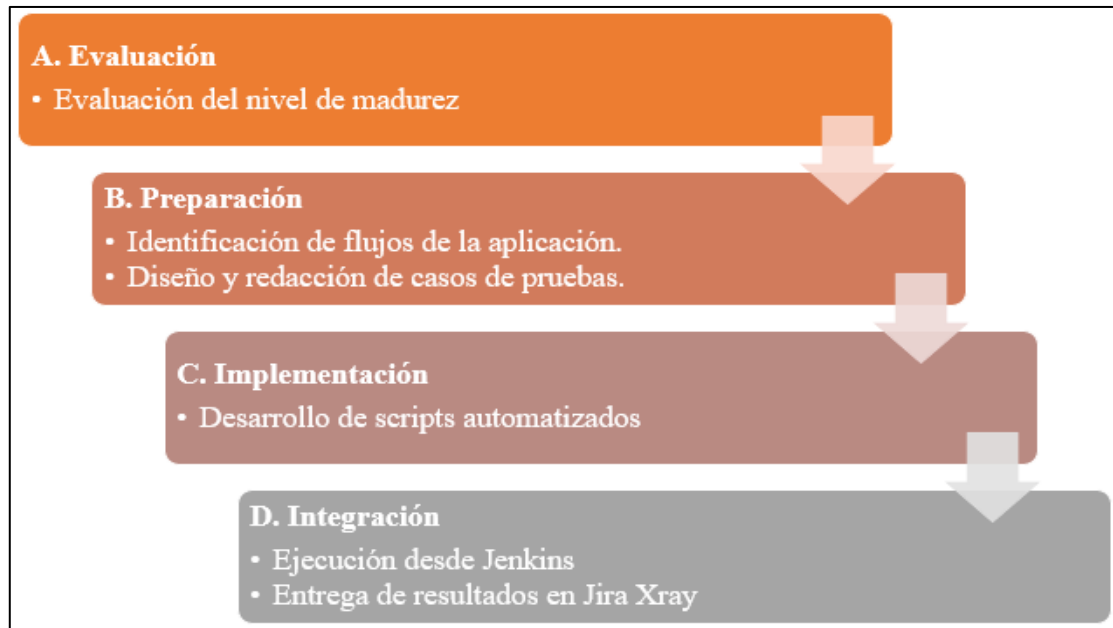
Fuente: Elaboración propia.

3.2.3.4 Marco de trabajo de Testing Continuo

Se definió un marco de trabajo de Testing Continuo basado en la realidad de los equipos de la entidad financiera y siguiendo las buenas prácticas de la metodología. Este marco abarcaba 4 fases: primero una evaluación de las aplicaciones, seguido por un análisis y diseño de los flujos de cada aplicativo. Luego se realizó la implementación de la automatización de las pruebas, y finalmente la ejecución de pruebas automatizadas y la integración con herramientas DevOps.

Figura 5

Fases para la adopción de Testing Continuo



Nota. La figura representa las fases definidas dentro del marco de trabajo de Testing Continuo para que las aplicaciones de la entidad financiera adopten las nuevas prácticas en el flujo SDLC. Fuente: Elaboración propia.

A. Evaluación

Durante la etapa de evaluación, las aplicaciones fueron medidas a través de encuestas que estuvieron basadas en cuatro pilares de Testing Continuo para poder identificar el nivel de madurez que tenía cada una. El resultado de la evaluación permitió conocer si las aplicaciones aplicaban o no para el uso de Testing Continuo. Mientras más maduro era un sistema, costó menos la adopción e implementación de Testing Continuo en el equipo. Y también se identificaron equipos que no pudieron adoptar el marco de trabajo por la naturaleza de sus pruebas que requerían intervención humana durante la ejecución de estas.

Figura 6

Los 4 pilares del marco de trabajo de Testing Continuo



Nota. La figura representa los 4 pilares en los cuales se basa el marco de trabajo de Testing Continuo diseñado para la entidad financiera. Fuente: Elaboración propia.

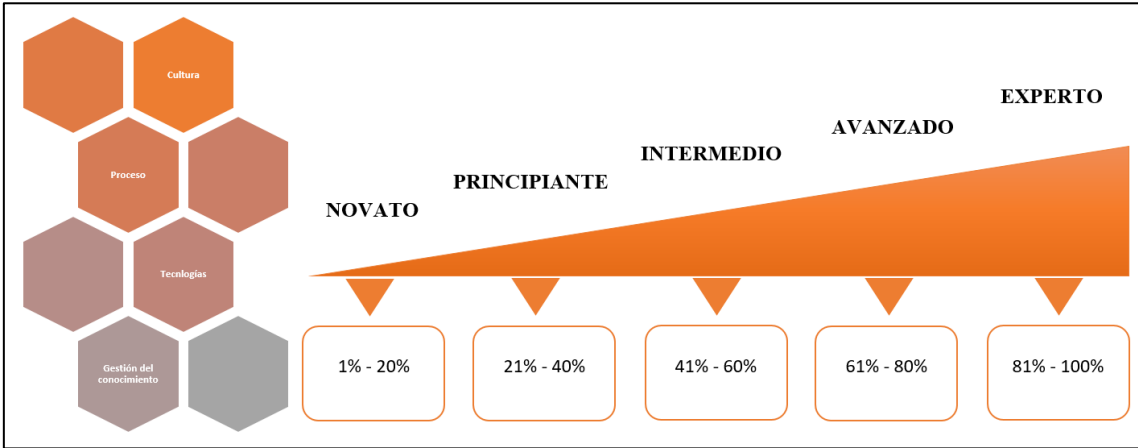
El marco de trabajo de Testing Continuo se basa en 4 pilares:

- **Cultura:** El marco de trabajo de Testing Continuo es parte de la cultura DevOps, por lo cual es importante el cambio de mentalidad sobre conceptos ágiles y de DevSecOps y guiar a los equipos en este camino.
- **Procesos:** Testing Continuo permite mejorar y optimizar los procesos del flujo SDLC, reduciendo tiempos en la etapa de pruebas.
- **Tecnologías:** Para la implementación del marco de Testing Continuo se utiliza tecnologías y herramientas que sumen a la automatización de las actividades.
- **Gestión del conocimiento:** Gestión del conocimiento y técnicas para el análisis y diseño de pruebas.

La evaluación de acuerdo con los pilares permitió categorizar a los aplicativos en 5 niveles de madurez, lo cual ayudó a segmentar a los equipos y planificar su adopción:

Figura 7

Niveles de madurez de los aplicativos



Nota. La figura representa los niveles de madurez para clasificar a los aplicativos de la entidad financiera. Fuente: Elaboración propia.

B. Preparación

Durante esta etapa los equipos identificaron y priorizaron los flujos de la aplicación de acuerdo con la criticidad para el negocio. Para poder priorizar los flujos, nos basamos en la matriz de Risk Based Test (RBT) que permitía identificar los escenarios críticos según la evaluación de riesgos y de esta manera definir cuál era el orden de los flujos a automatizar y también definir una mejor estrategia de pruebas de acuerdo con los escenarios más críticos para el negocio. La matriz permitió evaluar cada flujo en base al impacto que tenía en el negocio, la probabilidad de fallo, el nivel de control que se tenía sobre el flujo y el nivel de confianza, que en conjunto otorgaban una clasificación de riesgo extremo, controlado o bajo.

Figura 8

Matriz de evaluación Risk Based Test

RISK BASED TEST – PRUEBAS BASADA EN RIESGO

ID	Lista de escenarios de pruebas	Condición de la prueba / Caso de prueba / Escenario a evaluar	Evaluación de riesgos					Observación
			Impacto	Probabilidad	Nivel de Control	Nivel de Confianza	Factor de Riesgo	
1	Flujo 1	Error operativo	5	2	1	4	12.0	Riesgo extremo
2	Flujo 2	Error funcional	4	3	2	3	8.4	Riesgo controlado

Nota. La figura muestra un ejemplo de uso de la matriz RBT para dos flujos de un aplicativo. Fuente: Elaboración propia.

Luego de que los equipos priorizaron el orden de las pruebas que iban a realizar de manera automatizada, se procedía a diseñar los casos de prueba, y luego cada caso era redactado siguiendo la sintaxis del lenguaje Gherkin. Finalmente, los casos de prueba eran importados en el tablero del equipo dentro de la herramienta Jira Xray de forma individual o masiva.

Figura 9

Redacción en Gherkin

```

Feature: búsqueda en google
  Como usuario quiero buscar opciones en google para resolver mis dudas.

  Scenario: búsqueda de restaurantes
    Given abrir la página de google en un navegador web
    When se introduce la palabra de búsqueda "restaurantes"
    Then se muestra el resultado de los restaurantes
    And los resultados solo muestran los restaurantes de "Lima Metropolitana"
    But los resultados no incluyen a los distritos de "Ate", "Villa el Salvador" y "Puente Piedra"
  
```

Nota. La figura muestra un ejemplo de cómo redactar un caso de prueba para realizar búsquedas en la Web de Google, siguiendo la sintaxis del lenguaje Gherkin. Fuente: Elaboración propia.

Figura 10

Creación de un caso de prueba en Jira Xray

The screenshot shows the 'Create Issue' form in Jira Xray. The 'Project' field is set to 'MI PROYECTO'. The 'Issue Type' is 'Test'. The 'Summary' field is empty. The 'Description' field has a rich text editor with a toolbar. Below the description, the 'Tipo de Test' is 'Gherkin', and the 'Cucumber Type' is 'Scenario'. The 'Escenario en Gherkin' field is empty. At the bottom, there are buttons for 'Create another', 'Create', and 'Cancel'.

Nota. La figura muestra un ejemplo de cómo crear un caso de prueba en Jira Xray. Fuente: Elaboración propia.

C. Implementación

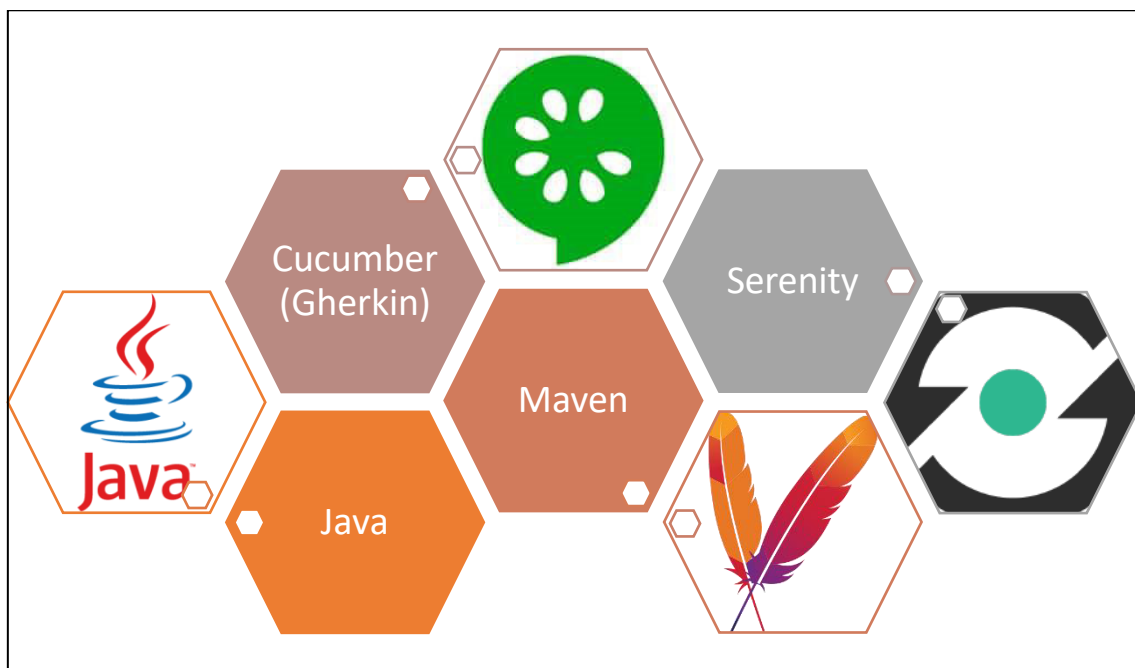
Durante la etapa de implementación, cada QE de los equipos fue responsable de desarrollar los scripts de automatización de pruebas en base a los casos de pruebas redactados en Gherkin. Adicionalmente, como parte del marco de trabajo de Testing Continuo se desplegó un framework de automatización de pruebas que resumía métodos en común para realizar pruebas de aplicaciones Web, API y Mainframe.

El framework de automatización de pruebas se basó en las siguientes tecnologías:

- **Java:** el código se encuentra desarrollado en Java 8 y se expone como dependencias para que los equipos lo puedan reutilizar en sus proyectos de automatización.
- **Maven:** es un utilitario para proyectos Java que permite compilar, empaquetar y ejecutar los proyectos.
- **Cucumber:** es una herramienta que permite implementar pruebas basándose en la metodología Behaviour Driven Development (BDD) describiendo los casos de pruebas en Gherkin, un lenguaje lógico y fácil de entender por todos los stakeholders del proyecto.
- **Serenity:** es una librería de código abierta que permite escribir pruebas automatizadas.

Figura 11

Framework de automatización de pruebas funcionales



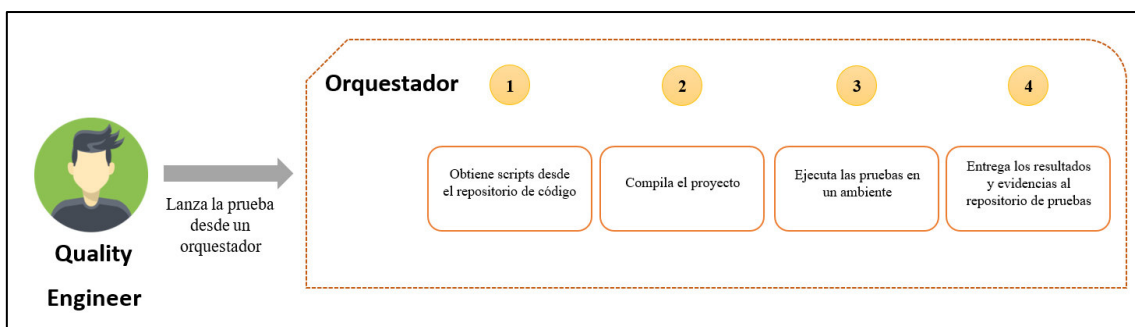
Nota. La figura muestra la composición de las tecnologías del framework de automatización de pruebas que fue desplegado a los equipos de la División de TI. Fuente: Elaboración propia.

D. Integración

Finalmente, en la etapa de integración, cada QE pudo ejecutar las pruebas de forma automatizada y a demanda desde el orquestador Jenkins y al finalizar pudieron visualizar los resultados en un reporte en la misma herramienta de Jenkins y las evidencias fueron cargadas automáticamente en la herramienta de gestión de casos de pruebas Jira Xray.

Figura 12

Integración de las pruebas con herramientas DevOps



Nota. La figura muestra la propuesta de integración de las pruebas automatizadas con herramientas DevOps. El detalle de las herramientas se describe en el punto 3.2.5. Fuente: Elaboración propia.

3.2.4 Fundamentos utilizados

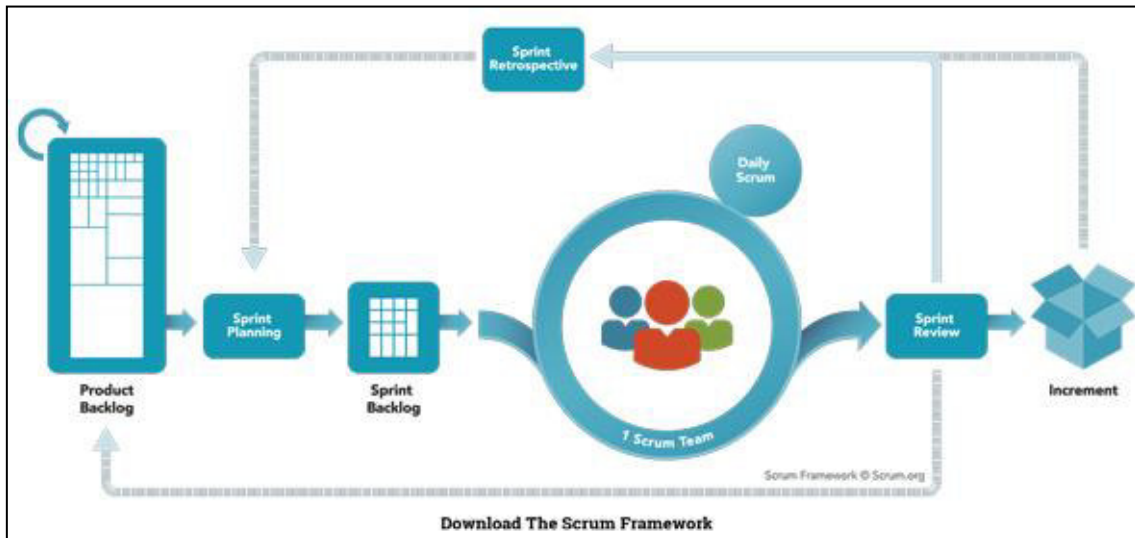
3.2.4.1 Desarrollo Ágil - SCRUM

Es una metodología de trabajo que prioriza la entrega de valor temprano y constante y anticipa la necesidad de flexibilidad durante la etapa de desarrollo. Según los creadores de SCRUM, podemos definir Scrum como:

“El marco de trabajo Scrum consiste en los Equipos Scrum y sus roles, eventos, artefactos y reglas asociadas. Cada componente dentro del marco de trabajo sirve a un propósito específico y es esencial para el éxito de Scrum y para su uso.” (Schwaber & Sutherland, 2016)

Figura 13

Marco de trabajo SCRUM



Nota. La figura muestra el flujo del marco de trabajo de SCRUM, sus artefactos y ceremonias. Fuente: (SCRUM ORG, 2020)

La metodología SCRUM clasifica los roles en dos categorías:

Roles centrales

Son los roles principales e indispensables del proyecto y de quienes depende el éxito de este.

- Product Owner:
- Scrum Master
- Scrum Team

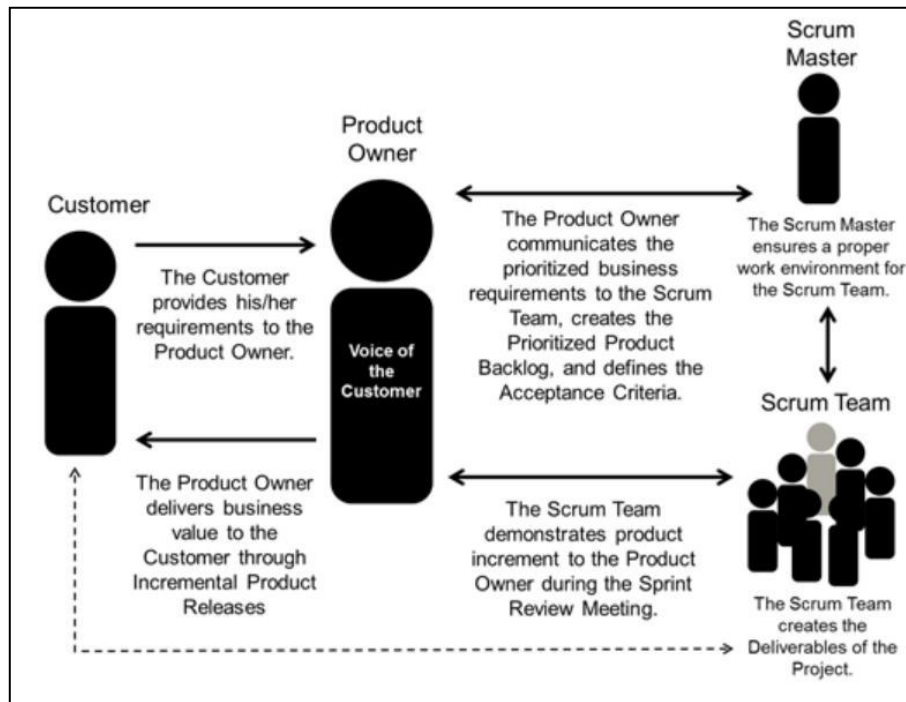
Roles no centrales

Son roles importantes ya que participan en decisiones claves del proyecto, pero el éxito del proyecto no depende de su participación.

- Stakeholders (clientes, usuarios)

Figura 14

Roles en SCRUM



Nota. La figura muestra los roles definidos en SCRUM y cómo interactúan entre ellos.

Fuente: (Scrum Study, s.f.)

La metodología brinda las siguientes ceremonias que deben tener los equipos:

- Sprint Planning
- Daily Meeting
- Sprint Review
- Sprint Retrospective

La metodología brinda los siguientes artefactos de apoyo a los equipos:

- Product Backlog
- Sprint Backlog

3.2.4.2 Testing Continuo

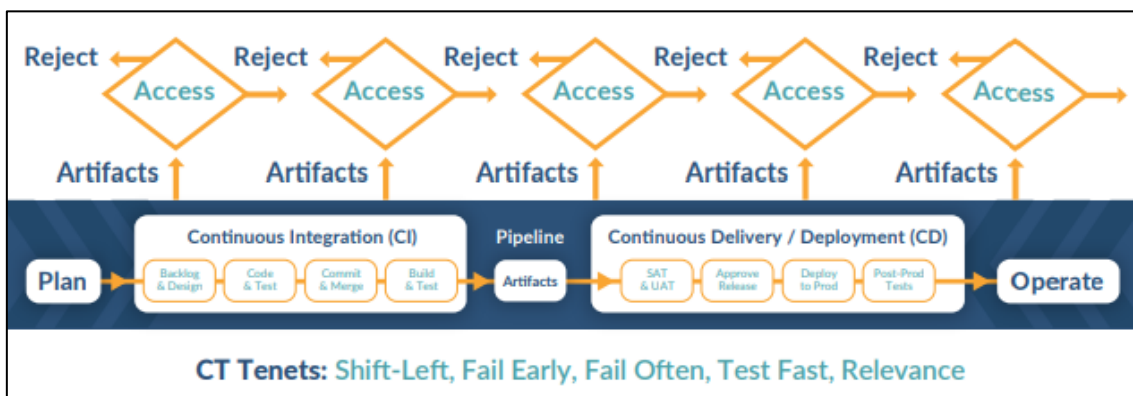
Es el proceso de poder ejecutar pruebas automatizadas como parte del ciclo de desarrollo de software y así obtener feedback temprano y reducir los riesgos, además, las pruebas son automatizadas e integradas con la infraestructura DevOps.

“Son las pruebas que permiten lanzamientos más rápidos, de mayor calidad y reduce costos al integrarse perfectamente con el proceso de entrega continua para identificar y abordar los riesgos en todas las etapas de desarrollo minimizando riesgos comerciales y el impacto a los clientes”. (Platz, n.d.)

“Las evaluaciones de Testing Continuo se ejecutan de manera continua en cada cambio incremental del producto utilizando un ambiente equivalente al productivo para cada etapa del pipeline DevOps desde el inicio, pasando por Integración Continua hasta Entrega y Despliegue Continuo.” (DevOps Institute)

Figura 15

Estrategia de pruebas Testing Continuo



Nota. La figura muestra la estrategia de pruebas de Testing Continuo dentro del flujo DevOps, en donde los artefactos de cada etapa son evaluados para continuar o no. Fuente: (DevOps Institute)

Conceptos de Testing Continuo:

Como parte de la definición de Testing Continuo, se introducen los conceptos que se detallan a continuación.

Figura 16

Conceptos Testing Continuo



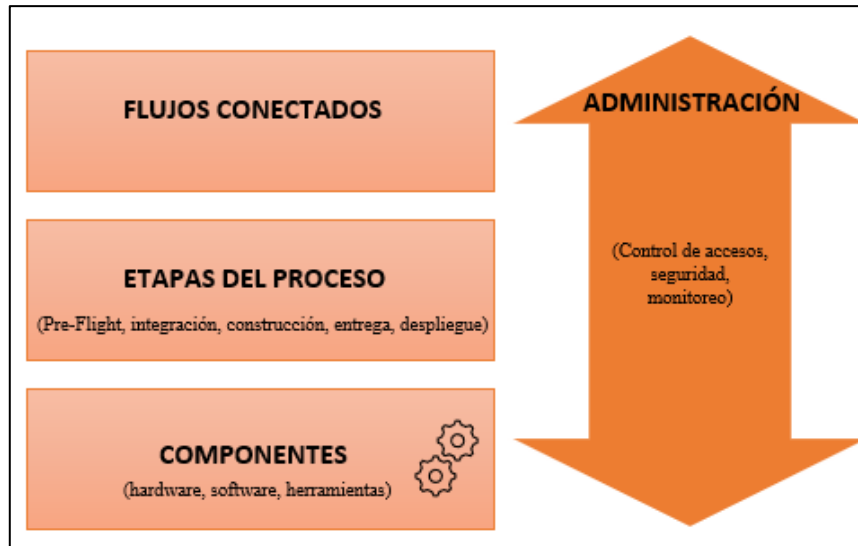
Nota. La figura detalla los conceptos que se abarcan dentro de Testing Continuo según el Instituto DevOps. Fuente: Elaboración propia.

– Infraestructura DevOps

La infraestructura DevOps hace referencia a que se deben contar con diferentes componentes del tipo hardware, software, herramientas, cloud, data, y otros, que soporten todas las etapas del proceso SDLC, desde el análisis, integración, construcción, entrega, hasta el despliegue; y a su vez estos procesos se encuentren conectados, y gestionados bajo controles de seguridad y sean monitoreados de manera constante.

Figura 17

Arquitectura DevOps



Nota. La figura detalla en qué se basa la infraestructura DevOps según el Instituto DevOps. Fuente: Elaboración propia.

- **Procesos: Pre-Flight, Continuous Integration (CI), Continuous Delivery (CD)**

El Pre-Flight Testing se refiere a la evaluación del diseño y código de software en etapas previas del diseño y construcción de un aplicativo y antes de realizar la integración con un pipeline DevOps de CI y CD.

- **Automatización de pruebas**

Las pruebas en Testing Continuo son automatizadas de tal manera que se puedan ejecutar rápidamente.

- **Múltiples etapas**

Testing Continuo no es una fase del ciclo de desarrollo de software, por el contrario, se implementa en todas las etapas del pipeline de DevOps

- **Ambiente de pruebas equivalente a producción**

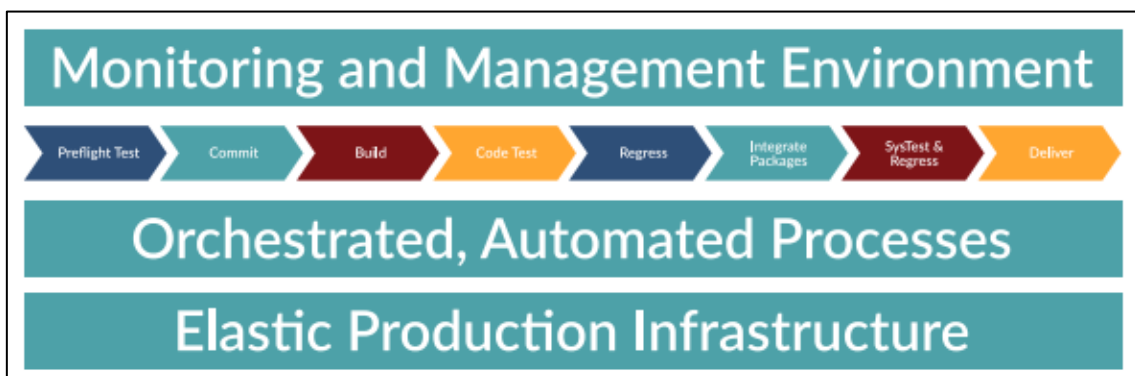
Se refiere a tener un ambiente valido que sea equivalente al ambiente productivo y sea resultado de la combinación de los siguientes factores: casos de uso (tipos de usuarios y flujos), entorno (condiciones externas), sistemas (versiones de sistemas operativos y otros servicios y/o programas) y configuraciones (data, topología), y en el cual se ejecuten las pruebas.

- **Alcance DevOps end to end**

“En DevOps, las pruebas se integran de manera horizontal en cada etapa del pipeline end-to-end, y de manera vertical en todas las capas de la herramienta de entrega continua e infraestructura.” (Hornbeek, 2018)

Figura 18

End-to-End DevOps Testing



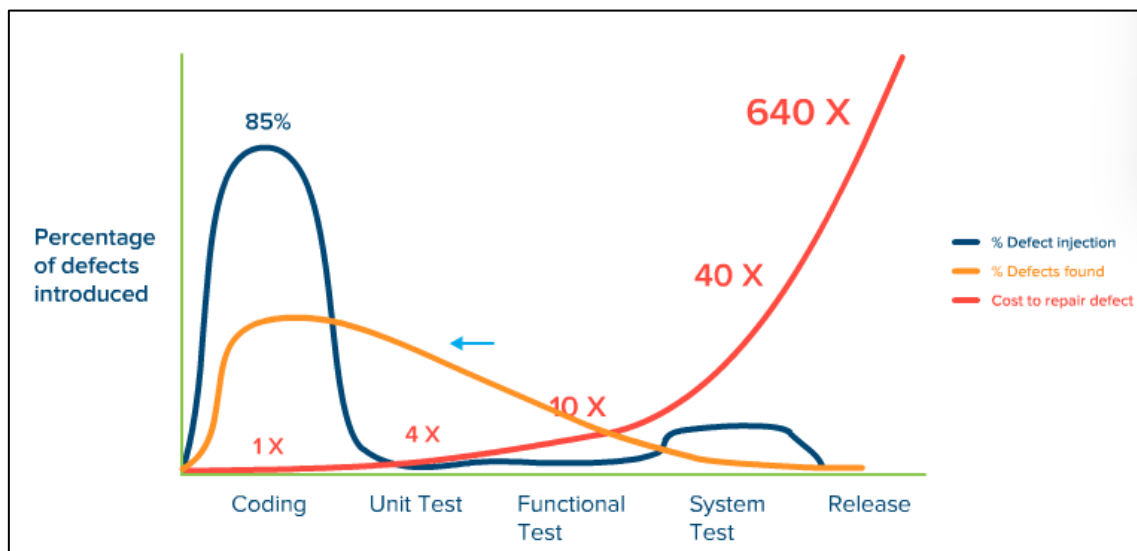
Nota. La figura muestra un ejemplo de las etapas de un pipeline end-to-end, en donde Testing participa en cada una de ellas. Fuente: (Hornbeek, 2018).

Testing Continuo se basa en cinco principios que resaltan las mejores prácticas para realizar DevOps Testing.

- **Shift Left:** es un principio que indica que las pruebas se deben realizar desplazándose hacia la izquierda en etapas tempranas de desarrollo.
- **Fail Early:** es un principio que indica que mientras más temprano ocurra un error, más temprano se podrá aprender del mismo y por ende desarrollar una solución tan pronto sea posible.
- **Fail Often:** acompaña al principio anterior y añade la habilidad de reponerse cuantas veces ocurra un error.
- **Test Fast:** es un principio que indica que se deben programar las pruebas para que se puedan ejecutar en ciclos rápidos.
- **Relevance:** el ultimo principio resalta en enfoque en las pruebas y resultados más importantes.

Figura 19

Shift Left Testing



Nota. La figura muestra el principio de Shift Left y cómo mientras las pruebas se desplacen más hacia la izquierda, más temprano se encontrarán defectos y el costo por resolverlos será más económico que en etapas avanzadas a la derecha. Fuente: (Jones, 2008)

3.2.5 Implementación de las áreas, procesos, sistemas y buenas prácticas

Para llevar a cabo la implementación del proyecto se siguieron las siguientes tareas:

- Definición de la arquitectura para la ejecución de pruebas dentro de la herramienta orquestador de automatizaciones.
- Configuración de los nuevos ejecutores definidos en la arquitectura para las pruebas automatizadas.
- Diseño y desarrollo de imágenes Docker para la ejecución de pruebas.
- Diseño y desarrollo de scripts para la ejecución de pruebas automatizadas desde el orquestador.
- Definición de lineamientos para el uso de un repositorio de casos de pruebas automatizados.
- Definición de lineamientos para el uso de un repositorio de scripts automatizados.
- Definición de un perfilado para que cada QE pueda acceder a las diferentes herramientas con los permisos correspondientes.
- Elaboración de la documentación del Modelo Operativo dentro del repositorio de documentos para su uso público.
- Seguimiento y monitoreo a las aplicaciones que participaron de la adopción del nuevo marco de trabajo de Testing.

La siguiente lista describe las historias de usuario priorizadas en el Product Backlog para el proyecto y agrupadas por épicas que luego fueron divididas en historias y tareas más específicas dado la magnitud de cada una.

Tabla 5

Product Backlog del proyecto priorizado por el Product Owner

Épica	ID	Historia
Implementación de	SQTC-1	Definir la arquitectura
	SQTC-2	Solicitar el aprovisionamiento de la arquitectura

infraestructura DevOps	SQTC-3	Habilitar comunicación con herramientas DevOps
	SQTC-4	Configurar el nodo Linux de Jenkins para Testing
	SQTC-5	Configurar el nodo Windows de Jenkins para Testing
	SQTC-6	Habilitar la comunicación con ambientes para pruebas
Integración con herramientas DevOps	SQTC-7	Definir el perfilado para el uso de herramientas DevOps
	SQTC-8	Analizar y diseñar una librería para ejecutar pruebas automatizadas
	SQTC-9	Implementar imágenes Docker para ejecutar pruebas automatizadas.
	SQTC-10	Implementar librería para ejecutar pruebas automatizadas
	SQTC-11	Desplegar imágenes Docker
	SQTC-12	Desplegar librería para ejecutar pruebas automatizadas
Adopción de Testing Continuo	SQTC-13	Elaborar el modelo operativo
	SQTC-14	Desplegar el modelo operativo
	SQTC-15	Realizar pilotos de adopción con los equipos.

Nota. La tabla lista las historias de usuario definidas para el proyecto, priorizadas por el Product Owner del proyecto y agrupadas por épicas. Fuente: Elaboración propia.

Cada una de las historias de usuario (HU) fueron desglosadas y detalladas en el tablero del equipo dentro de Jira y trabajadas según la priorización definida en cada sprint del proyecto.

Tabla 6

HU: SQTC-1 - Definir la arquitectura

ID	SQTC-1
Resumen	Definir la arquitectura
Descripción	Definir la arquitectura para la integración de las pruebas con las herramientas DevOps y la ejecución de las pruebas de pruebas automatizadas.
Criterios de aceptación	Diagrama de la arquitectura de la infraestructura y de redes aprobado por Seguridad y Arquitectura de la entidad financiera.
Épica	Implementación de infraestructura DevOps
Release	Infraestructura DevOps
Estimación	5

Nota. Detalle de la HU para definir la arquitectura. Fuente: Elaboración propia.

Tabla 7

HU: SQTC-2 - Solicitar aprovisionamiento de la arquitectura

ID	SQTC-2
Resumen	Solicitar el aprovisionamiento de la arquitectura
Descripción	Solicitar los nuevos recursos identificados en la arquitectura.
Criterios de aceptación	Nuevos servidores aprovisionados.
Épica	Implementación de infraestructura DevOps
Release	Infraestructura DevOps
Estimación	2

Nota. Detalle de la HU para solicitar el aprovisionamiento de la nueva arquitectura.
Fuente: Elaboración propia.

Tabla 8

HU: SQTC-3 - Habilitar comunicación con herramientas DevOps

ID	SQTC-3
Resumen	Habilitar comunicación con herramientas DevOps
Descripción	Habilitar la comunicación entre los nuevos servidores y las otras herramientas DevOps para la integración.
Criterios de aceptación	Nuevos servidores con comunicación con las herramientas DevOps
Épica	Implementación de infraestructura DevOps
Release	Infraestructura DevOps
Estimación	3

Nota. Detalle de la HU para habilitar la comunicación con las herramientas DevOps.

Fuente: Elaboración propia.

Tabla 9

HU: SQTC-4 - Configurar nodo Linux

ID	SQTC-4
Resumen	Configurar el nodo Linux de Jenkins para Testing
Descripción	Instalar y configurar los utilitarios y servicios en el servidor Linux, y configurar el servidor como nodo de Jenkins para la ejecución de pruebas automatizadas.
Criterios de aceptación	Servidor configurado en Jenkins.
Épica	Implementación de infraestructura DevOps
Release	Jenkins Testing
Estimación	5

Nota. Detalle de la HU para configurar el nuevo nodo Linux de Jenkins para Testing.

Fuente: Elaboración propia.

Tabla 10

HU: SQTC-5 - Configurar nodo Windows

ID	SQTC-5
Resumen	Configurar el nodo Windows de Jenkins para Testing
Descripción	Instalar y configurar los utilitarios en el servidor Windows y configurar el servidor como esclavo de Jenkins.
Criterios de aceptación	Servidor configurado en Jenkins.
Épica	Implementación de infraestructura DevOps
Release	Jenkins Testing
Estimación	5

Nota. Detalle de la HU para configurar el nuevo nodo Windows de Jenkins para Testing.
Fuente: Elaboración propia.

Tabla 11

HU: SQTC-6 - Habilitar comunicación con ambiente pruebas

ID	SQTC-6
Resumen	Habilitar la comunicación con ambientes para pruebas
Descripción	Habilitar la comunicación entre los nuevos servidores con las aplicaciones que van a ser probadas de manera automatizada.
Criterios de aceptación	Servidores con comunicación con los ambientes de pruebas de los equipos (desarrollo y certificación).
Épica	Implementación de infraestructura DevOps
Release	Jenkins Testing
Estimación	3

Nota. Detalle de la HU para habilitar la comunicación con los aplicativos que participan de las pruebas. Fuente: Elaboración propia.

Tabla 12

HU: SQTC-7 - Definir perfilado

ID	SQTC-7
Resumen	Definir el perfilado para el uso de herramientas DevOps
Descripción	Definir los roles y permisos para que los QE puedan utilizar las herramientas DevOps.
Criterios de aceptación	Perfilado para que los QE utilicen las herramientas DevOps, aprobado por Seguridad.
Épica	Integración con herramientas DevOps
Release	Perfilado Herramientas Testing
Estimación	2

Nota. Detalle de la HU para definir el perfilado de las herramientas DevOps en Testing.
Fuente: Elaboración propia.

Tabla 13

HU: SQTC-8 - Analizar y diseñar librería ejecución pruebas

ID	SQTC-8
Resumen	Analizar y diseñar una librería para ejecutar pruebas automatizadas
Descripción	Realizar el análisis y diseño de una librería para ejecutar pruebas automatizadas.
Criterios de aceptación	Diseño de una librería para ejecutar pruebas automatizadas desde Jenkins.
Épica	Integración con herramientas DevOps
Release	Pruebas Automatizadas
Estimación	5

Nota. Detalle de la HU para analizar y diseñar una librería para la ejecución de pruebas automatizadas. Fuente: Elaboración propia.

Tabla 14

HU: SQTC-9 - Implementar imágenes Docker

ID	SQTC-9
Resumen	Implementar imágenes Docker para ejecutar pruebas automatizadas.
Descripción	Elaborar el código de las imágenes Docker definidas en el análisis y diseño.
Criterios de aceptación	Desarrollo de imágenes Docker para compilar aplicaciones Java y ejecutar pruebas automatizadas de APIs y aplicaciones Web en el navegador Chrome.
Épica	Integración con herramientas DevOps
Release	Pruebas Automatizadas
Estimación	5

Nota. Detalle de la HU para implementar imágenes Docker. Fuente: Elaboración propia.

Tabla 15

HU: SQTC-10 - Implementar librería para ejecutar pruebas

ID	SQTC-10
Resumen	Implementar librería para ejecutar pruebas automatizadas
Descripción	Elaborar el código fuente de la librería para ejecutar pruebas automatizadas desde Jenkins.
Criterios de aceptación	Desarrollo de una librería pública en Groovy para ejecutar pruebas automatizadas desde
Épica	Integración con herramientas DevOps
Release	Pruebas Automatizadas
Estimación	5

Nota. Detalle de la HU para implementar una librería para la ejecución de pruebas automatizadas. Fuente: Elaboración propia.

Tabla 16

HU: SQTC-11 - Desplegar imágenes Docker

ID	SQTC-11
Resumen	Desplegar imágenes Docker
Descripción	Realizar el despliegue en producción de las imágenes Docker.
Criterios de aceptación	Imágenes Docker desplegadas en la rama de trabajo protegida “master” y cargadas en el repositorio de imágenes Azure Container Registry (ACR).
Épica	Integración con herramientas DevOps
Release	Pruebas Automatizadas
Estimación	3

Nota. Detalle de la HU para desplegar en producción las imágenes Docker. Fuente: Elaboración propia.

Tabla 17

HU: SQTC-12 - Desplegar librería para ejecutar pruebas

ID	SQTC-12
Resumen	Desplegar librería para ejecutar pruebas automatizadas
Descripción	Realizar el despliegue en producción de la librería para ejecutar pruebas automatizadas.
Criterios de aceptación	Librería de Testing desplegada en la rama de trabajo protegida “master”.
Épica	Integración con herramientas DevOps
Release	Pruebas Automatizadas
Estimación	3

Nota. Detalle de la HU para desplegar en producción la librería para la ejecución de pruebas. Fuente: Elaboración propia.

Tabla 18

HU: SQTC-13 - Elaborar Modelo Operativo

ID	SQTC-13
Resumen	Elaborar el modelo operativo
Descripción	Documentar los lineamientos, buenas prácticas, requisitos, roles y procedimientos definidos para adoptar el marco de trabajo de Testing Continuo.
Criterios de aceptación	Lineamientos y buenas prácticas documentadas en un espacio público de Confluence.
Épica	Adopción de Testing Continuo
Release	Modelo Operativo
Estimación	5

Nota. Detalle de la HU para elaborar el modelo operativo. Fuente: Elaboración propia.

Tabla 19

HU: SQTC-14 - Desplegar Modelo Operativo

ID	SQTC-14
Resumen	Desplegar el modelo operativo
Descripción	Desplegar el modelo operativo a través de los diferentes canales internos de la entidad financiera para que sea visible por los equipos.
Criterios de aceptación	Modelo operativo desplegado a través de comunicados por correos y WorkPlace a los equipos.
Épica	Adopción de Testing Continuo
Release	Modelo Operativo
Estimación	3

Nota. Detalle de la HU para desplegar el modelo operativo con los equipos internos de la entidad financiera. Fuente: Elaboración propia.

Tabla 20

HU: SQTC-15 - Realizar pilotos de adopción

ID	SQTC-15
Resumen	Realizar pilotos de adopción con los equipos.
Descripción	Realizar un seguimiento a los equipos mientras realizan la adopción del nuevo marco de trabajo y medir los resultados.
Criterios de aceptación	Resultados de las pruebas realizadas por los equipos siguiendo el modelo operativo desplegado.
Épica	Adopción de Testing Continuo
Release	Adopción
Estimación	5

Nota. Detalle de la HU para realizar el piloto con los equipos. Fuente: Elaboración propia.

Las herramientas utilizadas para el proyecto fueron las siguientes:

- **Confluence:** herramienta para compartir todo tipo de información dentro de la entidad financiera, relacionada a lineamientos, estándares, herramientas y procesos. Dentro de Confluence se publicó el modelo operativo de manera pública para la adopción de Testing Continuo en las aplicaciones de la entidad financiera.
- **Artifactory:** herramienta que sirve como repositorio oficial de artefactos, contiene releases desarrollados dentro de la empresa y dependencias externas para que sean utilizados por los equipos en sus proyectos. La librería e imágenes Docker desarrollados fueron desplegados en Artifactory.
- **Bitbucket:** herramienta que se utiliza como repositorio oficial de fuentes dentro de la entidad financiera. En Bitbucket se almacenaron los scripts de automatización de pruebas de los equipos.
- **Jira:** herramienta oficial dentro de la entidad financiera para la gestión de proyectos y permite realizar seguimientos a las actividades de los equipos en cada sprint y los pases a producción.
- **Jira Xray:** Plugin de Jira que permite la gestión de pruebas. Esta extensión le dio la capacidad a Jira de comportarse como el repositorio de casos de pruebas automatizadas.

- **Jenkins:** herramienta estandarizada dentro de la entidad financiera como orquestador de las automatizaciones. A través de Jenkins es cómo se ejecutaron las pruebas automatizadas.

Las tecnologías utilizadas para el proyecto fueron las siguientes:

- **Docker:** es un software que permite automatizar el despliegue de aplicaciones dentro de contenedores y gestionarlos. Para el proyecto, se utilizó Docker para la ejecución de pruebas automatizadas dentro de contenedores en Jenkins.
- **Groovy:** es un lenguaje de programación orientado a objetos desarrollado sobre la plataforma Java que permite desarrollar scripts de automatización.
- **Ansible:** es un software que permite automatizar la configuración de servidores, despliegue y gestión de aplicaciones. Durante el proyecto fue utilizado para la configuración de los servidores de Jenkins aprovisionados exclusivamente para las automatizaciones de Testing.
- **Cucumber:** es una herramienta que permite implementar pruebas basándose en la metodología Behaviour Driven Development (BDD) describiendo los casos de pruebas en un lenguaje lógico y fácil de entender por todos los stakeholders.
- **Serenity:** es una librería de código abierta que permite escribir pruebas automatizadas.
- **Emulador x3270:** Emulador de IBM que permite simular el comportamiento de un sistema core bancario.

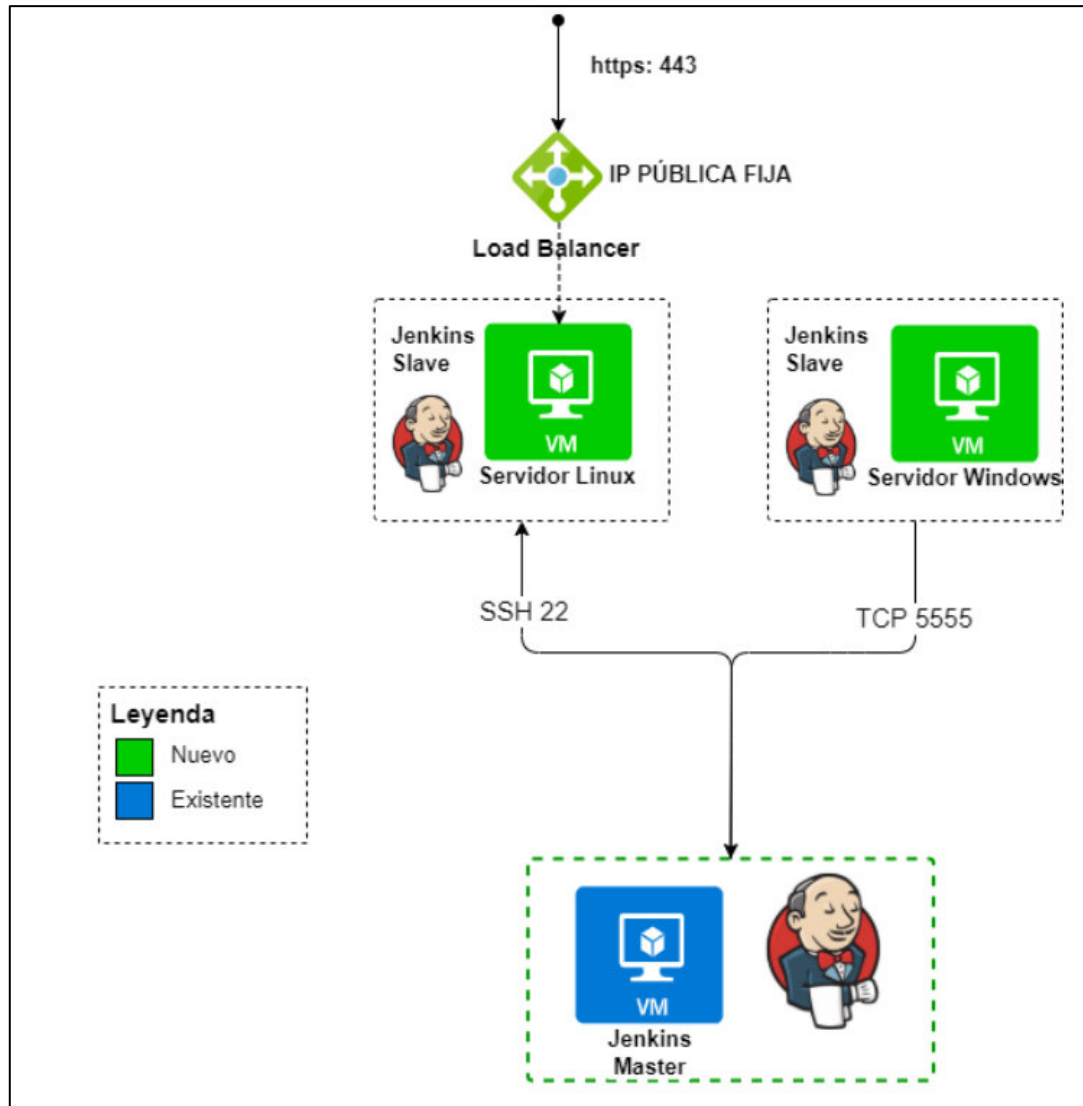
3.2.5.1 Implementación de infraestructura DevOps

Definición de la arquitectura

Dado que la entidad financiera ya contaba con una suite de herramientas, se definió agregar dos servidores exclusivos para realizar pruebas automatizadas y así los pipelines de Testing no compitan por recursos con los pipelines de despliegue y poder tener un mejor control de la gestión de los recursos de estos servidores exclusivos.

Figura 20

Diagrama de arquitectura Jenkins



Nota. La figura detalla la arquitectura definida para el uso de Jenkins con Testing. Fuente: Elaboración propia.

Los componentes nuevos se definieron para cubrir las siguientes necesidades:

- **Servidor Linux:** Máquina virtual de Azure que se conectó como un nodo de Jenkins a través del puerto 22 y funcionó como soporte para la ejecución de pruebas de aplicaciones Web y APIs sobre contenedores Docker.
- **Servidor Windows:** Máquina virtual On Premise que se conectó como un nodo de Jenkins a través del puerto 5555 y permitió la ejecución de pruebas de

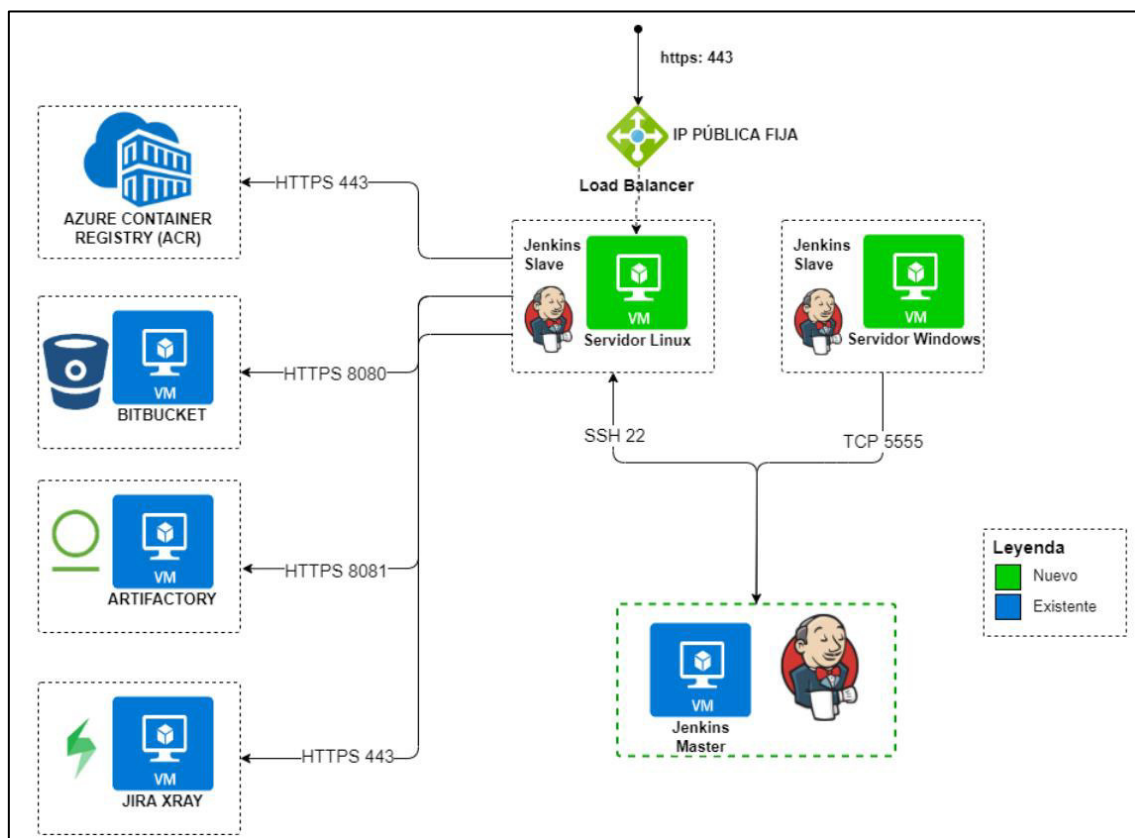
aplicaciones Mainframe, directamente sobre el emulador x3270 instalado en el mismo servidor

- **Load Balancer:** Recurso de Azure para la comunicación con Web Apps desde el servidor Linux configurado como nodo de Jenkins.

Los siguientes diagramas grafican la arquitectura de redes requerida para la comunicación entre las herramientas DevOps y las aplicaciones que participaron de la ejecución de pruebas automatizadas.

Figura 21

Diagrama de redes - Herramientas DevOps



Nota. La figura detalla cómo los servidores de Jenkins se conectan con las otras herramientas DevOps. Fuente: Elaboración propia.

Componentes:

- **Bitbucket**

Repositorio oficial de las fuentes que contienen los scripts automatizados de pruebas y que son descargados al momento de ejecutar el pipeline de Testing desde Jenkins a través del puerto 8080.

- **Jira Xray**

Repositorio oficial para almacenar los casos de pruebas, gestionar el plan de pruebas e importar los resultados de las pruebas. Al finalizar la ejecución de las pruebas, los resultados y las evidencias fueron importados a través del API de Jira Xray por el puerto 443 desde Jenkins.

- **Artifactory**

Repositorio oficial de artefactos que se utilizan durante la ejecución de las pruebas. Al ejecutar el pipeline de Testing, durante la compilación del proyecto, las dependencias indicadas en el pom.xml son descargadas desde el repositorio de Artifactory.

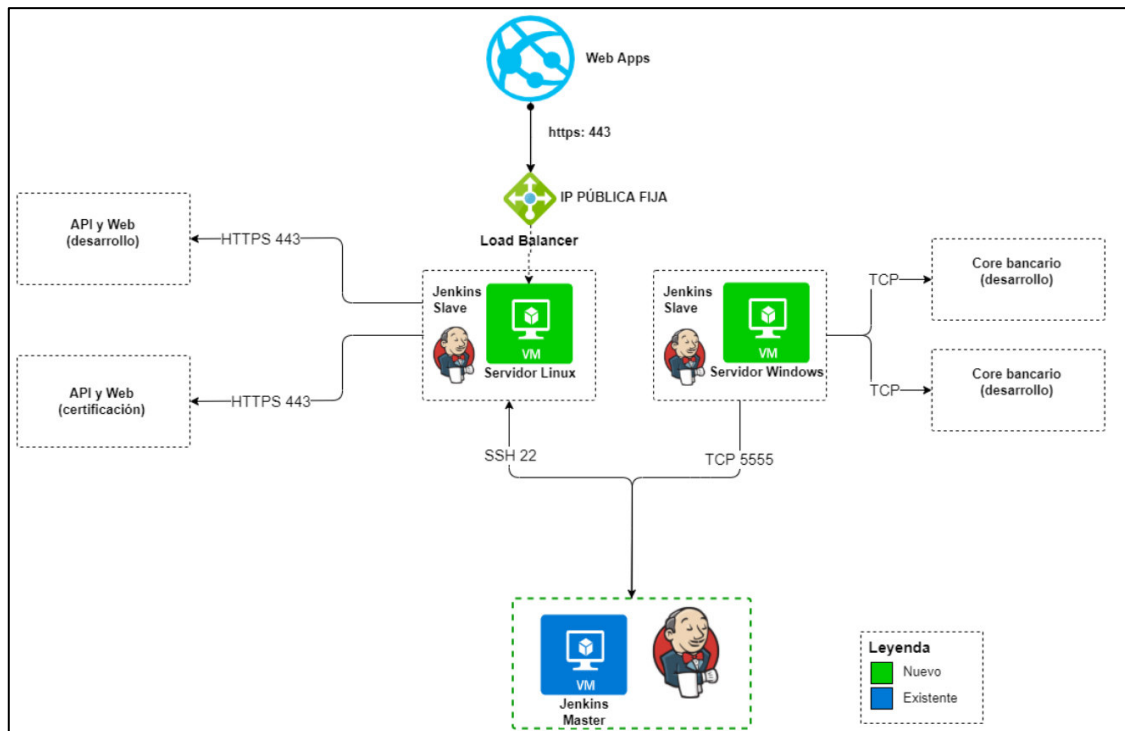
- **Azure Docker Registry**

Repositorio de las imágenes Docker que se utilizan durante la ejecución de las pruebas. Al ejecutar el pipeline de Testing, las imágenes Docker son descargadas desde el ACR accediendo con un usuario y token que cuente con los permisos de descarga.

Para realizar las pruebas de los aplicativos API y Web desde Jenkins se estableció la comunicación por el puerto seguro 443 hacia cada uno de estos aplicativos. Y para el caso de los aplicativos mainframe, la comunicación se estableció desde el servidor Windows dado que se encontraban en un segmento de red distinto a los servidores de Azure.

Figura 22

Diagrama redes – Aplicativos



Nota. La figura detalla cómo las aplicaciones del tipo API, Web y Mainframe se conectan con los esclavos de Jenkins. Fuente: Elaboración propia.

Implementación de la arquitectura

Luego de haber definido los componentes de la arquitectura y la comunicación necesaria entre ellos con la aprobación de Arquitectura y Seguridad, se proyectó que tendríamos una carga de hasta 4 aplicativos realizando pruebas en simultaneo para el caso de APIs y Web en el servidor Linux, y hasta 2 aplicativos mainframe en simultaneo en el servidor Windows.

Para poder cubrir esta necesidad de infraestructura, se definieron que los servidores deberían cumplir con las siguientes capacidades mínimas para soportar la carga de trabajo estimada y de ser necesario se realizaría un incremento luego de iniciar la operativa con los equipos.

Tabla 21

Infraestructura nueva solicitada

Recurso	Tipo	Sistema Operativo	Descripción
Virtual Machine Linux	Cloud	RHEL 7.8	1 D4s v4 (4 vCPUs, 16 GB RAM) x 730 Hours; Linux – Red Hat Enterprise Linux
Virtual Machine Windows	On Premise	Windows 2019	Server CPU 8 RAM 16 GB RAM
Load Balancer	Cloud	-	Standard Tier: 5 Rules, 3,000 GB Data Processed

Nota. La tabla describe los recursos solicitados para implementar la arquitectura definida en Jenkins. Fuente: Elaboración propia.

El presupuesto para los recursos de Tipo Cloud, fueron estimados en base a la calculadora del Portal de Azure, mientras que los recursos On Premise se estimaron en base a la calculadora interna de la entidad financiera.

Cada recurso fue llevado a un comité para sustentar el presupuesto requerido y la necesidad de cada uno y luego de la aprobación, se registraron las solicitudes de aprovisionamiento.

Luego de la confirmación del aprovisionamiento de los recursos, se procedió a configurar internamente cada uno de ellos para que cumplan su función de acuerdo con la arquitectura definida.

Load Balancer: Se configuró como único Backend Pool al servidor Linux Azure por el puerto 443.

Servidor Linux: la configuración de este servidor se realizó de manera automatizada a través de un script de Ansible que incluyó las siguientes tareas:

- Instalación de utilitarios base Linux.
- Creación de carpetas conforme la estructura de trabajo definido para Jenkins.
- Configuración de los niveles de permisos de las carpetas de Jenkins con el usuario y grupo de usuario definido como administrador para esta herramienta.
- Instalación y configuración de Java.
- Instalación y configuración de Maven.
- Instalación y configuración de GIT.
- Instalación y configuración de Docker como servicio dentro del servidor.

Servidor Windows: la configuración de este servidor se realizó de manera manual y se ejecutaron las siguientes tareas:

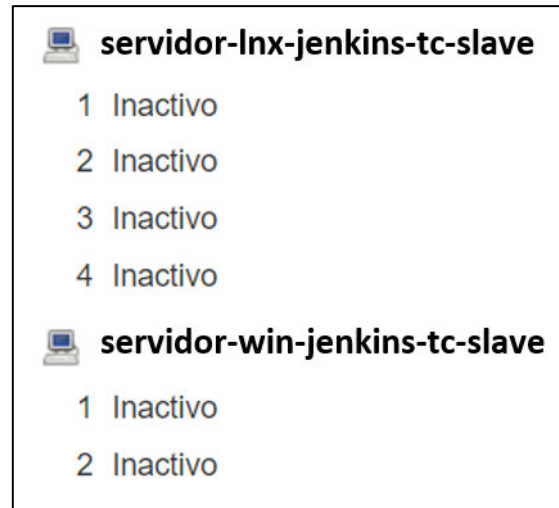
- Configuración del usuario administrador de Jenkins dentro del servidor.
- Creación de carpetas de acuerdo con la estructura definida para Jenkins.
- Instalación y configuración de Java.
- Instalación y configuración de los certificados internos de la entidad financiero en el Cacerts de Java.
- Instalación y configuración de Maven.
- Instalación y configuración de GIT.
- Instalación y configuración del emulador x3270.

Finalmente, cada servidor fue configurado desde la vista de administrador de la herramienta Jenkins y conectado como nodos con la restricción de que solo se puedan ejecutar pipelines de Testing en ellos.

Dada las características del servidor Linux, fue configurado con 4 ejecutores, mientras que el servidor Windows fue configurado con 2 ejecutores para atender las peticiones en paralelo.

Figura 23

Servidores Jenkins configurados



Nota. La figura muestra los servidores conectados a Jenkins luego de haber sido configurados. Fuente: Elaboración propia.

3.2.5.2 Integración con herramientas DevOps

Definición de perfilado

Se definieron perfiles por cada herramienta con diferentes niveles de permisos, manteniendo los lineamientos de seguridad, de acuerdo con las actividades que los QE necesitaron realizar en cada una de ellas.

Para el caso de Jira Xray, al ser el repositorio oficial de casos de pruebas, se vio la necesidad de que los QE cuenten con permisos avanzados para gestionar las pruebas: Por lo cual se otorgaron permisos de escritura para gestionar todo tipo de issue dentro de Jira, por ejemplo, crear, editar, eliminar, importar y exportar; y visualizar reportes. Todos los permisos otorgados estaban limitados solo al tablero del equipo al cual perteneció el QE, por ende, no podían ver reportes o pruebas de otros equipos.

Figura 24

Perfilado Jira Xray

JIRA XRAY	
PERFIL	PERMISO
QE/ Chapter Leader QE	Gestionar issues (crear, editar: Test, Test Set, Test Execution, Test Plan)
	Exportar e importar issues
	Cargar evidencias de los resultados de las pruebas

Nota. La figura muestra el detalle de los permisos establecido para el uso de Jira Xray por los QE y Chapter Leader QE dentro de los tableros del equipo al cual pertenecían. Fuente: Elaboración propia.

En el caso de Bitbucket, se restringieron los permisos para que los QE puedan acceder solo a los repositorios de scripts de pruebas que se encontraban dentro del proyecto de Bitbucket del equipo, con un perfil similar al de un desarrollador: Adicionalmente, se definió que los Chapter Leader QE cumplirían con el rol de revisor de scripts y brindar su conformidad y aprobar los PR para realizar pruebas automatizadas en el ambiente de certificación.

Figura 25

Perfilado Bitbucket

BITBUCKET	
PERFIL	PERMISO
QE	Crear, eliminar y escribir sobre ramas de trabajo del tipo "feature".
	Solicitar aprobación de Pull Request (PR)
Chapter Leader QE	Aprobar PR

Nota. La figura muestra el detalle de los permisos definidos para el uso de Bitbucket por los QE y Chapter Leader QE. Fuente: Elaboración propia.

Para el caso de Jenkins, se definieron permisos diferenciados por cada entorno de pruebas, en donde se brindaron mayores permisos para el entorno de desarrollo, mientras que en el entorno de certificación solo se otorgaron permisos de ejecución dado que era un ambiente controlado.

Figura 26:

Perfilado Jenkins

JENKINS		
ENTORNO	PERFIL	PERMISO
Desarrollo	QE/ Chapter Leader QE	Configurar Jobs
		Configurar credenciales
		Ejecutar Jobs
Certificación		Ejecutar Jobs

Nota. La figura muestra el detalle de los permisos definidos para el uso de Jenkins por los QE y Chapter Leader QE. Fuente: Elaboración propia.

Análisis y diseño de librería para ejecutar pruebas

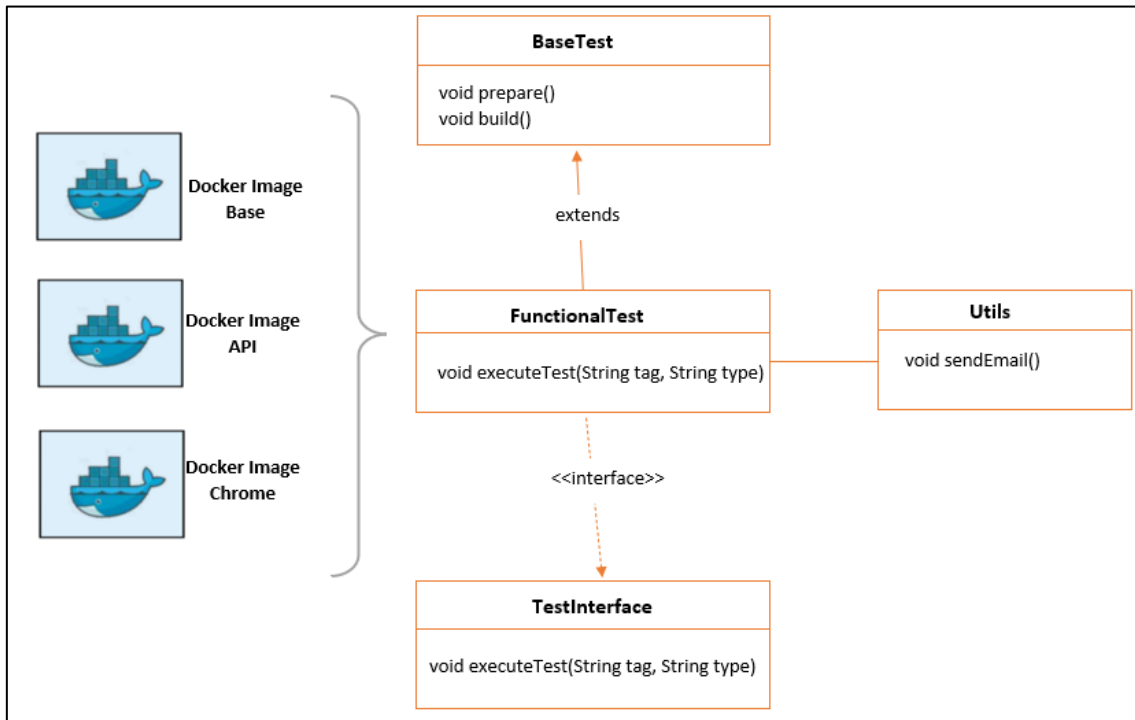
Para poder ejecutar las pruebas automatizadas desde Jenkins se definió una librería compartida de Testing que los equipos puedan utilizar en sus pruebas.

La librería compartida siguió las siguientes especificaciones:

- Desarrollada en Groovy.
- Permitió la ejecución de pruebas del tipo API desde Jenkins.
- Permitió la ejecución de pruebas de aplicaciones Web sobre el navegador Chrome en las versiones 96, 97 y 98 desde Jenkins.
- Para la ejecución recibió como parámetros de entrada el TAG de la prueba a ejecutar, el tipo de prueba.
- Las contraseñas que se utilizaban durante la ejecución se obtuvieron de la bóveda de secretos de la entidad financiera: Hashicorp Vault.
- La librería podía ser ejecutada desde el folder de Jenkins de todos los aplicativos.

Figura 27

Diseño librería ejecución pruebas



Nota. La figura muestra el diseño de las clases definidas para la librería de Testing y las imágenes Dockers identificadas para la compilación y ejecución de las pruebas. Fuente: Elaboración propia.

Como resultado del diseño se definieron 4 clases para la ejecución de pruebas automatizadas:

- **BaseTest:** Clase base que contuvo métodos en común para compilar el proyecto, preparar el espacio de trabajo y realizar validaciones mínimas necesarias para la ejecución de pruebas.
- **TestInterface:** Interface con los métodos para ejecutar pruebas.
- **FunctionalTest:** Clase que implementó los métodos definidos en la interface TestInterface.
- **Utils:** Clase con métodos utilitarios como el envío de correo para las notificaciones.

Adicionalmente se definieron 3 imágenes Docker para la ejecución de las pruebas:

- **Imagen Docker Base:** Imagen basada en Ubuntu con Java y Maven para compilar el proyecto de Testing.
- **Imagen Docker API:** Imagen basada en Ubuntu con Java y Maven para la ejecución del proyecto de Testing con pruebas de APIs.
- **Imagen Docker Chrome:** Imagen basada en Ubuntu con Java y Maven y el driver de Chrome para la ejecución del proyecto de Testing con pruebas de Web.

Implementación de librería para ejecutar pruebas

La librería fue desarrollada en Groovy y el método principal para ejecutar las pruebas del tipo Web y API, iniciaba un contenedor Docker y ejecutaba las pruebas dentro del contenedor con comandos Maven. Mientras que el método para ejecutar pruebas del tipo Mainframe utilizaba comandos batch de Windows para iniciar el emulador x3270.

Figura 28

Librería Ejecución Pruebas Automatizadas

```
this.script.steps.sh "echo 'Executing TAG: ${tag}'"  
this.script.steps.sh ""docker run --network host --cpus 2 --name 'C_TESTING' --rm  
-v /maven/repository:/usr/.m2/repository  
-v /java/jdk/jre/lib/security/cacerts:/usr/lib/jvm/jdk/jre/lib/security/cacerts  
-v '${script.env.WORKSPACE}:/project/data' -w /project/data acr.azurecr.io/chrome_v96:1.0.0  
mvn clean verify -Dcucumber.filter.tags=\"${tag}\" -Dweb.custom.driverPath=/etc/selenium/browser_driver  
-Dweb.browser.headless=true -Dweb.custom.driverPath.absolute=true ""
```

Nota. La figura muestra un ejemplo del método principal para ejecutar pruebas desde la librería de Testing. Fuente: Elaboración propia.

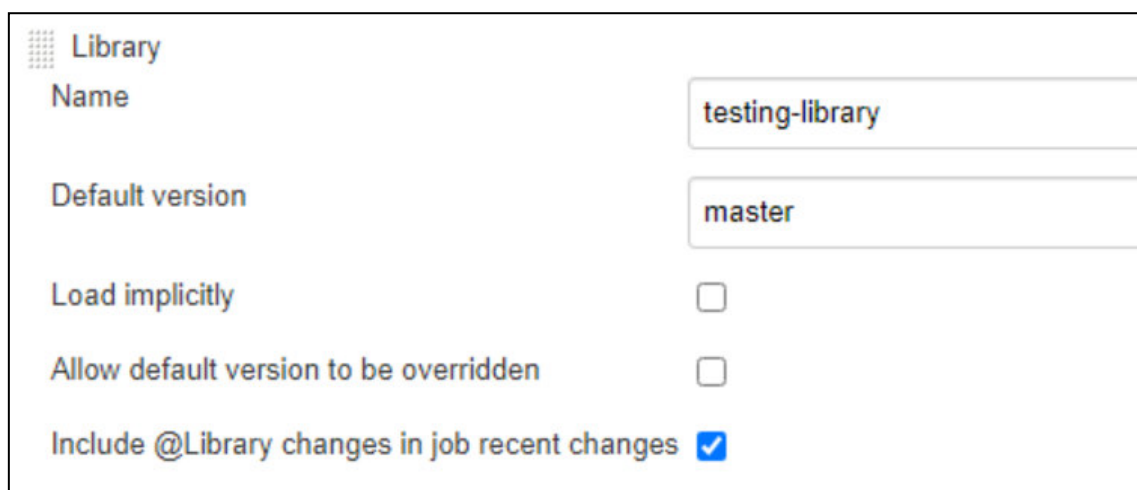
Al finalizar el desarrollo y obtener el conforme tras la revisión de pares con el Líder Técnico, se realizó el despliegue en producción para que la librería esté disponible para todos los equipos, lo cual consistió en promover los cambios en el repositorio desde la

rama de trabajo “develop” hacia la rama protegida “master” e importar el artefacto en Artifactory.

Una vez realizado el despliegue, la librería fue configurada desde la pestaña administrador de Jenkins como librería compartida para que sea reutilizada en los pipelines de Testing de los equipos.

Figura 29

Configuración librería Jenkins



Library	
Name	testing-library
Default version	master
Load implicitly	☐
Allow default version to be overridden	☐
Include @Library changes in job recent changes	☒

Nota. La figura muestra un ejemplo de la configuración de librería de Testing en Jenkins.
Fuente: Elaboración propia.

De la misma manera, la rama de trabajo del repositorio, que contenía el código fuente de los Docker Files, fue promovida a la rama protegida “master” y las imágenes Docker fueron construidas y desplegadas en el repositorio Azure Docker Registry para que puedan ser utilizadas por los equipos durante la ejecución de sus pruebas.

Finalmente se elaboró un proyecto ejemplo que fue desplegado como guía para que los equipos lo reutilicen durante la adopción del marco de trabajo. El proyecto ejemplo contenía un pipeline del tipo declarativo con los métodos para utilizar la librería compartida de Testing. Y también se habilitó una plantilla en Jenkins.

Figura 30

Ejemplo uso librería Testing

```
@Library('testing-library')
import library.FunctionalTest
def testing = new FunctionalTest(this)
try {
    node {
        stage("Preparation") {
            checkout scm
            testing.prepare()
        }
        stage("Build") {
            testing.build()
        }
        stage("Test") {
            testing.executeTest("@LOGIN", "web")
        }
    }
} catch (Exception e) {
    node {
        throw e
    }
}
```

Nota. La figura muestra un ejemplo del pipeline de Testing importando la librería compartida desarrollada que los equipos usaron en sus proyectos. Fuente: Elaboración propia.

3.2.5.3 Adopción de Testing Continuo

Elaboración y despliegue de Modelo Operativo

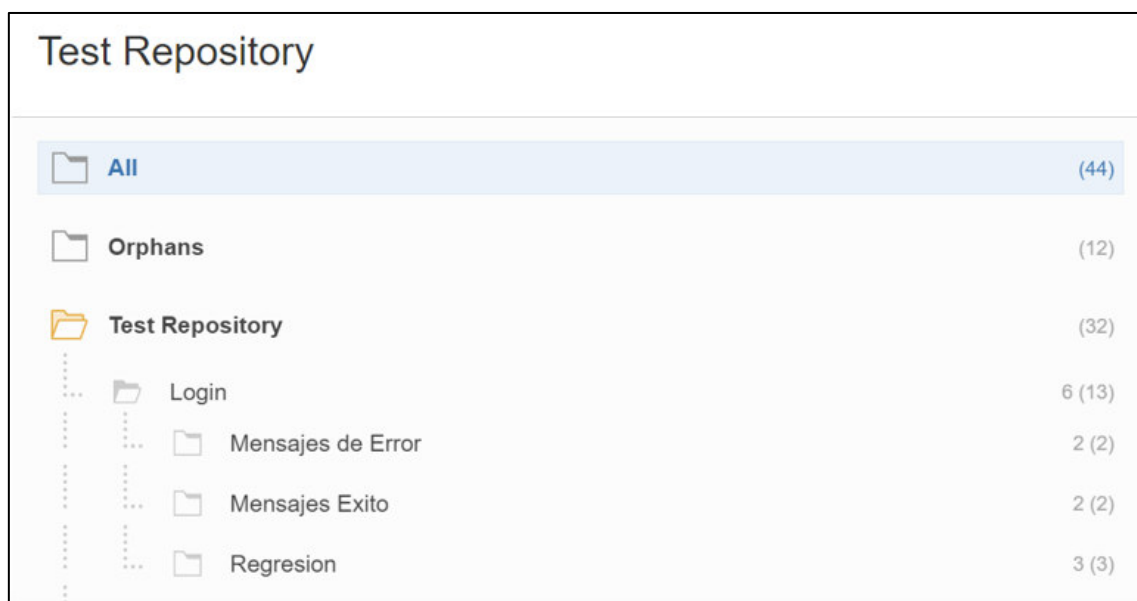
Para que los equipos pudieran utilizar las herramientas se definieron lineamientos y buenas prácticas que acompañaron al perfilado definido en el punto 3.2.5.2, los cuales fueron publicados en Confluence y compartidos a través de comunicados por los canales internos de la División de TI.

Jira Xray

Al ser Jira Xray un complemento de la herramienta Jira, el primer lineamiento fue que todo equipo debió contar con un tablero en esta herramienta. Los equipos que utilizaban metodología de trabajo cascada crearon tableros del tipo Kanban, mientras que los equipos ágiles, ya contaban con tableros del tipo Scrum. Dentro del proyecto de Jira, se visualizaba el repositorio de pruebas, el cual siguió la siguiente estructura de carpetas:

Figura 31

Estructura proyectos Jira Xray



Nota. La figura muestra la estructura de proyectos dentro del repositorio Jira Xray.

Fuente: Elaboración propia.

En donde:

- Dentro del repositorio, se creó una carpeta por cada funcionalidad de la aplicación y dentro subcarpetas por cada escenario de prueba a cubrir.
- La descripción del escenario de pruebas era redactada en el lenguaje Gherkin.

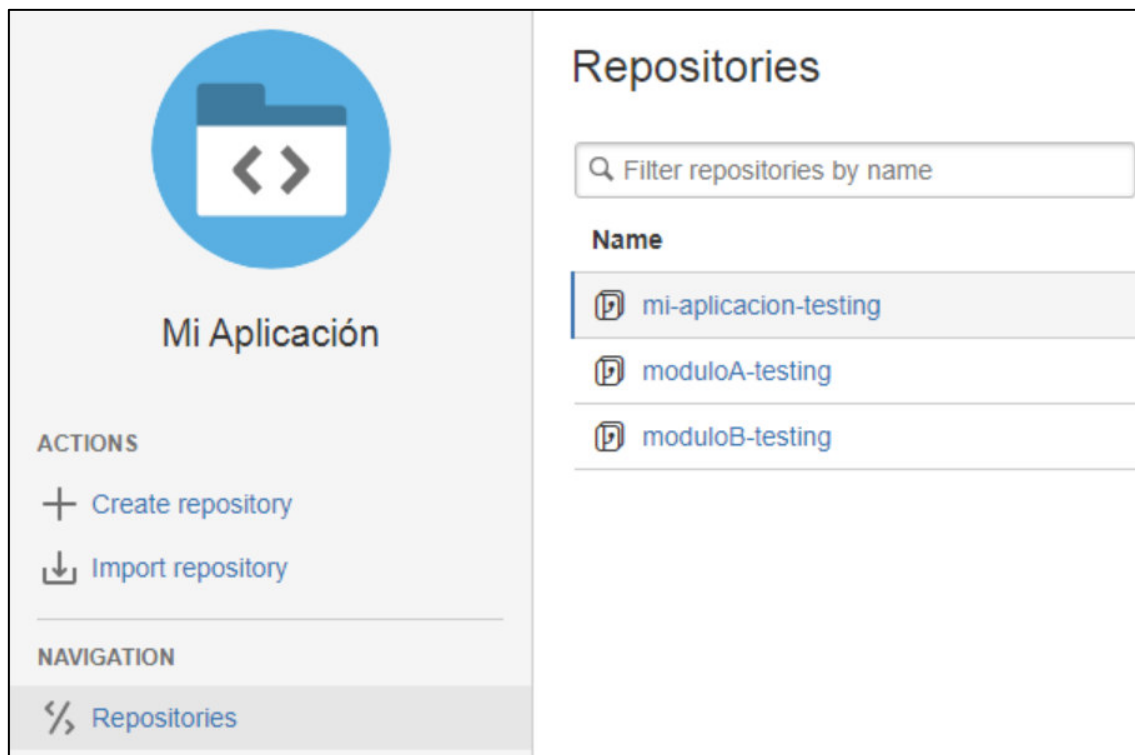
- Al momento de crear los issues en Jira, cada issue del tipo Test era agrupado en un issue del tipo Test Set, el cual equivalía a una funcionalidad del aplicativo.
- De acuerdo con la estrategia de pruebas definida por el QE, se creaban Test Execution con las pruebas a realizar en cada ciclo de pruebas.
- El Text Execution debía existir previamente a la ejecución de la prueba para tener la trazabilidad desde la ejecución hasta la entrega de los resultados en Jira Xray.

Bitbucket

Para el caso de Bitbucket, se definió que los repositorios de scripts de automatización de pruebas se crearían dentro del proyecto de la aplicación que ya contenía los otros repositorios de desarrollo Y dentro del proyecto de Bitbucket, se siguió la siguiente estructura para los repositorios de pruebas:

Figura 32

Estructura proyectos Bitbucket



Nota. La figura muestra la estructura de proyectos dentro de Bitbucket. Fuente: Elaboración propia.

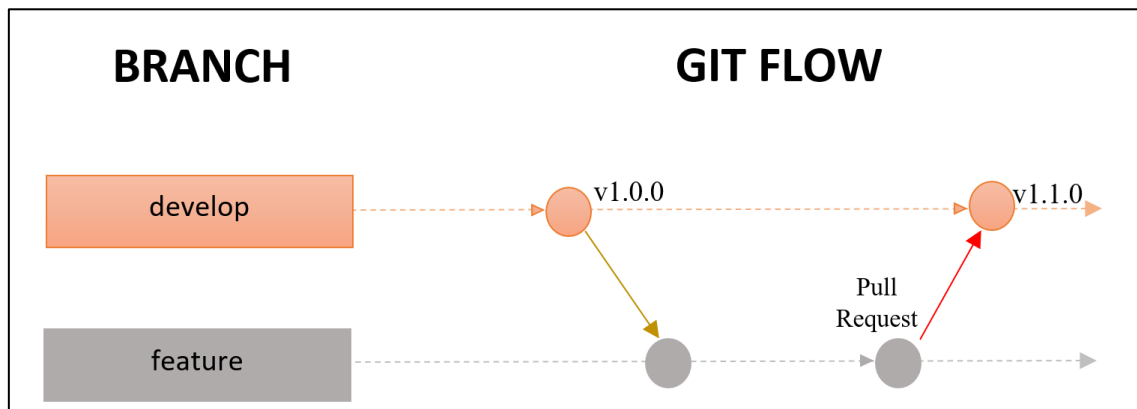
En donde:

- Como principal lineamiento, dentro del proyecto de la aplicación, se creó un único repositorio para los scripts de pruebas de toda la aplicación que siguió la nomenclatura “[nombre_mi_aplicacion]-testing”.
- Sin embargo, en algunos casos, la aplicación estaba disgregada por módulos que se probaban de manera independiente, por lo cual, se crearon repositorios por cada uno de ellos.

Adicionalmente, se definió el siguiente manejo de ramas del repositorio para que los QE pudieran tener flexibilidad y autocontrol sobre los cambios que realizaban y a su vez cuenten con una rama de trabajo protegida (develop) donde se versionaba el estado total del repositorio:

Figura 33

Flujo GIT Flow



Nota. La figura muestra el flujo Git Flow definido para los repositorios de scripts automatizados de pruebas de los equipos que solo contaba con dos tipos de ramas de trabajo: develop (protegida) y feature. Fuente: Elaboración propia.

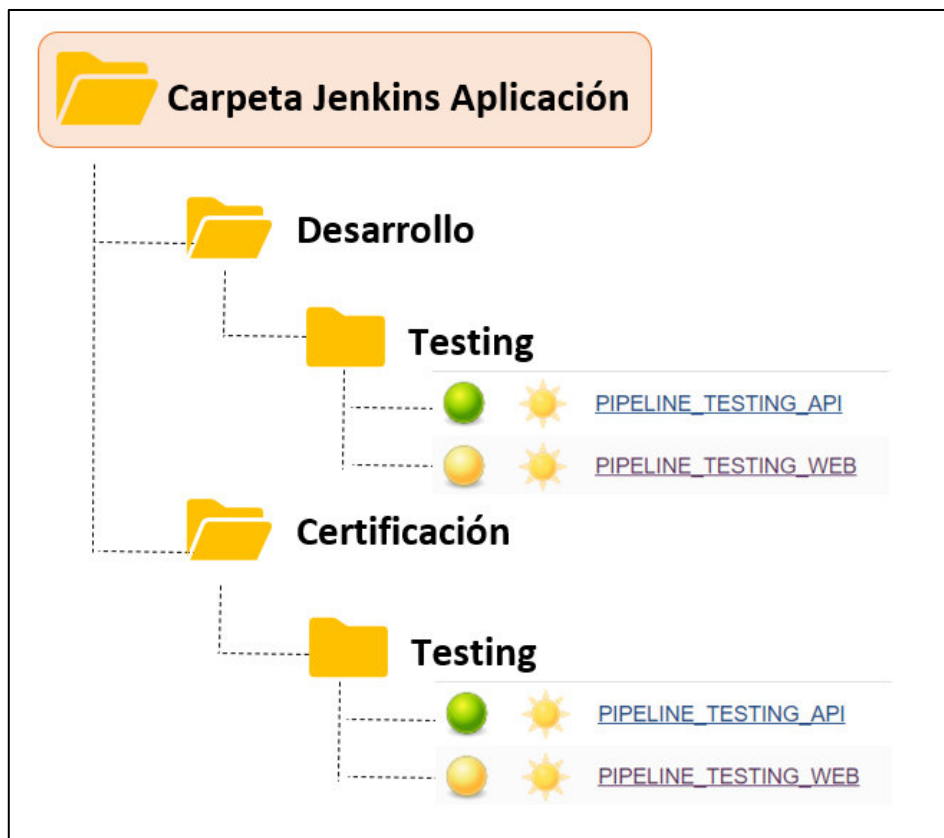
Jenkins

En el caso de Jenkins, se definió la siguiente estructura de proyectos por cada aplicativo, en donde se creó una carpeta con el nombre “Testing” dentro de las carpetas que ya se tienen separadas por ambiente, desarrollo y certificación. Al interior de la carpeta Testing, se configuraron los Jobs para la ejecución de pruebas.

Fue importante seguir la nomenclatura, dado que los nodos configurados exclusivamente para Testing tenían patrones de exclusividad basándose en el nombre de las carpetas.

Figura 34

Estructura carpetas Jenkins



Nota. La figura muestra la estructura de carpetas dentro de Jenkins por cada entorno de un aplicativo para organizar los pipelines de Testing. Fuente: Elaboración propia.

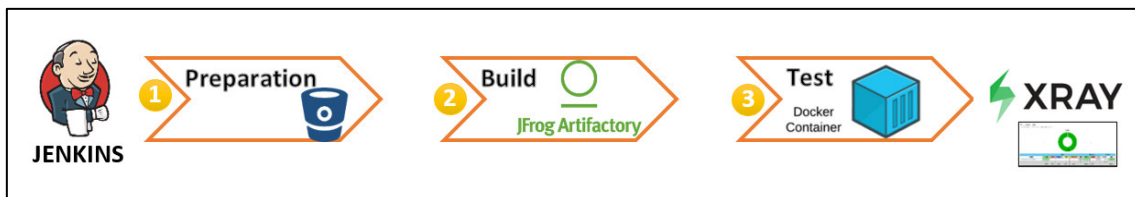
Adicionalmente, se proporcionó un pipeline de Testing para que los equipos lo ejecuten desde Jenkins con 3 etapas.

- **Preparation:** En esta etapa Jenkins descargaba las fuentes desde Bitbucket y validaba los parámetros de entrada.
- **Build:** En esta etapa Jenkins descargaba las dependencias del proyecto definidas en el archivo pom.xml desde Artifactory y compilaba las fuentes descargadas en la etapa previa.
- **Test:** En esta etapa Jenkins realizaba alguno de estos dos posibles escenarios:
 - **Pruebas de API y/o Web:** Iniciaba un contenedor Docker y ejecutaba las pruebas dentro del contenedor.
 - **Pruebas de API y/o Web: Se conectaba al servidor Windows,** iniciaba el emulador x3270 y ejecutaba las pruebas desde el mismo servidor.

Finalmente generaba un reporte de Cucumber e importaba los resultados a Jira Xray.

Figura 35

Pipeline de Testing



Nota. La figura muestra las etapas del pipeline de Testing. Fuente: Elaboración propia.

Adopción

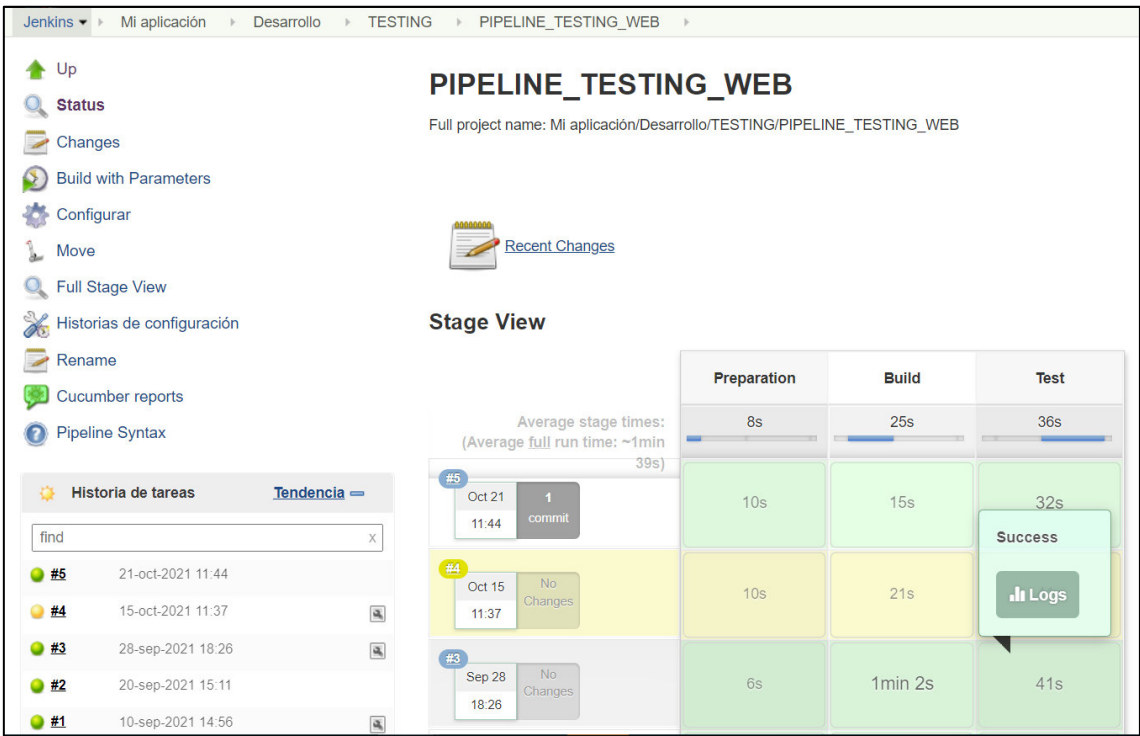
El marco de trabajo de Testing Continuo fue adoptado de manera incremental por los equipos a los cuales se realizó un seguimiento y acompañamiento durante la etapa de pilotos para que luego puedan aplicar el modelo operativo por sí mismos. Cada equipo adoptó Testing Continuo según su propio marco de trabajo, ágil o cascada, en donde los

primeros presentaron resultados por sprint, mientras que los segundos de acuerdo con su cronograma de trabajo.

Cada equipo pudo realizar la integración con las herramientas, gestionar su código fuente en Bitbucket, ejecutar sus pruebas desde Jenkins y visualizar los resultados en el reporte de Cucumber en Jenkins y en Jira Xray como se visualiza a continuación.

Figura 36

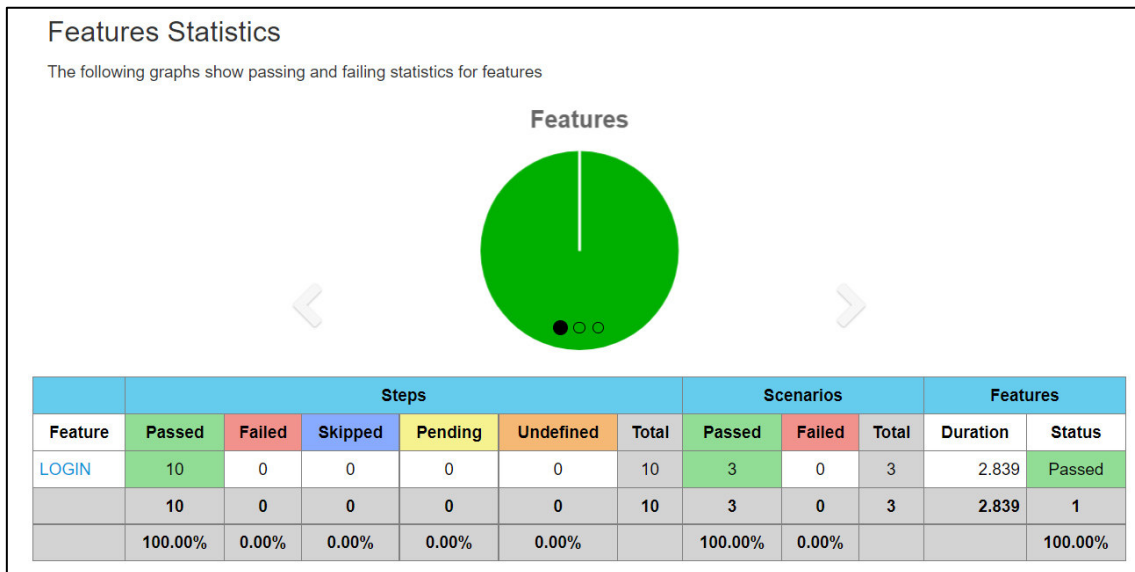
Ejecución pipeline Testing desde Jenkins



Nota. La figura muestra un ejemplo de la ejecución de un pipeline en Jenkins en donde se visualiza las etapas por las pasa el job, los tiempos que toma cada etapa y el estado del resultado de la ejecución. Fuente: Elaboración propia.

Figura 37:

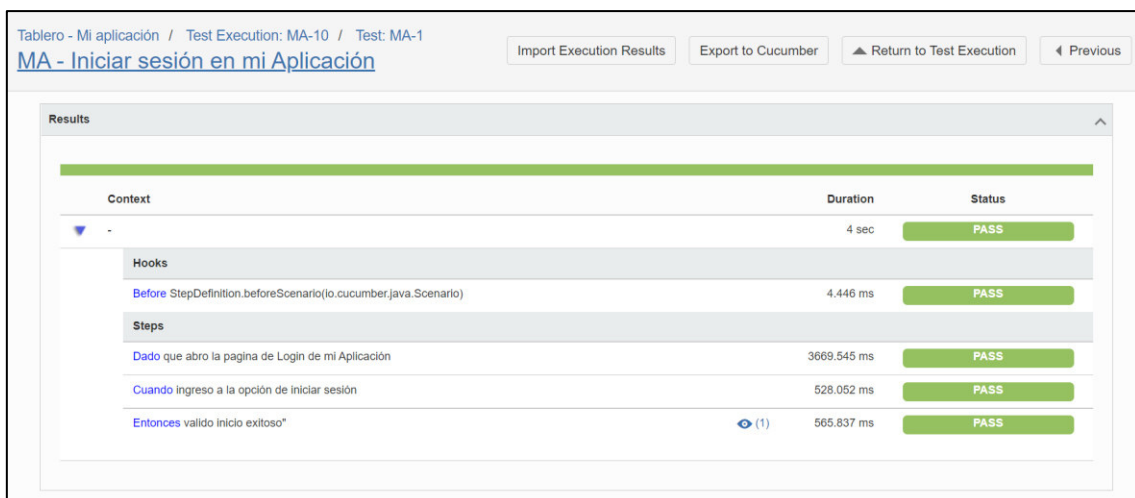
Reporte Cucumber Jenkins



Nota. La figura muestra un ejemplo del reporte Cucumber en Jenkins. Los valores son referenciales. Fuente: Elaboración propia.

Figura 38

Resultados Jira Xray

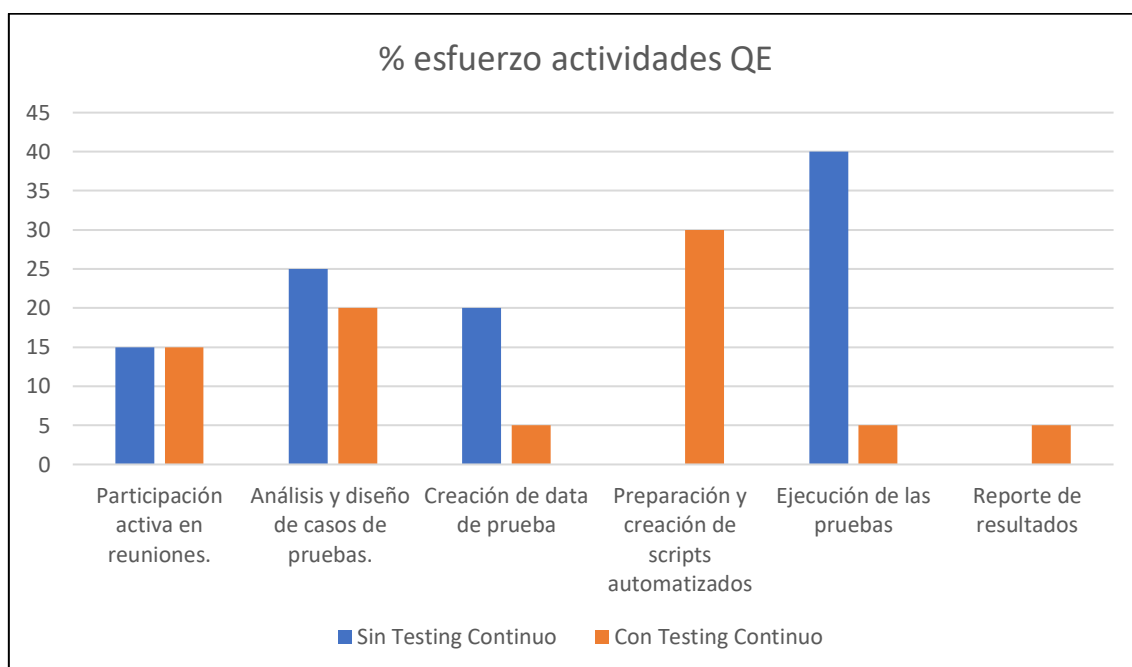


Nota. La figura muestra un ejemplo de los resultados en Jira Xray. Los valores son referenciales. Fuente: Elaboración propia.

Los resultados obtenidos por los equipos luego de la adopción fueron los siguientes y varían de acuerdo con la realidad de cada aplicativo:

Figura 39

Optimización de las tareas ejecutadas por los QE



Nota. La figura muestra la diferencia entre el porcentaje de esfuerzo de realizar las actividades de un QE sin Testing Continuo versus aplicando las prácticas de Testing Continuo. Fuente: Elaboración propia.

En promedio los QEs evidenciaron un ahorro del 20% de esfuerzo de sus actividades automatizadas versus las manuales.

Tabla 22

Tiempo ejecución pruebas manuales

Aplicativo	Número de casos de pruebas	Cantidad de casos ejecutados por día (8 horas)	Tiempo aprox. ejecución (días)
Aplicativo Web	280	20	14
Aplicativo Mainframe	390	26	15

Nota. La tabla muestra el tiempo aproximado de ejecución de pruebas manuales por día de dos tipos de aplicativos. Fuente: Elaboración propia.

Tabla 23

Tiempo ejecución pruebas automatizadas

Aplicativo	Número de casos de pruebas	Cantidad de casos ejecutados por día (8 horas)	Tiempo aprox. ejecución (días)
Aplicativo Web	280	280	1
Aplicativo Mainframe	390	195	2

Nota. La tabla muestra el tiempo aproximado de ejecución de pruebas automatizadas por día de dos tipos de aplicativos. Fuente: Elaboración propia.

El aplicativo web pudo ejecutar 280 pruebas automatizadas por día, incrementando en un 1400% su capacidad de ejecución de pruebas; mientras que el aplicativo mainframe pasó a ejecutar 195 pruebas automatizadas por día, incrementando en un 750% su capacidad de ejecución de pruebas.

Tabla 24

Ahorro tiempos ejecución pruebas

Aplicativo	% reducción tiempos	Antes Tiempo de ejecución (días)	Ahora Tiempo de ejecución (días)
Aplicativo Mainframe	85%	60.0	8.8
Aplicativo Web/API	73%	6.0	1.6

Nota. La tabla muestra el ahorro en tiempos de ejecución de pruebas de dos tipos de aplicativos. Fuente: Elaboración propia.

Los aplicativos obtuvieron los siguientes beneficios luego de una fase 1 de adopción:

- Pudieron automatizar en promedio el 49% de sus flujos.
- Obtuvieron en promedio el 79% de reducción de tiempos durante la etapa de pruebas.
- Pudieron reutilizar al 100% los scripts de pruebas automatizadas realizadas.

CAPÍTULO IV REFLEXIÓN CRÍTICA DE LA EXPERIENCIA

- La participación de la autora en el proyecto fue como Ingeniera DevOps, lo cual permitió afianzar sus conocimientos en estas prácticas y enfocarlas en el área de Testing.
- Aunque el pipeline de Testing definido ayudó a la ejecución de pruebas automatizadas, se necesitó de la intervención humana de un QE para que pueda ser ejecutado, lo cual generó una dependencia con la disponibilidad del QE para realizar pruebas a pesar de que ya había una estrategia definida y solo era necesario esperar los resultados.
- Se pudo automatizar la creación de data de prueba que dependía de APIs o portales Web, sin embargo, muchos datos core se encontraban a nivel de BD o dependían de rutinas batch para generar datos de escenarios específicos que no pudieron ser automatizados con scripts.
- Aunque se evidenció una notable reducción de tiempos en la ejecución de pruebas, se encontró que muchas de las API core eran inestables en los ambientes de desarrollo y certificación, lo cual, en algunos casos, detuvo las pruebas hasta en 8 días aproximadamente mientras se encontraba una solución al incidente.
- Al inicio del proyecto se planteó un orden de actividades a seguir durante la implementación del framework, sin embargo, dado que los equipos tenían diferentes necesidades y puntos de dolor, los esfuerzos no fueron destinados de la manera más optima.

- Es importante realizar un plan de adopción para que los equipos empiecen a utilizar el marco de trabajo dado que no solo implicó actividades operativas, sino que representó un cambio de cultura en la empresa.
- El perfilado definido se basó en el modelo de “desarrolladores” y se definieron permisos por aplicativo, sin embargo, el área de Testing es cross a diferentes aplicativos y en la mayoría de los casos pueden rotar de equipos, por lo cual, los QE tuvieron que solicitar diferentes permisos, volviendo inviable un perfilado diferenciado.

CAPÍTULO V CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- Se logró que los equipos que ya estaban familiarizados con un marco de trabajo ágil y DevOps adopten el marco de trabajo de Testing Continuo con facilidad debido al nivel de madurez que ya tenían este tipo de prácticas y cultura.
- Se logró que los equipos que siguen un marco de trabajo cascada adopten el marco de trabajo de Testing Continuo dado que la automatización de las pruebas se trabajó como un proyecto independiente al proyecto de desarrollo.
- Se logró una reducción en promedio del 75% en el tiempo de la ejecución de pruebas debido a que la ejecución de los pasos funcionales se realizó de manera automática sin intervención humana.
- Se logró implementar scripts de automatización de pruebas de los aplicativos que realizaban pruebas de tipo Web, API y Mainframe.
- Se definieron lineamientos para la gestión de casos de pruebas dentro de repositorios cuyo gobierno y control será administrado por el CoE de Testing.
- Se evidenció un incremento de hasta 1000% en la capacidad de ejecución de pruebas por día dado que ahora los equipos cuentan con un repositorio de casos de pruebas y scripts de automatización que pueden ser reutilizados en diferentes ciclos de prueba, además de las ejecuciones automáticas y la reducción de tiempos de ejecución.

- Al reducir las pruebas que se ejecutaban de manera manual, se evidenció una reducción de incidentes en producción de al menos 30% debido a que había menos probabilidad de tener falsos positivos por errores humanos durante las pruebas.
- Se consiguió un ahorro a nivel de costos en promedio de 16 mil soles por cada despliegue de las aplicaciones que adoptaron el framework debido a los tiempos de ahorro durante la ejecución de las pruebas.

5.2 RECOMENDACIONES

De la experiencia adquirida durante el proyecto, se recomienda lo siguiente:

- Optimizar la ejecución de pruebas desde Jenkins para que los pipelines de Testing se puedan ejecutar de forma programada en una fecha y hora específica según la necesidad de la aplicación durante sus pruebas de regresión.
- Coordinar y definir en conjunto con el squad de Despliegue Continuo la integración con el pipeline de Testing para que se ejecute de forma automática luego de cada despliegue tanto en los ambientes de desarrollo como de certificación.
- Evaluar la implementación de una herramienta para la gestión de datos de prueba de forma automatizada, segura y consistente.
- Evaluar la implementación de una herramienta para la virtualización de servicios de las APIs más inestables y así las pruebas no se vean afectadas y poder obtener feedback temprano.
- Las actividades propuestas por el marco de trabajo de Testing Continuo se deben aplicar de acuerdo con la necesidad y realidad de cada equipo, en donde, luego de la evaluación de puntos de dolor, el equipo deberá priorizar las actividades a

realizar para automatizar las pruebas. Por ejemplo: la generación de data de pruebas, la inestabilidad de ambientes previos o elaboración de scripts de automatización.

- Definir un plan de comunicación para compartir los nuevos lineamientos con los diferentes equipos y realizar evaluaciones para identificar puntos débiles que pueden ser resueltos con capacitaciones.
- Definir lineamientos que permitan asegurar que los nuevos aplicativos nazcan con los lineamientos de Testing Continuo.
- Definir un perfilado que permita que los QE puedan ejecutar pruebas de diferentes aplicativos y contar con un perfil con permisos de autorizador que permite que un QE líder pueda validar los scripts y plan de pruebas de los QE.

5.3 FUENTES DE INFORMACIÓN

DevOps Institute. (n.d.). Retrieved from DevOps Institute:

<https://www.devopsinstitute.com/wp-content/uploads/2020/11/CTF-Blueprint.pdf>

Hornbeek, M. (2018, Marzo). *DevOps Institute*. Retrieved from DevOps Institute:

<https://devopsinstitute.com/wp-content/uploads/2018/03/DevOps-testing-ebook-online.pdf>

Jones, C. (2008). *Applied Software Measurement: Global Analysis of Productivity and Quality*.

Platz, W. (n.d.). *Tricentis*. Retrieved from Tricentis: <https://www.tricentis.com/products/what-is-continuous-testing/>

Schwaber, K., & Sutherland, J. (2016, Julio). *Scrum ORG*. Retrieved from Scrum ORG:

<https://scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-Spanish.pdf>

SCRUM ORG. (2020). *SCRUM.org The home of Scrum*. Retrieved from SCRUM.org The home of Scrum: <https://www.scrum.org/resources/what-is-scrum>

Scrum Study. (n.d.). *Scrum Study global accreditation body*. Retrieved from Scrum Study global accreditation body: <https://www.scrumstudy.com/whyscrum/scrum-organization>

Superintendencia de Banca, Seguros y AFP. (2020, Setiembre). *Portal Superintendencia de Banca, Seguros y AFP*. Retrieved from Portal Superintendencia de Banca, Seguros y

AFP:

https://www.sbs.gob.pe/Portals/0/jer/LEY_GENERAL_SISTEMA_FINANCIERO/2020-09-Ley-26702.pdf