



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

**Automatización del procedimiento de despliegue de
APIs en la plataforma Azure API Management para
una entidad del sector financiero**

TRABAJO DE SUFICIENCIA PROFESIONAL

Para optar el Título Profesional de Ingeniero de Software

AUTOR

Jorge Martin ALAMO CASTILLO

ASESOR

Mg. William Martin ENRÍQUEZ MAGUIÑA

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

Alamo, J. (2022). *Automatización del procedimiento de despliegue de APIs en la plataforma Azure API Management para una entidad del sector financiero*. [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	Jorge Martin Alamo Castillo
Tipo de documento de identidad	DNI
Número de documento de identidad	48003399
URL de ORCID	https://orcid.org/0000-0002-8426-7815
Datos de asesor	
Nombres y apellidos	William Martin Enríquez Maguiña
Tipo de documento de identidad	DNI
Número de documento de identidad	06179457
URL de ORCID	https://orcid.org/0000-0003-1819-191X
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	Raúl Marcelo Armas Calderón
Tipo de documento	DNI
Número de documento de identidad	07156168
Miembro del jurado 1	
Nombres y apellidos	Arturo Alejandro Bartra More
Tipo de documento	DNI
Número de documento de identidad	40233946
Miembro del jurado 2 (Asesor)	
Nombres y apellidos	William Martin Enríquez Maguiña
Tipo de documento	DNI
Número de documento de identidad	06179457
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 21:04 horas del día 25 de agosto del año 2022, se reunieron virtualmente los docentes designados como Miembros de Jurado del Trabajo de Suficiencia Profesional, presidido por el Ing. Armas Calderón Raúl Marcelo (Presidente), Ing. Bartra More Arturo Alejandro (Miembro) y el Mg. Enríquez Maguiña William Martin (Miembro Asesor), usando la plataforma Meet (<https://meet.google.com/hqu-jkha-bft>), para la sustentación virtual del Trabajo de Suficiencia Profesional intitulado: **“AUTOMATIZACIÓN DEL PROCEDIMIENTO DE DESPLIEGUE DE APIS EN LA PLATAFORMA AZURE API MANAGEMENT PARA UNA ENTIDAD DEL SECTOR FINANCIERO”**, por el Bachiller **Alamo Castillo Jorge Martin**; para obtener el Título Profesional de Ingeniero de Software.

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

El 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, el Bachiller obtuvo la nota de **17 (DIECESIETE)**.

A continuación el Presidente de Jurados el Ing. Armas Calderón Raúl Marcelo, declara al Bachiller **Ingeniero de Software**.

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

Presidente

Ing. Armas Calderón Raúl Marcelo

Miembro

Ing. Bartra More Arturo Alejandro

Miembro Asesor

Mg. Enríquez Maguiña William Martin



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
N° 016-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: "Automatización del procedimiento de despliegue de APIs en la plataforma Azure API Management para una Entidad del Sector Financiero"
5. Autores del documento:	Alamo Castillo, Jorge Martin
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	Ocho por ciento (8%)
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:	13/11/2022

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

DEDICATORIA

Este trabajo está dedicado a mis padres y hermana por el apoyo y respaldo brindado para lograr mis objetivos profesionales.

AGRADECIMIENTO

Agradezco a mi asesor Mag. William Martin Enríquez Maguiña por sus consejos y orientación en la elaboración del presente Trabajo de Suficiencia Profesional.

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

Automatización del procedimiento de despliegue de APIs en la plataforma Azure
API Management para una Entidad del Sector Financiero

Autor: Alamo Castillo, Jorge Martin
Asesor: Enríquez Maguiña, William Martin
Título: Trabajo de Suficiencia Profesional para optar el Título Profesional de
Ingeniero de Software
Fecha: Julio de 2022

RESUMEN

El presente Trabajo de Suficiencia Profesional describe la Automatización del procedimiento de despliegue de APIs en la plataforma Azure API Management para una Entidad Financiera, a fin de reducir el tiempo de atención para las solicitudes de despliegue de APIs en el API Management y los posibles errores producto de configuraciones manuales aplicadas incorrectamente. Para solucionar esta problemática, se desarrolló e implementó un pipeline que ejecute las tareas manuales del procedimiento de forma automatizada utilizando herramientas de integración y entrega continua. La gestión del proyecto se realizó utilizando la metodología ágil Kanban, la cual permitió establecer un flujo de trabajo constante y desarrollo evolutivo de la solución. Finalmente, se tiene como resultado un pipeline automatizado que permite reducir el tiempo de espera para el despliegue de APIs en el API Management y minimizar la ejecución de configuraciones manuales en la plataforma.

Palabras claves: Automatización, API Management, Integración continua, Entrega continua, Metodología Kanban

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

**Automation of the API deployment procedure in the Azure API Management
platform for a Financial Sector Entity**

Author: Alamo Castillo, Jorge Martin

Advisor: Enríquez Maguiña, William Martin

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

Date: July 2022

ABSTRACT

This Professional Sufficiency Work describes the Automation of the API deployment procedure in the Azure API Management platform for a Financial Institution, in order to reduce the attention time for API deployment requests in the API Management and the possible errors resulting from incorrectly applied manual configurations. To solve this problem, a pipeline was developed and implemented to execute the manual tasks of the procedure in an automated way using continuous integration and delivery tools. The project was managed using the agile Kanban methodology, which allowed to establish a constant workflow and evolutive development of the solution. Finally, the result is an automated pipeline that reduces the waiting time for API deployment in API Management and minimizes the execution of manual configurations on the platform.

Keywords: Automation, API Management, Continuous Integration, Continuous Delivery, Kanban Methodology.

TABLA DE CONTENIDOS

RESUMEN	v
INTRODUCCIÓN	1
CAPÍTULO I – TRAYECTORIA PROFESIONAL.....	3
CAPÍTULO II – CONTEXTO EN EL QUE SE DESARROLLA LA EXPERIENCIA	11
2.1 EMPRESA – ACTIVIDAD QUE REALIZA.....	11
2.2 VISIÓN	11
2.3 MISIÓN.....	11
2.4 ORGANIZACIÓN DE LA EMPRESA	12
2.5 ÁREA, CARGO Y FUNCIONES DESEMPEÑADAS.....	12
2.6 EXPERIENCIA PROFESIONAL REALIZADA EN LA ORGANIZACIÓN	13
CAPÍTULO III – ACTIVIDADES DESARROLLADAS.....	14
3.1 SITUACIÓN PROBLEMÁTICA	14
3.1.1 <i>DEFINICIÓN DEL PROBLEMA</i>	16
3.2 SOLUCIÓN.....	16
3.2.1 <i>OBJETIVOS</i>	16
3.2.2 <i>ALCANCE</i>	17
3.2.3 <i>ETAPAS Y METODOLOGÍA</i>	18
3.2.4 <i>FUNDAMENTOS UTILIZADOS</i>	24
3.2.5 <i>IMPLEMENTACIÓN DE LAS ÁREAS, PROCESOS, SISTEMAS Y BUENAS PRÁCTICAS</i>	33
3.3 EVALUACIÓN.....	66
3.3.1 <i>EVALUACIÓN COSTO-BENEFICIO</i>	67
3.3.2 <i>INTERPRETACIÓN DEL VAN Y DEL TIR</i>	70
CAPÍTULO IV – REFLEXIÓN CRÍTICA DE LA EXPERIENCIA	72
CAPÍTULO V – CONCLUSIONES Y RECOMENDACIONES	74
5.1 CONCLUSIONES	74

5.2 RECOMENDACIONES	75
5.3 FUENTES DE INFORMACIÓN.....	76
5.4 GLOSARIO.....	78
ANEXOS.....	79
ANEXO 1 – CRONOGRAMA DEL PROYECTO.....	79
ANEXO 2 – COMPARACIÓN ENTRE EL PROCEDIMIENTO MANUAL Y EL PROCEDIMIENTO AUTOMATIZADO	80

ÍNDICE DE TABLAS

Tabla 1 Experiencia Profesional.....	3
Tabla 2 Formación Académica.....	9
Tabla 3 Cursos.....	10
Tabla 4 Certificaciones.....	10
Tabla 5 Idiomas	10
Tabla 6 Roles del Proyecto.....	19
Tabla 7 Etapas del Proyecto	20
Tabla 8 Carriles para la Construcción	21
Tabla 9 Carriles para el Despliegue.....	22
Tabla 10 Requerimientos funcionales del Proyecto	34
Tabla 11 Distribución del Equipo de Proyecto.....	35
Tabla 12 Pipelines para despliegue de APIs en Azure API Management.....	42
Tabla 13 Costo total del proyecto.....	67
Tabla 14 Costo por el tiempo de espera en la atención de solicitudes	68
Tabla 15 Costo por el tiempo para la solución de incidentes	69
Tabla 16 Costo del procedimiento automatizado	69
Tabla 17 Atención de solicitudes de despliegue.....	81
Tabla 18 Atención de incidentes	82

ÍNDICE DE FIGURAS

Figura 1 Organigrama de la Entidad Financiera.....	12
Figura 2 Proceso de Publicación de APIs.....	14
Figura 3 Procedimiento de Despliegue de APIs en Azure API Management	15
Figura 4 Áreas y Equipos beneficiados	18
Figura 5 Kanban workflow para la Construcción.....	21
Figura 6 Kanban workflow para Despliegue	22
Figura 7 Etapas para la Gestión del Proyecto.....	33
Figura 8 Épica del Proyecto.....	34
Figura 9 Historias de usuario para el Proyecto.....	36
Figura 10 Diseño General de la Solución	37
Figura 11 Rama de construcción en repositorio Shared Library	40
Figura 12 Rama de construcción en repositorio Templates	40
Figura 13 Repositorio Project y rama de construcción.....	41
Figura 14 Pipeline de despliegue en API Management de desarrollo	42
Figura 15 Configuración de pipelines de despliegue en desarrollo.....	43
Figura 16 Archivo de despliegue Jenkinsfile de desarrollo.....	44
Figura 17 Método buildContract	45
Figura 18 Plantilla de políticas XML	46
Figura 19 Método buildPolicy	47
Figura 20 ARM template para despliegue de APIs	49
Figura 21 ARM template para despliegue de políticas.....	50
Figura 22 Registro de service principal	51
Figura 23 Método apiDeployment.....	52
Figura 24 Método apiDeletion.....	53
Figura 25 Ticket de cambio	55
Figura 26 Pipeline de entrega en repositorio Shared Library de certificación	56
Figura 27 Pipeline de entrega en repositorio Templates de certificación.....	57
Figura 28 Pipeline de despliegue en API Management de certificación.....	57
Figura 29 Log de pipeline de entrega en repositorio Shared Library de certificación ...	58
Figura 30 Log de pipeline de entrega en repositorio Templates de certificación.....	59
Figura 31 Log de pipeline de despliegue en API Management de certificación	59

Figura 32 API desplegada en API Management de certificación.....	60
Figura 33 Pipeline de entrega en repositorio Shared Library de producción	61
Figura 34 Pipeline de entrega en repositorio Templates de producción.....	62
Figura 35 Pipeline de despliegue en API Management de producción.....	62
Figura 36 Log de pipeline de entrega en repositorio Shared Library de producción	63
Figura 37 Log de pipeline de entrega en repositorio Templates de producción.....	64
Figura 38 Log de pipeline de despliegue en API Management de producción.....	64
Figura 39 API desplegada en API Management de producción.....	65
Figura 40 Ticket de cambio finalizado	66
Figura 41 Relación entre VAN y TIR	71
Figura 42 Cronograma del Proyecto.....	79
Figura 43 Reporte de solicitudes e incidentes	80
Figura 44 Tiempos de ejecución en pipeline de despliegue	83

INTRODUCCIÓN

El presente Trabajo de Suficiencia Profesional describe la automatización de las tareas manuales del procedimiento de despliegue de APIs en la plataforma Azure API Management de una Entidad Financiera utilizando herramientas de integración y entrega continua.

La necesidad de automatizar este procedimiento surgió a raíz de los constantes reclamos por el tiempo que debían esperar los equipos para la atención de las solicitudes de despliegue de sus APIs en el API Management, así como la posibilidad de ocasionar algún problema en la plataforma producto de configuraciones mal ejecutadas por un operario.

Como solución a esta problemática, se implementó un pipeline automatizado que permite reducir el tiempo de espera de los usuarios para el despliegue de sus APIs en el API Management y minimizar el ingreso a la plataforma para la ejecución de tareas manuales.

Para la gestión de este proyecto, se utilizó la metodología ágil Kanban, la cual permitió establecer un flujo de trabajo constante y el desarrollo evolutivo e incremental de la solución utilizando buenas prácticas durante la implementación.

A continuación, se presenta la estructura en la que se encuentra organizado el trabajo:

- En el CAPÍTULO I, se desarrolla la trayectoria profesional del autor, la cual evidencia la experiencia profesional lograda.
- En el CAPÍTULO II, se describen los aspectos relevantes de la entidad en la que se desarrolló la experiencia profesional, tales como misión, visión, organización, cargo, funciones desempeñadas, así como la experiencia del autor trabajando en la entidad.
- En el CAPÍTULO III, se detalla las cada una de las actividades realizadas en el presente trabajo, tales como la problemática, objetivos, alcance, metodología,

solución, implementación y evaluación económica.

- En el CAPÍTULO IV, se describe la reflexión crítica del autor respecto a los desafíos que se presentaron y la experiencia profesional adquirida durante el desarrollo del proyecto.
- En el CAPÍTULO V, se brindan las conclusiones y recomendaciones del autor respecto a la experiencia profesional realizada.

CAPÍTULO I – TRAYECTORIA PROFESIONAL

El autor del presente Trabajo de Suficiencia Profesional es Bachiller en Ingeniería de Software de la Universidad Nacional Mayor de San Marcos. Cuenta con experiencia trabajando en empresas de tecnologías de información, sector financiero y entidades del estado, donde ha participado en proyectos de desarrollo de software, pruebas QA y automatización de procesos. Actualmente, se desempeña en proyectos de análisis, diseño e implementación de software en una de las entidades financieras más grandes del Perú en el área de Infraestructura y Operaciones TI.

En la Tabla 1 se detalla la experiencia profesional del autor:

Tabla 1

Experiencia Profesional

Período	Experiencia Profesional
Abril 2022 – Actualidad	Empresa: Entidad Financiera Cargo: Analista Programador II Funciones: - Crear componentes de software y dar mantenimiento a las aplicaciones de la Entidad Financiera acorde a los requerimientos definidos por el negocio, asegurando un nivel óptimo de disponibilidad. - Efectuar el diseño técnico de las soluciones de software a implementar de acuerdo con los requerimientos definidos, asegurando que cumplan con los estándares tecnológicos, y sean construidas según los lineamientos de seguridad y arquitectura definidos por la Entidad Financiera. - Atender los requerimientos del negocio ponderando brindar la mejor experiencia a los

usuarios finales, desarrollo de componentes de integración robustos y escalables (APIs/microservicios).

- Identificar riesgos potenciales y problemas que pueden afectar la estabilidad operativa de las aplicaciones a cargo, proponiendo alternativas de mitigación y solución.
- Dar seguimiento post-instalación de los requerimientos desplegados en el ambiente de producción.

Logros:

- Desarrollo de microservicio para la exposición del catálogo interno de APIs de la Entidad Financiera.
- Análisis y desarrollo de requerimientos para la aplicación Portal de APIs interno de la Entidad Financiera.

Diciembre 2020 –
Abril 2022

Empresa: Entidad Financiera

Cargo: Analista Programador III

Funciones:

- Elaborar documentos de análisis y diseño para los requerimientos solicitados, detallando las especificaciones necesarias para el despliegue en los ambientes de desarrollo, certificación y producción.
 - Desarrollo de componentes de software acorde a los requerimientos definidos, considerando lo especificado en los documentos de análisis y diseño, aplicando los estándares definidos por la Entidad Financiera.
 - Realizar las pruebas a los componentes de software creados o modificados, participar durante las
-

pruebas de integración con las aplicaciones impactadas según el alcance de la implementación.

- Identificar riesgos potenciales y problemas que pueden afectar la estabilidad operativa de las aplicaciones a cargo, proponiendo alternativas de mitigación y solución.
- Dar seguimiento post-instalación de los requerimientos desplegados en el ambiente de producción.

Logros:

- Programación de automatizaciones para el despliegue de archivos de configuración en componente Cloud y On Premise.
- Implementación de automatizaciones para el despliegue de artefactos en componentes Cloud.
- Implementación de encriptación a credenciales en base de datos de componente API Gateway.

Diciembre 2019 –
Diciembre 2020

Empresa: Entidad Financiera

Cargo: Analista Programador IV

Funciones:

- Validar que los documentos de análisis y diseño contengan las especificaciones necesarias para el despliegue en los ambientes de desarrollo, certificación y producción.
 - Desarrollo de componentes de software acorde a los requerimientos establecidos, considerando lo especificado en los documentos de análisis y diseño, aplicando los estándares definidos por la Entidad Financiera.
 - Realizar las pruebas a los componentes de software creados o modificados, participar durante las
-

	pruebas de integración con las aplicaciones impactadas según el alcance de la implementación. - Dar seguimiento post-instalación de los requerimientos desplegados en el ambiente de producción. Logros: - Diseño e implementación de módulo web para proporcionar data de prueba para las APIs internas de la Entidad Financiera. - Programación de procesos batch para la recolección, tratamiento e inserción en base de datos de la data generada en las consultas realizadas a las APIs internas de la Entidad Financiera . - Programación de automatizaciones para la configuración de parámetros en la Auditoría Funcional de Microservicios. - Instalación de agente para monitoreo en servidores de API Gateway en las suscripciones Cloud y On Premise para realizar labores de monitoreo.
Diciembre 2018 – Septiembre 2019	Entidad del Estado: Ministerio de la Mujer y Poblaciones Vulnerables - Programa Nacional Contra la Violencia Familiar y Sexual (PNCVFS) Cargo: Asistente para la Administración de Plataformas virtuales Funciones: - Desarrollar códigos de programación para añadir nuevas funcionalidades a los sistemas de Registros Administrativos del PNCVFS. - Programar y ejecutar rutinas de migración de datos y exportación de las bases de datos de los Registros Administrativos del PNCVFS.

-
- Realizar la explotación de datos correspondientes a los Registros Administrativos del PNCVFS a través de herramientas tecnológicas.
 - Generar reportes de seguimiento en relación con los datos recolectados en el Sistema de Registros Administrativos de PNCVFS.

Logros:

- Desarrollo e implementación de scripts de programación para automatizar y optimizar los procesos de recojo de datos de los Sistemas de Registros Administrativos de PNCVFS.
- Diseño y programación de módulos de registro e importación de datos para los Sistemas de Registros Administrativos de PNCVFS.
- Propuesta de diseño de interfaz de usuario para brindar un enfoque inclusivo, es decir que permita a los/as hablantes de las lenguas quechua y aimara, así como a las personas con deficiencia auditiva un fácil acceso a la información publicada en el Portal Web Estadístico del PNCVFS.

Julio 2018 –
Noviembre 2018

Entidad del Estado: Ministerio de la Mujer y Poblaciones Vulnerables – Programa Nacional contra la Violencia Familiar y Sexual (PNCVFS)

Cargo: Analista Estadístico de Base de Datos

Funciones:

- Desarrollar códigos de programación para evitar errores y omisiones en la recolección de datos de los Registros Administrativos del PNCVFS.
 - Documentar los sistemas de información de la entidad, considerando incluir descripción, ubicación de funciones o rutinas y procesos de
-

preparación de datos.

- Programar y ejecutar rutinas de migración de datos de los Registros Administrativos del PNCVFS.
- Elaborar informes de análisis estadístico correspondiente a los datos recolectados de los Registros Administrativos del PNCVFS.

Logros:

- Mantenimiento y adición de nuevas funcionalidades requeridas en el sistema de Registros Administrativos del PNCVFS, específicamente al módulo de Acciones Preventivas Promocionales.

Mayo 2018 –
Julio 2018

Empresa: Hiper S.A.

Cargo: Tester QA

Funciones:

- Ejecutar los casos de prueba correspondientes con la finalidad de asegurar la calidad de los productos de software de la empresa.
- Definir casos de prueba considerando los requerimientos funcionales, no funcionales y técnicos definidos por el negocio.
- Documentar de forma detallada los casos de pruebas.
- Registrar los incidentes identificados al ejecutar los casos de prueba, así como dar seguimiento y asegurar una adecuada corrección de los defectos encontrados.

Logros:

- Detectar fallos en la programación de versiones estables y certificadas previamente por QA de productos de la empresa.

Enero 2018 –	Empresa: Hiper S.A.
Abril 2018	Cargo: Programador Java Junior
	Funciones:
	- Ejecutar el mantenimiento y programación de componentes de software correspondientes a los proyectos y productos de la empresa. - Identificar problemas en los procesos de TI y analizar las causas relacionadas. - Analizar los incidentes que se presenten en los aplicativos informáticos. - Realizar seguimiento a los despliegues en entornos de certificación y producción.
	Logros:
	- Desarrollo de software para la simulación de ventanilla en la gestión de la red de atención a clientes con la finalidad de servir como muestra para nuevos clientes de la empresa.

Nota. Elaboración propia.

En la Tabla 2 se detalla la formación académica del autor:

Tabla 2

Formación Académica

Período	Formación Académica
2013 – 2017	Grado Académico de Bachiller en Ingeniería de Software Escuela Académico Profesional de Ingeniería de Software Facultad de Ingeniería de Sistemas e Informática Universidad Nacional Mayor de San Marcos

Nota. Elaboración propia.

En la Tabla 3 se detallan los cursos realizados por el autor:

Tabla 3

Cursos

Período	Curso	Institución
2021	AZ-104 Microsoft Azure Administrator	New Horizons
2021	AZ-900 MS Azure Fundamentals	New Horizons
2019	Android 8.0 Mobile Developer	Cibertec
2018	Big Data	Cibertec
2014	Herramientas Informáticas	Universidad Nacional Mayor de San Marcos

Nota. Elaboración propia.

En la Tabla 4 se detallan las certificaciones obtenidas por el autor:

Tabla 4

Certificaciones

Período	Certificación	Entidad
2021	Microsoft Certified: Azure Fundamentals	Microsoft
2022	Microsoft Certified: Azure Data Fundamentals	Microsoft

Nota. Elaboración propia.

En la Tabla 5 se detallan los idiomas conocidos por el autor:

Tabla 5

Idiomas

Período	Idioma	Nivel	Institución
2015 – 2018	Inglés	Avanzado	ICPNA

Nota. Elaboración propia.

CAPÍTULO II – CONTEXTO EN EL QUE SE DESARROLLA LA EXPERIENCIA

2.1 EMPRESA – ACTIVIDAD QUE REALIZA

La empresa es una sólida institución creada con capitales privados que forma parte del sector financiero peruano, la cual cuenta con una amplia trayectoria de más de 120 años que la ha posicionado como una de las más importantes del país e incluso se cotiza en la Bolsa de Valores de Nueva York. Su crecimiento continuo y sostenible se basa entre otros aspectos al respeto de la institucionalidad, a la búsqueda de la estandarización de sus procesos, priorizar las acciones que permitan el crecimiento de sus clientes, así como lograr estándares internacionales de responsabilidad social.

Brinda productos y servicios financieros, tanto a empresas como a personas naturales, a través de sus divisiones de Banca Corporativa, Banca Empresas y Banca Minorista, teniendo un alcance geográfico que abarca a todas las regiones del País. En los últimos años para mantenerse acorde con los avances tecnológicos, la empresa ha invertido en potenciar su área de investigación de TI, la misma que se encarga de impulsar innovaciones tecnológicas de acuerdo con las necesidades del mercado.

2.2 VISIÓN

La visión de la empresa está en relación con posicionarse como líder en el sector financiero del país con productos y servicios innovadores en el sector.

2.3 MISIÓN

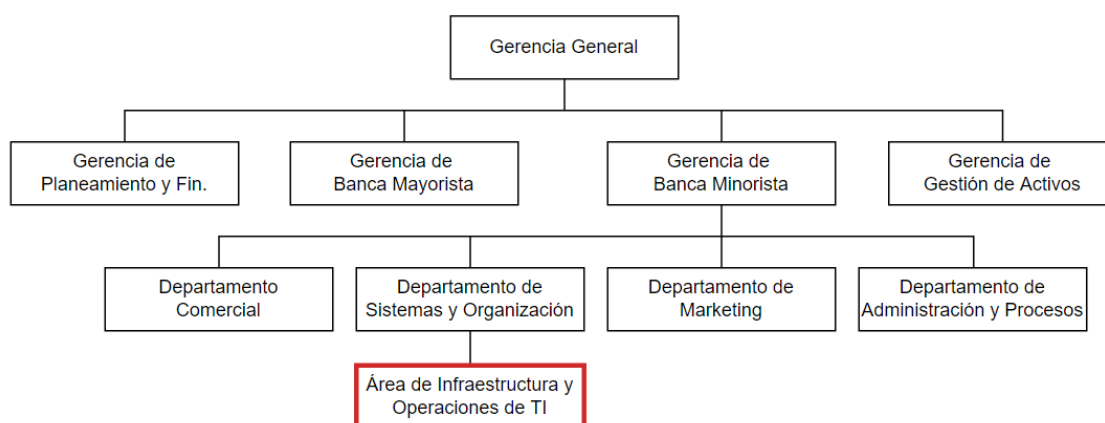
La misión de la empresa está en relación con lograr la satisfacción de sus clientes, brindándoles un soporte financiero personalizado que permita el logro de sus metas. Asimismo, busca generar valor impulsando el desarrollo personal y profesional de sus colaboradores.

2.4 ORGANIZACIÓN DE LA EMPRESA

El organigrama presentado en la Figura 1 ha sido enfocado en el área de TI en la cual se ha desarrollado el presente Trabajo de Suficiencia Profesional. La experiencia se desarrolló en el Área de Infraestructura y Operaciones de TI en el Departamento de Sistemas y Organización.

Figura 1

Organigrama de la Entidad Financiera



Nota. La figura representa la estructura organizacional de la empresa. Tomado de la Entidad Financiera, 2019.

2.5 ÁREA, CARGO Y FUNCIONES DESEMPEÑADAS

El autor del presente Trabajo de Suficiencia Profesional, durante el tiempo de implementación del proyecto, se desempeñaba en el cargo de Analista Programador en el Área de Infraestructura y Operaciones de TI. Las principales funciones que se desempeñaron fueron:

- Elaborar documentos de análisis y diseño para los requerimientos solicitados, detallando las especificaciones necesarias para el despliegue en los ambientes de desarrollo, certificación y producción.
- Desarrollar componentes de software acorde a los requerimientos establecidos, considerando lo especificado en los documentos de análisis y diseño, aplicando

los estándares definidos por la organización.

- Implementar la automatización de procesos según los requerimientos establecidos, asegurando que la entrega cumpla con los estándares tecnológicos, lineamientos de seguridad y buenas prácticas definidas por la organización.
- Realizar las pruebas a los componentes de software creados o modificados, participando durante las pruebas de integración con las aplicaciones impactadas según el alcance de la implementación.
- Identificar riesgos potenciales y problemas que pueden afectar la estabilidad de las aplicaciones a cargo, proponiendo alternativas de mitigación y solución.
- Dar seguimiento post-instalación de los requerimientos desplegados en el ambiente de producción.

2.6 EXPERIENCIA PROFESIONAL REALIZADA EN LA ORGANIZACIÓN

Durante la experiencia profesional del autor como colaborador en la Entidad Financiera, ha participado en diversas iniciativas tecnológicas, siendo las más resaltantes:

- Implementación de módulo en plataforma web para almacenar data de prueba para el consumo de APIs.
- Programación de procesos batch para la recolección, tratamiento y carga de datos de auditoría de microservicios en repositorio de data de prueba.
- Programación de pipeline para la configuración de parámetros para auditoría de microservicios.
- Instalación de agente de monitoreo en servidores Cloud y On Premise.
- Programación de pipeline para el despliegue de archivos de configuración en componentes Cloud y On Premise.
- Programación de pipeline para el despliegue de APIs en API Management Cloud.
- Encriptación de credenciales a nivel de base de datos.
- Desarrollo de microservicio para la consulta del catálogo interno de APIs.

CAPÍTULO III – ACTIVIDADES DESARROLLADAS

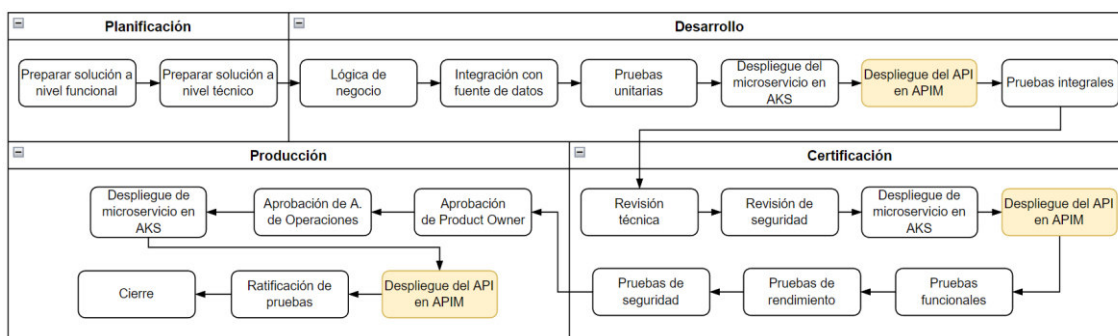
3.1 SITUACIÓN PROBLEMÁTICA

La Entidad Financiera, en la cual se ha desarrollado el presente Trabajo de Suficiencia Profesional, al ser una de las más importantes del Perú en su sector, cuenta con una alta demanda para la publicación de APIs que se integren a las diferentes aplicaciones y servicios que ofrece.

El proceso que maneja la organización para la publicación de APIs, específicamente APIs de tipo REST, considera como parte de los procedimientos a realizar, el despliegue del API en su plataforma Azure API Management, lo cual permite unificar la publicación de APIs a través de un único punto de entrada hacia los servicios backend para que sean consumidos por otras aplicaciones y equipos. En la Figura 2 se muestra el flujo trabajo básico de este proceso.

Figura 2

Proceso de Publicación de APIs

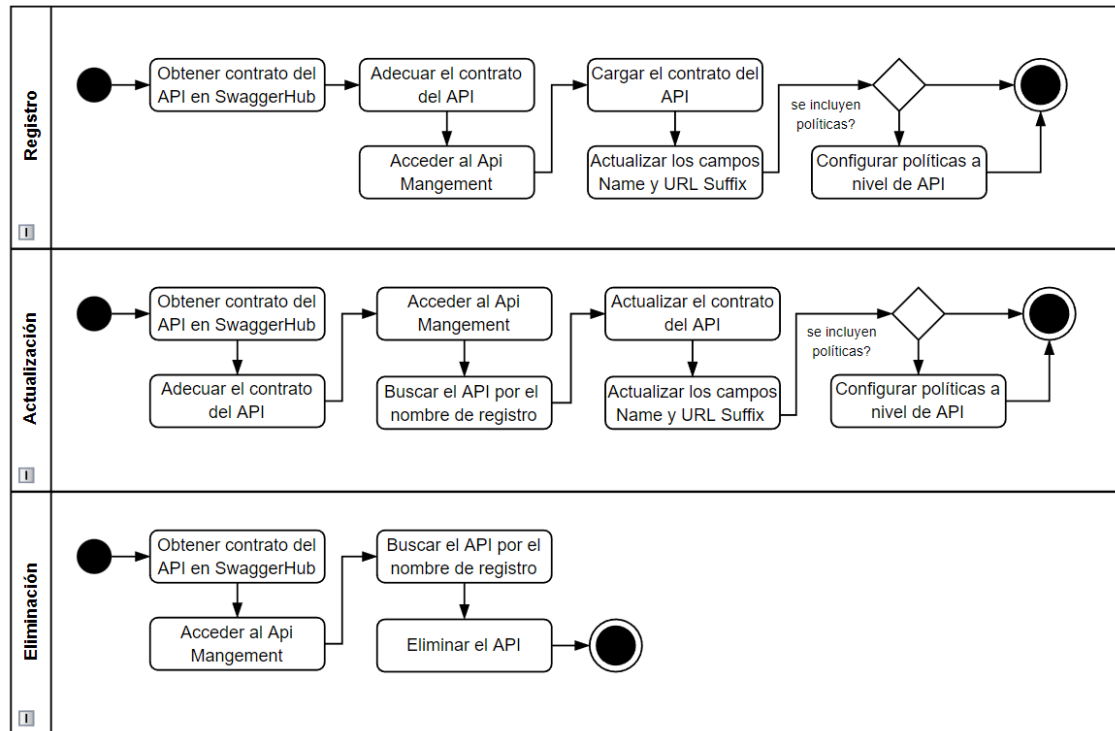


Nota. La figura representa el flujo de base del Proceso de Publicación de APIs. Tomado de la Entidad Financiera, 2021.

De lo mostrado en la Figura 2, el procedimiento para el despliegue de APIs en Azure API Management es parte importante del proceso de creación de APIs en la Entidad Financiera. Las actividades que forman parte de este procedimiento se describen en la Figura 3.

Figura 3

Procedimiento de Despliegue de APIs en Azure API Management



Nota. La figura representa el flujo de actividades del Procedimiento de Despliegue de APIs en Azure API Management. Tomado de la Entidad Financiera, 2021.

El procedimiento era realizado en el ambiente de desarrollo por el equipo de API Platform, responsable de la administración, soporte y planificación de iniciativas para la plataforma Azure API Management, y por el Área de Operaciones en los ambientes de certificación y producción. En ambos casos, el personal designado accedía directamente a la plataforma para ejecutar las configuraciones manualmente.

Este esquema de trabajo generaba retrasos en las actividades de los equipos de desarrollo de APIs debido a que el equipo de API Platform priorizaba la atención de actividades críticas que podían comprometer la estabilidad de sus servicios, mientras que postergaba la atención de las solicitudes para el registro, actualización y eliminación de APIs en el API Management. Bajo este escenario, el proceso de publicación de APIs se veía obstaculizado para los equipos de desarrollo que no podían finalizar sus actividades

en el ambiente de desarrollo y, en consecuencia, se retrasaba el inicio de sus actividades en los ambientes de certificación y producción respectivamente.

Previo al inicio del proyecto, se contaba con una automatización para este procedimiento, sin embargo, no era utilizada ya que no satisfacía las necesidades de la Entidad Financiera. La herramienta únicamente permitía realizar el registro de APIs, pero no cubría las funcionalidades de actualización y eliminación de APIs, las mismas que son requeridas en la organización para la publicación de automatizaciones a fin de que cuenten con la capacidad de revertir los cambios efectuados en caso de alguna contingencia.

3.1.1 DEFINICIÓN DEL PROBLEMA

La generación de retrasos en los compromisos pactados por los equipos de desarrollo para la publicación de APIs y posibles errores producto de configuraciones realizadas incorrectamente en la plataforma Azure API Management debido a la práctica de actividades manuales.

3.2 SOLUCIÓN

Como solución se planteó la automatización de las tareas manuales en el procedimiento de despliegue de APIs en la plataforma Azure API Management.

3.2.1 OBJETIVOS

3.2.1.1 Objetivo General.

Aplicar herramientas de integración y entrega continua en la automatización de tareas manuales en el procedimiento de despliegue de APIs en la plataforma Azure API Management de la Entidad Financiera para reducir los tiempos de atención del procedimiento y disminuir la exposición de la plataforma a configuraciones manuales.

3.2.1.2 *Objetivos específicos.*

A partir del objetivo general, se tienen los siguientes objetivos específicos:

- Entender el procedimiento manual realizado para el despliegue de APIs en Azure API Management.
- Definir y planificar las tareas a partir del análisis de los requerimientos funcionales de la solución.
- Definir el diseño de la solución y las herramientas a utilizar.
- Orquestar y automatizar la ejecución de tareas utilizando pipelines de integración y entrega continua en Jenkins.
- Aplicar los lineamientos y buenas prácticas para la implementación de automatizaciones definidos por la Entidad Financiera.

3.2.2 *ALCANCE*

3.2.2.1 *Alcance Funcional.*

El alcance funcional del proyecto consideró la automatización de las tareas manuales de registro, actualización y eliminación del procedimiento de despliegue de APIs en la plataforma Azure API Management.

3.2.2.2 *Alcance Organizacional.*

A nivel organizacional, este proyecto benefició a los siguientes equipos y áreas:

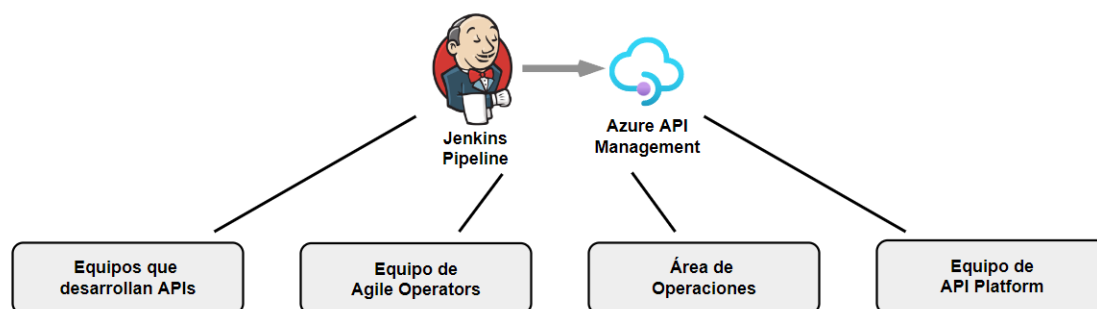
- Equipos que desarrollan APIs: usan el pipeline para desplegar sus APIs en Azure API Management a demanda en el ambiente de desarrollo.
- Equipo de API Platform: el uso del pipeline reduce las solicitudes de soporte relacionadas al despliegue de APIs en Azure API Management.
- Equipo de Agile Operators: usan el pipeline para desplegar APIs en Azure API Management en los ambientes de certificación y producción de acuerdo con lo solicitado.

- Área de Operaciones: validan que las actividades a ejecutar no comprometan la estabilidad de los componentes, el uso del pipeline reduce la exposición de la plataforma Azure API Management a errores producto de configuraciones manuales incorrectas.

En la Figura 4 se representan las áreas y equipos beneficiados directa e indirectamente de la solución.

Figura 4

Áreas y Equipos beneficiados



Nota. Elaboración propia.

3.2.3 ETAPAS Y METODOLOGÍA

La metodología que se utilizó para el desarrollo del presente proyecto fue la metodología ágil Kanban. Esta metodología fue seleccionada por el Área de Infraestructura y Operaciones TI como estándar para la construcción y despliegue de automatizaciones.

3.2.3.1 Roles.

Los roles que intervinieron en las diferentes etapas durante el desarrollo del proyecto son descritos en la Tabla 6.

Tabla 6

Roles del Proyecto

Rol	Descripción
Team Member	Miembros del equipo encargados del análisis, diseño, construcción y despliegue de los requerimientos según su especialidad.
Líder Técnico	Team Member que supervisa el trabajo del resto de team members y brinda apoyo como referente técnico del equipo.
Product Owner	Encargado de determinar las iniciativas en las que trabajará el equipo, así como definir los requerimientos del producto y brindar la conformidad de los entregables.
Analista de QA	Team Member encargado de realizar las pruebas y validaciones de las soluciones en los entornos de desarrollo, certificación y producción.
Analista de Seguridad	Parte del personal del Área de Infraestructura y Operaciones de TI, encargado de validar e identificar riesgos de seguridad para las soluciones y procedimientos de despliegue en los ambientes de certificación y producción.
Agile Operator	Personal tercerizado del Área de Operaciones encargado de ejecutar los manuales de despliegue de las soluciones en los ambientes de certificación y producción.

Nota. Elaboración propia. Basado en el framework de gestión de proyectos de la Entidad Financiera, 2021.

3.2.3.2 Etapas del Proyecto.

Las etapas que fueron requeridas en la realización del proyecto son descritas en la Tabla 7.

Tabla 7

Etapas del Proyecto

Etapas	Descripción
Planificación	Se establecieron los requerimientos funcionales para el pipeline automatizado y se seleccionaron a los integrantes que participaron del proyecto.
Análisis	Se analizaron los requerimientos funcionales y se definieron las historias de usuario que conformaron el backlog del proyecto.
Diseño	Se realizó el diseño de la solución de acuerdo con los lineamientos definidos por la Entidad Financiera.
Construcción	Se realizó el desarrollo de la solución siguiendo las buenas prácticas definidas por la Entidad Financiera.
Despliegue	Se gestionó el despliegue en los ambientes de certificación y producción de la solución, coordinando la ejecución de las pruebas de calidad en cada entorno.

Nota. Elaboración propia. Basado en el framework de gestión de proyectos de la Entidad Financiera, 2021.

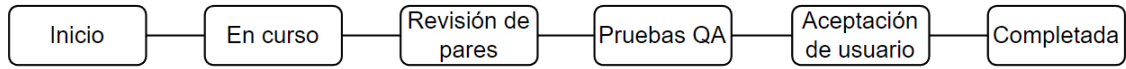
3.2.3.3 Uso de la Metodología.

La metodología Kanban es una metodología ágil utilizada ampliamente en proyectos de TI. Para este proyecto, la organización de las tareas se definió a través de flujos de Kanban utilizando la herramienta de administración de tareas Jira.

Para el desarrollo de las historias de usuario en la etapa de construcción, se siguió el flujo de Kanban mostrado en la Figura 5.

Figura 5

Kanban workflow para la Construcción



Nota. Elaboración propia. Basado en el tablero Kanban para tareas de construcción en la herramienta Jira interna de la Entidad Financiera, 2021.

En la Tabla 8 se describen los carriles del workflow que siguieron de las historias de usuario en la etapa de construcción.

Tabla 8

Carriles para la Construcción

Carril	Descripción
Inicio	Carril inicial con el que se crea la historia de usuario en Jira.
En curso	En este carril se inicia el desarrollo de la historia de usuario.
Revisión de pares	En este carril se realiza la revisión de pares del entregable generado en la historia de usuario por parte de otro miembro del equipo, a fin de asegurar que se siguieron las buenas prácticas durante el desarrollo.
Pruebas QA	En este carril se realizan las pruebas y validaciones de calidad por parte del Analista de QA. Dependiendo del tipo de historia de usuario se admite la validación de evidencia para dar la aprobación. En caso de existir observaciones, la historia retorna al estado En curso para la corrección.
Aceptación de usuario	En este carril el Product Owner valida el entregable generado. En caso de no ser conforme, la historia retorna al estado En curso o Pruebas QA para la corrección de acuerdo con la necesidad.

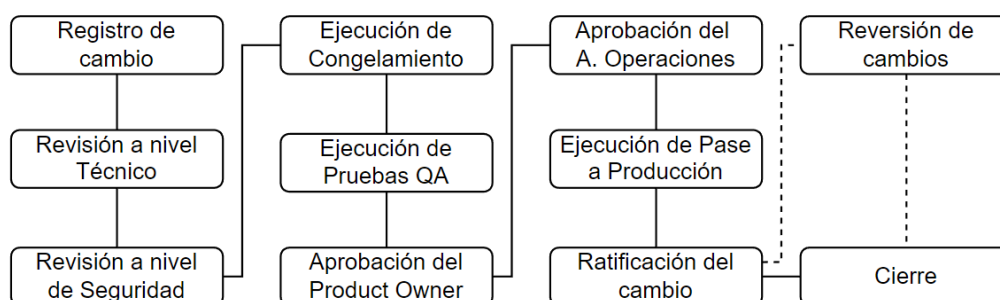
Completada	Carril que marca la finalización de la historia de usuario.
------------	---

Nota. Elaboración propia. Basado en el tablero Kanban de tareas de construcción en la herramienta Jira interna de la Entidad Financiera, 2021.

Para el desarrollo de las historias de usuario en la etapa de despliegue, se siguió el flujo de Kanban mostrado en la Figura 6.

Figura 6

Kanban workflow para Despliegue



Nota. Elaboración propia. Basado en el tablero Kanban de pase a producción en la herramienta Jira interna de la Entidad Financiera, 2021.

En la Tabla 9 se describen los carriles del workflow que siguieron de las historias de usuario en la etapa de despliegue.

Tabla 9

Carriles para el Despliegue

Carril	Descripción
Registro de cambio	Carril inicial asignado al crear el ticket de cambio en Jira. Se debe documentar en el ticket los pasos a ejecutar para el despliegue del cambio en los ambientes de certificación y producción, así como el procedimiento de reversión en caso de contingencia.

Revisión a nivel Técnico	El Líder Técnico realiza la validación de los cambios a desplegar para asegurar que cumpla los lineamientos y buenas prácticas de desarrollo definidas por la organización. De no obtener conformidad, se retorna el ticket a Registro de cambio.
Revisión a nivel de Seguridad	El Analista de Seguridad realiza la validación de los cambios a desplegar para asegurar que cumplan con los lineamientos de seguridad definidos por la organización. De no obtener conformidad, se retorna el ticket a Registro de cambio.
Ejecución de Congelamiento	El Agile Operator asignado ejecuta los pasos indicados en el ticket para realizar el despliegue de cambios en el ambiente de certificación. De producirse algún error al ejecutar los pasos, se retorna el ticket a Registro de cambio.
Ejecución de Pruebas QA	El Analista de QA realiza las validaciones necesarias para asegurar que los cambios en el ambiente de certificación fueron desplegados correctamente. De haber observaciones, se retorna el ticket a Registro de cambio.
Aprobación del Product Owner	El Product Owner brinda su conformidad para que los cambios validados en certificación sean desplegados en el ambiente de producción. De no ser conforme, se retorna el ticket a Registro de cambio.
Aprobación del Área de Operaciones	El Líder Técnico adjunta la conformidad del Área de Operaciones para el despliegue de los cambios en el ambiente de producción. Posteriormente, registra la fecha y hora del pase a producción.
Ejecución de Pase a Producción	El Agile Operator asignado ejecuta los pasos indicados en el ticket para realizar el despliegue de cambios en el ambiente de producción.
Ratificación del cambio	El Analista de QA realiza las validaciones necesarias para asegurar que los cambios en el ambiente de

	certificación fueron desplegados correctamente. En caso de haber observaciones, se envía el ticket a Reversión de cambios, caso contrario se da cierre al ticket de cambio.
Reversión de cambios	Este carril es asignado únicamente cuando no se da conformidad a la ratificación del pase. El Agile Operator ejecuta los pasos indicados en el ticket para realizar la reversión de los cambios en el ambiente de producción.
Cierre	Carril que marca la finalización del flujo de cambios, considerando si se dio el despliegue correctamente en el ambiente de producción o si se revirtió el cambio.

Nota. Elaboración propia. Basado en el tablero Kanban de pase a producción en la herramienta Jira interna de la Entidad Financiera, 2021.

3.2.4 FUNDAMENTOS UTILIZADOS

3.2.4.1 Metodología Kanban.

La Metodología Kanban tiene su origen en el Sistema de Producción de Toyota (TPS) y en el último tiempo se ha convertido en una de las metodologías ágiles más utilizadas para el desarrollo de software. Esta metodología propone un flujo de trabajo visual para gestionar la realización de tareas pendientes haciendo un uso óptimo de los recursos disponibles y a diferencia de otras metodologías ágiles, Kanban cuenta con mayor flexibilidad ya que no impone roles o equipos específicos a la organización. (Subra y Vannieuwenhuyse, 2018).

Subra y Vannieuwenhuyse (2018) señalan adicionalmente que la metodología Kanban aplicada al desarrollo de software se puede resumir en las siguientes prácticas:

- Visualizar el workflow: se organiza el trabajo en pequeñas tareas manejables que se describen en tarjetas (historias de usuario), las cuales se muestra en un tablero dividido por columnas que representan cada una de las etapas del workflow que

deben seguir las tarjetas para su realización (tablero Kanban).

- Limitar el trabajo actual: se establece un límite a la cantidad de historias de usuario a trabajar en cada etapa del workflow. Si alguna etapa alcanza el límite, los responsables pasan a apoyar al resto del equipo y no reciben historias nuevas hasta que se restaure el flujo de trabajo.
- Medir el lead time y cycle time: se mide el tiempo promedio en resolver una historia desde que ingresó a la cola de atención (lead time) y se mide el tiempo promedio en resolver una historia desde el momento que es tomada e iniciada (cycle time), esto a fin de buscar formas de optimizar los procesos.

3.2.4.2 *Jira.*

Jira es una solución para la administración de proyectos de todo tipo con especial énfasis en los que están relacionados con el desarrollo de software. Combina el soporte para las metodologías ágiles Scrum y Kanban con el manejo de flujos de trabajo flexibles para ofrecer a los equipos la experiencia de administrar sus proyectos aplicando agilidad. (Li, 2018).

ATLASSIAN (2022a) señala que para los equipos que utilizan metodologías ágiles , Jira facilita el uso de tableros de Scrum y Kanban de acuerdo con las necesidades del proyecto. Estos tableros permiten visualizar el estado del trabajo que está realizando el equipo y centralizar la administración de las tareas del proyecto a través de flujos de trabajo personalizables,

Li (2018) señala de forma complementaria que las tareas en Jira son denominadas issues y puede ser de diferentes tipos:

- Épica: es un issue que representa una gran cantidad de trabajo que puede ser dividido en múltiples issues de tipo historia más pequeños y manejables.
- Historia: es un issue que representa un requerimiento único y breve, generalmente se utilizan para realizar la captura de requerimientos desde el punto de vista del usuario final.

- Error: es un issue que representa un problema que necesita ser revisado y corregido oportunamente.
- Tarea: es un issue que representa una actividad genérica que no entra en la categoría de historia o error, pero implica la realización de un cierto nivel de esfuerzo para ser completada.

3.2.4.3 *Azure CLI.*

Azure Command-Line Interface (CLI) es un software utilitario que permite comunicarse de forma remota con la nube de Azure a través de una interfaz en línea de comandos y ejecutar instrucciones administrativas sobre los recursos de Azure por medio de comandos o scripts automatizados. (MICROSOFT, 2022a).

3.2.4.4 *ARM templates.*

Azure Resource Manager Templates (ARM Templates) son plantillas que implementan la práctica de infraestructura como código, es decir, permiten definir la infraestructura para una determinada solución basada en Azure a través de un archivo en formato Json donde se especifican los recursos, propiedades y configuraciones requeridas. Bajo este esquema, se implementa la plantilla por medio de un solo comando, sin la necesidad de tener que ejecutar una serie de sentencias en consola para la realizar la implementación. (MICROSOFT, 2022b).

MICROSOFT (2022b) menciona adicionalmente que las plantillas ARM pueden ser incorporados a herramientas basadas en integración y entrega continua, ofreciendo la posibilidad de automatizar tareas de configuración y gestión de recursos en Azure de forma ágil y segura.

3.2.4.5 *Azure Container Registry.*

Azure Container Registry (ACR) es un servicio de alojamiento privado y gestión de imágenes de contenedores basados en la tecnología Docker, la cual permite realizar compilación y despliegue de aplicaciones contenerizadas de forma segura utilizando los comandos estandarizados de Docker. (MICROSOFT, 2022c).

3.2.4.6 Azure API Management.

Azure API Management es una plataforma orientada a la administración de APIs, la cual permite exponer las APIs a través de una interfaz de entrada segura hacia los servicios backend, así como configurar fácilmente aspectos como la seguridad, monitoreo y autenticación sin la necesidad de modificar el código fuente del API. (Barbieri y Bonanni, 2019).

Según Budzynski (2019), el uso de la plataforma Azure API Management supone para la organización los siguientes beneficios:

- Se abstrae el acceso a las APIs de sus respectivos servicios de backend internos y se exponen a través del dominio del API Management.
- Al no se exponen directamente los servicios backend internos de las APIs registradas en el API Management, se reduce el riesgo de posibles ataques.
- El API Management permite configurar la autenticación de las solicitudes entrantes a las APIs y definir filtros para limitar el tráfico únicamente a direcciones IP confiables.
- Se pueden configurar políticas a nivel global, por producto, por API y por operación, las cuales son aplicadas a las solicitudes entrantes, así como a las respuestas salientes que pasen por el API Management.
- El API Management proporciona métricas del consumo de las APIs registradas permitiendo monitorear el tráfico de entrada y salida, así como facilitar las tareas de debugging.

3.2.4.7 Políticas en Azure API Management.

Las políticas son archivos en formato XML que especifican una serie de sentencias declarativas para aplicar cambios sobre las solicitudes de entrada y respuestas de salida. Estas políticas se pueden configurar en el API Management a nivel global (aplica a todas las APIs), de producto, de API y de Operación dependiendo de la necesidad. (MICROSOFT, 2022d).

MICROSOFT (2022d) señala de manera complementaria que las declaraciones XML que conforman la política se dividen en 4 secciones diferenciadas:

- Inbound: en esta sección se declaran las políticas a aplicar a las solicitudes entrantes.
- Backend: en esta sección se declaran las políticas a aplicar antes que las solicitudes de entrada sean enviadas al backend.
- Outbound: en esta sección se declaran las políticas a aplicar a las respuestas salientes.
- On-error: en esta sección se declaran las políticas a aplicar en caso de producirse algún error.

3.2.4.8 SwaggerHub.

SwaggerHub es una plataforma de trabajo colaborativo para el diseño, construcción, documentación e implementación de APIs basadas en la especificación OpenAPI, la cual ofrece una serie de capacidades para facilitar la administración del API durante todo su ciclo de vida. (SMARTBEAR, 2022).

SMARTBEAR (2022) menciona adicionalmente que la plataforma SwaggerHub ofrece las siguientes funcionalidades:

- Diseñar los contratos de las APIs de forma colaborativa.
- Almacenar los contratos de las APIs en repositorio centralizado
- Gestionar el versionamiento de las APIs
- Compartir las APIs de manera pública o privada.

3.2.4.9 Jenkins.

Jenkins es un servidor de automatización de fuente abierta basado en java, el cual cuenta con una amplia gama de plugins que le permiten integrarse con diferentes herramientas y servicios a fin de automatizar las tareas presentes en las etapas de desarrollo, pruebas y despliegue que forman parte del ciclo de vida de una aplicación de

software. Dispone de soporte con Docker para facilitar su adopción en múltiples entornos de forma rápida y sencilla. (Patil y Soni, 2021).

3.2.4.10 Bitbucket.

Bitbucket es una herramienta para la administración de repositorios basada en Git como sistema versionamiento, la cual proporciona un entorno seguro para centralizar el almacenamiento de código fuente y trabajar de forma colaborativa a través del flujo de desarrollo de una solución de software. Ofrece la posibilidad de integrarse con otras herramientas para favorecer la creación de código de calidad a través de la ejecución de pruebas automatizadas. (ATLASSIAN, 2022b).

3.2.4.11 Docker.

Docker es una plataforma de fuente abierta para implementar y ejecutar aplicaciones sobre un entorno ligero que cuenta con todas las configuraciones y librerías necesarias para su ejecución, el cual es denominado contenedor. El uso de contenedores desacopla la aplicación de la infraestructura permitiendo que sea desplegada en diferentes entornos sin la necesidad de preocuparse por realizar instalaciones previas, todo lo necesario se encuentra en el contenedor. (DOCKER, INC., 2022).

DOCKER, INC. (2022) señala adicionalmente los siguientes conceptos claves respecto a Docker:

- Registry: es un repositorio para el almacenamiento de imágenes Docker, el cual puede ser público o privado.
- Dockerfile: es un archivo con las definiciones y pasos a realizar para la creación y ejecución de una imagen.
- Imagen: las imágenes son plantillas que contienen las especificaciones necesarias para la creación de un contenedor. Se pueden crear imágenes personalizadas a partir de un Dockerfile o reutilizar alguna que se encuentre publicada en un registry.
- Contenedor: es una instancia en ejecución definida a partir de una imagen Docker.

3.2.4.12 Artifactory.

Artifactory es una herramienta orientada a la administración de repositorios para el versionamiento de archivos binarios y artefactos, la cual proporciona un entorno seguro para el almacenamiento de los productos de compilación de software siendo compatible con los principales formatos de paquetes utilizados en la industria de desarrollo de software. (JFROG LTD., 2022).

3.2.4.13 Groovy.

Groovy es un lenguaje de programación implementado sobre la plataforma Java, el cual cuenta con soporte para programación orientada y una sintaxis muy parecida a la que se maneja programando en Java, pero con reglas de codificación menos estrictas como el tipado dinámico, inferencia de tipos, entre otras. (Davis, 2019).

3.2.4.14 API.

Application Programming Interface (API) es un mecanismo conformado por una serie de protocolos y especificaciones utilizado para el diseño e integración de aplicaciones basadas en software. Las APIs permiten la comunicación entre sistemas sin importar la forma en la que están implementados, en ese sentido las API representan un contrato de mutuo acuerdo entre las partes, si una parte lanza una solicitud bajo una estructura en específico, esta estructura define cómo debe responder la otra parte. (RED HAT, INC., 2017).

IBM (2021) señala que una API REST es una API que se encuentra alineada con los 6 principios de diseño definidos por la arquitectura REST:

- Interfaz uniforme: las solicitudes al API dirigidas a un mismo recurso deben ser iguales, sin importar su procedencia.
- Separación cliente-servidor: el diseño de API REST implica que las aplicaciones con los roles de cliente y servidor sean totalmente independientes entre sí, no deben interactuar de ninguna forma más allá de entregar los datos solicitados.
- Sin estado: cada una de las solicitudes al API deben contener toda la información

necesaria para su procesamiento, el servidor no almacena datos que guarden relación con la solicitud entrante.

- Almacenamiento en caché: los recursos deben tener la posibilidad de ser almacenado en memoria caché de lado del cliente o del servidor con la finalidad de mejorar las interacciones.
- Sistema basado en capas: el diseño de un API REST las llamadas y respuestas deben pasar por diferentes capas, de tal manera que el cliente no pueda percibir si se comunica directamente con el servidor o con un intermediario.
- Código a demanda: se debe poder enviar recursos ejecutables desde el servidor hacia el cliente bajo demanda.

3.2.4.15 Shared Library.

Shared Library o librería compartida es un conjunto de recursos de programación disponibles para ser utilizados por diferentes aplicativos sin la necesidad de tener que implementarlos nuevamente. Bajo este esquema si se actualizan las fuentes de la librería compartida, todas las aplicaciones que la están referenciando toman los cambios aplicados inmediatamente. (IBM, 2022).

3.2.4.16 CI/CD.

La CI/CD consiste en una serie de métodos para la distribución de aplicaciones de forma periódica a través de la automatización del ciclo de vida de las aplicaciones de software. Este esquema busca solucionar los problemas que implica integrar código nuevo a una aplicación de forma ágil. (RED HAT, INC., 2018).

A. Integración Continua.

Continuous Integration (CI) o Integración Continua es la práctica a través de la cual los miembros de un equipo de desarrollo integran periódicamente los cambios en su código a un repositorio centralizado, tras lo cual se realiza la compilación y ejecución de pruebas unitarias automáticamente. Se tiene como objetivo identificar errores de forma oportuna, incrementar la calidad del software y reducir los tiempos para la publicación y validación de actualizaciones en el software. (AWS, INC., 2022a).

B. Entrega Continua.

Continuous Delivery (CD) o Entrega Continua es la práctica que extiende a la integración continua a través de la cual se realiza la compilación y ejecución pruebas automáticamente (de carga, de integración, etc.) de los cambios en el código sobre un ambiente pre productivo con el objetivo de tener un artefacto listo para ser publicado en producción en cualquier momento. El paso final para actualizar los cambios en el ambiente productivo requiere ejecución manual a fin de poder controlar en qué momento se realiza el despliegue, a diferencia de la implementación continua que realiza la publicación en producción automáticamente. (AWS, INC., 2022b).

3.2.4.17 Service Principal.

Un Service Principal representa una entidad de seguridad en Azure Active Directory en la cual se definen perfiles para la autorización y autenticación sobre un determinado recurso en la nube de Azure. (MICROSOFT, 2022e).

3.2.4.18 Pipeline.

Un pipeline en Jenkins es un conjunto de pasos que permiten definir flujos basados en integración y entrega continua, disponiendo de un amplio catálogo de complementos y herramientas compatibles. Para describir cada uno de los pasos o fases asociadas al pipeline se puede utilizar una sintaxis declarativa escrita en Groovy o directivas DSLs propias de Jenkins, en ambos casos se define un archivo llamado Jenkinsfile con todas las especificaciones del flujo a implementar. (Jenkins.io, s.f.).

3.2.5 IMPLEMENTACIÓN DE LAS ÁREAS, PROCESOS, SISTEMAS Y BUENAS PRÁCTICAS

La automatización de tareas manuales en el procedimiento de despliegue de APIs en Azure API Management a través de pipelines en Jenkins surgió como una iniciativa del Área de Infraestructura y Operaciones TI, específicamente del equipo de API Platform, para mejorar el procedimiento manual que venía utilizando la organización.

En la Figura 7 se muestran las etapas para la gestión del proyecto que siguió el equipo de API Platform. El detalle del cronograma de cada una de estas etapas durante el desarrollo del proyecto se encuentra en el Anexo 1.

Figura 7

Etapas para la Gestión del Proyecto



Nota. Elaboración propia. Basado en el framework de gestión de proyectos de la Entidad Financiera, 2021.

Las etapas, actividades, herramientas y tecnologías que se utilizaron para la implementación de este proyecto, se describen a continuación:

3.2.5.1 Etapa de Planificación.

En esta etapa el Product Owner definió los requerimientos funcionales que debía satisfacer la solución, para lo cual creó un Épica en Jira sobre la que se manejó el backlog del proyecto. Como se puede apreciar en la Figura 8, para este proyecto se creó la épica “Agregar nuevas funcionalidades – Pipeline Azure API Management”.

Figura 8

Épica del Proyecto



Nota. La figura muestra la épica creada para el proyecto de automatización. Tomado de la herramienta Jira interna de la Entidad financiera, 2021.

Para la épica del proyecto se definieron posteriormente los requerimientos funcionales que debía satisfacer la solución, los cuales se describen en la Tabla 10.

Tabla 10

Requerimientos funcionales del Proyecto

Requerimiento	Descripción
Automatización del flujo de eliminación de APIs	El pipeline debe tener la capacidad de eliminar APIs que ya se encuentren desplegadas en el API Management.
Automatización del flujo de registro y actualización de APIs	El pipeline debe tener la capacidad de registrar APIs que ya no se encuentren desplegadas en el API Management, en caso el API ya se encuentre desplegada, se debe actualizar.
Integración con repositorio de contratos de APIs	El pipeline debe tener la capacidad de descargar el contrato del API a desplegar directamente desde SwaggerHub.

Automatización del flujo de configuración de políticas a nivel de API	El pipeline debe tener la capacidad de configurar políticas a nivel de API para las APIs a desplegar en API Management.
---	---

Nota. Elaboración propia. Basado en los requerimientos definidos para la épica del proyecto en la herramienta Jira interna de la Entidad financiera, 2021.

En esta etapa, el Product Owner también seleccionó a los integrantes del equipo de desarrollo. En la Tabla 11 se muestra la distribución de los team members y el nivel de asignación que les correspondió en este proyecto.

Tabla 11

Distribución del Equipo de Proyecto

Rol	Asignación
Analista Programador	Total
Analista de QA	Parcial
Arquitecto de Tecnología	Parcial

Nota. Elaboración propia. Basado en los roles y personas asignadas a la épica del proyecto en la herramienta Jira interna de la Entidad financiera, 2021.

Se designó como responsable del desarrollo e implementación de la solución al team member con rol de Analista Programador. La participación del resto del equipo fue a tiempo parcial apoyando en tareas y definiciones puntuales dependiendo de la necesidad.

3.2.5.2 Etapa de Análisis.

En esta etapa el Analista programador definió y creó en Jira las historias de usuario a partir del análisis de los requerimientos funcionales definidos previamente por el Product Owner.

En la Figura 9 se puede apreciar las historias de usuario que conformaron el backlog para la épica del proyecto.

Figura 9

Historias de usuario para el Proyecto

Incidencias en épica				
✓		Preparación del entorno de desarrollo	RESUELTA	Jorge Alamo Castillo
✓		Integración para obtener contratos desde SwaggerHub	RESUELTA	Jorge Alamo Castillo
✓		Desarrollo de funcionalidad para registro y actualización de APIs	RESUELTA	Jorge Alamo Castillo
✓		Desarrollo de funcionalidad para eliminación de APIs	RESUELTA	Jorge Alamo Castillo
✓		Desarrollo de funcionalidad para aplicar políticas a nivel de API	RESUELTA	Jorge Alamo Castillo
✓		Actualización de la librería compartida	RESUELTA	Jorge Alamo Castillo
✓		Revisión de código	RESUELTA	Jorge Alamo Castillo

Nota. La figura muestra las historias de usuario del proyecto de automatización. Tomado de la herramienta Jira interna de la Entidad financiera, 2021.

3.2.5.3 Etapa de Diseño.

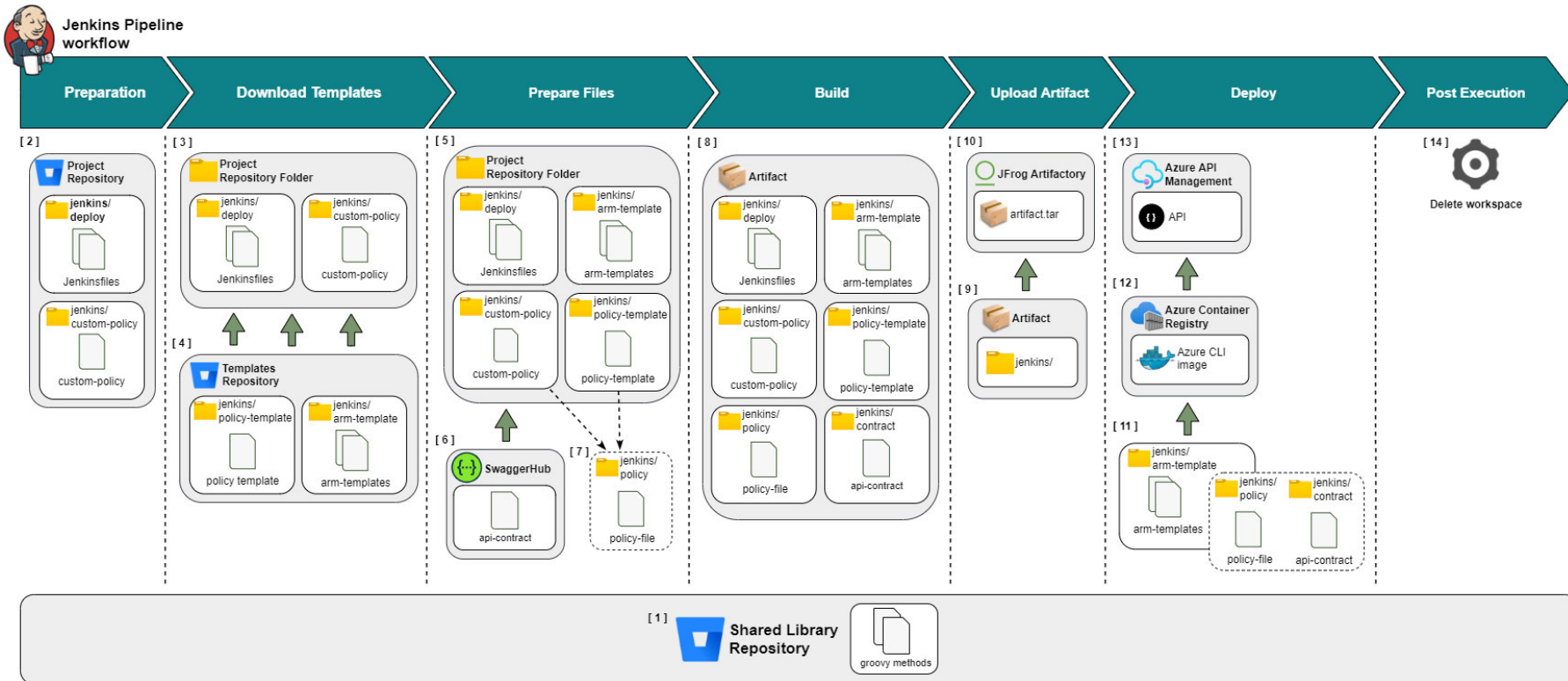
En esta etapa el Arquitecto de Tecnología con el apoyo del Analista Programador, realizó el diseño de la solución tomando como base la automatización que existía previamente y considerando la necesidad de integrarse con la herramienta de uso interno SwaggerHub como repositorio para obtener los contratos de las APIs a desplegar en Azure API Management.

El diseño de la solución se realizó siguiendo los lineamientos y tecnologías definidos por la Entidad Financiera para la automatización de procesos y procedimientos.

En la Figura 10 se muestra el diseño general de la solución para automatizar el despliegue de APIs en la plataforma Azure API Management.

Figura 10

Diseño General de la Solución



Nota. Elaboración propia. Basado en el diseño de solución propuesto por el Arquitecto de Tecnología de la Entidad Financiera, 2021.

En el diseño de la solución se definió el uso de pipelines automatizados en Jenkins bajo un esquema descentralizado, es decir, cada equipo que desee utilizar la automatización debe preparar su propio repositorio y los pipelines asociados. Las funcionalidades se implementaron sobre una librería compartida que centraliza las funcionalidades utilizadas por los pipelines.

De forma complementaria, se proporcionó el acceso a los pipelines y el repositorio que fueron utilizados en la etapa de despliegue a manera de plantillas que cualquier equipo puede replicar fácilmente y utilizar la automatización.

A fin de describir el flujo seguido por la automatización, se asume el escenario de un equipo que ha implementado su propio repositorio (el cual denominaremos Project) y los pipelines asociados. Las etapas generales definidas en estos pipelines para el despliegue de APIs en Azure API Management se describen a continuación:

- Preparation: Se ejecuta el pipeline desde Jenkins apuntando a un archivo Jenkinsfile almacenado en el repositorio Project [2], este repositorio incluye el archivo custom-policy con políticas personalizadas a configurar en el API (su uso es opcional). Posteriormente se cargan variables de ambiente y se importan los métodos que utilizará el pipeline desde el repositorio Shared Library [1], el cual centraliza los métodos en lenguaje Groovy, utilizados durante el despliegue.
- Download Templates: En este punto se descargan los archivos ARM templates y la plantilla de políticas (policy-template) almacenados en el repositorio Templates [4]. Estas plantillas son guardadas en la carpeta del repositorio Project [3] en runtime.
- Prepare Files: Se descarga el contrato del API desde SwaggerHub [6], dentro del archivo se añade el host del servicio backend asociado al API y posteriormente se guarda en la carpeta del repositorio Project [5], los valores necesarios para identificar el contrato a descargar se especifican en el archivo Jenkinsfile. Adicionalmente, se construye el archivo policy-file a partir de la plantilla policy-template, en la cual se insertan las políticas personalizadas definidas en el archivo custom-policy [7].

- Build: Se empaquetan los archivos que se encuentran en la carpeta del repositorio Project [8] en runtime, al archivo resultante se le denomina artefacto.
- Upload Artifact: El artefacto generado [9] es versionado en el repositorio de Artifactory [10].
- Deploy: El despliegue del API se realiza a través de un contenedor creado en base a una imagen Docker de Azure CLI almacenada en el servicio Azure Container Registry de la organización [12]. Sobre el contenedor se ejecutan los archivos ARM templates, los cuales en sus especificaciones utilizan el contrato del API y el archivo de políticas (policy-file) [11]; estos archivos ARM templates definen una serie de instrucciones a ejecutar sobre el recurso API Management en Azure para realizar el despliegue del API [13], utilizando un service principal como método de autenticación.
- Post execution: Se liberan los recursos utilizados por el pipeline [14].

3.2.5.4 *Etapas de Construcción.*

En esta etapa el Analista Programador inició el desarrollo de las historias de usuario, De acuerdo con el tablero Kanban definido para las tareas de construcción en la Tabla 8, la revisión de pares se realizó con el Arquitecto de Tecnología, la validación de las historias de usuario fue realizada por el Analista de QA y el Product Owner se encargó brindar la conformidad de la parte usuaria sobre los entregables de las historias de usuario.

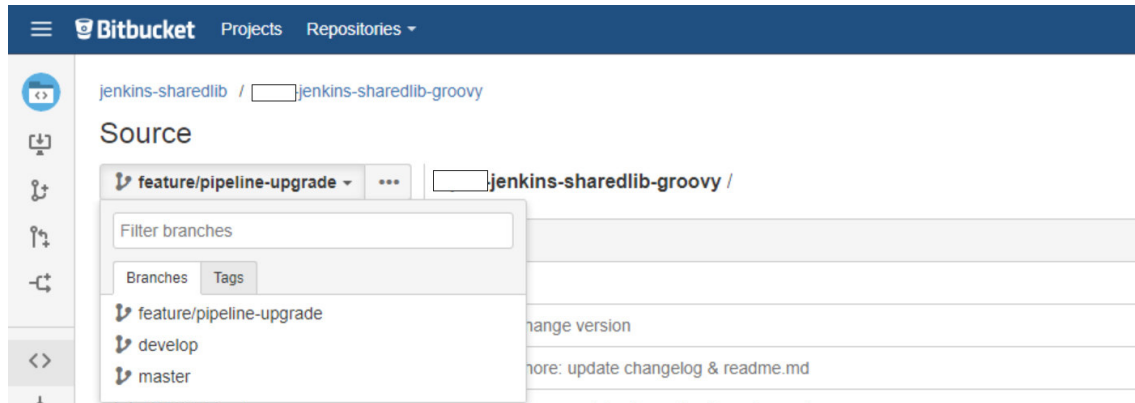
A continuación se detallan las actividades que fueron realizadas durante esta etapa del proyecto:

A. Creación de repositorios en Bitbucket.

Como se mencionó, el equipo de API Platform contaba previamente con un repositorio Shared Library para centralizar la lógica de programación utilizada por sus pipelines. Sobre este repositorio se creó la rama de construcción feature/pipeline-upgrade para el desarrollo de las funcionalidades requeridas, como se muestra en la Figura 11.

Figura 11

Rama de construcción en repositorio Shared Library

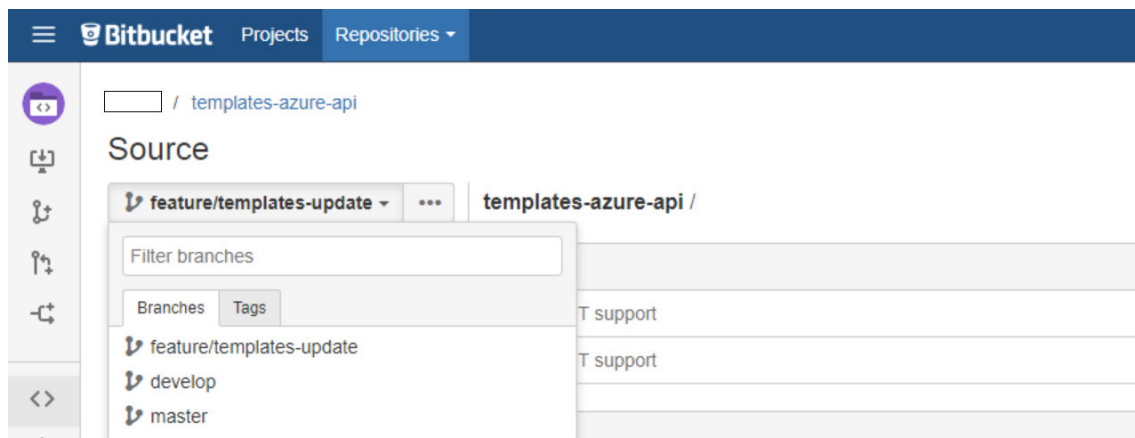


Nota. La figura muestra la rama de construcción creada en el repositorio Shared Library. Tomado de la herramienta Bitbucket interna de la Entidad Financiera, 2021.

De forma similar, se disponía del repositorio Templates como parte de un proyecto previo. Sobre este repositorio se creó la rama de construcción feature/templates-update para almacenar la plantilla base de políticas y los archivos ARM templates, tal como se muestra en la Figura 12.

Figura 12

Rama de construcción en repositorio Templates

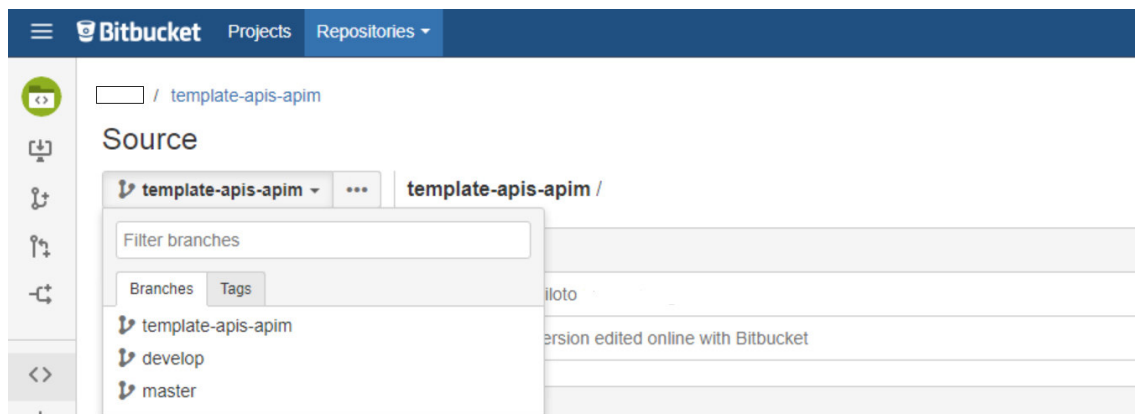


Nota. La figura muestra la rama de construcción creada en el repositorio Templates. Tomado de la herramienta Bitbucket interna de la Entidad Financiera, 2021.

Por último, se solicitó al Líder Técnico la creación del repositorio `template-apis-apim` (que denominaremos Project) para alojar los archivos de despliegue `Jenkinsfile` y el archivo con las políticas personalizadas. Sobre este repositorio se creó la rama de desarrollo `feature/template-jenkins`, como se muestra en la Figura 13.

Figura 13

Repositorio Project y rama de construcción



Nota. La figura muestra el repositorio Project y la rama de construcción creados. Tomado de la herramienta Bitbucket interna de la Entidad Financiera, 2021.

Este repositorio se utilizó para probar las funcionalidades desarrolladas sobre el repositorio Shared Library.

B. Creación de pipelines en Jenkins.

Para los repositorios Shared Library y Templates, previamente se contaba con pipelines de entrega creados para promover los cambios en el código fuente hacia los ambientes de certificación y producción. Este tipo de pipelines, cuya finalidad es promover las fuentes, no requieren ser ejecutados en el ambiente de desarrollo.

Para probar las funcionalidades implementadas en el repositorio Shared Library respecto al despliegue de APIs en Azure API Management, se crearon pipelines en Jenkins asociados al repositorio Project para los ambientes de desarrollo, certificación y producción, como se describe en la Tabla 12.

Tabla 12

Pipelines para despliegue de APIs en Azure API Management

Pipeline	Función	Repositorio
template-deploy-apis-apim-dev	Registro y actualización de APIs	Project
template-deploy-apis-apim-cert	Registro y actualización de APIs	Project
template-deploy-apis-apim-prod	Registro y actualización de APIs	Project
template-delete-apis-apim-dev	Eliminación de APIs	Project
template-delete-apis-apim-cert	Eliminación de APIs	Project
template-delete-apis-apim-prod	Eliminación de APIs	Project

Nota. Elaboración propia. Basado en los pipelines para despliegue de APIs en Azure API Management creados en la herramienta Jenkins interna de la Entidad Financiera, 2021.

En la Figura 14 se muestra como referencia, uno de los pipelines creados para el despliegue de APIs en Azure API Management del ambiente de desarrollo.

Figura 14

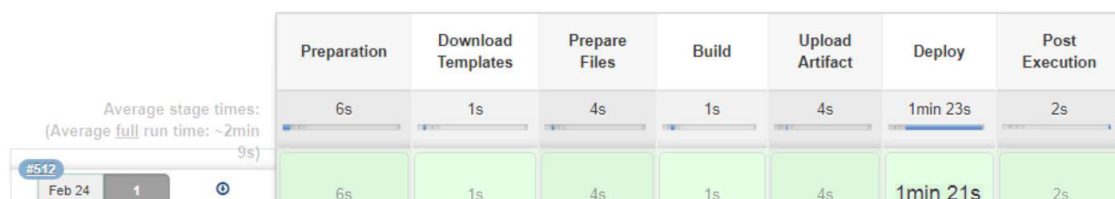
Pipeline de despliegue en API Management de desarrollo

Pipeline TEMPLATE-DEPLOY-APIS-APIM-DEV

Full project name: TEMPLATE-DEPLOY-APIS-APIM-DEV



Stage View



Nota. La figura muestra uno de los pipelines utilizados para realizar el despliegue de APIs en Azure API Management del ambiente de desarrollo. Tomado de la herramienta Jenkins interna de la Entidad Financiera, 2021.

Estos pipelines se asociaron al repositorio de Project configurando su URL y la rama a ejecutar, así como la ruta del archivo Jenkinsfile de despliegue. En la Figura 15 se muestra como referencia, la configuración de los pipelines en el ambiente de desarrollo.

Figura 15

Configuración de pipelines de despliegue en desarrollo

The image shows a Jenkins configuration interface. On the left, there are labels for 'Repositories', 'Branches to build', 'Navegador del repositorio', 'Additional Behaviours', and 'Script Path'. The 'Repositories' section contains a 'Repository URL' field with the value 'https://bitbucket[redacted]/scm/[redacted]/template-apis-apim.git' and a 'Credentials' dropdown menu showing 'DES/[redacted] bitbucket-user'. The 'Branches to build' section has a 'Branch Specifier (blank for \'any\')' field with the value 'feature/template-jenkins'. The 'Navegador del repositorio' is set to '(Auto)'. The 'Additional Behaviours' section has a text input field and an 'Añadir' button. The 'Script Path' is set to '[redacted]/Jenkinsfile-delivery-dev-apim-pipeline.groovy'.

Nota. La figura muestra la configuración realizada en los pipelines para despliegue de APIs en Azure API Management del ambiente de desarrollo. Tomado de la herramienta Jenkins interna de la Entidad Financiera, 2021.

C. Preparación de archivos de despliegue.

De forma complementaria a los pipelines y el repositorio utilizados para validar las funcionalidades implementadas en el repositorio Shared Library, se crearon archivos de despliegue denominados Jenkinsfiles, los cuales fueron almacenados en el repositorio Project y configurados en los pipelines de Jenkins para ejecutar los despliegues.

En estos archivos se especifican las etapas que conforman el workflow del pipeline que se definieron en el diseño de la solución (mostrado en la Figura 10), y adicionalmente se incluyen los siguientes apartados:

- Referencia al repositorio Shared Library, utilizado para importar las clases y métodos necesarios para realizar el despliegue.

- Nombre y versión del API a desplegar, utilizados para obtener el contrato del API desde SwaggerHub.
- Nombre del archivo de políticas, indica el archivo dentro del repositorio que contiene las políticas personalizadas que se configuran al desplegar el API. En caso no se declare, se realiza el despliegue sin considerar las políticas.
- URL del backend, se utiliza para indicar a qué servicio debe redireccionar las solicitudes el Azure API Management. Este valor se añade en el contrato del API.
- En etapa de Deploy se puede utilizar el método `deployApiApim` (para registrar o actualizar APIs) o `deleteApiApim` (para eliminar APIs) según se requiera.

En la Figura 16 se muestra como referencia, el archivo Jenkinsfile utilizado en uno de los pipelines de despliegue en el ambiente de desarrollo.

Figura 16

Archivo de despliegue Jenkinsfile de desarrollo

```
@Library('jenkins-sharedlib')
import sharedlib.Jenkinsfile
def [ ] = [ ] Jenkinsfile (steps, this)
def project = " "
def backendUrl = "${project}"
def parameters = [
    swaggerHubParameters : [
        apiName: " ",
        apiVersion: " "
    ], policyParameters : [
        customPolicyFile: " "
    ]
]
try {
    node {
        stage('Preparation') {
            checkout scm
            env.project = "${project}"
            env.backendUrl = "${backendUrl}"
            .prepare()
        }
        stage('Download Templates') {
            .downloadTemplate(" ", "template-deploy-azure-api")
        }
        stage('Prepare files') {
            .buildContract(parameters)
            .buildPolicy(parameters, )
        }
        stage('Build') {
            .build(' ', 'tar')
        }
        stage('Upload Artifact') {
            .deployArtifact(' ', 'tar')
        }
        stage('Deploy') {
            .deployApiApim(parameters)
        }
        stage('Post Execution') {
            .executePostExecutionTasks()
        }
    }
}
```

Nota. La figura muestra el archivo Jenkinsfile utilizado para el despliegue de APIs en Azure API Management en el ambiente de desarrollo. Tomado de la herramienta Bitbucket interna de la Entidad Financiera, 2021.

D. Integración con SwaggerHub.

Se programó la capacidad que permite obtener el contrato del API a ser desplegada en Azure API Management directamente desde SwaggerHub.

La integración se realizó a través de una URL que expone SwaggerHub, la cual permite obtener el contrato de una determinada API en formato Json a partir del nombre y la versión con la que fue registrada en la herramienta. El formato de URL utilizado fue *{hostname}/v1/apis/{grupo}/{nombre_api}/{version_api}?format=json&resolved=true*, donde los valores de hostname y grupo son propios de la Entidad Financiera.

Se implementó el método *buildContract* en el repositorio Shared Library, como se muestra en la Figura 17.

Figura 17

Método buildContract

```
public void buildContract(script, steps, swaggerHub) {
    try {
        steps.sh
            "mkdir ${script.██████████}${swaggerHub.contractFolder}"
        steps.sh
            "chmod -R ██████████ ${script.██████████}${swaggerHub.contractFolder}/"
        steps.sh
            """curl -X GET
            '${swaggerHub.uri}${swaggerHub.apiName}/${swaggerHub.apiVersion}${swaggerHub.uriParams}'
            -o ${script.██████████}${swaggerHub.contractFolder}/${swaggerHub.contractFileName}"""

        String contractTypeName =
            steps.sh script:
                """cat ${script.██████████}${swaggerHub.contractFolder}/${swaggerHub.contractFileName}
                | grep '██████████' | awk -F '\"' '██████████' | head -n1""", returnStdout: true
        contractTypeName = contractTypeName.trim()

        if (contractTypeName != "██████████" && contractTypeName != "██████████") {
            String curlResponse =
                steps.sh script:
                    """cat ${script.██████████}${swaggerHub.contractFolder}/${swaggerHub.contractFileName}""",
                    returnStdout: true
            throw new Exception("Ocurrió un error al obtener el contrato desde SwaggerHub: ${curlResponse}")
        }
    } catch (Exception e) {
        throw e
    }
}
```

Nota. La figura muestra el método *buildContract* implementado en el repositorio Shared Library para obtener los contratos de las APIs desde SwaggerHub. Tomado de la herramienta Bitbucket interna de la Entidad Financiera, 2021.

El método *buildContract* crea la carpeta */contract* a la que se le asignan permisos de escritura para guardar el contrato del API que se descarga utilizando la URL que expone SwaggerHub. Los valores de *nombre_api* y *version_api* se definen en el archivo Jenkinsfile.

Este contrato se utiliza dentro del flujo del pipeline para desplegar APIs en Azure API Management.

E. Funcionalidad para configuración de políticas.

Se programó la funcionalidad que permite configurar políticas a nivel de API al momento de registrar y actualizar un API en Azure API Management. Las políticas que admite el pipeline son de tipo inbound, es decir, se aplican sobre las solicitudes entrantes al API.

Para este desarrollo, se añadió al repositorio Templates una plantilla XML con la estructura base para incluir las políticas a configurar en el API dentro de la carpeta */policy-template*, como se muestra en la Figura 18.

Figura 18

Plantilla de políticas XML

```
<policies>
  <inbound>
    <!-- insert custom policies -->
  </inbound>
  <backend>
    <base />
  </backend>
  <outbound>
    <base />
  </outbound>
  <on-error>
    <base />
  </on-error>
</policies>
```

Nota. La figura muestra la plantilla con la estructura base para incluir políticas a configurar en el API al ser desplegada en Azure API Management. Tomado de la herramienta Bitbucket interna de la Entidad Financiera, 2021.

Se implementó el método *buildPolicy* en el repositorio Shared Library, como se muestra en la Figura 19.

Figura 19

Método buildPolicy

```
public void buildPolicy(script, steps, policy) {  
    try {  
        steps.sh "mkdir ${script.██████████} ${policy.policyFolder}"  
        steps.sh "chmod -R ██████████ ${script.██████████} ${policy.policyFolder}/"  
  
        String policyString =  
            steps.sh script: "cat ${script.██████████} ${policy.deployPolicy}/██████████.xml",  
            returnStdout: true  
  
        script.steps.writeFile(  
            file: "${script.██████████} ${policy.policyFolder}/${policy.policyFile}",  
            text: policyString  
        )  
  
        if(policy.customPolicy) {  
            steps.sh "sed ██████████ '/<!--Insert custom policies-->/{r  
                ${script.██████████} ${policy.customPolicyFolder}/${policy.customPolicyFile}' -e 'd}'  
                ${script.██████████} ${policy.policyFolder}/${policy.policyFile}""  
        } else {  
            steps.sh "sed ██████████ '/<!--Insert custom policies-->/d'  
                ${script.██████████} ${policy.policyFolder}/${policy.policyFile}""  
        }  
    }  
}
```

Nota. La figura muestra el método *buildPolicy* implementado en el repositorio Shared Library para construir el archivo con la política a configurar en el API al ser desplegada en Azure API Management. Tomado de la herramienta Bitbucket interna de la Entidad Financiera, 2021.

El método *buildPolicy* crea la carpeta */policy* a la que se le asignan permisos de escritura para guardar el archivo resultante de políticas, el cual se genera siguiendo los siguientes pasos:

- Se descarga la plantilla base de políticas que se encuentra en la carpeta */policy-template* del repositorio Templates.
- Sobre la plantilla base de políticas se insertan las políticas personalizadas que se encuentran almacenadas en la carpeta */custom-policy* del repositorio Project, obteniendo como resultado el archivo final de políticas.

Estas políticas se utilizan dentro del flujo del pipeline para desplegar APIs en Azure API Management.

F. Preparación de ARM templates.

Se definieron ARM templates para realizar el despliegue de APIs en Azure API Management, estas plantillas permiten definir configuraciones a aplicar sobre un recurso de Azure a través un archivo Json. Para esta implementación se definió utilizar dos plantillas para el despliegue de APIs y políticas a nivel de API respectivamente.

Para el despliegue de APIs, el archivo ARM template requiere los siguientes parámetros de entrada:

- Instancia de API Management, identificador del recurso API Management en Azure. Este valor se obtiene al cargar las variables de ambiente en el pipeline.
- Identificador del API, utilizado para poder identificar el API en el API Management. Este valor se genera a partir del contrato del API.
- Nombre del API, indica el nombre con el que se mostrará el API en el API Management. Este valor se obtiene del contrato del API.
- Descripción, indica la descripción del API que se mostrará en el API Management. Este valor se obtiene del contrato del API.
- Protocolo de comunicación, utilizado para especificar si el protocolo de transferencia de datos es http o https. Este valor se obtiene del contrato del API.
- Path del API, representa la dirección que utiliza el API para exponer sus funcionalidades. Este valor se obtiene del contrato del API.
- Contrato del API, este archivo se obtiene a través del método *buildContract* directo de SwaggerHub.

En la Figura 20 se muestra el ARM template para el despliegue de APIs en Azure API Management.

ARM template para despliegue de APIs

```
{
  "$schema": "http://schema.management.azure.com/schemas/managementTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
  },
  "resources": [
    {
      "type": "Microsoft.ApiManagement",
      "name": "[concat(parameters('apiInstanceName'), '/', parameters('apiName'))]",
      "apiVersion": "2018-01-01",
      "scale": null,
      "properties": {
        "description": "[parameters('apiDescription')]",
        "type": "http",
        "displayName": "[parameters('apiDisplayName')]",
        "path": "[parameters('apiPath')]",
        "protocols": [
          "https"
        ],
        "value": "[parameters('apiContract')]",
        "format": "openapi+json",
        "subscriptionRequired": "[parameters('subscriptionRequired')]"
      },
      "resources": []
    }
  ],
  "outputs": {}
}
```

Nota. La figura muestra el archivo ARM template utilizado para el despliegue de APIs en Azure API Management. Tomado de la herramienta Bitbucket interna de la Entidad Financiera, 2021.

Para el despliegue de políticas a nivel de API, el archivo ARM template requiere los siguientes parámetros de entrada:

- Instancia de API Management, identificador del recurso API Management en Azure. Este valor se obtiene al cargar las variables de ambiente en el pipeline.
- Identificador del API, utilizado para poder identificar el API a la que se deben asociar las políticas. Este valor se genera a partir del contrato del API.
- Archivo de política en formato XML, este archivo se obtiene a través del método *buildPolicy*.

En la Figura 21 se muestra el ARM template para el despliegue de políticas a nivel de API en Azure API Management.

Figura 21

ARM template para despliegue de políticas

```
{
  "$schema": "http://schema.management.azure.com/schemas/management-template.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
  },
  "resources": [
    {
      "type": "Microsoft.ApiManagement",
      "name": "[concat(parameters('apimInstanceName'), '/', parameters('apiName'), '/policy')]",
      "apiVersion": "2018-01-01",
      "properties": {
        "value": "[replace(parameters('apiPolicy'), '{', parameters('apiName') + ')']]",
        "format": "rawxml"
      },
      "dependsOn": [
      ]
    }
  ],
  "outputs": {
  }
}
```

Nota. La figura muestra el archivo ARM template para el despliegue de políticas a nivel de API en Azure API Management. Tomado de la herramienta Bitbucket interna de la Entidad Financiera, 2021.

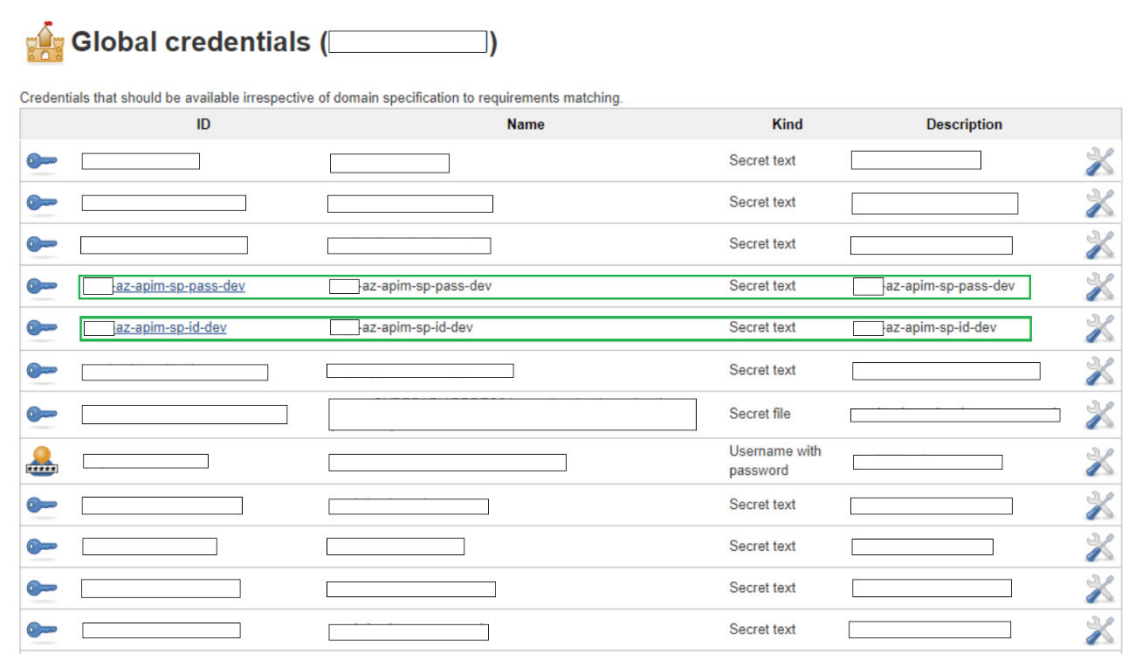
G. Configuración de service principal.

Como método de autenticación se utilizó una identidad de seguridad en Azure denominada service principal, la cual permitió definir capacidades de autenticación y autorización sobre el recurso API Management. Para esta implementación se consideraron los roles necesarios para el registro, actualización y eliminación de APIs.

La creación del service principal se solicitó al equipo de Administración de Accesos, indicando los roles que se debían configurar y posteriormente se registraron los valores del identificador y contraseña asociados al service principal en la bóveda de credenciales que dispone Jenkins para almacenar los secretos utilizados en los pipelines.

Estos secretos son accedidos por el pipeline para conectarse al API Management y ejecutar las acciones necesarias para realizar el despliegue de APIs. El registro de las credenciales del service principal se muestra en la Figura 22.

Figura 22
Registro de service principal



Global credentials ()

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
		Secret text	
		Secret text	
		Secret text	
az-apim-sp-pass-dev	az-apim-sp-pass-dev	Secret text	az-apim-sp-pass-dev
az-apim-sp-id-dev	az-apim-sp-id-dev	Secret text	az-apim-sp-id-dev
		Secret text	
		Secret file	
		Username with password	
		Secret text	
		Secret text	
		Secret text	
		Secret text	

Nota. La figura muestra el registro de las credenciales asociadas al service principal utilizado para el despliegue de APIs en Azure API Management en la bóveda de secretos de Jenkins. Tomado de la herramienta Jenkins interna de la Entidad Financiera, 2021.

H. Funcionalidad para registro y actualización de APIs.

Se desarrollaron las funcionalidades para realizar el registro y actualización de APIs en el API Management de Azure. En este desarrollo se adaptó la funcionalidad de registro de APIs para que en caso el contrato del API ya se encuentre desplegado en el API Management, se realice la actualización del API.

Se implementó el método *apiDeployment* en el repositorio Shared Library para automatizar el registro y actualización de APIs, como se muestra en la Figura 23.

Figura 23

Método *apiDeployment*

```
public void apiDeploymeny(script, steps, docker, configuration, jenkinsCredentialId,
contract, contractString, product) {

    DeployParametersManager deployParametersManager = new DeployParametersManager()
    DeployParameters deployParameters = deployParametersManager.generateDeployParameters(script, configuration)

    docker.withRegistry(██████████, DOCKER_REGISTRY_URL) { docker.withServer(██████████, DOCKER_HOST) {
        steps.withCredentials([
            [$class: "StringBinding", credentialsId: "${██████████}", variable: "██████████"],
            [$class: "StringBinding", credentialsId: "${██████████}", variable: "██████████"],
            [$class: "StringBinding", credentialsId: "${██████████}", variable: "██████████"}]) {

            String azcliMessage = ""
            String azCliDockerId =
                this.prepareDockerContainer(steps, deployParameters.dockerParameters, deployParameters.dockerVolumes)

            try {
                azcliMessage = "Error al conectarse al apim ${script.env.██████████}."
                this.azureConnection(script, steps, azCliDockerId, deployParameters.loginArgs)

                azcliMessage = "Error al crear o actualizar el api ${contract.██████████}."
                String azureApiDeployCommand = this.prepareApiDeployment(script, configuration, contract)
                this.dockerExec(steps, azCliDockerId, azureApiDeployCommand)

                azcliMessage = "Error al crear o actualizar la politica en el api ${contract.██████████}."
                String azurePolicyDeployCommand = this.preparePolicyDeployment(script, configuration, contract)
                this.dockerExec(steps, azCliDockerId, azurePolicyDeployCommand)

                script.steps.echo "El api ${contract.██████████} fue creada/actualizada exitosamente."
            }
        }
    }
}
```

Nota. La figura muestra el método *apiDeployment* implementado en el repositorio Shared Library para registrar y actualizar APIs en Azure API Management. Tomado de la herramienta Bitbucket interna de la Entidad Financiera, 2021.

El método *apiDeployment* ejecuta los siguientes pasos:

- Accede al servicio Azure Container Registry de la organización utilizando los métodos *docker.withRegistry* y *docker.withServer*, propios de Jenkins.
- Crea un contenedor Docker instanciando la imagen Azure CLI almacenada en el registry a través del método interno *prepareDockerContainer*.
- Invoca al método interno *azureConnection* para establecer conexión con el recurso Azure API Management a través del contenedor Docker. Se utiliza un Service Principal para como método de autenticación hacia el API Management.
- Invoca al método interno *prepareApiDeployment*, el cual recibe como entrada el contrato del API y genera el comando para el despliegue del API utilizando ARM templates.
- Invoca al método interno *preparePolicyDeployment*, el cual utiliza el contrato del API para identificar sobre qué elemento se tiene que asociar el archivo de políticas

y se genera el comando para el despliegue de las políticas a nivel de API utilizando ARM templates.

- Luego invoca al método interno *dockerExec* para ejecutar los comandos generados con los métodos *prepareApiDeployment* y *preparePolicyDeployment* en el contenedor Docker, el cual se encuentra conectado al API Management a través de Azure CLI.

En caso de producirse algún error durante la ejecución se imprime en el log del pipeline el detalle de lo ocurrido.

1. Funcionalidad para eliminación de APIs.

Se desarrolló la funcionalidad para realizar la eliminación de APIs en el API Management de Azure. En este desarrollo se implementó el método *apiDeletion* en el repositorio Shared Library para automatizar la eliminación de APIs, como se muestra en la Figura 24.

Figura 24

Método *apiDeletion*

```
public void apiDeletion(script, steps, docker, configuration,
jenkinsCredentialId, contract) {

    DeployParametersManager deployParametersManager = new DeployParametersManager()
    DeployParameters deployParameters = deployParametersManager.generateDeployParameters(script, configuration)

    docker.withRegistry(██████████ DOCKER_REGISTRY_URL) {
        docker.withServer(██████████ DOCKER_HOST) {
            steps.withCredentials([
                [$class: "StringBinding", credentialsId: "${██████████}", variable: "██████████"],
                [$class: "FileBinding", credentialsId: "${██████████}", variable: "██████████"],
                [$class: "StringBinding", credentialsId: "${██████████}", variable: "██████████"}]]) {

                String azcliMessage = ""
                String azCliDockerId =
                    this.prepareDockerContainer(steps, deployParameters.dockerParameters, deployParameters.dockerVolumes)

                try {
                    azcliMessage = "Error al conectarse al apim ${script.env.██████████}."
                    this.azureConnection(script, steps, azCliDockerId, deployParameters.loginArgs)

                    azcliMessage = "Error al eliminar el api ${contract.██████████}."
                    String azureApiDeleteCommand = this.prepareApiDeletion(script, configuration, contract)
                    this.dockerExec(steps, azCliDockerId, azureApiDeleteCommand)

                    script.steps.echo "El api ${contract.██████████} fue eliminada exitosamente."
```

Nota. La figura muestra el método *apiDeletion* implementado en el repositorio Shared Library para eliminar APIs en Azure API Management. Tomado de la herramienta Bitbucket interna de la Entidad Financiera, 2021.

El método *apiDeletion* ejecuta los siguientes pasos:

- Accede al servicio Azure Container Registry de la organización utilizando los métodos *docker.withRegistry* y *docker.withServer*, propios de Jenkins.
- Crea un contenedor Docker instanciando la imagen Azure CLI almacenada en el registry a través del método interno *prepareDockerContainer*.
- Invoca al método interno *azureConnection* para establecer conexión con el recurso Azure API Management a través del contenedor Docker. Se utiliza un Service Principal para como método de autenticación hacia el API Management.
- Invoca al método interno *prepareApiDeletion*, el cual recibe como entrada el contrato del API y genera el comando para la eliminación del API utilizando ARM templates.
- Luego invoca al método interno *dockerExec* para ejecutar el comando generado con el método *prepareApiDeletion* en el contenedor Docker, el cual se encuentra conectado al API Management a través de Azure CLI.

En caso de producirse algún error durante la ejecución se imprime en el log del pipeline el detalle de lo ocurrido.

3.2.5.5 *Etapas de Despliegue.*

En esta etapa el Analista Programador gestionó el congelamiento y pase a producción de la solución. Se coordinó con el equipo de Gobierno de APIs el uso de un API y contrato de prueba para realizar el despliegue en Azure API Management y validar las funcionalidades implementadas en el repositorio Shared Library durante esta etapa.

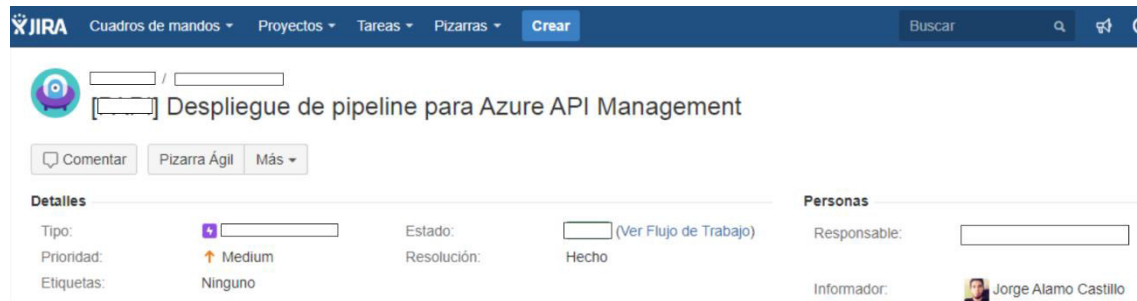
A. Registro de cambio.

El Analista Programador creó el ticket de cambio en Jira para gestionar el despliegue de la solución, en el cual se detallaron los pasos a seguir para ejecutar el despliegue en los ambientes de certificación y producción, así como el procedimiento de reversión en caso se presente una contingencia. Al finalizar se movió el ticket al carril de Revisión a nivel Técnico.

En la Figura 25 se muestra el ticket de cambio con el que se gestionó el despliegue de la automatización.

Figura 25

Ticket de cambio



Nota. La figura muestra el ticket de cambio creado para realizar el despliegue de la automatización. Tomado de la herramienta Jira interna de la Entidad financiera, 2021.

B. Revisión a nivel Técnico.

El Líder Técnico del equipo revisó que cada uno de los pasos definidos en el ticket para realizar el despliegue en los ambientes de certificación y producción, así como el procedimiento de reversión cumplan con los lineamientos y buenas prácticas de desarrollo definidos por la Entidad Financiera. Se hizo especial énfasis en asegurar la correcta especificación de los pasos a seguir en caso de reversión.

Al no encontrarse observaciones, el Líder Técnico movió el ticket al carril de Revisión a nivel de Seguridad.

C. Revisión a nivel de Seguridad.

El Analista de Seguridad validó que el cambio a desplegar en los ambientes de certificación y producción, así como el procedimiento de reversión cumpla con los lineamientos de seguridad de la Entidad Financiera. Se requirió realizar la explicación del cambio desde el punto de vista técnico y bajo qué escenarios se utilizará la automatización.

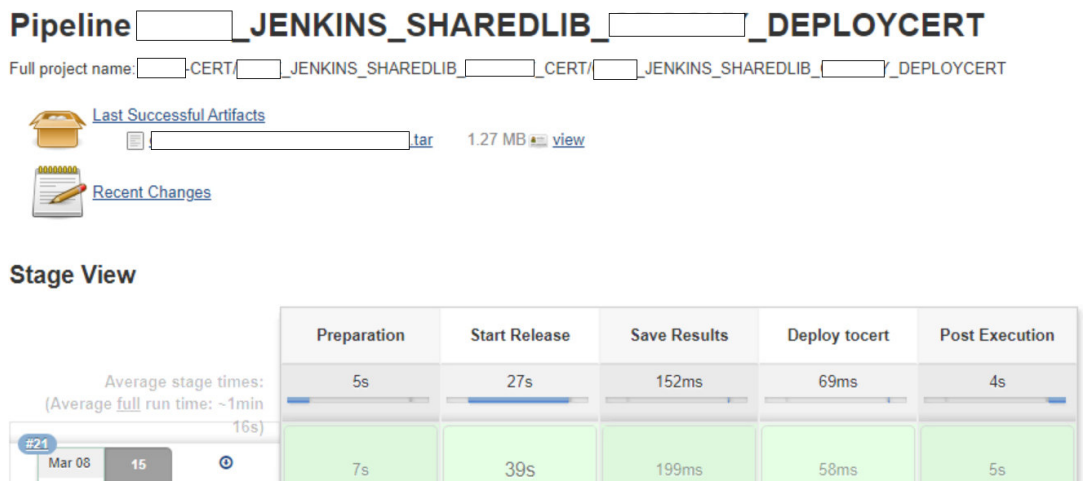
Se finalizó sin observaciones, así que el Analista de Seguridad movió el ticket al carril de Ejecución de Congelamiento.

D. Ejecución de Congelamiento.

El Agile Operator asignado al congelamiento, ejecutó los pasos indicados en el ticket para desplegar la solución en el ambiente de certificación. El despliegue consistió en ejecutar los pipelines para promoción de fuentes en los repositorios Shared Library y Templates, y posteriormente el pipeline para despliegue de APIs en Azure API Management.

En la Figura 26 se muestra la ejecución exitosa del pipeline para la promoción de fuentes en el repositorio Shared Library del ambiente de certificación.

Figura 26
Pipeline de entrega en repositorio Shared Library de certificación

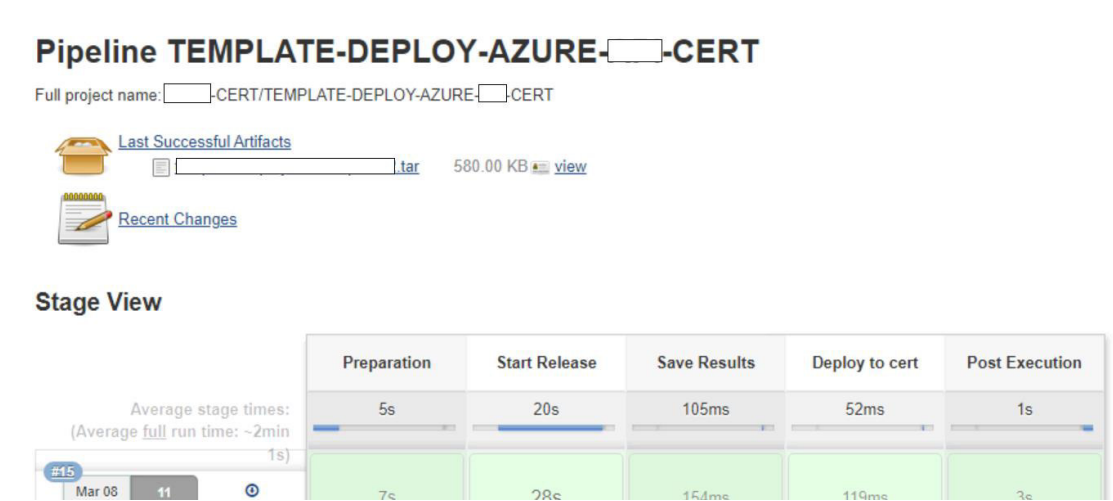


Nota. La figura muestra la ejecución del pipeline para la promoción de fuentes en el repositorio Shared Library del ambiente de certificación. Tomado de la herramienta Jenkins interna de la Entidad Financiera, 2021.

En la Figura 27 se muestra la ejecución exitosa del pipeline para la promoción de fuentes en el repositorio Templates del ambiente de certificación.

Figura 27

Pipeline de entrega en repositorio Templates de certificación

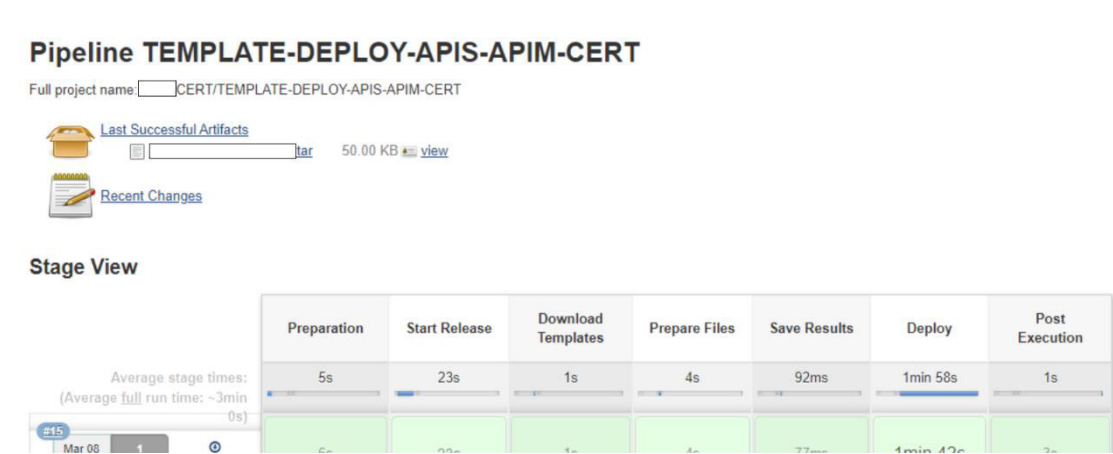


Nota. La figura muestra la ejecución del pipeline para la promoción de fuentes en el repositorio Templates del ambiente de certificación. Tomado de la herramienta Jenkins interna de la Entidad Financiera, 2021.

En la Figura 28 se muestra la ejecución exitosa del pipeline para el despliegue de APIs en Azure API Management del ambiente de certificación.

Figura 28

Pipeline de despliegue en API Management de certificación



Nota. La figura muestra la ejecución del pipeline para el despliegue de APIs en Azure API Management del ambiente de certificación. Tomado de la herramienta Jenkins interna de la Entidad Financiera, 2021.

Los pipelines se ejecutaron correctamente y se adjuntó al ticket Jira, la evidencia del despliegue del API en el API Management de certificación. Al finalizar sin observaciones, el Agile Operator movió el ticket al carril de Ejecución de Pruebas QA.

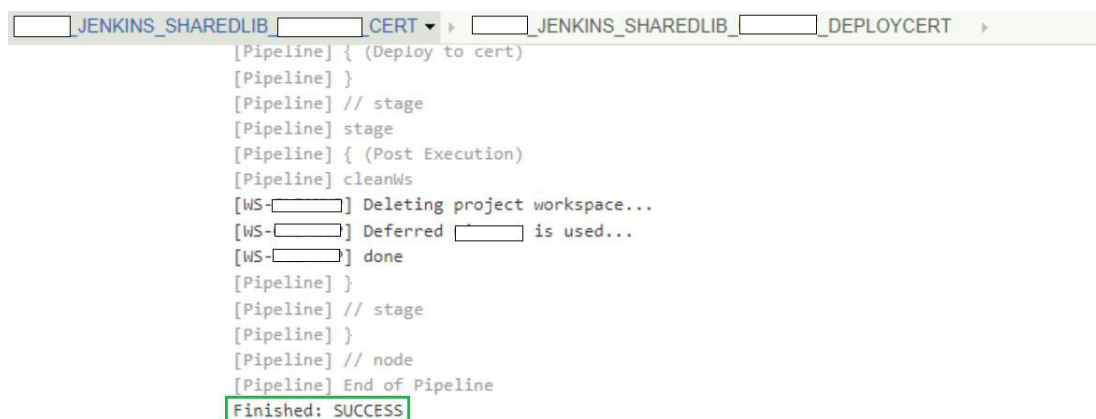
E. Ejecución de Pruebas QA.

El Analista de QA del equipo validó la correcta ejecución de los pipelines en el ambiente de certificación revisando los logs generados en Jenkins. Adicionalmente se revisó la evidencia adjuntada al ticket por parte del Agile Operator, en la cual se verificó que el API se había desplegado correctamente en el API Management de Azure de certificación.

En la Figura 29 se muestra el log del pipeline para la promoción de fuentes en el repositorio Shared Library del ambiente de certificación.

Figura 29

Log de pipeline de entrega en repositorio Shared Library de certificación



```
[Pipeline] { (Deploy to cert)
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Post Execution)
[Pipeline] cleanWs
[WS- ] Deleting project workspace...
[WS- ] Deferred  is used...
[WS- ] done
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Nota. La figura muestra el log de la ejecución satisfactoria del pipeline para la promoción de fuentes en el repositorio Shared Library del ambiente de certificación. Tomado de la herramienta Jenkins interna de la Entidad Financiera, 2021.

En la Figura 30 se muestra el log del pipeline para la promoción de fuentes en el repositorio Templates en el ambiente de certificación.

Figura 30

Log de pipeline de entrega en repositorio Templates de certificación

```
TEMPLATE-DEPLOY-AZURE--CERT ▾ ▶
[Pipeline] { (Deploy to cert)
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Post Execution)
[Pipeline] cleanWs
[WS-] Deleting project workspace...
[WS-] Deferred  is used...
[WS-] done
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Nota. La figura muestra el log de la ejecución satisfactoria del pipeline para la promoción de fuentes en el repositorio Templates del ambiente de certificación. Tomado de la herramienta Jenkins interna de la Entidad Financiera, 2021.

En la Figura 31 se muestra el log del pipeline para el despliegue de APIs en Azure API Management del ambiente de certificación.

Figura 31

Log de pipeline de despliegue en API Management de certificación

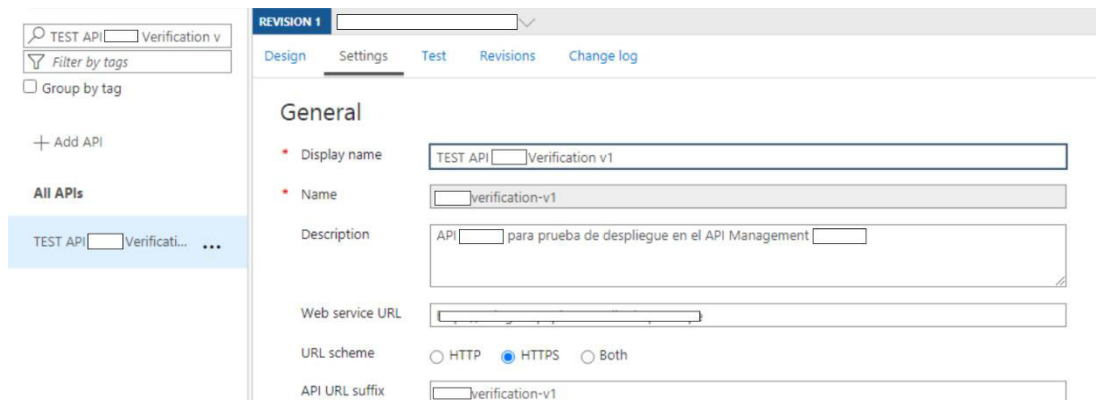
```
TEMPLATE-DEPLOY-APIS-APIM-CERT ▾ ▶
El api  -verification-v1 fue creada/actualizada exitosamente.
[Pipeline] sh
+ docker rm -f 
[Pipeline] sh
+ rm -rf *****
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // wrap
[Pipeline] sh
+ set +x
[Pipeline] }
```

Nota. La figura muestra el log de la ejecución satisfactoria del pipeline para el despliegue de APIs en Azure API Management del ambiente de certificación. Tomado de la herramienta Jenkins interna de la Entidad Financiera, 2021.

En la Figura 32 se muestra el API de prueba desplegada en Azure API Management del ambiente de certificación.

Figura 32

API desplegada en API Management de certificación



Nota. La figura muestra la API desplegada a través del pipeline en el API Management del ambiente de certificación. Tomado de la plataforma Azure API Management de la Entidad Financiera, 2021.

Realizadas las revisiones, no se encontraron observaciones y el Analista de QA movió el ticket al carril de Aprobación de Product Owner.

F. Aprobación de Product Owner.

El Product Owner del equipo revisó los resultados de las pruebas en certificación y verificó el cumplimiento de los requisitos funcionales solicitados para la solución desarrollada. Sin esta aprobación, no se hubiera sido posible realizar el pase a producción.

El Product Owner dio su conformidad a la solución implementada y movió el ticket al carril de Aprobación del Área de Operaciones.

G. Aprobación del Área de Operaciones.

El Líder Técnico del equipo solicitó la conformidad del Área de Operaciones para realizar el pase a producción indicando la fecha y hora en la que se programado llevar a cabo el pase. Luego de revisar la disponibilidad para el pase, el Área de

Operaciones brindó su conformidad y asignó a un Agile Operator responsable de ejecutar las tareas del pase a producción.

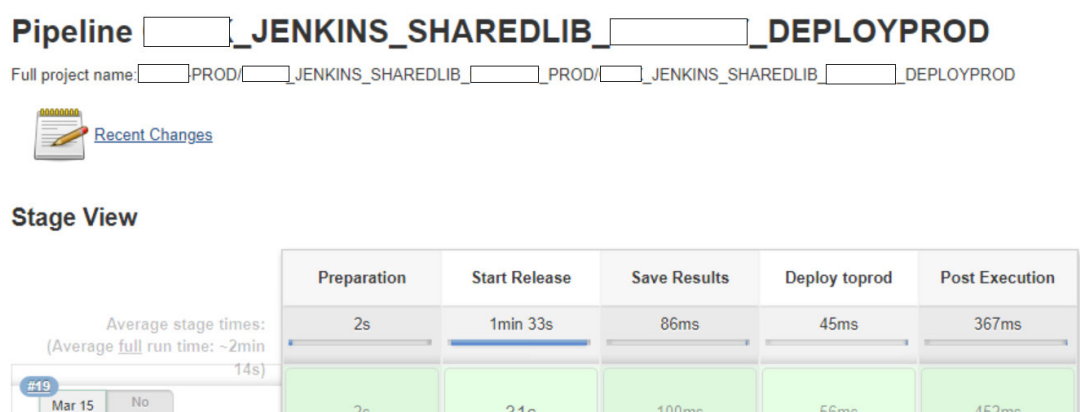
Obtenida la conformidad, el Líder Técnico registró la fecha y hora programada para el pase en el ticket de Jira, y adjuntó la conformidad del Área de Operaciones. Al culminar, movió el ticket al carril de Ejecución de Pase a Producción.

H. Ejecución de Pase a Producción.

El Agile Operator asignado al pase, ejecutó los pasos indicados en el ticket para desplegar la solución en el ambiente de Producción. El despliegue consistió en ejecutar los pipelines de promoción de fuentes para los repositorios Shared Library y Templates, y posteriormente el pipeline para despliegue de APIs en Azure API Management.

En la Figura 33 se muestra la ejecución exitosa del pipeline para la promoción de fuentes en el repositorio Shared Library del ambiente de producción.

Figura 33
Pipeline de entrega en repositorio Shared Library de producción

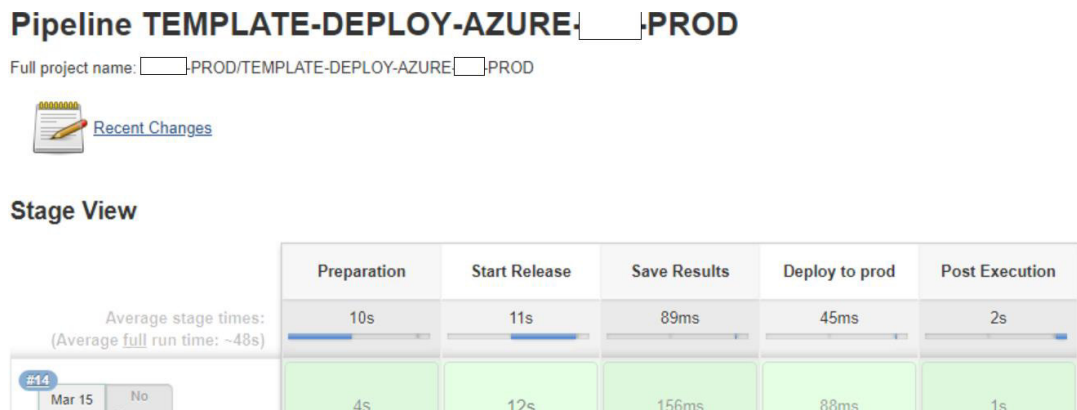


Nota. La figura muestra la ejecución del pipeline para la promoción de fuentes en el repositorio Shared Library del ambiente de producción. Tomado de la herramienta Jenkins interna de la Entidad Financiera, 2021.

En la Figura 34 se muestra la ejecución exitosa del pipeline para la promoción de fuentes en el repositorio Templates del ambiente de producción.

Figura 34

Pipeline de entrega en repositorio Templates de producción

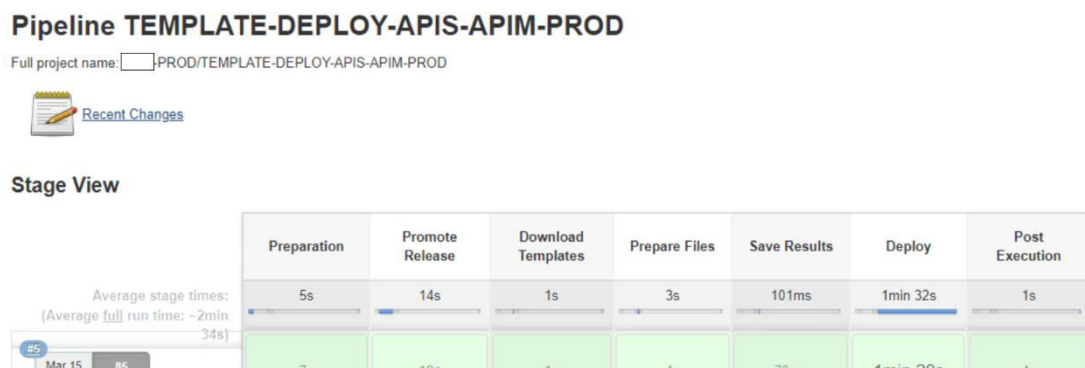


Nota. La figura muestra la ejecución del pipeline para la promoción de fuentes en el repositorio Templates del ambiente de producción. Tomado de la herramienta Jenkins interna de la Entidad Financiera, 2021.

En la Figura 35 se muestra la ejecución exitosa del pipeline para el despliegue de APIs en Azure API Management en el ambiente de producción.

Figura 35

Pipeline de despliegue en API Management de producción



Nota. La figura muestra la ejecución del pipeline para el despliegue de APIs en Azure API Management del ambiente de producción. Tomado de la herramienta Jenkins interna de la Entidad Financiera, 2021.

Los pipelines se ejecutaron correctamente y se adjuntó al ticket Jira evidencia del despliegue del API en el API Management de Producción. Al finalizar todo conforme, el Agile Operator movió el ticket al carril de Ratificación del cambio.

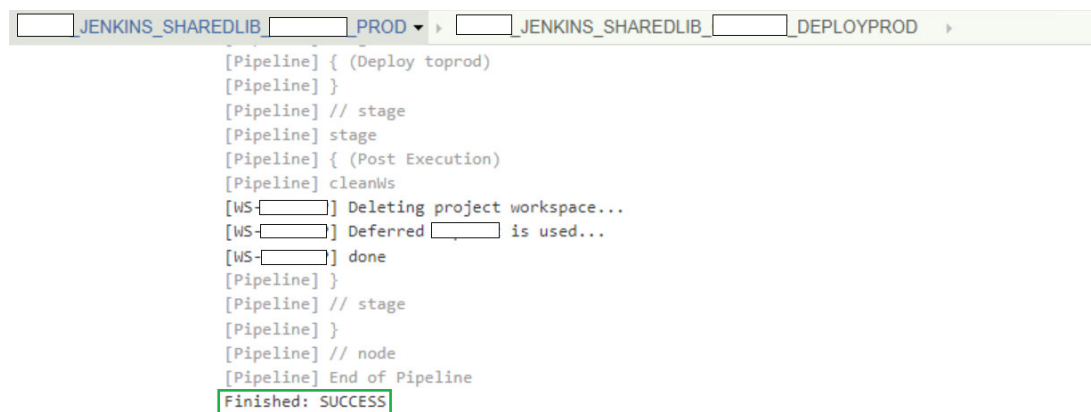
I. Ratificación.

El Analista de QA del equipo ratificó la correcta ejecución de los pipelines, esta vez en el ambiente de producción revisando los logs generados en Jenkins. Adicionalmente se revisó la evidencia adjuntada al ticket por parte del Agile Operator, en la cual se verificó que el API se había desplegado correctamente en el API Management de Azure en producción.

En la Figura 36 se muestra el log del pipeline para la promoción de fuentes en el repositorio Shared Library en el ambiente de producción.

Figura 36

Log de pipeline de entrega en repositorio Shared Library de producción



```
[Pipeline] { (Deploy toprod)
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Post Execution)
[Pipeline] cleanWs
[WS-] Deleting project workspace...
[WS-] Deferred is used...
[WS-] done
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Nota. La figura muestra el log de la ejecución satisfactoria del pipeline para la promoción de fuentes en el repositorio Shared Library en el ambiente de producción. Tomado de la herramienta Jenkins interna de la Entidad Financiera, 2021.

En la Figura 37 se muestra el log del pipeline para la promoción de fuentes en el repositorio Templates en el ambiente de producción.

Figura 37

Log de pipeline de entrega en repositorio Templates de producción

```
TEMPLATE-DEPLOY-AZURE- PROD ▾ ▶  
[Pipeline] { (Deploy to prod)  
[Pipeline] }  
[Pipeline] // stage  
[Pipeline] stage  
[Pipeline] { (Post Execution)  
[Pipeline] cleanWs  
[WS-] Deleting project workspace...  
[WS-] Deferred  is used...  
[WS-] done  
[Pipeline] }  
[Pipeline] // stage  
[Pipeline] }  
[Pipeline] // node  
[Pipeline] End of Pipeline  
Finished: SUCCESS
```

Nota. La figura muestra el log de la ejecución satisfactoria del pipeline para la promoción de fuentes en el repositorio Templates en el ambiente de producción. Tomado de la herramienta Jenkins interna de la Entidad Financiera, 2021.

En la Figura 38 se muestra el log del pipeline para el despliegue de APIs en Azure API Management en el ambiente de producción.

Figura 38

Log de pipeline de despliegue en API Management de producción

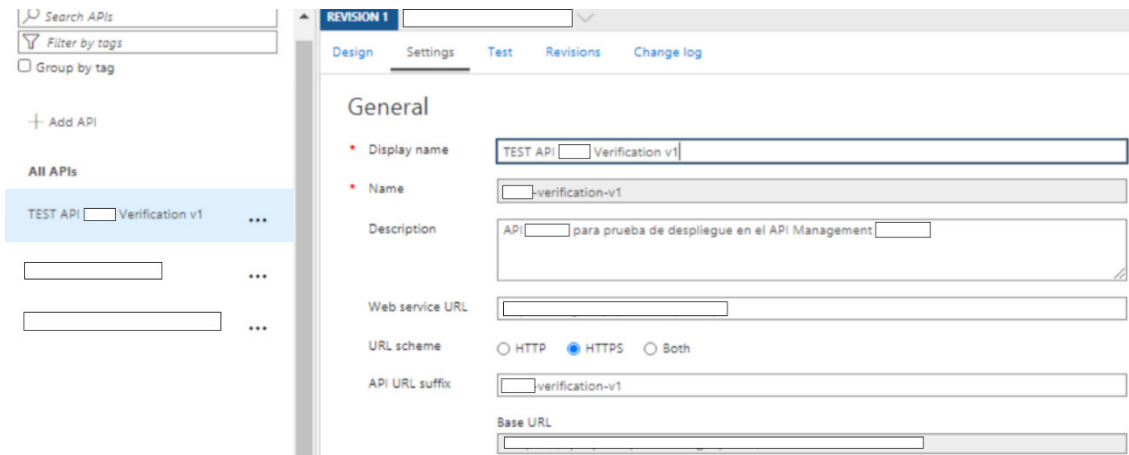
```
TEMPLATE-DEPLOY-APIS-APIM-PROD ▾ ▶  
responseCode: 201  
[Pipeline] echo  
El api -verification-v1 fue creada/actualizada exitosamente.  
[Pipeline] sh  
+ docker rm -f   
[Pipeline] sh  
+ rm -rf ****  
[Pipeline] }  
[Pipeline] // withCredentials  
[Pipeline] }  
[Pipeline] // withDockerServer  
[Pipeline] }  
[Pipeline] // withDockerRegistry
```

Nota. La figura muestra el log de la ejecución satisfactoria del pipeline para el despliegue de APIs en Azure API Management en el ambiente de producción. Tomado de la herramienta Jenkins interna de la Entidad Financiera, 2021.

En la Figura 39 se muestra el API de prueba desplegada en Azure API Management del ambiente de producción.

Figura 39

API desplegada en API Management de producción



Nota. La figura muestra la API desplegada a través del pipeline en el API Management en el ambiente de producción. Tomado de la plataforma Azure API Management de la Entidad Financiera, 2021.

Finalizadas las revisiones, no se encontraron observaciones y el Analista de QA movió el ticket al carril de Cierre.

J. Reversión de cambios.

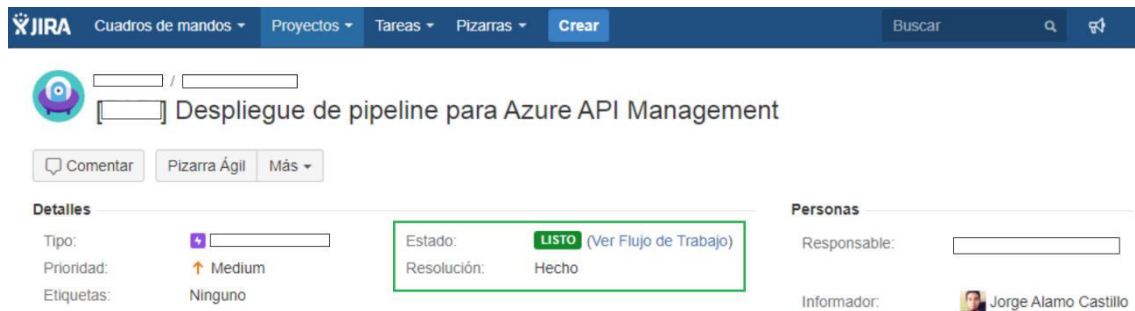
El ticket de cambio no pasó por este carril ya que los pipelines se ejecutaron correctamente en el ambiente de producción y el Analista de QA dio conformidad a la ratificación del cambio.

K. Cierre.

Finaliza el pase a producción de los cambios y el ticket en Jira se coloca en estado listo, como se muestra en la Figura 40.

Figura 40

Ticket de cambio finalizado



Nota. La figura muestra el estado listo en el ticket de cambio utilizado en el despliegue de la solución. Tomado de la herramienta Jira interna de la Entidad financiera, 2021.

Finalizado el pase a producción, se publicó y notificó a los equipos internos de la organización respecto al documento del modelo operativo para el pipeline de despliegue de APIs en Azure API Management, en el cual se indican los pasos que deben seguir los equipos de desarrollo para utilizar la automatización.

En el modelo operativo se hace referencia al repositorio Project y los pipelines de despliegue asociados que fueron utilizados en la implementación, y se detalla los roles utilizados en el service principal para autenticarse al API Management en Azure. De esta forma, los equipos de desarrollo pueden utilizarlos como guía para replicar el pipeline automatizado y realizar el despliegue de sus APIs en Azure API Management.

3.3 EVALUACIÓN

La automatización de tareas manuales en el procedimiento de despliegue de APIs en la plataforma Azure API Management descrita en el presente Trabajo de Suficiencia Profesional, no implicó un ingreso económico propiamente para la Entidad Financiera, sin embargo, representó un ahorro de tiempo con respecto al uso del procedimiento manual. A fin de realizar la evaluación económica de la solución implementada, se ha optado por cuantificar el valor monetario de este ahorro de tiempo en base al sueldo del personal involucrado en el procedimiento.

Se debe considerar que los valores reflejados en este análisis económico son referenciales, ya que no se tiene conocimiento exacto de los valores reales.

3.3.1 EVALUACIÓN COSTO-BENEFICIO

3.3.1.1 Análisis de costos.

El proyecto de automatización se llevó a cabo en un periodo de 3 meses, finalizando en el primer trimestre del año 2021. La inversión realizada se sustentó únicamente en el pago al personal implicado en el desarrollo de la solución, no se consideran los costos asociados a licencias de software o herramientas tecnológicas debido a que estas habían sido adquiridas previamente por la Entidad Financiera.

El costo total requerido para llevar a cabo el proyecto fue de S/ 25,500, tal como se detalla en la Tabla 13.

Tabla 13

Costo total del proyecto

Personal	Asignación	Sueldo mensual	Sueldo por 3 meses
Analista Programador	100%	S/ 6,000	S/ 18,000
Arquitecto de Tecnología	25%	S/ 7,000	S/ 5,250
Analista de QA	15%	S/ 5,000	S/ 2,250
Costo total			S/ 25,500

Nota. Elaboración propia.

3.3.1.2 Análisis de beneficios.

Los beneficios del proyecto se calcularon a partir de estimar el valor monetario que genera el ahorro de tiempo al utilizar el procedimiento automatizado en lugar del procedimiento manual, como se puede apreciar en la Ecuación 1.

$$Beneficio_{total} = Costo\ proc.\ manual_{total} - Costo\ proc.\ auto_{total} \quad (1)$$

Para calcular el costo total asociado a la utilización del procedimiento manual, se consideraron los siguientes escenarios comentados previamente como parte de la situación problemática:

- El tiempo de espera para la atención de las solicitudes de despliegue de APIs en el API Management dirigidas al equipo de API Platform en el ambiente de desarrollo. Esta espera afecta a los Analistas Programadores encargados de realizar el desarrollo de las APIs.
- Posibles errores en la plataforma API management producto de configuraciones manuales ejecutadas incorrectamente. Esto afecta a los Arquitectos de Tecnología del equipo API Platform debido a que en caso de producirse algún error, se les generará un incidente para solucionar el problema.

De los escenarios señalados, se tiene que el costo total asociado al procedimiento manual se calcula a partir de sumar el costo total que implica el tiempo de espera (T_e) para la atención de solicitudes de despliegue de APIs y el costo total que implica el tiempo requerido para solucionar los incidentes (T_s). Esta relación se describe en la Ecuación 2.

$$\text{Costo proc. manual}_{total} = \text{Costo } T_{e_{total}} + \text{Costo } T_{s_{total}} \quad (2)$$

En el cálculo del costo total del T_e se consideró el tiempo promedio de espera (T_{pe}) para la atención de solicitudes generadas en el ambiente de desarrollo para el despliegue de APIs en el API Management y la cantidad promedio de solicitudes (P_s) recibidas por el equipo de API Platform mensualmente, como se detalla en la Tabla 14.

Tabla 14

Costo por el tiempo de espera en la atención de solicitudes

Personal afectado	Sueldo mensual	Sueldo mensual/ hr.	T_{pe} (hrs.)	P_s mensual	Costo T_e total mensual
Anl. Programador	S/ 6,000	S/ 25	4	32	S/ 3,200

Nota. Elaboración propia.

En el cálculo del costo total del Ts se consideró el tiempo promedio para la solución de incidentes (Tps) generados por errores en configuraciones manuales sobre el API Management y la cantidad promedio de incidentes (Pi) recibidos por el equipo de API Platform mensualmente, tal como se detalla en la Tabla 15.

Tabla 15

Costo por el tiempo para la solución de incidentes

Personal afectado	Sueldo mensual	Sueldo mensual/hr.	Tps (hrs.)	Pi mensual	Costo Ts total mensual
Arq. de Tecnología	S/ 7,000	S/ 29.17	1.5	7	S/ 306.25

Nota. Elaboración propia.

De acuerdo con la Ecuación 2, se tiene que el costo total mensual asociado a la utilización del procedimiento manual fue de S/ 3,506.25.

Para calcular el costo total asociado a la utilización del procedimiento automatizado, se consideró el tiempo promedio para la ejecución (Tpx) del procedimiento de despliegue de APIs en el API Management utilizando el pipeline y la cantidad promedio de ejecuciones (Px) del pipeline que para efectos del análisis se le asignó el mismo valor del Ps, como se detalla en la Tabla 16.

Tabla 16

Costo del procedimiento automatizado

Personal involucrado	Sueldo mensual	Sueldo mensual/hr.	Tpx (hrs.)	Px mensual	Costo proc. auto. total mensual
Anl. Programador	S/ 6,000	S/ 25	0.05	32	S/ 40

Nota. Elaboración propia.

Finalmente, según la Ecuación 1 se tiene que el beneficio total del proyecto se estima en S/ 3,466.25 mensuales, lo que vendría a ser S/ 41,595 anuales.

3.3.1.3 Evaluación del resultado.

La relación costo-beneficio se calculó a partir de los valores estimados al primer año del proyecto y para efectos de este análisis se ha considerado una tasa de descuento del 8% anual.

Con estas consideraciones se tiene que la relación costo-beneficio es de 1.51, a partir de lo cual se puede concluir que por cada S/ 1 invertido se está obteniendo un beneficio de S/ 1.51 haciendo que el proyecto sea rentable económicamente dentro del primer año.

3.3.2 INTERPRETACIÓN DEL VAN Y DEL TIR

De la evaluación costo-beneficio se tiene como estimación que para el proyecto se invirtió un total de S/ 25,500, obteniendo S/ 41,595 de beneficio en el primer año. A partir de estos valores se realizó el cálculo del VAN y TIR considerando una tasa de descuento del 8 % anual para fines de este análisis.

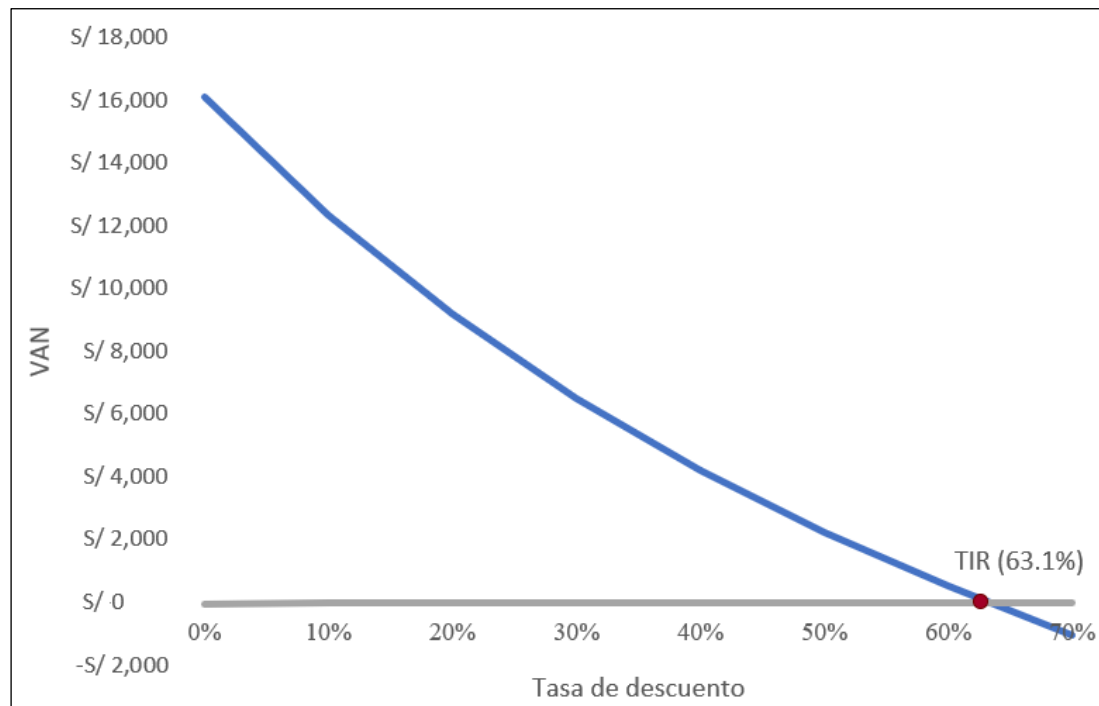
En el caso del VAN se calcula un valor de S/ 13,013.89, de lo cual se puede interpretar que el proyecto generó beneficios dentro primer año por encima de la inversión realizada.

Para el TIR se calcula un valor del 63.1%, de lo cual se puede interpretar que la tasa interna de retorno es mayor a la tasa de descuento considerada en el análisis, es decir, el proyecto genera una rentabilidad que es superior a la mínima requerida dentro del primer año.

En la Figura 41 se muestra la relación existente entre el VAN y el TIR aplicados al proyecto de automatización.

Figura 41

Relación entre VAN y TIR



Nota. Elaboración propia.

CAPÍTULO IV – REFLEXIÓN CRÍTICA DE LA EXPERIENCIA

El autor del presente Trabajo de Suficiencia Profesional ejerció el rol de Analista Programador durante el proyecto, lo que significó una oportunidad para adquirir nuevas competencias, afianzar conocimientos de programación y comprender los procesos internos de la Entidad Financiera. Del aprendizaje y experiencia obtenidos se destacan los siguientes:

- Este proyecto representó la primera oportunidad que el autor tuvo de encabezar la implementación de una solución tecnológica dentro de la Entidad Financiera, para lo cual el uso de la metodología ágil Kanban fue importante al permitir gestionar las actividades del proyecto de manera fluida y expresar claramente los progresos conseguidos a través de indicadores visuales.
- La comunicación y feedback obtenido por parte del equipo permitió incorporar sus experiencias trabajando anteriormente con proyectos de automatización respecto a cómo implementar los lineamientos y estándares tecnológicos definidos por la Entidad Financiera durante el desarrollo de la solución.
- Una lección aprendida en el equipo fue la importancia de hacer un debido seguimiento a los accesos necesarios para el desarrollo de las actividades, a fin de poder gestionar su renovación oportunamente cuando estos estén próximos a vencer. Los problemas de accesos ocasionaron retrasos al inicio del proyecto hasta que pudieron ser renovados.
- Durante el despliegue en el ambiente de certificación, se presentaron problemas al versionar el contrato del API, esto fue revisado con el equipo de DevOps y se determinó que el inconveniente era causado por una incorrecta configuración en el archivo de despliegue del pipeline. La respuesta rápida para solucionar el problema y el apoyo del Líder Técnico del equipo permitieron evitar retrasos en los compromisos de entrega.
- La implementación del pipeline automatizado tuvo una buena recepción por parte de los equipos de desarrollo y significó un aporte beneficioso para mejorar el esquema de trabajo que se venía utilizando respecto al despliegue de APIs en el API Management de Azure.

- Participar en este proyecto permitió al autor crecer profesionalmente al adquirir experiencia trabajando con componentes en la nube de Azure, herramientas de integración y entrega continua, y herramientas para la administración de proyectos, así como tener la oportunidad de aplicar el conocimiento adquirido en el desarrollo de una solución ampliamente utilizada en la Entidad Financiera.

CAPÍTULO V – CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- Se logró automatizar las tareas manuales de registro, actualización y eliminación de APIs en la plataforma Azure API Management a través de un pipeline automatizado, utilizando herramientas de integración y entrega continua.
- Se logró reducir en un 98.75% el tiempo de espera para el despliegue de APIs en la plataforma Azure API Management del ambiente de desarrollo. Los equipos de desarrollo de APIs disponen de un pipeline automatizado que pueden replicar fácilmente y ejecutar directamente en el momento que lo necesiten.
- Se logró disminuir la exposición de la plataforma Azure API Management a configuraciones manuales. El uso del pipeline automatizado eliminó la necesidad de que un operario intervenga para ejecutar las tareas de registro, actualización o eliminación de APIs en el API Management de forma manual, reduciendo en un 100% los posibles errores generados al realizar estas tareas manuales incorrectamente.
- Se analizó en detalle cada una de las actividades realizadas en el procedimiento manual para el despliegue de APIs en el API Management, lo cual facilitó el desarrollo de las funcionalidades del pipeline automatizado.
- Se realizó una adecuada planificación y definición de actividades a realizar, lo cual permitió disponer de un margen de tiempo para lidiar con las complicaciones que surgieron durante la implementación, esto sin afectar los compromisos de entrega del proyecto.
- Se diseñó la solución en cumplimiento de los lineamientos y buenas prácticas definidas por la Entidad Financiera. Se utilizaron las herramientas estandarizadas para la implementación de automatizaciones como son Jenkins, Bitbucket y Artifactory, aplicando buenas prácticas como el versionamiento de artefactos binarios y la utilización de una librería compartida para centralizar las funcionalidades del pipeline automatizado.

5.2 RECOMENDACIONES

- Difundir los principios y prácticas relacionadas a la integración y entrega continua de software a fin de facilitar su adopción al interior de los equipos internos de la Entidad Financiera.
- A efectos de mejorar la calidad del código en nuevos desarrollos, se recomienda incorporar una herramienta de análisis estático de código como SonarQube al pipeline de promoción de fuentes en el repositorio Shared Library (librería compartida).
- Para proyectos similares que requieran automatizar la administración de recursos en Azure, se sugiere aplicar la práctica de infraestructura como código. Este enfoque se basa en utilizar archivos de configuración (ARM templates) para implementar y administrar recursos en la nube de Azure de forma ágil y segura.
- En próximas versiones a implementar se recomienda evaluar la incorporación de Azure REST API como alternativa al uso de Azure CLI, lo cual permitirá disponer de las nuevas capacidades publicadas periódicamente por Azure para la administración de recursos en la nube sin la necesidad de gestionar la publicación de una nueva imagen Docker con una versión actualizada de Azure CLI.

5.3 FUENTES DE INFORMACIÓN

Subra, J., y Vannieuwenhuysen, A. (2018). Scrum: un método ágil para sus proyectos.

Li, P. (2018). Jira Software Essentials: Plan, track, and release great applications with Jira Software.

Atlassian (2022a). ¿Para qué sirve Jira? Recuperado de <https://www.atlassian.com/es/software/jira/guides/use-cases/what-is-jira-used-for#jira-for-agile-teams>

Microsoft (29 de mayo de 2022a). What is the Azure CLI? Recuperado de <https://docs.microsoft.com/en-us/cli/azure/what-is-azure-cli>

Microsoft (26 de mayo de 2022b). What are ARM templates? Recuperado de <https://docs.microsoft.com/en-us/azure/azure-resource-manager/templates/overview>

Microsoft (01 de junio de 2022c). Introduction to Container registries in Azure. Recuperado de <https://docs.microsoft.com/en-us/azure/container-registry/container-registry-intro>

Barbieri, L., y Bonanni, M. (2019). Mastering Azure Serverless Computing: A practical guide to building and deploying enterprise-grade serverless applications using Azure Functions.

Budzynski, M. (12 de febrero de 2019). Benefits of using Azure API Management with microservices. Recuperado de <https://azure.microsoft.com/en-us/blog/benefits-of-using-azure-api-management-with-microservices/>

Microsoft (24 de marzo de 2022d). Policies in Azure API Management. Recuperado de <https://docs.microsoft.com/en-us/azure/api-management/api-management-howto-policies>

Smartbear (01 de junio de 2022). About SwaggerHub. Recuperado de <https://support.smartbear.com/swaggerhub/docs/about.html>

Patil, A., y Soni, M. (2021). Hands-on Pipeline as Code with Jenkins: CI/CD Implementation for Mobile, Web, and Hybrid Applications Using Declarative Pipeline in Jenkins.

Atlassian (2022b). A brief overview of Bitbucket. Recuperado de <https://bitbucket.org/product/guides/getting-started/overview#a-brief-overview-of-bitbucket>

Docker, Inc. (2021). Docker overview. Recuperado de <https://docs.docker.com/get-started/overview/>

JFrog Ltd. (2022). Universal - All Your Binaries In One Place. Recuperado de <https://jfrog.com/artifactory/features/>

Davis, A. (2019). Learning Groovy 3: Java-Based Dynamic Scripting.

Red Hat, Inc. (31 de octubre de 2017). ¿Qué es una API? Recuperado de <https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces>

IBM (06 de abril de 2021). ¿Qué es una API REST? Recuperado de <https://www.ibm.com/pe-es/cloud/learn/rest-apis>

IBM (07 de abril de 2022). Bibliotecas compartidas. Recuperado de <https://www.ibm.com/docs/es/integration-bus/10.0?topic=libraries-shared>

Red Hat, Inc. (31 de enero de 2018). ¿Qué son la integración y la distribución continuas (CI/CD)? Recuperado de <https://www.redhat.com/es/topics/devops/what-is-ci-cd>

AWS, Inc. (2022a). ¿Qué es la integración continua? Recuperado de <https://aws.amazon.com/es/devops/continuous-integration/>

AWS, Inc. (2022b). ¿Qué es la entrega continua? Recuperado de <https://aws.amazon.com/es/devops/continuous-delivery/>

Microsoft (01 de junio de 2022e). Application and service principal objects in Azure Active Directory. Recuperado de <https://docs.microsoft.com/en-us/azure/active-directory/develop/app-objects-and-service-principals>

Jenkins.io (s.f.). Getting started with Pipeline. Recuperado de <https://www.jenkins.io/doc/book/pipeline/getting-started/>

5.4 GLOSARIO

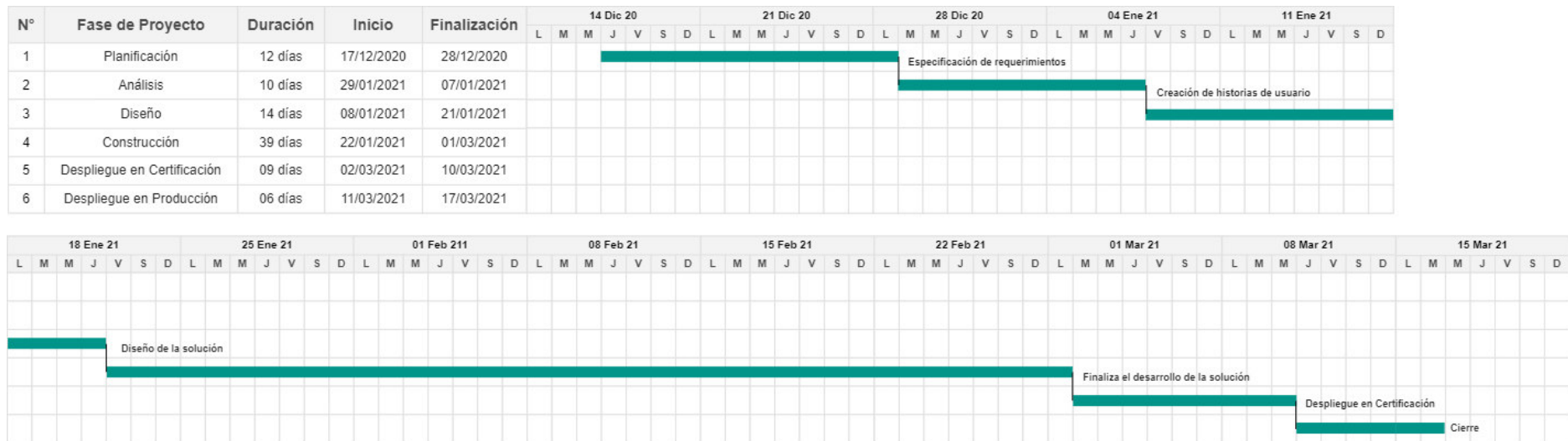
- Workflow: conjunto de tareas que guardan relación entre sí y se ejecutan siguiendo una secuencia de etapas o fases con la finalidad de automatizar un proceso o procedimiento.
- Backlog: lista de actividades o historias de usuario pendientes de realizar para el desarrollo de un proyecto, las cuales cuentan con una descripción, orden y estimación de tiempo para su realización.
- Historia de usuario: representa una descripción sencilla de los requerimientos de un producto de software desde el punto de vista del usuario.
- Template: archivo que define una estructura modelo que se puede utilizar como patrón para la creación de un elemento de software.
- Hostname: nombre asignado a la infraestructura que aloja un servicio para ser identificada dentro de una red informática respecto a otros servicios sin la necesidad de utilizar direcciones IP.
- Feedback: comunicación fluida entre los integrantes de equipo con la finalidad de compartir ideas y opiniones constructivas que sirvan para impulsar la mejora constante, el fortalecimiento de capacidades y el aumento de la productividad.
- Contrato: documento que describe el funcionamiento de un API y las operaciones que expone, definiendo la estructura para la comunicación con otros servicios a través de solicitudes y respuestas.
- Log: archivo de texto en el que se graban de forma secuencial los detalles de la ejecución de un procedimiento o sistema informático.
- Runtime: etapa que describe el periodo de tiempo en el que se ejecutan las instrucciones asociadas a un programa de software.

ANEXOS

ANEXO 1 – CRONOGRAMA DEL PROYECTO

Figura 42

Cronograma del Proyecto



Nota. Elaboración propia. Basado en el cronograma definido para el proyecto en la herramienta Jira interna de la Entidad financiera, 2021

ANEXO 2 – COMPARACIÓN ENTRE EL PROCEDIMIENTO MANUAL Y EL PROCEDIMIENTO AUTOMATIZADO

Para este análisis se ha considerado evaluar los escenarios descritos en la situación problemática, siendo el tiempo de espera para la atención de solicitudes de despliegue de APIs en Azure API Management y los errores generados producto de configuraciones manuales ejecutadas incorrectamente sobre la plataforma. De los resultados obtenidos se realizó la comparativa entre el procedimiento manual y el procedimiento automatizado.

1. PROCEDIMIENTO MANUAL

Para el procedimiento manual se consideró el tiempo de espera para la atención de las solicitudes generadas en el ambiente de desarrollo y el tiempo empleado en solucionar los incidentes causados por configuraciones manuales ejecutadas incorrectamente. Se analizaron las solicitudes e incidentes dirigidos al equipo de API Platform en los 6 meses previos a la implementación del procedimiento automatizado.

En la Entidad Financiera se cuenta con la herramienta Remedy para la gestión de tickets de soporte, de la cual se obtuvo el reporte de solicitudes e incidentes generados al equipo de API Platform en el periodo de septiembre del 2020 a febrero de 2021, como se muestra en la Figura 43.

Figura 43

Reporte de solicitudes e incidentes



ID	ID de petición de servicio	Resumen	Prioridad	Servicio	Estado	Estado de SLM	Fecha de inicio real	Fecha completada
0005590	000920	Registro de API's en e Media			Cerrado		10/09/2020 17:27:58	10/09/2020 17:27:58
0005583	000913	Registro Swagger del Baja			Cerrado		09/09/2020 14:15:00	09/09/2020 14:15:00
0005583	000913	Registro Swagger del Baja			Cerrado		09/09/2020 14:02:16	09/09/2020 14:02:16
0005579	000910	Consulta de API Manz Baja			Cerrado		09/09/2020 10:43:29	09/09/2020 10:46:56
0005568	000896	Validación de configur Baja			Cerrado		03/09/2020 17:41:14	03/09/2020 17:41:26
0005555	000883	Consulta de API Manz Baja			Cerrado		01/09/2020 9:38:19	01/09/2020 9:43:41
0005545	000872	Registro de API en el , Alta			Cerrado		27/08/2020 16:54:34	27/08/2020 16:54:34
0005539	000865	Registro de API en el , Alta			Cerrado		26/08/2020 16:03:17	26/08/2020 16:03:17
0005530	000856	Información sobre API Media			Cerrado		26/08/2020 9:30:19	26/08/2020 9:30:19
0005529	000856	Información de apis re Media			Cerrado		26/08/2020 9:29:33	26/08/2020 9:29:33
0005530	000856	Consulta de API Manz Baja			Cerrado		25/08/2020 14:52:05	26/08/2020 11:38:36
0005508	000833	Registro Swagger del Baja			Cerrado		18/08/2020 11:21:24	18/08/2020 11:21:24
0005503	000828	Consulta de API Manz Baja			Cerrado		17/08/2020 10:07:18	17/08/2020 12:16:17
0005501	000828	Registrar en el API M: Media			Cerrado		14/08/2020 13:23:10	14/08/2020 13:23:10

Nota. La figura muestra el reporte de solicitudes e incidentes generados al equipo de API Platform. Tomado de la herramienta interna Remedy de la Entidad Financiera.

De los reportes obtenidos se identificaron las solicitudes relacionadas al despliegue de APIs en el API Management, a partir de las cuales se estimó la cantidad promedio de solicitudes recibidas y el tiempo promedio de espera para la atención de estas solicitudes.

En la Tabla 17 se muestra el detalle de las solicitudes de despliegue y los tiempos de espera asociados.

Tabla 17

Atención de solicitudes de despliegue

Año	Mes	Solicitudes recibidas	Tiempo promedio de espera por solicitud (hrs.)
2020	Septiembre	40	5.2
	Octubre	33	4.3
	Noviembre	32	3.7
	Diciembre	28	4.1
2021	Enero	31	3.8
	Febrero	29	4.2
Promedio total		32	4

Nota. Elaboración propia.

Como resultado se tiene que dentro de los 6 meses evaluados, el equipo de API Platform recibió en promedio 32 solicitudes al mes para el despliegue de APIs en el API Management, teniendo cada una un tiempo de espera promedio de 4 horas para ser atendida desde que la solicitud fue recibida hasta que fue completada.

Adicionalmente, se identificaron los incidentes relacionados a configuraciones manuales de registro, actualización y eliminación de APIs aplicadas incorrectamente sobre la plataforma Azure API Management, a partir de las cuales se determinó la cantidad de incidentes generados y el tiempo promedio requerido para solucionarlos, como se muestra en la Tabla 18.

En la Tabla 18 se muestra el detalle de los incidentes reportados y tiempos requeridos para solucionarlos.

Tabla 18

Atención de incidentes

Año	Mes	Incidentes reportados	Tiempo promedio de resolución por incidente (hrs.)
2020	Septiembre	5	2
	Octubre	9	1.5
	Noviembre	6	1.7
	Diciembre	7	1.1
2021	Enero	4	1
	Febrero	10	1.5
Promedio total		7	1.5

Nota. Elaboración propia.

Como resultado se tiene que dentro de los 6 meses evaluados, se reportaron 7 incidentes en promedio por mes al equipo de API Platform relacionadas a configuraciones manuales aplicadas incorrectamente sobre la plataforma API Management, siendo cada uno resuelto en un tiempo promedio de 1.5 horas desde que el incidente fue tomado por un operario hasta que fue solucionado.

2. PROCEDIMIENTO AUTOMATIZADO

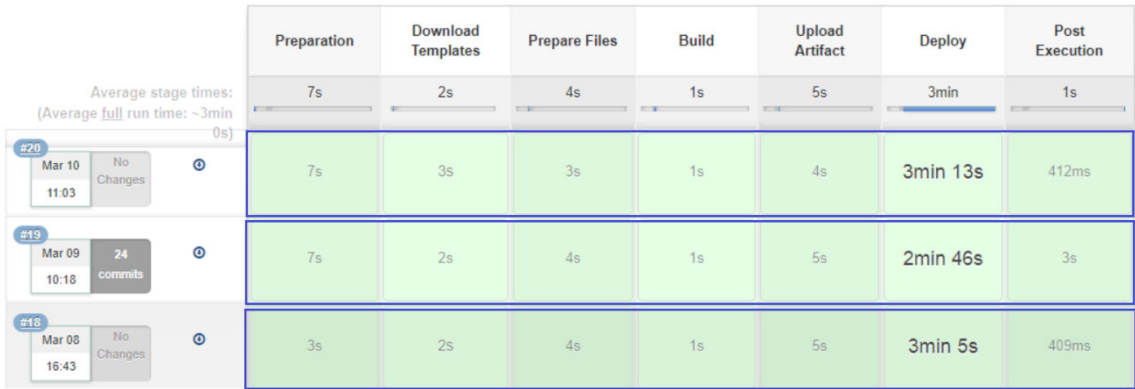
Para el procedimiento automatizado se ha considerado el tiempo de ejecución promedio desde que se inició el pipeline hasta que finalizó el despliegue.

De las ejecuciones del pipeline realizadas en el ambiente de desarrollo, se tiene que el tiempo promedio de ejecución empleado para realizar el despliegue de APIs en el API Management es de 3 minutos.

En la Figura 44 se muestran los tiempos de ejecución obtenidos al utilizar el pipeline de despliegue en desarrollo.

Figura 44

Tiempos de ejecución en pipeline de despliegue



Average stage times:
(Average full run time: ~3min 0s)

	Preparation	Download Templates	Prepare Files	Build	Upload Artifact	Deploy	Post Execution
#20 Mar 10 11:03 No Changes	7s	3s	3s	1s	4s	3min 13s	412ms
#19 Mar 09 10:18 24 commits	7s	2s	4s	1s	5s	2min 46s	3s
#18 Mar 08 16:43 No Changes	3s	2s	4s	1s	5s	3min 5s	409ms

Nota. La figura muestra el tiempo de ejecución empleado por el pipeline para realizar el despliegue de APIs en el API Management. Tomado de la herramienta Jenkins interna de la Entidad Financiera, 2021.

3. RESULTADOS OBTENIDOS

En el caso del tiempo de espera para la atención de solicitudes de despliegue de APIs en Azure API Management, se tienen los siguientes resultados:

- El tiempo promedio de espera asociado al uso del procedimiento manual fue de 4 horas. Este valor considera el tiempo transcurrido desde que se recibió la solicitud de despliegue hasta que fue atendida por el equipo de API Platform.
- El tiempo promedio de espera asociado al procedimiento automatizado es representado por el tiempo que le toma al pipeline ejecutar el despliegue, siendo en promedio 3 minutos.

De estos resultados se concluye que utilizar el procedimiento automatizado reduce en un 98.75% el tiempo de espera para la publicación de APIs en el API Management del ambiente de desarrollo.

Respecto a los errores generados producto de configuraciones manuales ejecutadas incorrectamente sobre la plataforma Azure API Management, se tienen los siguientes resultados:

- Utilizando el procedimiento manual, se reportaban en promedio 7 incidentes al mes relacionados con errores generados por configuraciones manuales realizadas incorrectamente para el registro, actualización y eliminación de APIs en el API Management, requiriendo cada uno 1.5 horas en promedio para ser resuelto por el equipo de API Platform.
- Utilizando el procedimiento automatizado no se presentan este tipo de errores, ya que la configuración es aplicada sin la intervención de un operario que ejecute las tareas manualmente.

De estos resultados se concluye que utilizar el procedimiento automatizado reduce en un 100% los posibles errores generados por configuraciones manuales realizadas incorrectamente para el registro, actualización y eliminación de APIs en el API Management.