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

Implementación de microservicios para integrar la información de productos en una empresa retail

TRABAJO DE SUFICIENCIA PROFESIONAL

Para optar el Título Profesional de Ingeniero de Sistemas

AUTOR

Juan Luis CASTRO QUEZADA

ASESOR

Mg. Santiago Domingo MOQUILLAZA HENRÍQUEZ

Lima, Perú

2022

Referencia bibliográfica

Castro, J. (2022). *Implementación de microservicios para integrar la información de productos en una empresa retail*. [Trabajo de suficiencia profesional de pregrado, Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería de Sistemas e Informática, Escuela Profesional de Ingeniería de Sistemas]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	JUAN LUIS CASTRO QUEZADA
Tipo de documento de identidad	DNI
Número de documento de identidad	48065902
URL de ORCID	https://orcid.org/0000-0003-2250-9564
Datos de asesor	
Nombres y apellidos	Santiago Domingo Moquillaza Henríquez
Tipo de documento de identidad	DNI
Número de documento de identidad	08280889
URL de ORCID	https://orcid.org/0000-0001-9531-881X
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	Frank Edmundo Escobedo Bailón
Tipo de documento	DNI
Número de documento de identidad	41671087
Miembro del jurado 1	
Nombres y apellidos	Rosa Menéndez Mueras
Tipo de documento	DNI
Número de documento de identidad	10246770
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 Sistemas

Acta Virtual de Sustentación
del Trabajo de Suficiencia Profesional

Siendo las 21:01 horas del día 17 de agosto del año 2022, se reunieron virtualmente los docentes designados como Miembros de Jurado del Trabajo de Suficiencia Profesional, presidido por el Dr. Escobedo Bailón Frank Edmundo (Presidente), Mg. Menéndez Mueras Rosa (Miembro) y el Mg. Moquillaza Henríquez Santiago Domingo (Miembro Asesor), usando la plataforma Meet (https://meet.google.com/tfs-rxrv-fza), para la sustentación virtual del Trabajo de Suficiencia Profesional intitulado: ***"IMPLEMENTACIÓN DE MICROSERVICIOS PARA INTEGRAR LA INFORMACIÓN DE PRODUCTOS EN UNA EMPRESA RETAIL",*** *por el Bachiller* ***Castro Quezada Juan Luis;*** *para obtener el Título Profesional de Ingeniero de Sistemas.*

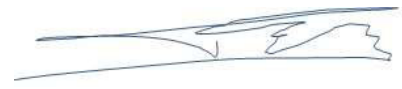
Acto seguido de la exposición del Trabajo de Suficiencia Profesional, el Presidente invitó al Bachiller a dar las respuestas a las preguntas establecidas por los miembros del Jurado.

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 ***19 (DIECINUEVE).***

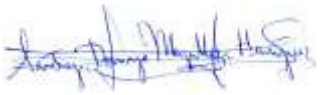
A continuación el Presidente de Jurados el Dr. Escobedo Bailón Frank Edmundo, declara al Bachiller ***Ingeniero de Sistemas****.*

Siendo las ***21:44*** *horas, se levantó la sesión.*



Presidente
Dr. Escobedo Bailón Frank Edmundo

Miembro
Mg. Menéndez Mueras Rosa



Miembro Asesor
Mg. Moquillaza Henríquez Santiago Domingo

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
Universidad del Perú, DECANA DE AMÉRICA
FACULTAD DE INGENIERIA DE SISTEMAS E INFORMATICA
Escuela Profesional de Ingeniería de Sistemas

INFORME DE EVALUACIÓN DE ORIGINALIDAD

1. Facultad de Ingeniería de Sistemas e Informática
2. Escuela Profesional de Ingeniería de Sistemas
3. Autoridad académica que emite el informe de originalidad
   Directora(e) de la EPIS
4. Apellidos y Nombres de la autoridad académica
   Dra. Luzmila E. Pró Concepción
5. Operador del programa informático de similitudes
   Dra. Luzmila E. Pró Concepción
6. Documento evaluado
   Título de pregrado: **"Implementación de Microservicios para Integrar la Información de Productos en una Empresa Retail"**
7. Autor del documento
   Bach. **Castro Quezada, Juan Luis**
8. Fecha de recepción del documento **10/09/**2022
9. Fecha de aplicación del programa informático de similitudes **10/09/**2022
10. Software utilizado
    - Turnitin
11. Configuración del programa detector de similitudes
    - Excluye textos entrecomillados
    - Excluye bibliografía
    - Excluye cadenas menores a 40 palabras
12. Porcentaje de similitudes según programa detector de similitudes **4 (cuatro)%**
13. Fuentes originales de las similitudes encontradas
    Se adjunta en el anexo 1
14. Observaciones
15. Calificación de originalidad
    - Documento cumple criterios de originalidad, sin observaciones [x]
    - Documento cumple criterios de originalidad, con observaciones [ ]
    - Documento no cumple criterios de originalidad [ ]
16. Fecha de informe **16/09/2022**

UNMSM
Firmado digitalmente por PRO CONCEPCION Luzmila Elisa FAU 20148092282 soft
Motivo: Soy el autor del documento
Fecha: 14.11.2022 12:57:08 -05:00

Firma de evaluador
Dra. Luzmila E. Pró Concepción
Directora (e) de la EPIS

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
Universidad del Perú, DECANA DE AMÉRICA
FACULTAD DE INGENIERIA DE SISTEMAS E INFORMATICA
Escuela Profesional de Ingeniería de Sistemas

ANEXO 1

Fuentes originales de las similitudes encontradas

1. hdl.handle.net: 3%
2. cybertesis.unmsm.edu.pe: <1%
2. www.scrumguides.org: <1%
3. sedici.unpl.edu.ar: <1%

UNMSM
Firmado digitalmente por PRO CONCEPCION Luzmila Elisa FAU 20148092282 soft
Motivo: Soy el autor del documento
Fecha: 14.11.2022 12:56:43 -05:00

Firma de evaluador
Dra. Luzmila E. Pró Concepción
Directora (e) de la EPIS

DEDICATORIA

Dedico este trabajo a mi familia, en especial a mi madre, y a mis amigos cercanos quienes siempre me impulsaron a mejorar día a día.

AGRADECIMIENTOS

A mis familiares y amigos cercanos por la confianza.

A los profesores y compañeros del trabajo y estudio que contribuyeron en mi crecimiento profesional.

A mi asesor por todo el apoyo brindado para realizar este trabajo.

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

Implementación de microservicios para integrar la información de productos en una empresa retail

Autor: Castro Quezada, Juan Luis
Asesor: Moquillaza Henríquez, Santiago Domingo
Título: Trabajo de Suficiencia Profesional para optar el Título Profesional de Ingeniero de Sistemas
Fecha: Junio de 2022

RESUMEN

El área comercial corporativa de un negocio retail necesitaba mantener actualizada la información de los productos administrados por su plataforma de importaciones con el catálogo de productos corporativo. El catálogo ofrecería beneficios tales como actuar de intermediario para intercambiar dicha información entre diferentes plataformas, además de ser una fuente centralizada, estandarizada y validada de la información de productos para diferentes unidades de negocio. Para ello era necesario integrar la información de productos entre la plataforma de importaciones y el catálogo de productos lo cual se hizo a través de un conjunto de microservicios y servicios de mensajería bajo el patrón publicador suscriptor. El marco de trabajo elegido para esta integración de sistemas fue Scrum debido a la incertidumbre que se tenía y la necesidad de reaccionar de forma rápida a los cambios que surgiesen en el proyecto. Se logró cumplir con los objetivos de implementar la arquitectura de integración, aprovisionar la infraestructura, mejorar la plataforma de importaciones y realizar las pruebas de la implementación de la integración que permitieron cumplir con el objetivo de integrar la plataforma de importaciones y el catálogo de productos corporativo de forma satisfactoria.

Palabras claves: integración de sistemas, productos, patrón publicador suscriptor, microservicios, scrum.

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

Implementation of microservices to integrate product information in a retail company

Author: Castro Quezada, Juan Luis
Advisor: Moquillaza Henríquez, Santiago Domingo
Title: Professional Sufficiency Work to opt for the Professional Title of Systems Engineer
Date: June 2022

ABSTRACT

The corporate commercial area of a retail business needed to keep the product information managed by its import platform up to date with the corporate product catalog. The catalog would offer benefits such as acting as an intermediary to exchange this information between different platforms, as well as being a centralized, standardized and validated source of product information for different business units. For this purpose, it was necessary to integrate the product information between the import platform and the product catalog, which was done through a set of microservices and messaging services under the subscriber publisher pattern. The framework chosen for this systems integration was Scrum due to the uncertainty and the need to react quickly to changes that arose in the project. The objectives of implementing the integration architecture, provisioning the infrastructure, improving the import platform and testing the implementation of the integration were achieved, which made it possible to meet the objective of integrating the import platform and the corporate product catalog in a satisfactory manner.

Keywords: system integration, products, Publisher Subscriber pattern, microservices, scrum.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En un negocio retail perteneciente al grupo empresarial en la cual se desarrolla la experiencia profesional del autor, se detectaron dificultades para conciliar la información de los productos que manejaban en sus sistemas de información. Esto evidencio algunos problemas que se tenían como el generar silos de información, mantener información no estandarizada y dificultades para integrar la información. Bajo ese contexto se decidió implementar un catálogo de productos corporativo (denominado RPC) que permitiese solucionar los problemas anteriormente expuestos.

Con el surgimiento del catálogo de productos, los sistemas de información de diferentes unidades de negocio se integraron con él a fin de aprovechar los beneficios que ofrece. Uno de estos sistemas era la plataforma de importaciones (denominada V2S) administrado por el área comercial corporativa del antes mencionado negocio retail. Entre las motivaciones que impulsaron la integración entre V2S y RPC la principal, era la posibilidad que ofrecía este como medio para integrar la información de productos de diferentes plataformas. Esto último debido a la necesidad de integrar la información de productos con el área de procesos comerciales a través de su sistema de gestión de maestros de productos corporativo (denominado PIM).

Para poder integrar ambos sistemas, se debía tener en cuenta su estado tecnológico, siendo RPC una solución orientada a microservicios y desplegada en la nube y V2S una plataforma legada sobre arquitectura on premise. La integración entre ambos sistemas debía estar lo más desacoplada posible de estas, en donde cada componente sea de fácil mantenimiento, escalable de acuerdo al volumen de información a integrar, con responsabilidades lo más acotadas posibles y mantener sincronizada la información en el menor tiempo posible. Estas características motivaron a que la integración tenga una arquitectura orientada a microservicios que sería la solución a esta necesidad.

Quedan fuera del alcance de este trabajo de suficiencia profesional la construcción de RPC y su integración con PIM.

Este trabajo está estructurado en 5 capítulos, de la siguiente manera:

En el CAPÍTULO I - TRAYECTORIA PROFESIONAL, se presentará cronológicamente la experiencia profesional del autor, su educación complementaria y sus roles, funciones y logros adquiridos hasta la fecha de redactado el presente trabajo.

En el CAPÍTULO II – CONTEXTO EN EL QUE SE DESARROLLÓ LA EXPERIENCIA, se presentará a la organización en la cual se llevó a cabo el proyecto presentado en este trabajo. Se detallará también la visión, misión, organigrama y las tareas desempeñadas en dicha organización.

En el CAPÍTULO III – ACTIVIDADES DESARROLLADAS, se presentará la situación problemática, los objetivos, el alcance, la metodología usada, fundamentos de la solución y una evaluación de la solución.

En el CAPÍTULO IV – REFLEXIÓN CRÍTICA DE LA EXPERIENCIA, se presentará una reflexión crítica sobre el proyecto trabajado.

En el CAPÍTULO V – CONCLUSIONES Y RECOMENDACIONES, se presentará las conclusiones recabadas del proyecto y las recomendaciones del mismo.

CAPÍTULO I - TRAYECTORIA PROFESIONAL

1.1 Presentación profesional

El autor del presente trabajo es bachiller de la Facultad de Ingeniería de Sistemas e Informática (FISI) en la Universidad Nacional Mayor de San Marcos (UNMSM) el cual cuenta con alrededor de 6 años de experiencia en la construcción de soluciones tecnológicas en entidades públicas y privadas en industrias de retail, programas de innovación y sectores de banca y seguros principalmente.

Se ha desenvuelto en diferentes roles, principalmente como desarrollador de software y analista funcional en proyectos ágiles en marcos de trabajo como Scrum y Kanban orientado al cumplimiento de objetivos. Esto ha permitido que el autor desarrolle sus habilidades blandas las cuales son necesarias para poder trabajar con diferentes equipos multidisciplinarios como lo demanda su profesión.

1.2 Formación académica

En la Tabla 1, se detalla la formación académica profesional recibida por el autor de este informe.

Tabla 1.
Formación Académica

Formación	Institución	Periodo
Educación Universitaria. Grado de Bachiller en Ingeniería de Sistemas	Universidad Nacional Mayor de San Marcos Facultad Ingeniería de Sistemas e Informática	2012 - 2017

Nota. Elaboración propia

En la Tabla 2, se detalla la formación académica complementaria recibida por el autor de este informe.

Tabla 2.

Formación Académica Complementaria

Formación	Institución	Horas	Periodo
Especialización en Business Intelligence	Data Mining Consulting	112	2021 - 2022
Especialización en Programación con Python	Coursera (Universidad de Michigan)	19	2021 - 2021
Curso Virtual Scrum	Red Agile Perú	16	2020 - 2020
ITIL Foundation Versión 3	Cibertec	24	2017 - 2017
Fundamentos Ágiles	Cibertec	32	2017 - 2017
Programa Formativo Data Training Center	Sapia	40	2017 - 2017

Nota. Elaboración propia

Finalmente, la Tabla 3 describe algunas certificaciones obtenidas por el autor.

Tabla 3.

Certificaciones

Certificación	Institución	Fecha
Scrum Master Profesional Certificate	Certiprof	2021
Scrum Product Owner Profesional Certificate	Certiprof	2021

Nota. Elaboración propia.

1.3 Experiencia profesional

En la Tabla 4, se detalla la formación laboral del autor destacando las organizaciones en las que participo, el cargo, periodo, proyectos y funciones. Cabe mencionar que la entidad en la cual se desarrolló la experiencia será nombrada como Entidad de TI.

Tabla 4.

Formación Laboral

Organización	Funciones
Entidad de TI enero 2022 – actualidad	**Puesto:** Software Developer Senior **Rol:** Fullstack Developer **Funciones:** - Desarrollar los componentes de Backend y Frontend asignados. - Apoyar al equipo de SRE en la habilitación de servicios en la nube y despliegue de piezas CI/CD. - Coordinar con otros equipos Scrum la integración de servicios Backend. - Participar en las definiciones técnicas con el equipo. - Brindar mantenimiento de código legado y apoyar a resolver incidencias en producción. - Apoyar en el onboarding de los nuevos miembros del equipo. - Proyectos: Integración RPC, VeV, V2S.
Dynamic Devs abril 2021 – diciembre 2022	**Puesto:** Software Engineer II **Rol:** Fullstack Developer **Funciones:** - Desarrollar los componentes de Backend y Frontend asignados. - Apoyar al equipo de SRE en la habilitación de servicios en la nube y despliegue de piezas CI/CD. - Participar en las definiciones técnicas con el equipo. - Brindar mantenimiento de código legado y apoyar a resolver incidencias en producción.

	- Proyectos: Embarque Cloud, Integración RPC, VeV, V2S.
Multiplica mayo 2018 – julio 2020	**Puesto:** .NET Developer **Rol:** Fullstack Developer / Functional Analyst / Technical Lead **Funciones:** - Liderar el análisis de las solicitudes del cliente pactando entregas semanales. - Informar el estado de los avances del proyecto a los stakeholder. - Facilitar las ceremonias de Scrum y resolver los impedimentos del equipo. - Coordinar con otros equipos Scrum las integraciones de los proyectos. - Diseñar la arquitectura de componentes de Backend. - Desarrollar componentes de Backend y Frontend. - Proyectos: Seguridad, Autoservicios, Mydream, Autoinspecciones, Soporte OIM.
Programa Nacional de Innovación en Pesca y Acuicultura (PNIPA) febrero 2018 – mayo 2018	**Puesto:** Soporte Informático **Rol:** Fullstack Developer / Functional Analyst **Funciones:** - Definir la estrategia de trabajo y entrega con el área técnica de usuarios para cumplir con los entregables por hito. - Levantar los requerimientos funcionales, definir la arquitectura, desarrollar los componentes del sistema y validarlos con los usuarios. - Capacitar a los usuarios en el uso del sistema en las conferencias del PNIPA. - Brindar soporte a las incidencias de la aplicación en producción.

	- Proyectos: Fondos Concursables, Panel Evaluadores Técnicos.
Programa Nacional de Innovación Agraria (PNIA) junio 2017 – octubre 2017	**Puesto:** Soporte Informático **Rol:** Fullstack Developer / Functional Analyst - **Principales Funciones:** - Definir la estrategia de trabajo y entrega con el área técnica de usuarios para cumplir con los entregables por hito. - Levantar los requerimientos funcionales, desarrollar los componentes del sistema y validarlos con los usuarios. - Brindar soporte a las incidencias de la aplicación en producción. - Proyectos: Becas y Pasantías, Fondos Concursables.
SkyBox Logistics febrero 2016 – julio 2016	**Puesto:** Practicante de Sistemas **Rol:** Developer Jr. **Principales Funciones:** - Participar en la documentación técnica del proyecto. - Desarrollar nuevas funcionalidades de código legado. - Optimizar los tiempos de respuesta. - Proyectos: SkyBox Marketplace.

Nota. Elaboración propia

CAPÍTULO II - CONTEXTO EN EL QUE SE DESARROLLO LA EXPERIENCIA

2.1 Empresa – actividad que realiza

La empresa en la que se desarrolló la experiencia profesional, a la que se le viene denominando como entidad de TI, pertenece a uno de los grupos empresariales más grandes a nivel nacional e internacional. El grupo empresarial tiene presencia geográfica en 7 países del continente como son Perú, Chile, Colombia, Argentina, Brasil, México y Uruguay, con diferentes negocios dedicados a:

- Retail.
- Mejoramiento del hogar.
- Supermercados.
- Servicios financieros.
- Inmobiliario.
- Marketplace.

A nivel corporativo el grupo maneja 4 principios que consisten en superar las expectativas del cliente, hacer que las cosas sucedan, crecer en base a nuestros logros, y actuar con sentido.

La entidad de TI, se encarga de brindar soporte informático a las demás empresas del grupo. En el contexto del presente trabajo de suficiencia profesional, la entidad de TI actúa como apoyo a la unidad de negocio dedicada al sector retail de tiendas de mejoramiento del hogar a nivel corporativo. La entidad de TI en este caso se encuentra en Perú, pero tiene su homologo a nivel corporativo.

2.2 Visión

La visión del grupo empresarial se comparte con la entidad de TI, y se guía por lo siguiente, tomado de uno de sus portales web, contribuir a mejorar la calidad de vida de nuestros clientes en cada comunidad en la que operamos.

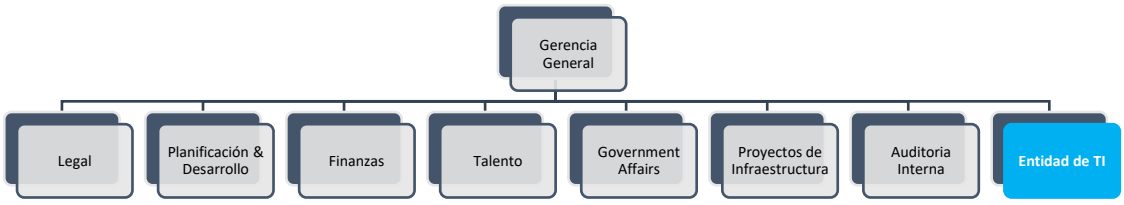
2.3 Misión

De igual manera que la visión, tenemos que la misión es satisfacer y superar las expectativas de los clientes a través de una experiencia de compra que combina lo mejor de los productos, servicios, entornos y convivencia para lograr que repitan sus preferencias.

2.4 Organización de la empresa

La Figura 1 muestra el organigrama del grupo empresarial para Perú.

Figura 1.
Organigrama del grupo en Perú.



Nota. Adaptado de la organización.

Este organigrama no representa necesariamente gerencias o áreas, representa otras empresas subsidiaras del grupo destinadas a realizar actividades comerciales según la descripción del diagrama anterior como es el caso de la entidad de TI.

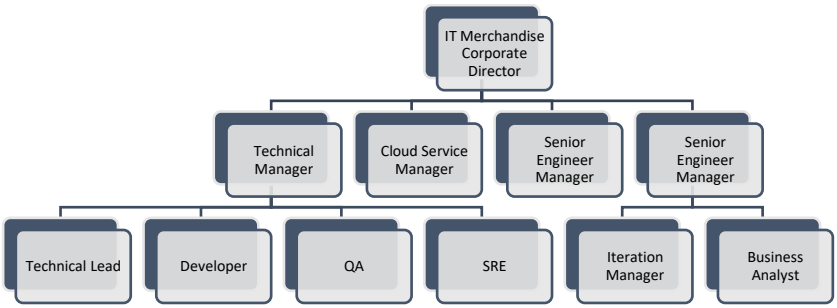
2.5 Área, cargo y funciones desempeñadas

Si bien en los puntos anteriores se menciona que la entidad de TI en la cual labora el autor pertenece a Perú, se debe indicar que sus actividades están netamente enfocadas en el apoyo a la sub gerencia de Merchandise perteneciente a la gerencia de Retail IT en su homologo corporativo.

La Figura 2 detalla la composición de los principales perfiles de la subgerencia de Merchandise.

Figura 2.

Organización de Merchandise.



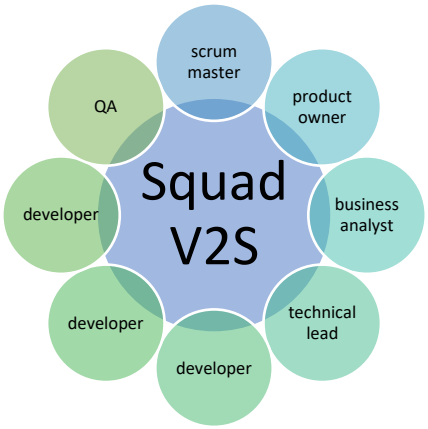
Nota. Adaptado de la organización.

El autor actualmente forma parte del squad de V2S conformado por múltiples perfiles de la subgerencia de Merchandise y de otras gerencias y subgerencias. Inicia sus actividades como miembro de la organización desde principios del 2022. Cabe resaltar que la relación con la organización inicia aproximadamente un año antes como miembro externo a través de una empresa consultora.

La composición del squad a fecha de redactado este trabajo consta de los siguientes miembros como se muestra en la Figura 3.

Figura 3.

Miembros del squad de V2S en la actualidad



Nota. Adaptado de la organización.

El autor del presente trabajo se desempeña como Senior Developer y entre las funciones que tiene a cargo se detallan la siguientes:

- Participar en todas las ceremonias agiles del equipo de manera activa y colaborativa.
- Participar en las coordinaciones con el equipo y otros equipos multidisciplinarios en todos los niveles para integrar y asegurar la calidad y seguridad de los desarrollos.
- Desarrollar los nuevos requerimientos entregados proponiendo soluciones creativas en el stack tecnológico acordado por el equipo.
- Brindar soporte al equipo de continuidad operacional en la resolución de incidencias en producción de las aplicaciones de la plataforma de importaciones legada V2S.
- Generar y compartir conocimiento para el equipo y la organización.

2.6 Experiencia profesional realizada en la organización

El squad de V2S se caracteriza por brindar apoyo a las necesidades que presenta la operación del negocio, lo que conlleva a que este siempre apoyando múltiples frentes de acuerdo a la prioridad de estas necesidades. Esto queda evidenciado durante los sprints en donde el equipo suele tener múltiples objetivos por lo que estos se subdividen para poder cumplirlos. Es así que el autor ha podido participar en diferentes iniciativas ligadas por lo general a integraciones con soluciones en la nube como el catálogo de productos y embarque, o el apoyo en el mantenimiento evolutivo de sistemas como venta en verde o la plataforma V2S e incidencias en ambientes productivos.

Entre los proyectos que forman parte de V2S y las integraciones con el mismo se han tenido las siguientes actividades:

- Apoyar en el análisis y resolución de incidencias en producción en los proyectos de compras, finanzas, campañas y productos.
- Crear nuevos features en los proyectos de productos y embarque legado.
- Desarrollar y participar en los proyectos de venta en verde, integraciones con RPC e integraciones con Embarque Cloud.

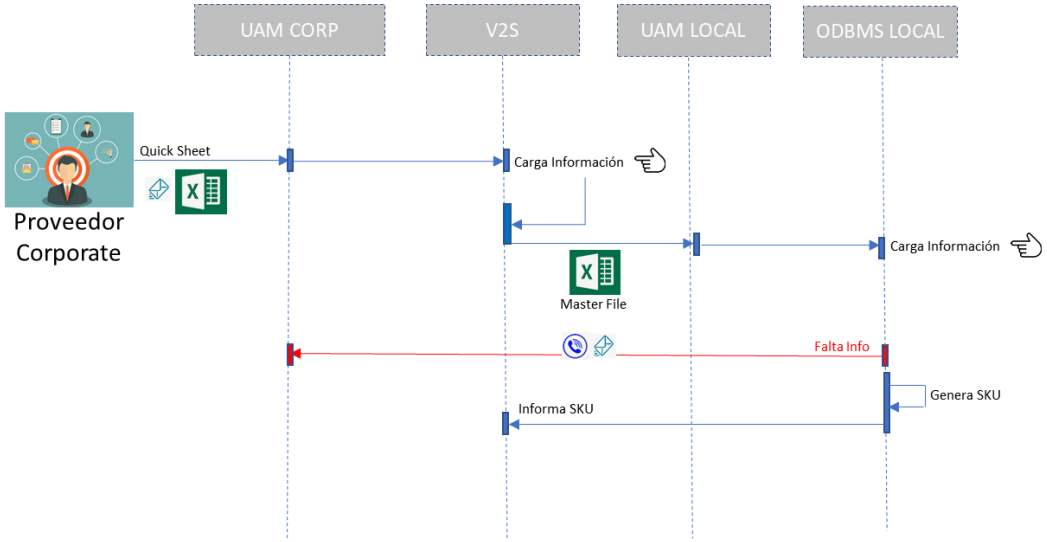
- Participar de las coordinaciones técnicas con el equipo de SRE la habilitación de servicios en la nube, automatizaciones y permisos sobre infraestructura.
- Apoyar en el onboarding de nuevos miembros del equipo en los proyectos a cargo del squad de V2S.
- Realizar las revisiones de código entre los miembros del equipo.

CAPÍTULO III - ACTIVIDADES DESARROLLADAS

3.1 Situación problemática

En la unidad del negocio retail en la cual se desarrolla la experiencia profesional, se detectaron dificultades para conciliar la información de los productos que manejaban las diferentes áreas en sus sistemas de información. Las conciliaciones se hacían de forma periódica mediante procedimientos como compartir dicha información a través de hojas de cálculo o abrir canales de comunicación por llamadas o email por falta de información; en otros casos implicaba el tener que hacer integraciones directas entre sistemas ad hoc lo cual terminaba resultando complejo. La Figura 4 muestra el flujo de intercambio de información a través de hojas de cálculo.

Figura 4.
Flujo de conciliación As Is.



Nota. Adaptado del proyecto

Esto evidencio algunos problemas que se tenían con respecto a la administración de la información de productos como el generar silos de información, mantener información no estandarizada y dificultades para integrar la información. Bajo el contexto anteriormente mencionado se decidió implementar un sistema de catálogo de productos

corporativo (RPC), que permitiese solucionar los problemas anteriormente expuestos ofreciendo beneficios relacionados principalmente al respaldo de procesos de creación, mantenimiento y validación de la información de productos, un almacenamiento centralizado y estandarizado, además de poder actuar como una plataforma intermediaria para las integraciones entre sistemas de diferentes áreas de una misma unidad de negocio.

El área comercial corporativa, que administra la información de productos desde la perspectiva de su compra e importación a través de la plataforma de importaciones (V2S), no era ajena a las dolencias solucionadas por RPC por lo que vio la necesidad de integrarse con él. Esto último impulsado principalmente por el área de procesos comerciales, encargada de administrar la información de productos para su venta a través de su plataforma de gestión de productos corporativo (PIM), que requería poder integrar la información de los productos importados en su sistema.

Desde la perspectiva del área comercial corporativa esto implicaba un nuevo reto dado que tenía que implementar una arquitectura que soportara la integración con RPC, teniendo en cuenta que esta debía estar lo más desacoplada posible ambos sistemas, en donde cada componente de la integración sea de fácil mantenimiento, escalable de acuerdo al volumen de información a integrar, con responsabilidades lo más acotadas posibles y mantener sincronizada la información en el menor tiempo posible.

3.1.1 Definición del problema

El problema principal radicaba en que no existía una integración de la información de productos de la plataforma de importaciones con el catálogo de productos corporativo a fin de que este último permita su almacenamiento centralizado y estandarizado, soporte de creación, mantenimiento y validación, además de servir como intermediario de integraciones.

3.1.1.1 Problema general

Ante la situación problemática y el problema expuestos en los puntos anteriores podemos formular el problema general como una pregunta:

¿De qué manera se puede implementar microservicios para integrar la información de productos en una empresa retail?

3.1.1.2 Problemas específicos

- ¿De qué manera se puede implementar la arquitectura de la integración entre la plataforma de importaciones y el catálogo de productos, para implementar microservicios para integrar la información de productos importados en una empresa retail?
- ¿Cómo se puede aprovisionar la infraestructura, para implementar microservicios para integrar la información de productos importados en una empresa retail?
- ¿De qué manera se pueden realizar las mejoras sobre los módulos de la plataforma de importaciones, para implementar microservicios para integrar la información de productos importados en una empresa retail?
- ¿Cómo se pueden realizar las pruebas de la integración entre la plataforma de importaciones y el catálogo de productos, para implementar microservicios para integrar la información de productos importados en una empresa retail?

3.2 Solución

Implementar un sistema de integración que permita sincronizar la información de los productos de la plataforma de importaciones con el catálogo de productos de acuerdo a la operación del negocio teniendo en cuenta que esta debía estar lo más desacoplada posible ambos sistemas, en donde cada componente de la integración sea de fácil mantenimiento, escalable de acuerdo al volumen de información a integrar, con responsabilidades lo más acotadas posibles y mantener sincronizada la información en el menor tiempo posible.

3.2.1 Objetivos

3.2.1.1 Objetivo general

Implementar microservicios para integrar la información de productos en una empresa retail.

3.2.1.2 Objetivos específicos

- Implementar la arquitectura de la integración entre la plataforma de importaciones y el catálogo de productos, para implementar microservicios integrar la información de productos importados en una empresa retail.
- Aprovisionar la infraestructura, para implementar microservicios para integrar la información de productos importados en una empresa retail.
- Realizar las mejoras sobre los módulos de la plataforma de importaciones, para implementar microservicios para integrar la información de productos importados en una empresa retail.
- Realizar las pruebas de la integración entre la plataforma de importaciones y el catálogo de productos, para implementar microservicios para integrar la información de productos importados en una empresa retail.

3.2.2 Alcance

3.2.2.1 Alcance funcional

El alcance funcional del proyecto contempla la integración entre la plataforma de importaciones y el catálogo de productos, además de las modificaciones necesarias sobre los módulos de la plataforma de importaciones encargados de administrar creaciones y actualizaciones de la información relacionadas al producto que sea relevante para el catálogo de productos. A continuación, se detalla el alcance de la integración y modificaciones.

Integración con el catálogo de productos:

- Administrar las solicitudes de sincronización con el catálogo.
- Obtener la información de productos e información relacionada.
- Publicar eventos de sincronización en los tópicos del catálogo.
- Escuchar eventos de sincronización de las suscripciones del catálogo.

Mejoras en la plataforma de importaciones:

- Visualizar las solicitudes de sincronización en el módulo de administración de productos.
- Administrar el envío de solicitudes de sincronización en los módulos:
    - Administración de productos
    - Campañas
    - Administración de proveedores.
- Modificar las funcionalidades de creación y modificación masiva e individual.

3.2.2.2 Alcance organizacional

El alcance organizacional comprende el área comercial corporativa de la unidad de negocio retail.

3.2.2.3 Alcance tecnológico

A nivel tecnológico se usaron las siguientes herramientas:

- Lenguaje de programación utilizado fue Java, usando el framework de Springboot para la construcción de Apis Rest.
- Para la mensajería se utilizó el servicio de Pub/Sub de GCP.
- A nivel de integración continua y despliegue continuo se utilizó Gitlab CI/CD.
- En cuanto a la infraestructura en donde se despliegan los servicios contenerizados es de carácter hibrido entre la nube (servicio de kubernetes engine) y on premise (plataforma rancher).
- A nivel de base de datos se usó Oracle. Finalmente, para la exposición de los servicios se utilizó Kong como Api Manager.

3.2.3 Etapas y metodología

La Tabla 5 muestra las etapas del proyecto asociado a los sprints que abarcaron, teniendo en cuenta que la ejecución de algunas etapas se trabajó en paralelo.

Tabla 5.

Etapas del proyecto

Etapa	Descripción	Sprints
Descubrimiento	Descubrimiento en base la necesidad presentada, se define equipo y metodología de trabajo y una definición inicial de requisitos.	Inception sprint 0
Definición de la arquitectura de integración	Definición y validación de la arquitectura de la integración.	1 - 2
Aprovisionamiento de la infraestructura	Actividades relacionadas a construir el soporte base de herramientas, servicios, entornos necesarios para soportar la integración.	1 - 2
Implementación de la integración	Construcción de los componentes de la integración, despliegues, ejecución de pruebas automatizadas y funcionales.	3 - 13
Mejoras sobre la plataforma V2S	Implementación de mejoras sobre la plataforma de importaciones necesarias para la integración.	3 - 13
Certificación	Pruebas de integración y de aceptación de usuario.	14 - 17
Mantenimiento	Mantenimiento correctivo, preventivo y evolutivo de la integración.	18 - actualidad

Nota. Elaboración propia.

En la etapa de certificación inicialmente estaban actividades como el paso a producción y marcha blanca, pero esto depende de factores externos al proyecto (paso de otras capacidades como PIM y ODBMS) por lo que no se presenta en este documento. La etapa de mantenimiento se comentará brevemente para dar una idea de lo que se está trabajando actualmente, pero queda excluido del alcance del proyecto.

A nivel metodológico se usó el marco de trabajo ágil Scrum, que ayuda a resolver problemas complejos a través de un proceso iterativo e incremental. Scrum maneja una serie de eventos, roles y artefactos que cooperan de forma organizada, los eventos son

cajas de tiempo en donde interactúan los roles con objetivos definidos por cada evento y los cuales pueden generar artefactos. (Schwaber & Sutherland, 2020)

La Tabla 6 da un detalle general de la aplicación de Scrum en el proyecto.

Tabla 6.
Scrum en el proyecto

Número de sprints	17
Duración de cada sprint	2 semanas
Miembros	8 miembros

Nota. Elaboración propia.

La etapa de inicio del proyecto podría corresponderse a lo que se conoce como inception o sprint 0, pero este término nunca es usado por Schwaber & Sutherland (2020) en la guía Scrum. Ambler (2019) sostiene por ejemplo que la fase de inicio (Inception) sirve para que el equipo se enfoque en hacer suficiente trabajo para organizarse y avanzar en la dirección correcta. Esta fase termina cuando se tiene una visión clara acordada con respecto a los resultados esperados por el equipo y como puede lograrse.

A continuación, se relata cómo se llevó cada evento de Scrum por los miembros del equipo en relación con los artefactos utilizados.

Planificación

La planificación, o sprint planning, era el primer evento en cada sprint en donde participaban principalmente el scrum team, y en algunas otras ocasiones miembros de equipos transversales como oyentes activos. Esta reunión tenía una duración aproximada 2 horas en donde el product owner explicaba las historias de usuario propuestas para el sprint. Posterior a ello el development team puntuaba el esfuerzo de las mencionadas historias en la escala de Fibonacci en el siguiente rango 1, 2, 3, 5, 8, 13 y 21.

En caso el equipo creyera conveniente podían dividir las historias en sub tareas. La descripción de las épicas, tareas y sub tareas son de formato libre, pero la descripción

de las historias de usuario sigue por lo general el siguiente formato que se muestra en la Tabla 7, las que no siguen este formato están ligadas a actividades técnicas, pruebas de concepto o de definición.

Tabla 7.

Descripción de historia de usuario.

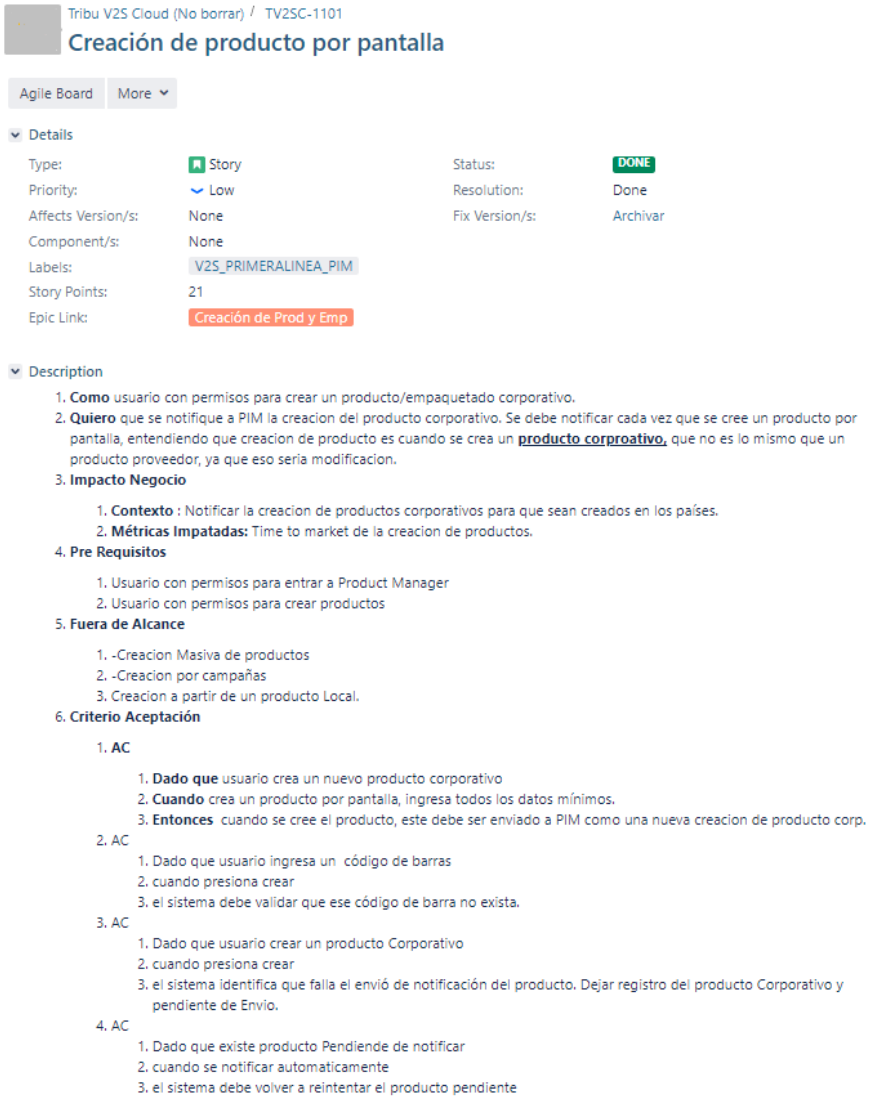
Campo	**Descripción**
Como	Descripción del rol que presenta la necesidad
Quiero	Deseable que debería cumplir la historia
Impacto de Negocio	Impacto positivo definido bajo un contexto y métricas a impactar
Pre Requisitos	Pre condiciones para abordar la historia.
Fuera de Alcance	Actividades que quedan fuera del alcance de la historia
Criterio de Aceptación	Criterios necesarios para brindar conformidad a la historia del usuario expresado en el formato - Dado que (acción realizada) - Cuando (sucede algún evento) - Entonces (que debería suceder)

Nota. Adaptado del proyecto.

La Figura 5 muestra un ejemplo de una historia de usuario registrada en Jira en donde se ve el formato de redacción definido, además de la puntuación en escala Fibonacci establecida, en conjunto con la épica a la cual pertenece.

Figura 5.

Ejemplo de historia de usuario.



Tribu V2S Cloud (No borrar) / TV2SC-1101

Creación de producto por pantalla

Agile Board More

Details

Type:	Story	Status:	DONE
Priority:	Low	Resolution:	Done
Affects Version/s:	None	Fix Version/s:	Archivar
Component/s:	None		
Labels:	V2S_PRIMERALINEA_PIM		
Story Points:	21		
Epic Link:	Creación de Prod y Emp		

Description

1. **Como** usuario con permisos para crear un producto/empaquetado corporativo.
2. **Quiero** que se notifique a PIM la creacion del producto corporativo. Se debe notificar cada vez que se cree un producto por pantalla, entendiendo que creacion de producto es cuando se crea un **producto corproativo,** que no es lo mismo que un producto proveedor, ya que eso seria modificacion.
3. **Impacto Negocio**
    1. **Contexto** : Notificar la creacion de productos corporativos para que sean creados en los países.
    2. **Métricas Impatadas:** Time to market de la creacion de productos.
4. **Pre Requisitos**
    1. Usuario con permisos para entrar a Product Manager
    2. Usuario con permisos para crear productos
5. **Fuera de Alcance**
    1. -Creacion Masiva de productos
    2. -Creacion por campañas
    3. Creacion a partir de un producto Local.
6. **Criterio Aceptación**
    1. **AC**
        1. **Dado que** usuario crea un nuevo producto corporativo
        2. **Cuando** crea un producto por pantalla, ingresa todos los datos mínimos.
        3. **Entonces** cuando se cree el producto, este debe ser enviado a PIM como una nueva creacion de producto corp.
    2. AC
        1. Dado que usuario ingresa un código de barras
        2. cuando presiona crear
        3. el sistema debe validar que ese código de barra no exista.
    3. AC
        1. Dado que usuario crear un producto Corporativo
        2. cuando presiona crear
        3. el sistema identifica que falla el envió de notificación del producto. Dejar registro del producto Corporativo y pendiente de Envio.
    4. AC
        1. Dado que existe producto Pendiende de notificar
        2. cuando se notificar automaticamente
        3. el sistema debe volver a reintentar el producto pendiente

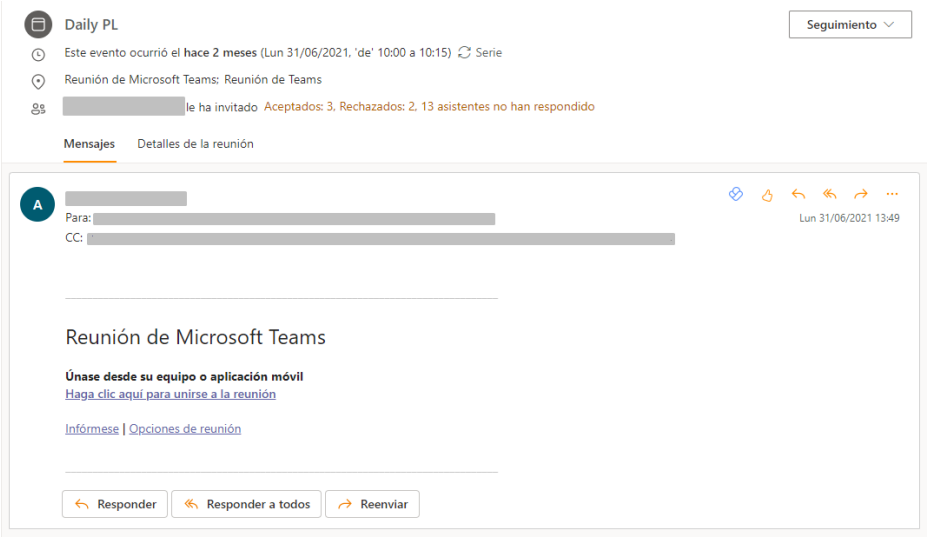
Nota. Adaptado del proyecto.

Reuniones diarias

Reuniones que el equipo mantenía de forma diaria por las mañanas con un máximo de 15 minutos de duración. Estas reuniones por lo general desencadenaban nuevas reuniones para resolver dudas puntuales del equipo. Las reuniones se agendaban periódicamente como se muestra en la Figura 6.

Figura 6.

Reunión de daily periódica.



Nota. Adaptado del proyecto.

Refinamiento

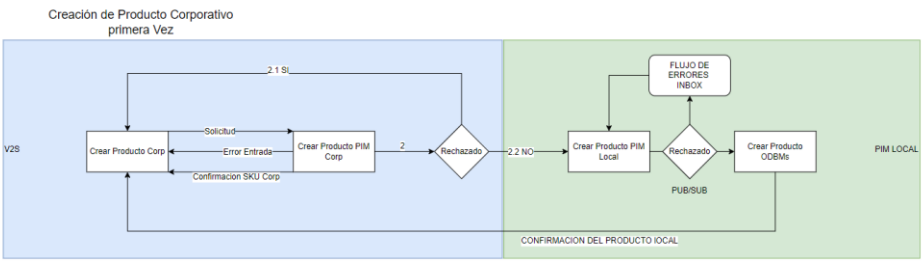
Las actividades de refinamiento que tuvo el equipo estuvieron ligadas a:

- Detallar, enriquecer y explorar las historias de usuario del product backlog.
- Estimar y priorizar las historias de usuario del product backlog.
- Realizar un diseño técnico de las historias más complejas.

Este tipo de actividades no siempre involucraba al equipo completo, para no impactar negativamente en los objetivos del sprint. El diseño técnico de la solución a alto nivel era uno de los puntos importantes a tratar como se muestra en la Figura 7.

Figura 7.

Diseños técnicos en el refinamiento.



Nota. Adaptado del proyecto.

Revisiones

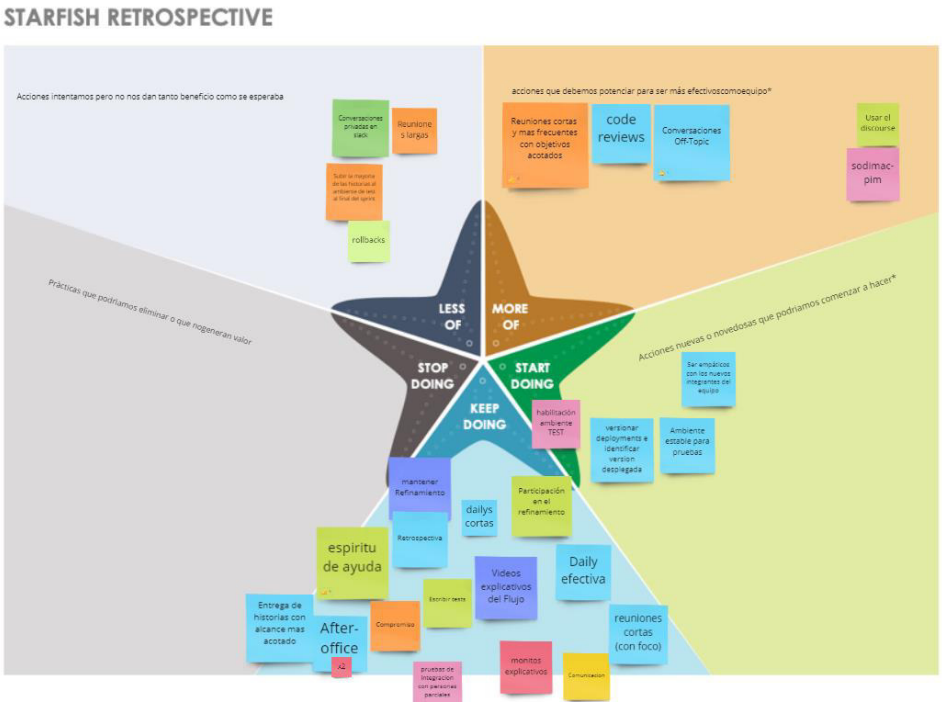
Las revisiones se hacían con el objetivo de ver los avances y los cambios que surgieron durante el sprint. Estas revisiones solían ser orientadas a la parte más técnica, en donde solo intervenía el equipo y otras más funcionales en donde intervenían los stakeholders del proyecto.

Retrospectivas

Las retrospectivas servían para que el equipo inspeccionara el sprint mediante el ejercicio de dinámicas que permitían al equipo evaluar su rendimiento, impedimentos y oportunidades de mejoras. Además de ello los miembros del equipo seleccionaban al integrante que más destaco durante el sprint.

La herramienta Miro, la cual es una pizarra digital, servía como soporte principal para esta actividad como se evidencia en la Figura 8.

Figura 8.
Actividad de retrospectiva.



Nota. Adaptado del proyecto.

3.2.4 Fundamentos utilizados

3.2.4.1 Sector retail - productos

De acuerdo a PerúRetail (2018) el retail, también conocido como comercio minorista, es un sector económico que abarca a las empresas dedicadas al comercio masivo de productos o servicios para grandes cantidades de clientes. Algunos ejemplos del sector están constituidos por supermercados, tiendas departamentales, farmacias, tiendas de mejoramiento del hogar entre otras en base a los productos que venden.

Los productos importados mencionados en este documento, hacen referencia a productos dentro de la clasificación de tiendas de mejoramiento del hogar. Estos productos pueden ser categorizados en grupos relacionados a:

- Servicios y otros.
- Productos madereros.
- Materiales de construcción.
- Ferretería.
- Jardín.
- Hogar.
- Productos de acabado.
- Productos de temporada.

3.2.4.2 Scrum

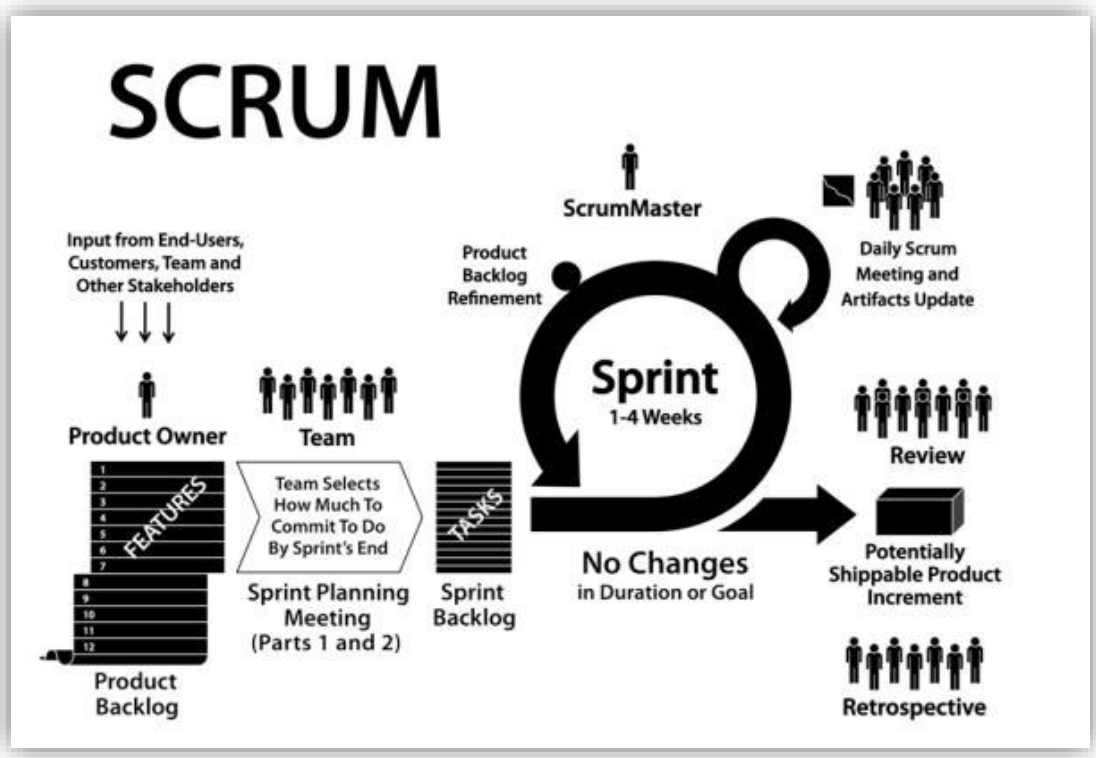
Scrum de acuerdo a la guía escrita por sus autores Schwaber & Sutherland (2020) representa un marco ligero que ayuda a personas, equipos y organizaciones en el cual se pueden abordar problemas complejos a la vez que se busca entregar el mayor valor posible.

Sutherland et al. (2007) mencionan que Scrum estructura el desarrollo en ciclos de trabajo denominados sprints, al comienzo de cada sprint un equipo multifuncional selecciona elementos de una lista priorizada los cuales se comprometen a completar al finalizar el sprint. Asimismo Sutherland et al. (2007) sostienen que durante el sprint todos

los días el equipo se reúne para inspeccionar su progreso y ajustar los pasos necesarios para completar el trabajo restante, finalmente el último día del sprint el equipo debe revisar el sprint con las partes interesadas y debe buscar retroalimentación para incorporar al siguiente sprint.

La Figura 9 demuestra lo descrito anteriormente de manera gráfica.

Figura 9.
Flujo de Scrum.



Nota. Fuente de (Sutherland et al., 2007)

A continuación, se describen los elementos que componen Scrum.

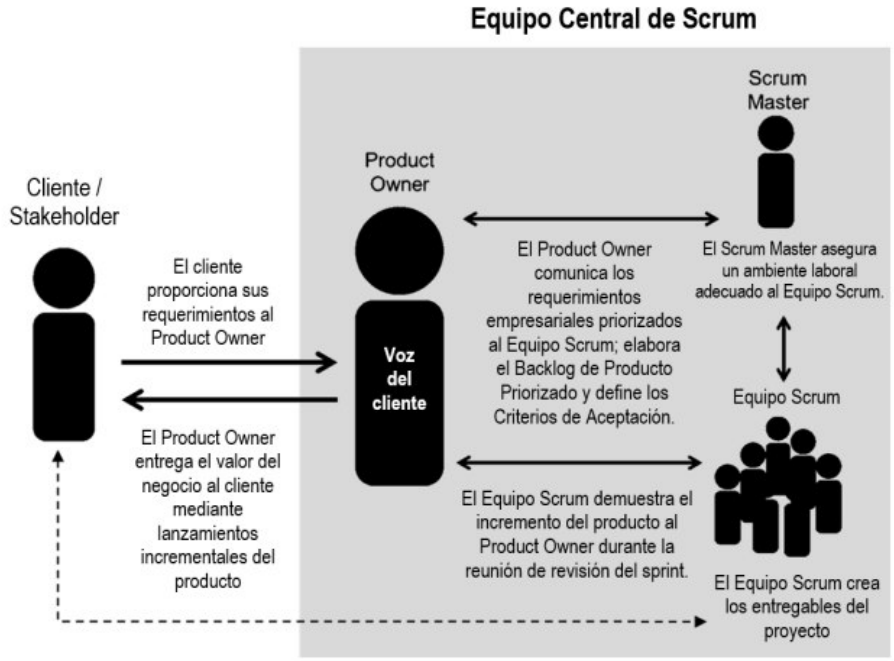
Roles

El Equipo Scrum (Scrum Team) se compone de tres roles bien definidos, el Scrum Master, el Product Owner y los Developers. El tamaño de este equipo debe ser el adecuado para permanecer ágil y poder cumplir con el objetivo en cada sprint, teniendo

en cuenta que son un equipo multidisciplinario que cuenta con las habilidades necesarias para cumplir dicho objetivo, que no hay jerarquías entre ellos y que son los únicos responsables de las actividades relacionadas con la entrega de valor. (Schwaber & Sutherland, 2020)

SCRUM STUDY (2017) Además de los roles propuestos por la guía los cuales categoriza como roles centrales, menciona un segundo grupo denominado roles no centrales en los cuales incluye miembros que tengan un interés en el proyecto como clientes, usuarios, o patrocinadores. La Figura 10. Relaciones de los roles de Scrum muestra una descripción general de cada rol, para ser posteriormente detallada.

Figura 10.
Relaciones de los roles de Scrum



Nota. Fuente de *(SCRUM STUDY, 2017)*

a) Scrum master.

De acuerdo a SCRUM STUDY (2017), es un facilitador encargado de asegurar que el equipo tenga un ambiente adecuado para el desarrollo exitoso de los incrementos

del producto. Schwaber & Sutherland (2020) indican que son responsables de las siguientes actividades:

- Planifica y asesora la implementación de Scrum.
- Apoya al equipo a entender de manera teórica y práctica Scrum.
- Apoya al equipo a enfocarse en crear valor y liberan impedimentos.
- Asegura el cumplimiento de los eventos de Scrum.
- Facilita la colaboración de las partes interesadas según sea necesario.

b) Product owner.

SCRUM STUDY (2017) menciona que es la persona responsable de maximizar el valor comercial del proyecto articulando las necesidades del cliente. De acuerdo a Schwaber & Sutherland (2020) entre las principales actividades de la cual es responsable, están:

- Explicar de forma explícita los objetivos del producto.
- Crear y comunicar de forma transparente los elementos del product backlog.
- Maximizar el valor resultante del trabajo del Equipo Scrum.

c) Developers.

De acuerdo a Schwaber & Sutherland (2020) entre las principales actividades de las cuales son responsables, se listan las siguientes:

- Encargados de crear el incremento de valor.
- Crear el plan del sprint (sprint backlog).
- Mantenerse acorde a la definición de hecho (DoD).

SCRUM STUDY (2017) denomina este rol como equipo scrum, para efectos de este documento, el equipo scrum es la agrupación de estos tres roles.

Eventos

Los eventos son oportunidades formales para realizar la inspección y adaptación de los artefactos de Scrum en búsqueda de la transparencia deseada y minimizando la

necesidad de reuniones innecesarias. Son en total 5 eventos los cuales se detallan a continuación. (Schwaber & Sutherland, 2020)

a) Sprint

Es el evento más importante de Scrum, permiten la previsibilidad al garantizar la inspección y la adaptación. Todo el trabajo necesario y los otros eventos se realizan dentro del sprint. Son de duración fija y se recomienda que su duración sea menor a un mes. Un sprint puede cancelarse si el objetivo del mismo se vuelve obsoleto. (Schwaber & Sutherland, 2020)

b) Sprint planning

Evento que inicia el sprint al planificar el trabajo que se realizara durante el mismo. Aborda las siguientes preguntas ¿Por qué es valioso este sprint?, ¿Qué se puede hacer en este sprint? y ¿Cómo se realizará el trabajo elegido? La respuesta a estas preguntas deriva en la elaboración del objetivo del sprint, los elementos del producto backlog para el sprint y el plan para completar estos elementos respectivamente que sumados se denominan sprint backlog. (Schwaber & Sutherland, 2020)

c) Daily scrum

Schwaber & Sutherland (2020) indican que el propósito del daily scrum es inspeccionar el progreso hacia el objetivo del sprint y adaptar el sprint backlog según sea necesario. Schwaber & Sutherland (2020) sostienen además que debe ser un evento diario y cuya duración sea de 15 minutos a la misma hora y en el mismo lugar en donde se comunica el trabajo a realizar durante el día.

d) Sprint review

Penúltimo evento del sprint, cuyo propósito es inspeccionar el resultado del sprint y poder determinar cambios futuros, el equipo scrum en su conjunto presentan los

resultados hacia las partes interesadas analizando el progreso hacia el objetivo. (Schwaber & Sutherland, 2020)

e) Sprint retrospective

Schwaber & Sutherland (2020) sostienen que el propósito de este evento es planificar formas de mejorar la calidad y eficacia a través de la inspección del último sprint con respecto a personas , interacciones herramientas y la definición de hecho.

Artefactos

Chapbell (2022) explica que los artefactos son elementos que ayudan a compartir con los involucrados información de vital importancia reforzando de ese modo los valores de Scrum ligados a la transparencia, inspección y adaptación. De acuerdo a Schwaber & Sutherland (2020) estos son tres pero la literatura muchas veces expone un conjunto extendido de artefactos como la definición de hecho (DoD), Burn-down chart (y similares) como indica Chapbell (2022).

a) Product backlog

De acuerdo a Schwaber & Sutherland (2020) el product backlog es una lista de elementos priorizada y representa la única fuente de trabajo para el equipo Scrum. Los elementos del product backlog que puede realizar el equipo Scrum dentro de un sprint se consideran listos para la selección en un evento de sprint planning como indican. (Schwaber & Sutherland, 2020)

Schwaber & Sutherland (2020) sostienen además lo siguiente con respecto a la actividad del refinamiento, la cual según mencionan tiene por objetivo desglosar y definir aún más los elementos del product backlog en elementos más acotados y precisos. Siendo una actividad continua de agregar detalle, descripción, orden y tamaño. Estas actividades en algunas ocasiones se representan como un evento independiente.

Por lo general el product backlog es representado a través de historias de usuario, que como menciona Chapbell (2022) describen como las funcionalidades del producto pueden ser útiles para el cliente, en donde cada historia tiene asignados unos puntos que sirven de métrica para la estimación de la dificultad de implementar dicha historia.

b) Sprint backlog

Chapbell (2022) menciona que el sprint backlog es una lista de tareas pendientes que describe la funcionalidad a desarrollar en un sprint en particular. Schwaber & Sutherland (2020) indican que se trata de un plan para los Developers en tiempo real, representando una imagen del trabajo que pueden realizar en un Sprint.

SCRUM STUDY (2017) añade que una vez comprometidos con el sprint backlog, no se debieran agregar nuevas historias de usuario; por otro lado, pueden detallarse tareas que pueden haberse pasado por alto al momento de dividir las historias durante el sprint planning. Esto último por lo menos en un caso ideal según la experiencia del autor.

c) Increment

El incremento representa un peldaño hacia el objetivo del producto como indican Schwaber & Sutherland (2020), además de ello también mencionan que se pueden generar varios incrementos en un mismo sprint y la suma de estos se presenta en el sprint review. Tener en cuenta que el incremento es una versión funcional y potencialmente entregable del producto, dado que antes de ser puesto en producción se debe verificar una definición de hecho por parte del Product Owner como indica Chapbell (2022).

3.2.4.3 Microservicios

Un microservicio es una pequeña aplicación que puede ser implementada, escalada y probada de forma independiente que se encuentra orientada a una única responsabilidad. (Thönes, 2015)

En esa misma línea Dragoni et al. (2017) expone una serie de características que deberían tener los microservicios.

- Tamaño: Menor con respecto a un servicio típico, si es demasiado grande puede perderse la granularidad.
- Contexto limitado: las funcionalidades que implementa se orientan a una única capacidad de negocio.
- Independencia: Cada microservicio es operacionalmente independiente de otros microservicios, la única forma de comunicación es a través de interfaces públicas.
- Flexibilidad: Mantenimiento rápido capaz de seguir el ritmo del entorno empresarial.
- Modularidad: Cada microservicio contribuye al comportamiento global del sistema, en lugar de un componente único.
- Evolución: Fácil de evolucionar en base a nuevas funcionalidades.

Las arquitecturas orientadas a microservicios poseen múltiples patrones de diseño que apoyan su construcción, entre los patrones de diseño más importantes tenemos a los relacionados a la gestión de datos, pruebas, seguridad, fiabilidad, descubrimiento de servicios, implementación, estilos de comunicación, descomposición entre otros como expone Richards (2018) en su libro Patrones de microservicios: con ejemplos en Java.

En cuanto al grupo de patrones relacionados al estilo de comunicación de microservicios, Richards (2018) expone dos que serán usados como base para el proyecto, el primero es el de invocación de procedimiento remoto, que se basa en un protocolo de solicitud/respuesta en donde una implementación es a través de REST y el otro en base a mensajería a través de canales de comunicación mayormente asíncronos, un ejemplo de implementación de este último es a través de Pub/Sub.

3.2.4.4 Sistemas legados

Como resumen Gholami, Daneshgar, Beydoun y Rabhi (2017) los sistemas legados son sistemas que usan tecnología obsoleta pero que aportan un valor importante a la organización razón por la cual es necesario mantenerlos. Strobl, Bernhart y Grechenig

(2020) indican que algunos representan grandes monolitos estrechamente acoplados o atolladeros con poco acoplamiento, pero con límites imperceptibles, sea cual sea el caso se identifican como altamente problemáticos, complejos para ser reemplazados y a su vez demasiados valiosos para el negocio.

Srinivas, Ramakrishna, Rajasekhara Rao y Suresh Babu (2016) describen los desafíos de mantener sistemas legados, los cuales se listan a continuación.

- Son lentos y costosos debido al hardware obsoleto.
- El mantenimiento es costoso debido a que la detección de errores es costosa y lleva mucho tiempo.
- La falta de interfaces proporciona mayor dificultad para integrarse con otros sistemas.
- La evolución de los sistemas heredados para proporcionar nuevas funcionalidades es muy difícil de realizar.

Para efectos prácticos de este trabajo definiremos como sistema legado a un sistema de alto valor para el negocio cuya tecnología presenta un desface para las nuevas propuestas tecnológicas de la organización y cuyo mantenimiento resulta ser complejo.

3.2.4.5 Computación en la nube

RED HAT (2018) menciona que la computación en la nube es la ejecución del trabajo informático en entornos de TI que extraen, agrupan y comparten recursos flexibles de red y que como tal este término no es una tecnología en sí mismo. Geewax (2018) define la nube como una colección de servicios que ayudan a los desarrolladores a centrarse en los proyectos en lugar de la infraestructura que lo impulsa.

De acuerdo al cuadrante de Gartner para el año 2021 los tres proveedores de nube líderes fueron Amazon, Microsoft y Google, como se evidencia en la Figura 11.

Figura 11.

Cuadrante de Gartner para infraestructura y plataforma en nube

Nota. Tomado de (GARTNER, 2021)

Para este trabajo, el proveedor de nube utilizado fue Google quien mediante GCP brinda un conjunto de herramientas, plataformas y servicios informáticos. A continuación, se describen algunos servicios de Google, que serán de utilidad para entender el presente trabajo.

Pub/Sub

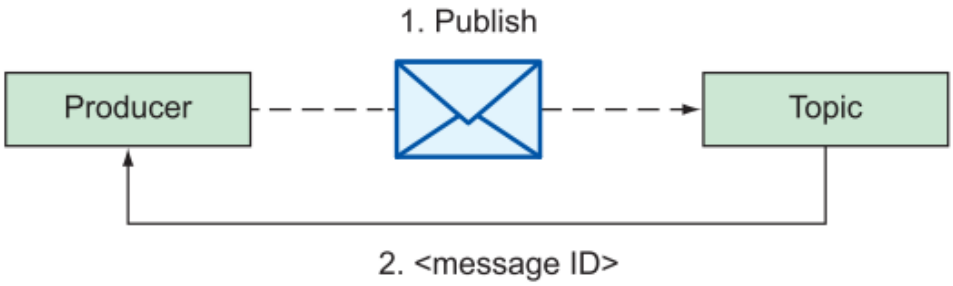
Geewax (2018) describe a Pub/Sub como un sistema de mensajeria totalmente administrado, por ende es altamente confiable y escalable en donde remitentes y

destinatarios estan separados. Algnos elementos de Pub/Sub son expuestos a continuación en base a la documentación oficial.

- Mensaje: Representa los datos que se comunican del emisor al receptor.
- Tópico: Entidad con nombre que representa una fuente de mensajes.
- Suscripción: Una entidad con nombre interesada en recibir mensajes de un tópico en particular.
- Publicador: Generador de mensajes y encargado de publicarlos en un tópico en particular.
- Suscriptor: Recibe los mensajes de una suscripción en particular.

Geewax (2018) describe el funcionamiento a alto nivel como se menciona a continuación. En un inicio un publicador (o también llamado productor) de mensajes envía un mensaje etiquetado o categorizado en un tópico (también denominado tema). Una vez llegado este mensaje al tópico recibirá un código identificador, devolviendo este código al productor del mensaje. La Figura 12 muestra esta parte del flujo.

Figura 12.
Publicación en Pub/Sub

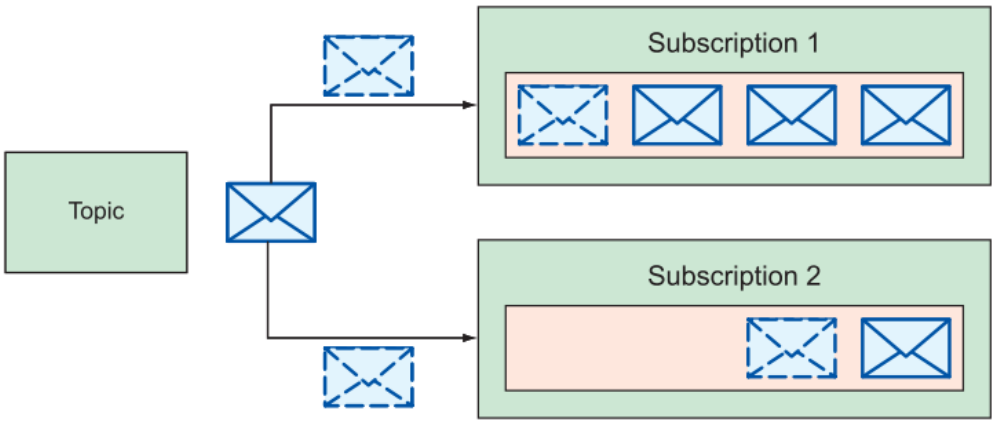


Nota. Fuente de (Geewax, 2018)

Una vez el mensaje fue recibido por el tópico, surge la interrogante de saber a quién le interesa el mensaje. Esto se resuelve mediante el concepto de suscripción, es así que se emite una copia del mensaje a las suscripciones que posee el tópico, estos se encolaran en cada suscripción a la espera de ser atendidos. Esta parte del flujo se refleja en la Figura 13.

Figura 13.

Recorrido del mensaje en Pub/Sub

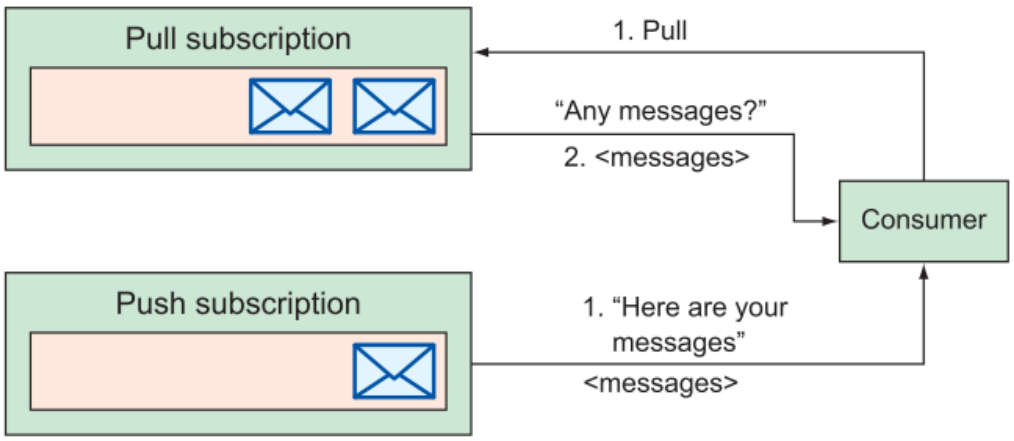


Nota. Fuente de (Geewax, 2018)

Como menciona Geewax (2018), hasta este punto el mensaje aún no se entrega pues debe llegar al último elemento que corresponde al suscriptor (o también llamado consumidor). Una vez el mensaje está en la suscripción puede llegar de dos maneras al suscriptor la primera es que la suscripción envíe el mensaje al suscriptor (tipo push) y la segunda es que el suscriptor extraiga el mensaje de la suscripción (tipo pull). Lo descrito en esta parte final del flujo se muestra de forma gráfica en la Figura 14.

Figura 14.

Suscripción a Pub/Sub



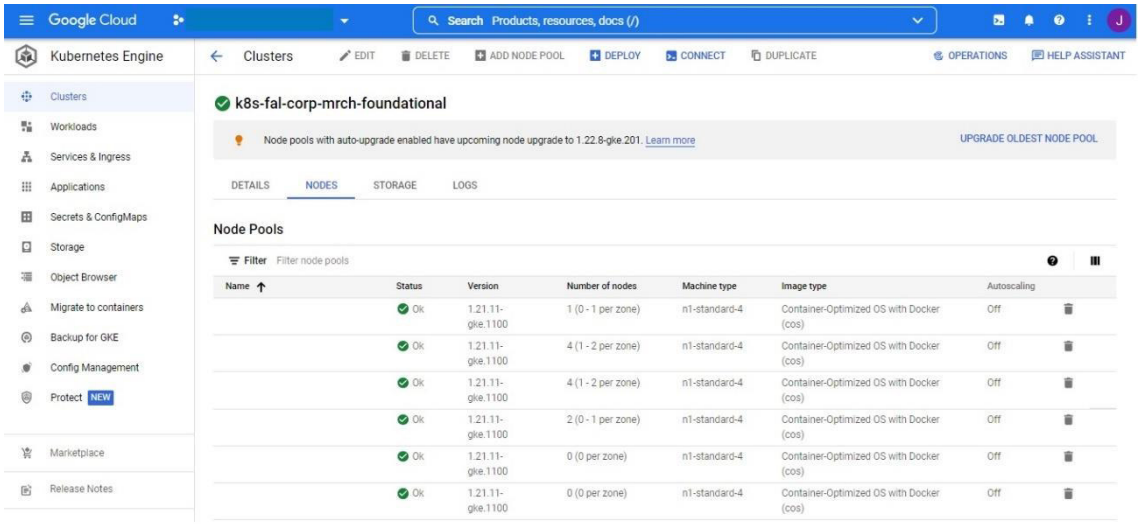
Nota. Fuente de *(Geewax, 2018)*

Kubernetes Engine

Geewax (2018) indica que es una implementación administrada y alojada de kubernetes en la nube de Google, usando las mismas herramientas que se usaría si se estuviese ejecutando kubernetes implementado en casa, con el agregado de que puede encargarse de operaciones administrativas mediante la API ofrecida por este servicio. A continuación, se muestra la consola de administración de Kubernetes Engine en GCP como se ve en la Figura 15.

Figura 15.

Consola de administración de kubernetes engine



Nota. Adaptado del proyecto

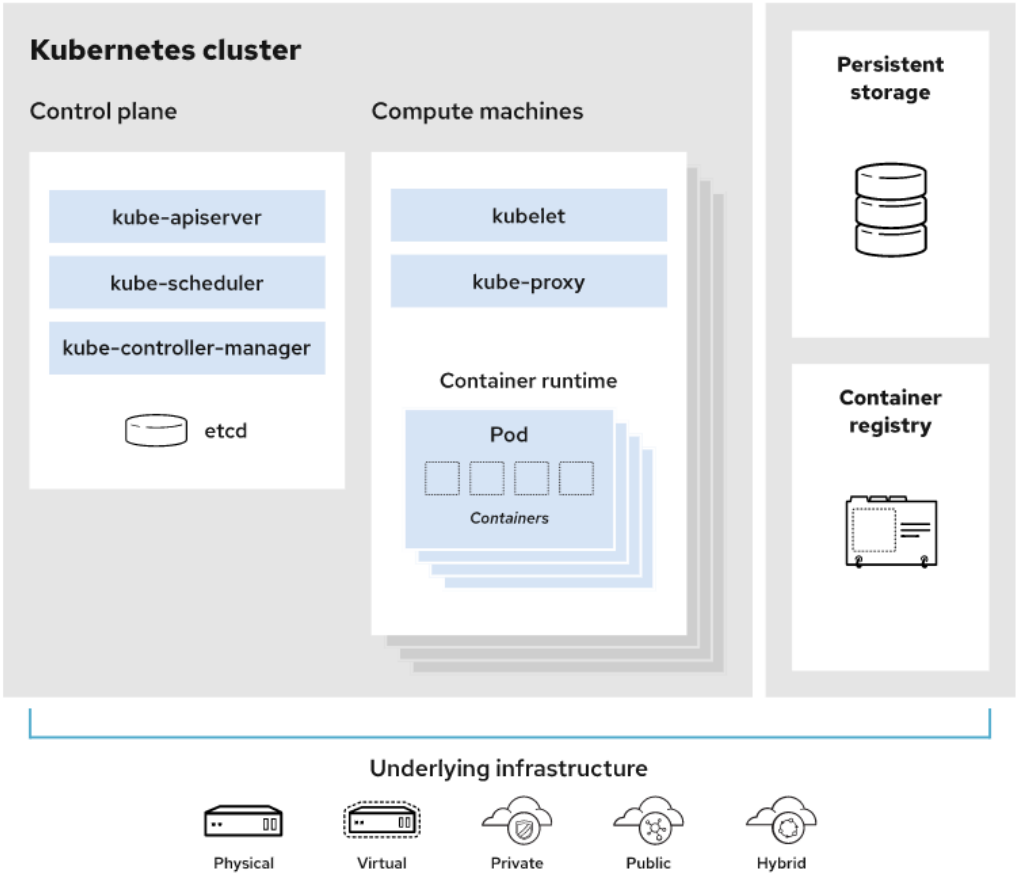
En este punto hemos mencionado kubernetes, pero ¿qué es exactamente? Bueno, Lukša (2020) lo define como un software encargado de automatizar la implementación y gestión de sistemas de gran escala compuesto por procesos informáticos ejecutados en contenedores. Geewax (2018) lo describe como un sistema que administra contenedores permitiendo dividir las cosas en partes que tengan sentido independiente del hardware subyacente.

RED HAT (2020a) describe algunos términos usados en kubernetes son los siguientes, como se mencionan a continuación.

- Plano de control. Conjunto de procesos encargados de controlar los nodos de kubernetes y se asignan las tareas.
- Nodo. Maquina que ejecuta las tareas asignadas por el control plane.
- Pod. Grupo de contenedores implementados en un solo nodo. Los contenedores de un pod comparten recursos.
- Kubelet. Servicio ejecutado en los nodos encargado de garamtizar el funcionamiento de los contenedores.
- Kubectl. Herramienta de linea de comandos de kubernetes.

Como menciona también RED HAT (2020a), una implementación de kubernetes funcionando se denomina clúster, en la Figura 16. Componentes de kubernetes se muestra un gráfico en donde se aprecia de algún modo lo descrito anteriormente.

Figura 16.
Componentes de kubernetes.



Nota. Fuente de (RED HAT, 2020a)

Un término recurrente al momento de definir Kubernetes fue el de contenedor, el cual según menciona Geewax (2018) es una herramienta de infraestructura destinada a facilitar el empaquetamiento de una aplicación, la configuración y cualquier otro tipo de dependencia de forma estándar. Una implementación del concepto de contenedores es Docker, el cual es un proyecto utilizado para la automatización de la implementación de aplicaciones basadas en contenedores que pueden ser ejecutadas en la nube o en instalaciones on premise. (MICROSOFT, 2022c)

Al igual que kubernetes engine ofrecido por Google, existen otras alternativas, un ejemplo concreto es el de rancher, el cual es un stack de software utilizado para poder gestionar clústeres de Kubernetes ya sea en nubes privadas, públicas o ambientes on premise.

3.2.4.6 Api REST

RED HAT (2020b) menciona que el término Api, proviene de interfaz de programación de aplicaciones y hace referencia a un conjunto de definiciones y protocolos usados para diseñar e integrar software, considerándose como un contrato necesario para un consumidor y un productor. Complementando lo anterior, MICROSOFT (2022b) indica que REST es un estilo arquitectónico independiente de cualquier protocolo sin estar vinculado a HTTP, a pesar de que la mayoría de las implementaciones usan dicho protocolo, además de presentar características como mantener una interfaz uniforme, tener un modelo de solicitud sin estado entre las más importantes.

De acuerdo a IBM (2021) la flexibilidad de REST en comparación a otras alternativas ha logrado que sea adoptada en gran medida en la actualidad, pero ello no quita que deban alinearse a ciertos principios de diseño. Alguno de los principios que expone se listan a continuación.

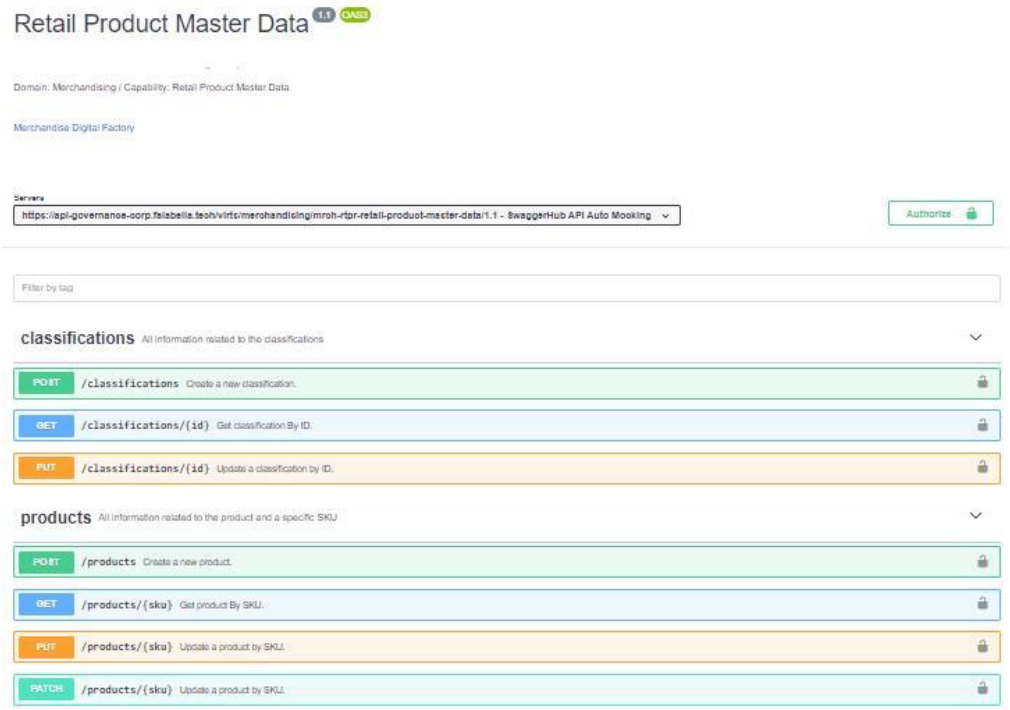
- Interfaz uniforme
- Desacoplamiento cliente-servidor
- Sin estado

- Capacidad de almacenamiento en caché
- Arquitectura de sistema en capas
- Código bajo demanda (opcional)

Swagger

Swagger es una herramienta de código abierto usada principalmente para documentar un Api REST, entre otras cosas. Puede configurarse de forma independiente o también incluirse en las apis mediante las librerías que ofrece. La Figura 17 muestra la interfaz de un Api de ejemplo documentada por Swagger.

Figura 17.
Api documentado en Swagger.



Nota. Adaptado del proyecto

Springboot

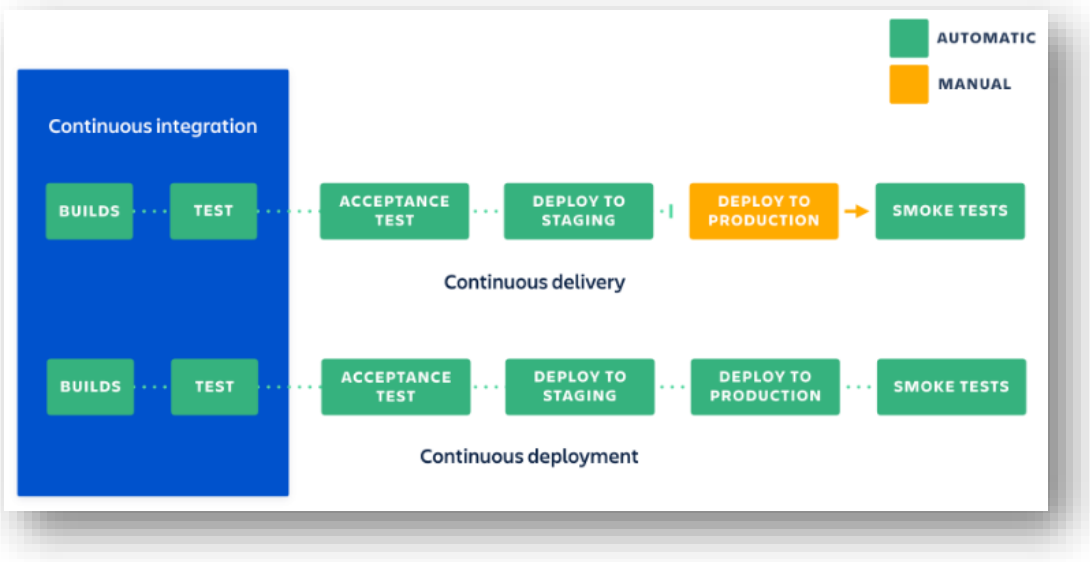
En la actualidad existen muchos frameworks de desarrollo que permiten implementar un Api REST de forma muy rápida. Uno de ellos diseñado para el lenguaje de programación Java es Springboot, el cual es un módulo de Spring framework que

permite la construcción de aplicaciones de forma rápida con una configuración por defecto que puede ser cambiada a demanda.

3.2.4.7 Integración, entrega y despliegue continuo

Este apartado está dedicado a las prácticas de integración continua, entrega continua y despliegue continuo que ayudan a la entrega de software de forma rápida. La Figura 18 muestra como están relacionadas estas prácticas y posteriormente se detalla cada una y los beneficios que generan.

Figura 18.
Integración, entrega y despliegue continuos.



Nota. Fuente de (ATLASSIAN, 2022)

Integración continua

Fowler (2006) indica que la integración continua (CI por sus siglas en inglés) es una práctica de desarrollo de software mediante la cual los miembros del equipo integran sus trabajos con frecuencia. Menciona también que cada integración se verifica mediante una compilación automatizada (incluyendo las pruebas) para detectar errores de integración lo más pronto posible.

Los beneficios obtenidos de esta práctica según ATLASSIAN (2022) son los siguientes:

- Menos errores enviados a ambientes productivos gracias a las pruebas automatizadas.
- Crear una nueva versión es fácil.
- Los costos relacionados a las pruebas se reducen drásticamente.
- El equipo de calidad puede dedicar menos tiempo a pruebas y centrarse en mejoras significativas de calidad.
- Menos cambios de contexto para los desarrolladores dado que pueden validar rápidamente la integración antes de pasar a otras tareas.

Entrega continua

La entrega continua (CD por sus siglas en inglés) está basada en la CI para garantizar el despliegue de nuevos cambios lo más pronto posible y de forma consistente, esto debido a que además de las compilaciones y pruebas automatizadas el proceso del despliegue esta automatizado de tal forma que se puede hacer presionando solo un botón. (MICROSOFT, 2022a)

Los beneficios obtenidos de esta práctica según ATLASSIAN (2022) son los siguientes:

- Se elimina la complejidad de implementar software. No hay necesidad de pasar días preparando el despliegue.
- Se puede desplegar más a menudo, acelerando procesos de retroalimentación.
- Se tiene menos presión sobre las decisiones de para pequeños cambios, por lo que fomenta una iteración más rápida.

Despliegue continuo

La implementación continua lleva la entrega continua más allá debido a que no se requiere de intervención humana para el despliegue. Esto significa que los cambios pasan por todas las etapas y si no falla nada se lanzan automáticamente a producción. (MICROSOFT, 2022a)

Los beneficios obtenidos de esta práctica según ATLASSIAN (2022) son los siguientes:

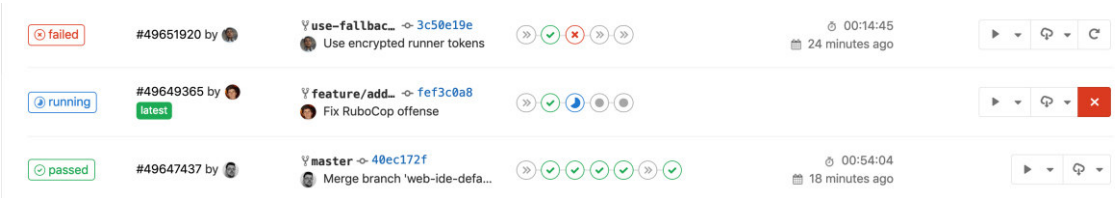
- Se puede desarrollar más rápido, ya que no hay necesidad de pausar el desarrollo para los despliegues.
- Los despliegues son menos riesgosos y fáciles de corregir en caso de problemas a medida que se implementan pequeños cambios
- Los clientes ven un flujo continuo de mejoras y la calidad aumenta a diario, en vez de cada mes, trimestre o año.

Gitlab CI/CD

Gitlab CI/CD es una herramienta para las prácticas de integración, despliegue y entrega continua. Como menciona Bustillos (2020), en un artículo escrito para Encora, para configurar un proyecto en Gitlab con CI/CD se debe colocar un archivo .gitlab-ci.yml en la raíz del repositorio el cual será ejecutado por unos runners (usados para ejecutar trabajos y enviar resultados a Gitlab). Los trabajos ejecutados por el runner por lo general se organizan por etapas y se organizan en un orden.

Gitlab CI/CD ofrece además una interfaz que perite visualizar lo que sucede durante la ejecución permitiendo obtener detalles y los resultados de la ejecución, como se muestra en la Figura 19.

Figura 19.
Pipeline de Gitlab CI/CD.



Nota. Fuente de (Bustillos, 2020)

3.2.4.8 Antecedentes

Monolithic to Microservices redesign of event driven integration platform
(Djogic, Ribic, & Donko, 2018)

En este artículo de investigación, los autores presentan un rediseño de la arquitectura de la plataforma de integración SOA basada en principios de diseño de microservicios para el sector inmobiliario. Esto encontraba justificación debido a que la plataforma de integración empezó a recibir mayor cantidad de mensajes que procesar en conjunto con una creciente cantidad de integraciones que motivaban un rediseño de dicha plataforma. Este rediseño es para proporcionar escalabilidad, una mejor gestión de recursos y facilidad de mantenimiento e implementación. Dadas las características anteriores, una arquitectura basada en microservicios es la elección.

Como conclusiones los autores relatan que el enfoque de integración orientado a microservicios brinda mayor agilidad y flexibilidad en comparativa con el enfoque SOA con respecto al mantenimiento, actualización e implementación de productos software. Además, como se podría pensar el rediseño basado en microservicios es solo un enfoque de muchos otros posibles y que este enfoque no solo representa una descomposición de los componentes, sino también la capacidad para organizarse en equipos más pequeños e independientes. Finalmente, como trabajo adicional relacionado proponen la investigación todos los patrones relacionados con la arquitectura de microservicios y comparar los beneficios para diferentes sectores intentando así una forma estandarizada de plantear la arquitectura.

Implementación de Middleware Publicador/Subscriptor para Aplicaciones Web de Monitoreo
(Defossé et al., 2017)

En este artículo, los autores relatan la prevalencia del estilo arquitectónico REST basado en el protocolo HTTP para la comunicación sobre otros modelos como SOAP, CORBA, etc. Resaltan uno de los problemas de la mecánica de comunicaciones que presenta REST la cual imposibilita que un cliente reciba las actualizaciones de un recurso si es que este no inicia una solicitud antes. De acuerdo a su estudio se elaboró una aplicación web de monitoreo de estaciones de transformación eléctricas compuesto por

una base de datos, un recolector de estados y eventos y finalmente, por el módulo de visualización. El problema surgió en la repetición de invocaciones que debían hacer para mantener actualizado el estado de la estación generando cargas adicionales de trabajo y posibles desfases y retrasos del estado real. Como solución a este problema aprovecharon la base de datos para publicar y suscribir mensajes y el uso de websockets para la actualización en tiempo real sin iniciar una solicitud desde el cliente.

Como conclusión los autores encontraron una mejora en el desempeño y la experiencia del usuario, solucionando así los problemas de desfase y retraso de la información.

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

En base al punto 3.2.3 correspondiente a la sección de etapas y metodología, se organizará esta sección. El autor fue parte del development team. Las actividades se dieron conforme al cronograma que se muestra en la Figura 20.

Figura 20.

Cronograma del proyecto

		Inicio	Desarrollo	Certificación	Mantenimiento
Diciembre	S1 - S2	sp 0			
	S3 -S4		sp1		
Enero	S1 - S2		sp2		
	S3 - S4		sp3		
Febrero	S1 - S2		sp4		
	S3 - S4		sp5		
Marzo	S1 - S2		sp6		
	S3 - S4		sp7		
Abril	S1 - S2		sp8		
	S3 - S4		sp9		
Mayo	S1 - S2		sp10		
	S3 - S4		sp11		
Junio	S1 - S2		sp12		
	S3 - S4		sp13		
Julio	S1 - S2			sp14	
	S3 - S4			sp15	
Agosto	S1 - S2			sp16	
	S3 - S4			sp17	
Setiembre	S1 - S2				sp18
	S3 - S4				...

Nota. Adaptado del proyecto.

3.2.5.1 Descubrimiento

Presentación de la necesidad

Esta iniciativa comienza por la necesidad de crear un MDM único que permita gestionar la información de productos importados y locales (por país) a fin de entregarla de forma correcta, completa, oportuna y consistente a diferentes áreas y canales de venta del negocio. Esta iniciativa y el producto a construir fueron denominados como PIM corporativo. Como se mencionó anteriormente esto involucraba a los productos importados que son gestionados por la plataforma V2S lo que conllevo a la necesidad de una integración entre V2S y PIM. Durante el planteamiento de esta necesidad, se creó RPC como plataforma intermediaria entre la integración V2S – PIM, surgiendo de ese modo la necesidad de integrar V2S y RPC.

Definición del equipo y metodología

El marco de trabajo seleccionado fue Scrum, contando con alrededor de 17 sprints, y un equipo conformado por alrededor de 8 personas. Se debe tener en cuenta que el equipo encargado fue el de Primera Línea, que a día de hoy recibe el nombre de squad V2S, y que como se mencionó en el punto 2.6 atiende las necesidades a ritmo del negocio y no tiene un único objetivo por sprint. La Tabla 8 muestra el equipo scrum conformado por el proyecto.

Tabla 8.

Equipo Scrum

Rol	Cantidad
Scrum Master	1
Product Owner	1
Developers	8

Nota. Elaboración propia.

Un punto importante a tener en cuenta es que algunos miembros encargados de la construcción de RPC apoyaron en un inicio la definición y construcción de la integración.

Definición inicial de requerimientos

Parte de esta etapa inicial consistía en armar los requerimientos (product backlog) a través de las historias de usuario y documentarlos mediante la herramienta Jira. La Figura 21 muestra la lista de épicas que se trabajaron a través de todo el proyecto.

Figura 21.

Lista de épicas del proyecto

T	Key ↑	Summary
	TV2SC-1102	Creación de Prod y Emp
	TV2SC-1103	Modificación de Productos
	TV2SC-1104	Creación de Proveedor
	TV2SC-1105	Modificación de Proveedor
	TV2SC-1106	Actualizacion de PIM a V2S
	TV2SC-1966	Legado Mama-Baby
	TV2SC-2218	Legado Modificación de Productos
	TV2SC-2643	Legado Mejora Campañas
	TV2SC-2644	Legado Creación de Productos
	TV2SC-2645	Legado Modificación de Proveedor
	TV2SC-2661	Legado Creación Proveedores
	TV2SC-2829	Mejoras Clacom-Marca
	TV2SC-2830	Requerimientos Tecnicos
	TV2SC-3260	Visor Integración
	TV2SC-3730	CV Digital
	TV2SC-3771	Codigos de Barra (Visualizar)
	TV2SC-3848	Actividades paso a produccion
	TV2SC-3906	Manage Product Mejoras
	TV2SC-3920	Campañas(UAT)

Nota. Adaptado del proyecto

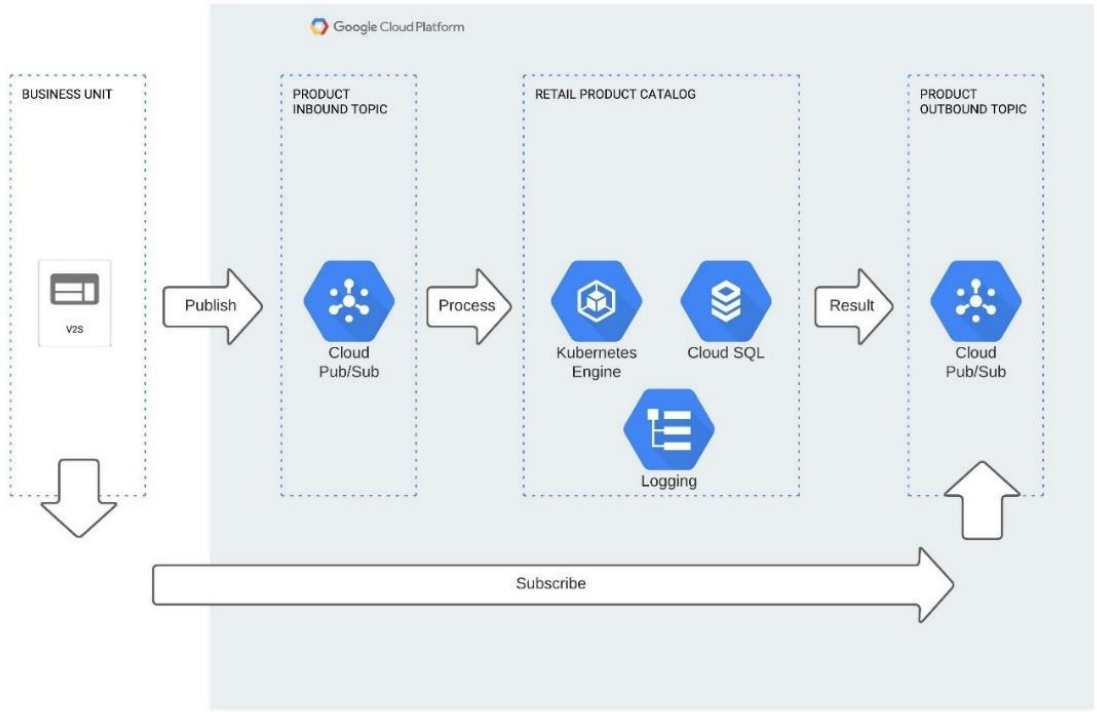
3.2.5.2 Definición de la arquitectura de integración

Para definir la arquitectura de integración, había que entender cómo funcionaba el catálogo de productos a alto nivel. La Figura 22 expresa al catálogo de productos

mediante dos interfaces, las cuales corresponden a un tópico de entrada y un tópico de salida (de nombres inbound y outbound respectivamente), entre ambas interfaces se encuentran las apis, base de datos, servicios de logs, entre otros que soportan las operaciones del catálogo. Para que una unidad de negocio pueda integrarse con el catálogo debe publicar un mensaje en su tópico de entrada y suscribirse al tópico de salida para saber si el mensaje se procesó con éxito.

Figura 22.

Diagrama a alto nivel de RPC.



Nota. Elaboración propia.

El mensaje que se entrega y devuelve en cada tópico corresponde a una estructura estándar de lo que se conoce como un evento corporativo. Este evento corporativo representa la estructura de un contrato de comunicación entre las plataformas que deseen integrarse con el catálogo, el cual se puede dividir en dos partes, la estructura correspondiente a los metadatos del evento y la estructura del mensaje como tal. La estructura del mensaje también debe seguir la estructura de un contrato de comunicación bien definido pues será procesado posteriormente por las apis del catálogo de productos.

La Figura 23 muestra la estructura de un evento corporativo en un formato json.

Figura 23.

Estructura de un evento corporativo.

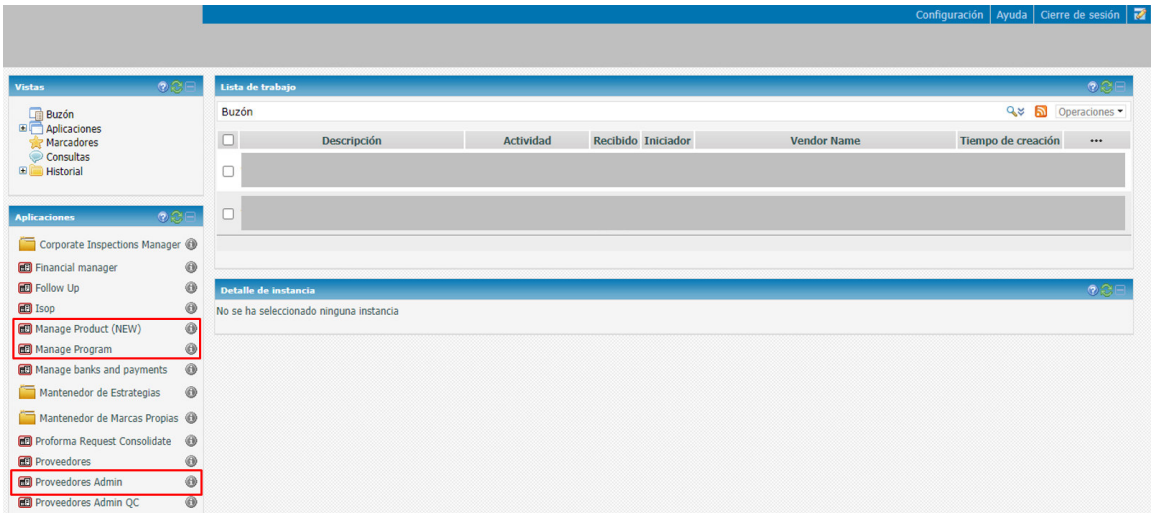
Nota. Adaptado del proyecto.

La definición del contrato de un evento se puede ver en el Anexo 1 y el contrato del mensaje se puede ver en el Anexo 2.más adelante

Una vez entendido el funcionamiento del catálogo quedaba revisar que interfaces de comunicación ofrecía la plataforma de importaciones, para ello revisemos un poco a alto nivel como se compone. La plataforma de importaciones es un sistema legado, compuesta entre una combinación de tecnologías con desfase tecnológico como módulos diseñados con primefaces y servicios SOAP; y tecnologías de más reciente uso como son los SPA construidos con angular y servicios REST. Cada módulo dentro de la plataforma se encuentra orquestado por la suite BPM de Oracle usada para administrar procesos empresariales ejecutados por los usuarios a través de la web. La Figura 24 muestra la plataforma de importaciones, resaltando los módulos encargados de administrar información relacionada a los productos importados directa e indirectamente.

Figura 24.

Página de inicio de V2S.



Nota. Adaptado del proyecto.

Los tres módulos afectados como se resalta en a figura anterior son:

- Administración de productos
- Campañas
- Administración de proveedores.

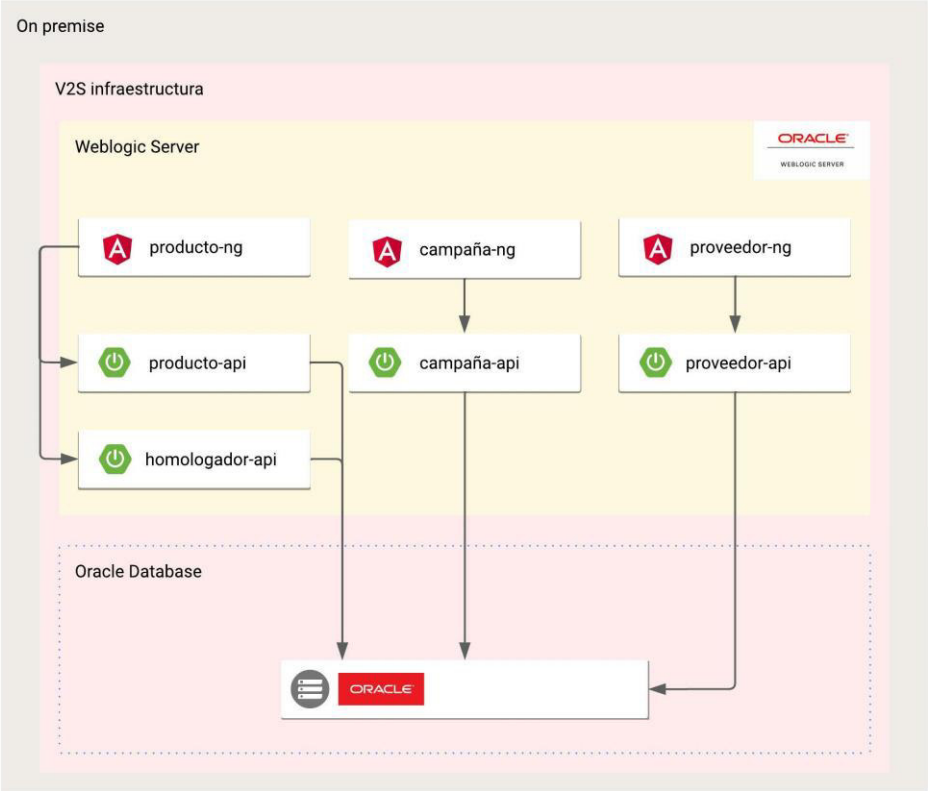
La Figura 25 muestra la arquitectura a alto nivel de los módulos que se deben de modificar en la plataforma de importaciones. Como se puede ver estos módulos se despliegan en un servidor weblogic en infraestructura on premise y se componen de servicios REST construidos en java con springboot en el backend, y angular para el frontend, conectados a una base de datos relacional Oracle. Siendo estos servicios y la base de datos las interfaces de comunicación que se expondrían desde esta plataforma para la integración.

Finalmente, una vez analizados el catálogo de productos y la plataforma de importaciones, quedaba diseñar la arquitectura, la cual a través de los 2 primeros sprint se fue iterando llegando a su primera versión la cual no cambiaría mucho después.

La arquitectura de la integración se definió como un conjunto de microservicios con responsabilidades independientes las cuales se comunicaban mediante patrones de solicitud/respuesta (microservicio - microservicio) y mensajería (microservicio – tópico

y suscripción - microservicio). Si bien supone una arquitectura de integración asíncrona, el tiempo entre que se genera una solicitud de integración y este se completa no toma mucho tiempo en comparación a otras arquitecturas como el uso de ETL.

Figura 25.
Diagrama de alto nivel de V2S.



Nota. Elaboración propia.

La integración obligo la construcción y administración de un total de 5 microservicios, 1 tópico, 2 suscripciones y 1 librería, a parte de las modificaciones sobre la plataforma de importaciones. Los entregables de la definición de arquitectura y la definición de responsabilidades de sus componentes pueden verse en el Anexo 3 y el **¡Error! No se encuentra el origen de la referencia.** respectivamente.

3.2.5.3 Aprovisionamiento de la infraestructura

El aprovisionamiento de la infraestructura consistió en todas aquellas actividades relacionadas a brindar soporte al desarrollo, despliegue y pruebas de los nuevos

componentes de la integración. Las actividades de aprovisionamiento se ejecutaron en su mayoría entre el sprint 1 y 2, en los siguientes sprints siguientes se limitó al acompañamiento, ajustes y refactorizaciones.

Entre estas actividades más importantes de este punto tenemos las relacionadas a la generación de arquetipos, habilitación de repositorios y herramientas CI/CD y la habilitación de la infraestructura base.

Arquetipos de microservicios

Los arquetipos son plantillas de proyectos con configuraciones, librerías y dependencias sobre el cual se pueden desarrollar proyectos de un mismo tipo. En este caso los arquetipos fueron del tipo api, publicador y suscriptor de acuerdo a la definición de la arquitectura. Además, los microservicios construidos en base a estos arquetipos debían tener en cuenta recomendaciones las cuales de la Tabla 9.

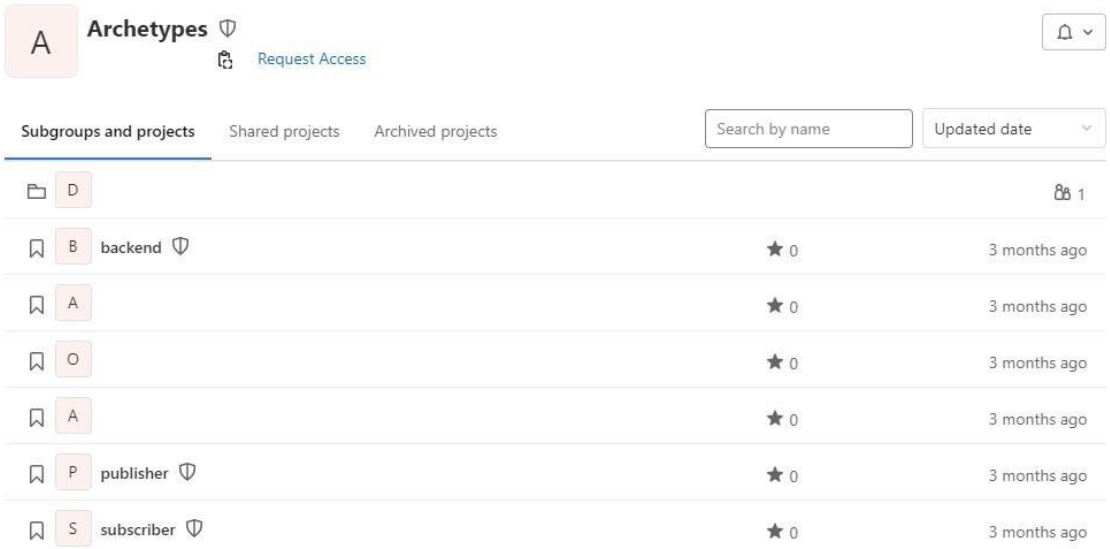
Tabla 9.
Recomendaciones de backend.

Recomendación	Implementación
Rastreo de errores	Cada solicitud http establece encabezados de transporte predefinidos cuyo valor debe ser el mismo en el ciclo de vida de la solicitud. Para el caso de los eventos es similar, solo que se establecen a nivel de atributos customizados.
Respuestas estándar	Los microservicios basados en api rest manejaron respuestas estándar basadas única y exclusivamente en los códigos HTTP. Los errores manejaban una respuesta estándar compuesta por el nombre del error y el mensaje asociado al error.
Autenticación	La autenticación se delega al api manager de kong y solo para los microservicios que necesiten estar expuestos a internet.
Encriptación	La encriptación de información sensible solo se aplicó a los mensajes que se entregaban en los tópicos de RPC.

Nota. Adaptado del proyecto.

La construcción de los arquetipos definiría la tecnología a usarse, la cual tuvo como lenguaje de programación java gestionado las dependencias con maven y el framework de springboot. El entregable relacionado a los arquetipos se muestra en la Figura 26.

Figura 26.
Entregable de arquetipos.



Nota. Adaptado del proyecto.

Repositorios y herramientas CI/CD

La herramienta usada para administrar las versiones de código fue Gitlab y su conjunto de herramientas para automatizar los procesos de integración y despliegue continuo. Los repositorios creados para cada microservicio siguen la siguiente nomenclatura:

[dominio].[capacidad].[capa].[componente].[recurso]

A continuación, se muestra la Tabla 10 con la descripción de cada elemento que compone el nombre de cada recurso en el repositorio.

Tabla 10.

Nombre estándar de repositorio.

Elemento	Descripción
dominio	Referencia a las siglas definidas en el modelo de capacidades retail
capacidad	Referencia a las siglas definidas en el modelo de capacidades retail
capa	Tipo de capa a la cual corresponde el componente
componente	Tipo de componente, con referencia a la tecnología principalmente
recurso	Identifica el recurso utilizado, lo más explícito posible

Nota. Adaptado del proyecto.

Además de ello se establece la estrategia de unión de cambios, descripción base de cada repositorio, permisos a los miembros del equipo, ramas por defecto y los flujos de integración y despliegue continuo.

Los flujos de integración y despliegue continuo usaron el Gitlab CI/CD para automatizar las tareas a través de canalizaciones de flujos de trabajos (pipelines). Para el caso de los servicios basados en java de este proyecto las tareas de los pipelines fueron clasificados en 3 grupos y son listados en la Tabla 11. Además, las tareas de cada grupo pueden verse en el Anexo 5.

Tabla 11.

Grupos de tarea del pipeline.

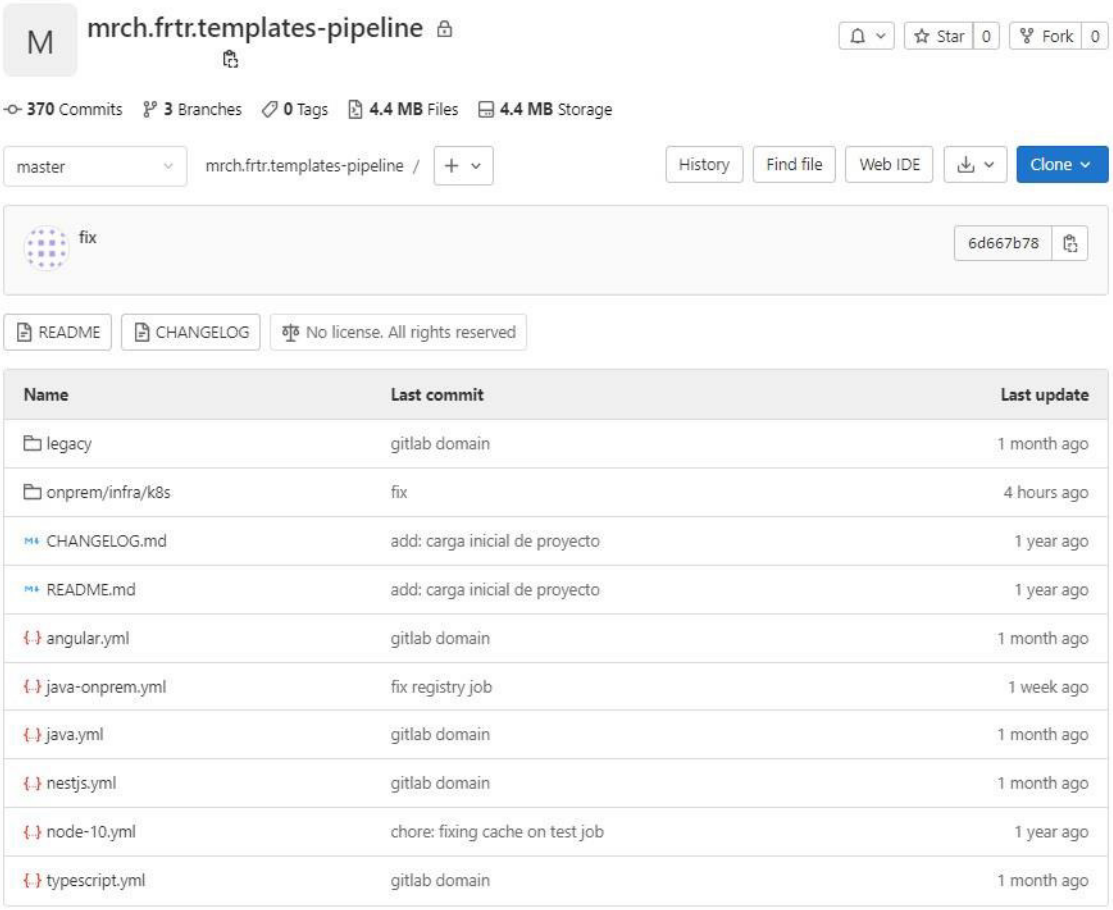
Grupo	Objetivo
Calidad de código	Análisis estático de código y pruebas del mismo.
Seguridad	Tareas relacionadas con el DevSecOps, automatizadas.
Construcción y despliegue	Compilación, empaquetado, registro y construcción de componentes software.

Nota. Adaptado del proyecto.

Estas tareas eran configuradas de acuerdo a la necesidad en un proyecto de configuración, los cuales existían tanto para las piezas de integración en cloud, como en on premise. La Figura 27 muestra uno de los entregables relacionados a los servicios on premise en general.

Figura 27.

Entregable de tareas de CI/CD.



Nota. Adaptado del proyecto.

Infraestructura base

Aprovisionamiento de los ambientes en donde existirán los componentes de la integración. De acuerdo al diagrama de arquitectura de la integración los componentes deben permanecer en un entorno on premise (Rancher) y nube (GCP) en donde la división de los ambientes fue la siguiente: desarrollo, pruebas, pruebas de usuario (UAT) y producción. A nivel de GCP la separación de los ambientes se hizo en base a la creación de diferentes proyectos, mientras que en on premise la separación fue una combinación entre diferentes servidores por ambiente de producción y los ambientes restantes, en donde estos se últimos se separaron por un espacio de nombres de Kubernetes.

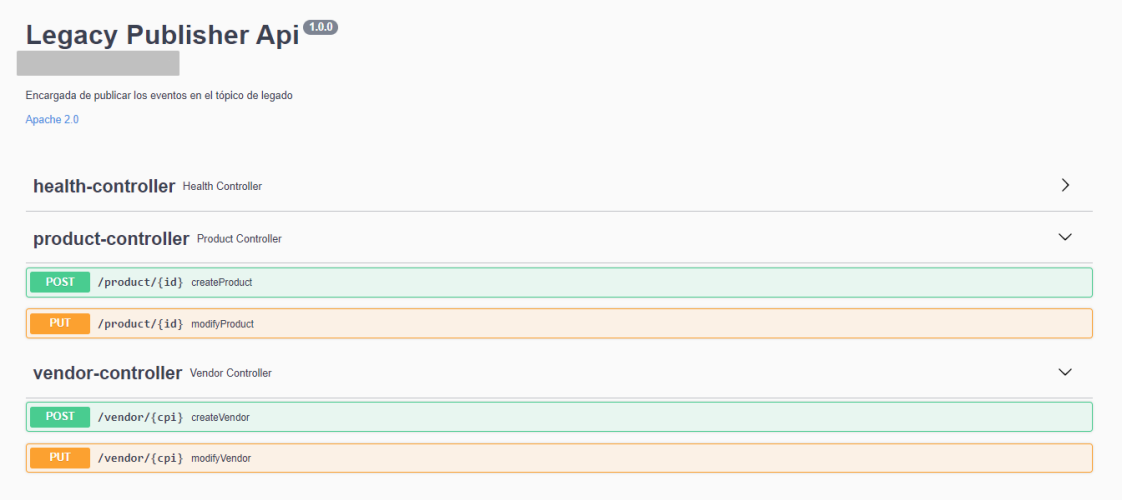
Además de ello por cada ambiente se tenía que configurar los tópicos y suscripciones, servicios de monitoreos, administración de imágenes, orquestación de contenedores y administradores de apis, todo esto gestionado por el equipo de DevOps.

3.2.5.4 Implementación de la integración

La implementación de la integración se dio principalmente entre los sprints 3 al 13 aproximadamente. Consistió en el desarrollo de los microservicios de integración, construcción de pruebas automatizadas, despliegue y configuración, solución de errores encontrados y mejoras del desarrollo en cada iteración.

La construcción de los microservicios, se hizo en base al tipo y a la infraestructura en donde se desplegará (on premise o nube). En términos generales la mayoría de los microservicios construidos exponen un conjunto de operaciones (endpoints), las cuales son documentadas a través de swagger como se observa en la Figura 28, en donde se toma como ejemplo al legacy publisher api.

Figura 28.
Contrato del legacy publisher api.



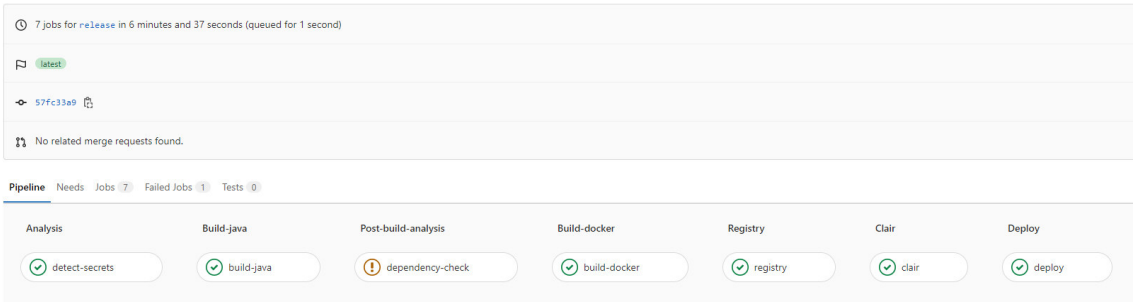
Nota. Adaptado del proyecto.

Además, en todos los microservicios se configuraba un archivo de nombre ". gitlab-ci.yml" el cual serviría para ejecutar los pipelines. La Figura 29 muestra el

despliegue automatizado del microservicio y su despliegue en el clúster de kubernetes on premise administrado por rancher.

Figura 29.

Pipeline del legacy publisher api.

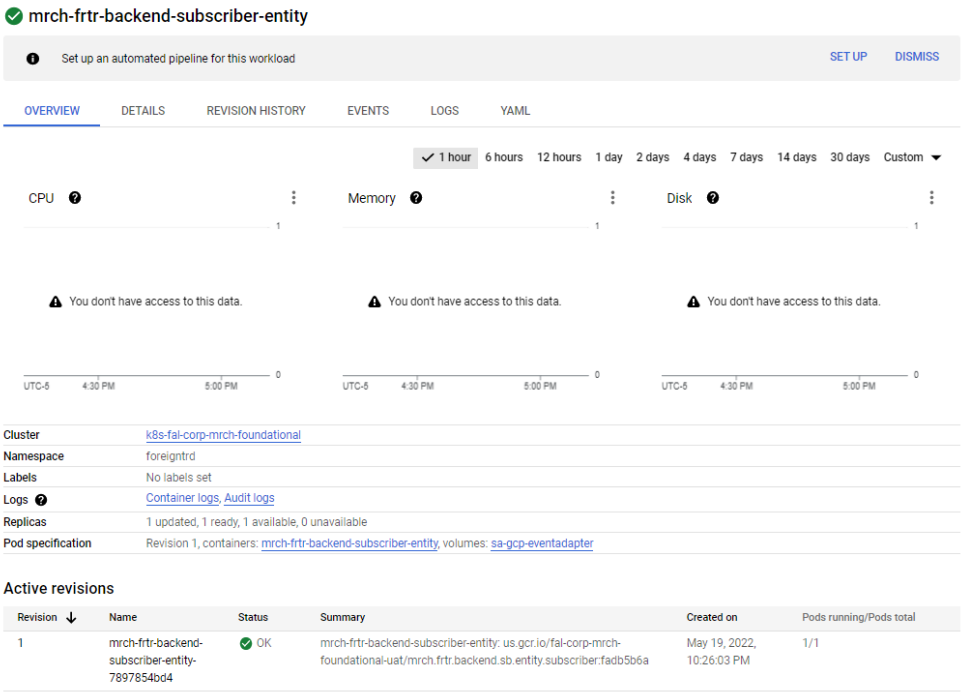


Nota. Adaptado del proyecto

El despliegue de los microservicios se hace en el clúster on premise administrado a través de rancher o en cloud administrado por kubernetes engine. La Figura 30 muestra el legacy subscriber api desplegado en el entorno cloud.

Figura 30.

Legacy subscriber api en kubernetes engine.

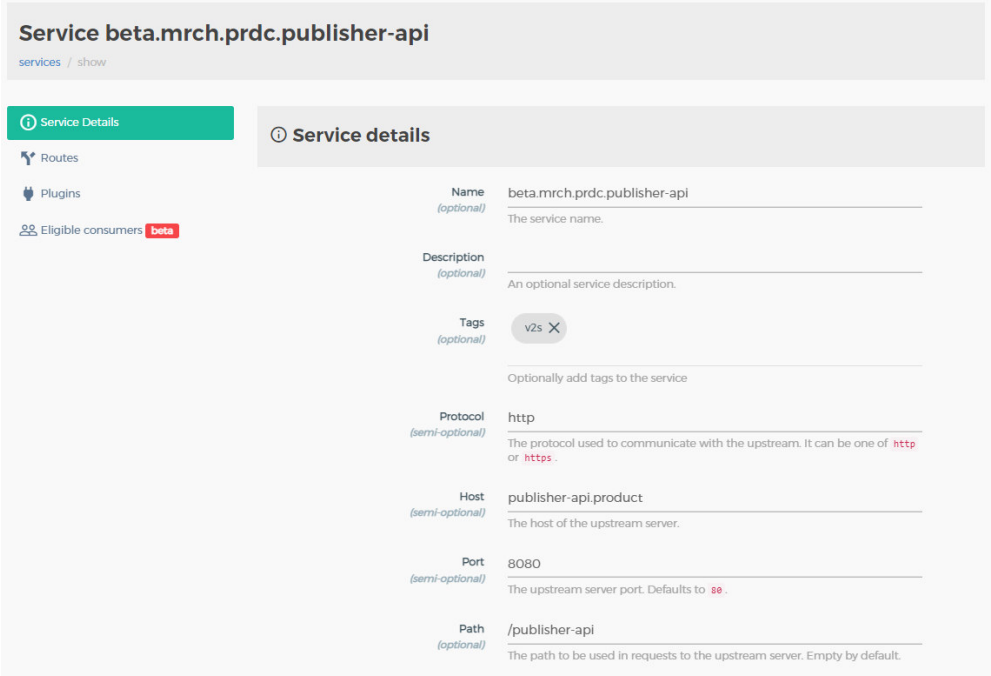


Nota. Adaptado del proyecto.

Desplegado en el clúster de kubernetes, queda exponerlo para que pueda ser usado por aplicaciones o microservicios externos. Para ello se usó kong y esto solo para los microservicios que exponen endpoints. La Figura 31 muestra la configuración.

Figura 31.

Exposición del legacy publisher api.

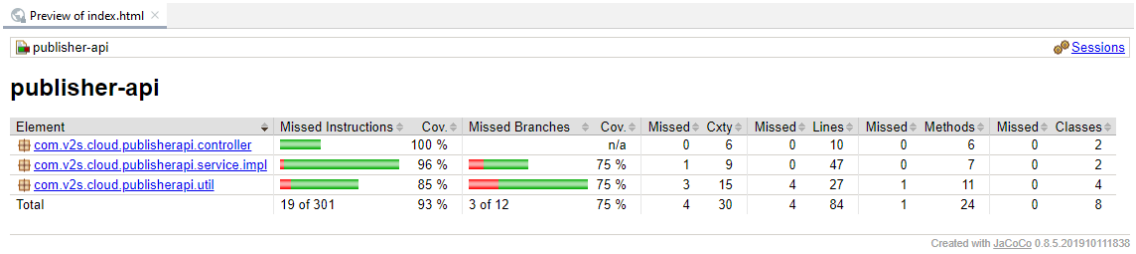


Nota. Adaptado del proyecto.

La construcción de cada microservicio implicaba la ejecución de pruebas unitarias y que estas tuviesen una cobertura mayor al 80%, la Figura 32 muestra el reporte generado por la herramienta de cobertura de código utilizada, para el legacy publisher api.

Figura 32.

Pruebas automatizadas del legacy publisher api.

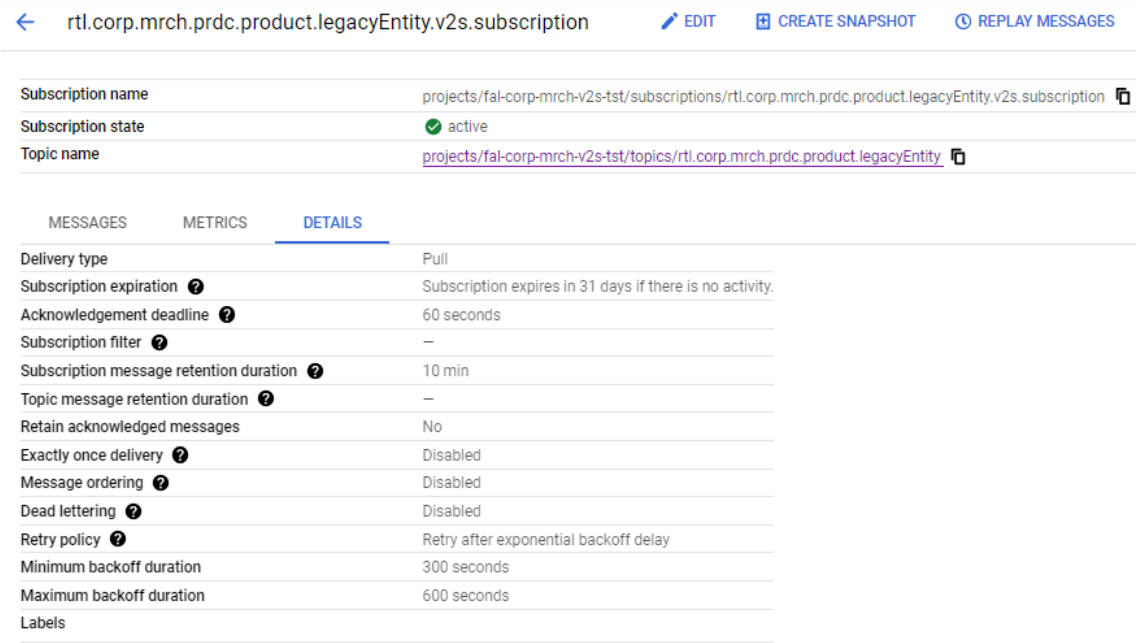


Nota. Adaptado del proyecto.

La configuración del tópico y suscripción se hacen de forma manual. La muestra la configuración de la suscripción y el tópico.

Figura 33.

Configuración de Pub/Sub.



Nota. Adaptado del proyecto.

Las suscripciones se configuraron para que los microservicios de tipo subscriber consultaran continuamente los cambios en el servicio de Pub/Sub.

La construcción de los microservicios, se hicieron en base a la arquitectura de la integración definida. A continuación, se describe la integración en base 3 flujos.

1. Notificación de la plataforma de importaciones
2. Publicación en catálogo de productos
3. Suscripción al catálogo de productos

Notificación de la plataforma de importaciones

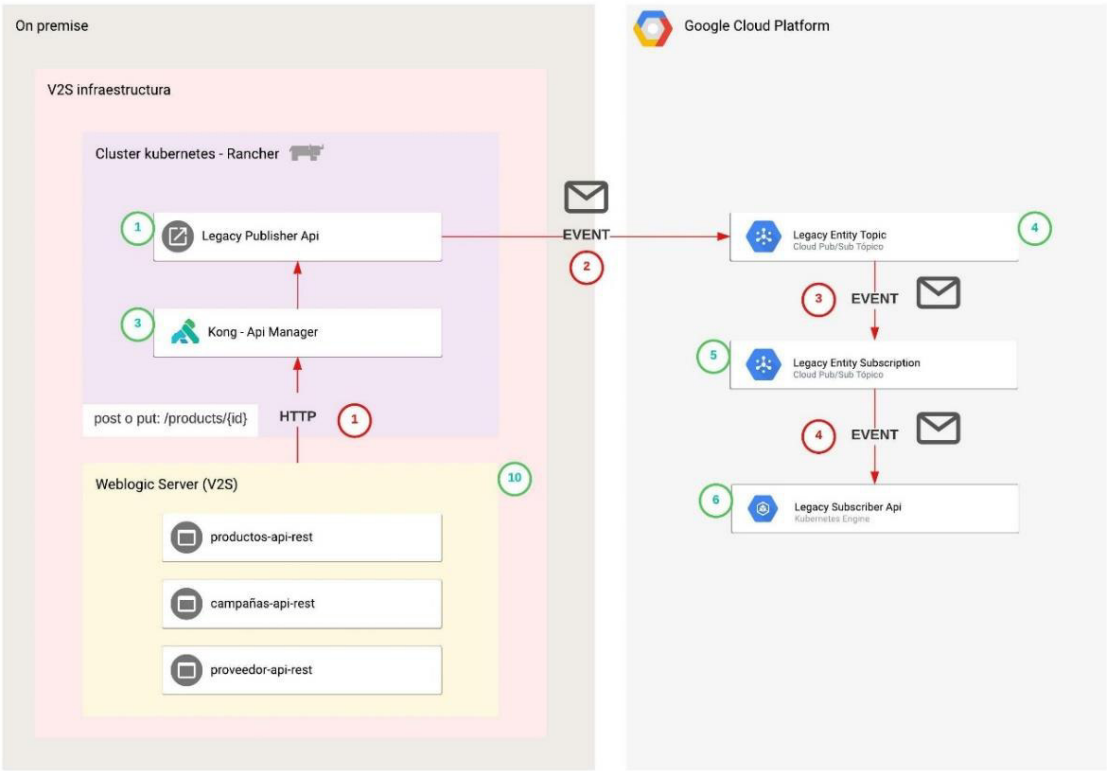
El flujo de comunicación del legado es descrito a través de los siguientes pasos.

1. Los procesos de creación o modificación de producto que se dan en la plataforma V2S gatillan una solicitud HTTP hacia el microservicio legacy publisher api (expuesto a través de kong).
2. Este a su vez publica un evento en el legacy entity topic.
3. El evento publicado en el legacy entity topic es encolado y posteriormente asignado al legacy entity subscription.
4. Finalmente, el legacy subscriber api está a la escucha de nuevos mensajes para procesarlos y enviarlos al entity sync (mostrado en el siguiente flujo).

La Figura 34 resume de forma gráfica lo descrito en los pasos anteriormente mencionados.

Figura 34.

Notificación de V2S.



Nota. Elaboración propia.

Publicación en catálogo de productos

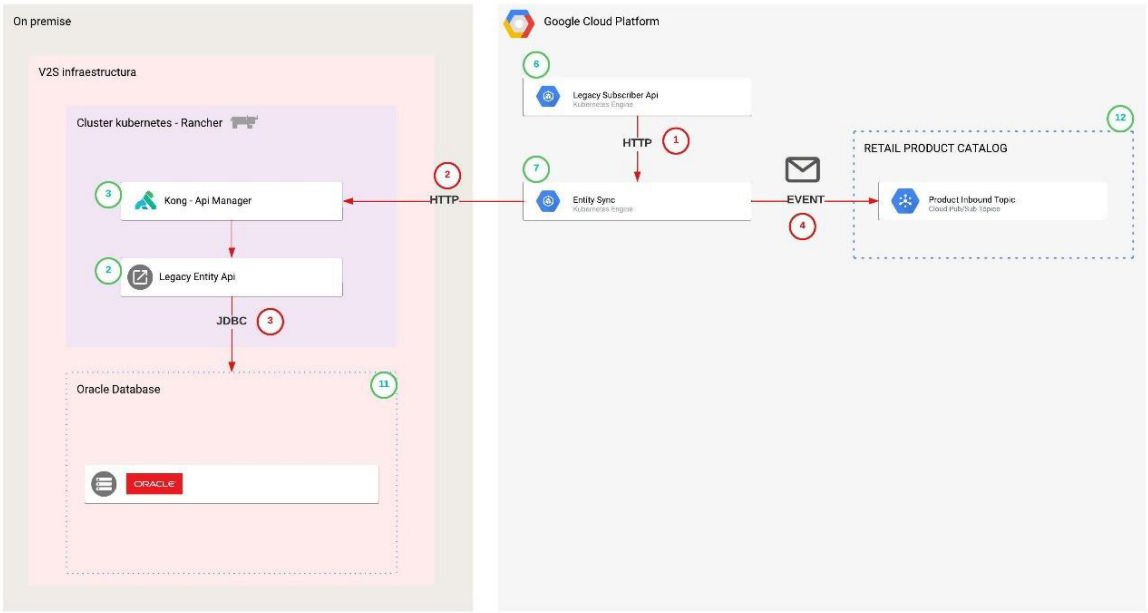
El flujo de publicación en el catálogo de productos se describe en los pasos enumerados a continuación.

1. El legacy subscriber api una vez recibe el mensaje del legacy entity subscription (flujo anterior) envía una solicitud HTTP hacia el entity sync con el id del producto cuya creación o modificación desea integrar.
2. El entity sync con el id de producto se encarga de hacer una nueva solicitud HTTP al legacy entity api para obtener una versión previa de la información que se debe enviar al catálogo.
3. El legacy entity api, recibe la solicitud del entity sync y con el id del producto recolecta la información necesaria del mismo desde la base de datos de la plataforma V2S.
4. Finalmente, el entity sync con la respuesta obtenida del legacy entity api publica el evento en el tópico product inbound topic del catálogo de productos.

La Figura 35 resume de forma gráfica lo descrito.

Figura 35.

Publicación en RPC.

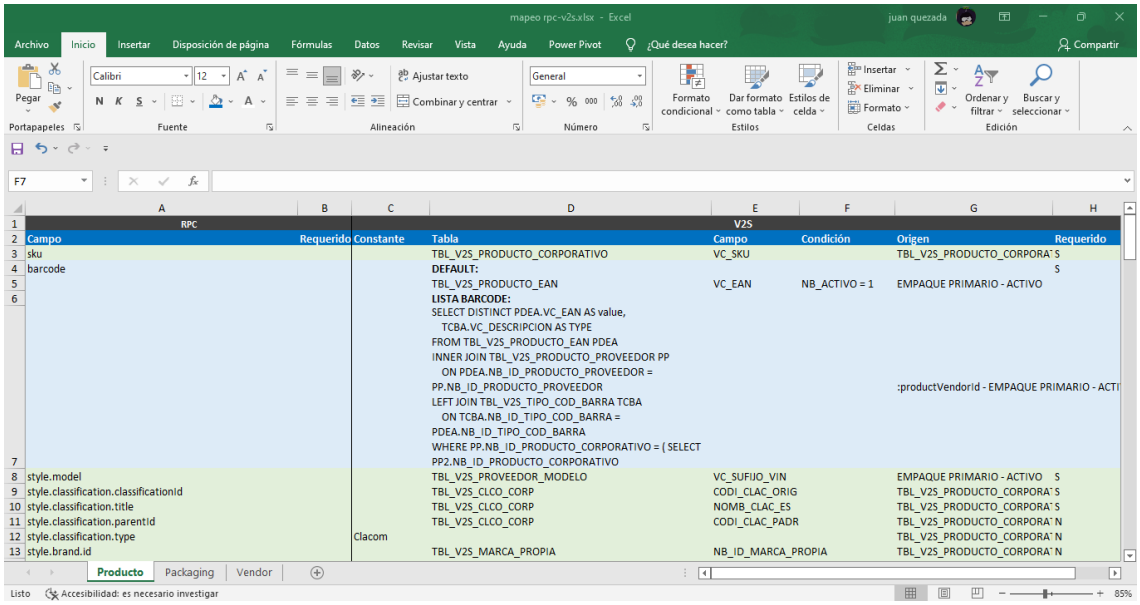


Nota. Elaboración propia.

En el paso 3 del proceso el legacy entity api hace una extracción de la información de producto, para que posteriormente en el paso 4 el entity sync haga una pequeña transformación de los datos y finalice cargando los mismos al tópico del catálogo de productos. Para ello era necesario identificar una correspondencia entre cada elemento del catálogo de productos con la plataforma V2S, como muestra en la Figura 36.

Figura 36.

Mapeo de datos entre V2S - RPC.



	A	B	C	D	E	F	G	H
1	RPC				V2S			
2	Campo	Requerido	Constante	Tabla	Campo	Condición	Origen	Requerido
3	sku			TBL_V2S_PRODUCTO_CORPORATIVO	VC_SKU		TBL_V2S_PRODUCTO_CORPORA	S
4	barcode			**DEFAULT:**				S
5				TBL_V2S_PRODUCTO_EAN	VC_EAN	NB_ACTIVO = 1	EMPAQUE PRIMARIO - ACTIVO	
6				**LISTA BARCODE:** SELECT DISTINCT PDEA.VC_EAN AS value, TCBA.VC_DESCRIPCION AS TYPE FROM TBL_V2S_PRODUCTO_EAN PDEA INNER JOIN TBL_V2S_PRODUCTO_PROVEEDOR PP ON PDEA.NB_ID_PRODUCTO_PROVEEDOR = PP.NB_ID_PRODUCTO_PROVEEDOR LEFT JOIN TBL_V2S_TIPO_COD_BARRA TCBA ON TCBA.NB_ID_TIPO_COD_BARRA = PDEA.NB_ID_TIPO_COD_BARRA WHERE PP.NB_ID_PRODUCTO_CORPORATIVO = (SELECT PP2.NB_ID_PRODUCTO_CORPORATIVO			:productVendorId - EMPAQUE PRIMARIO - ACTI	
7								
8	style.model			TBL_V2S_PROVEEDOR_MODELO	VC_SUFIJO_VIN		EMPAQUE PRIMARIO - ACTIVO	S
9	style.classification.classificationId			TBL_V2S_CLCO_CORP	CODI_CLAC_ORIG		TBL_V2S_PRODUCTO_CORPORA	S
10	style.classification.title			TBL_V2S_CLCO_CORP	NOMB_CLAC_ES		TBL_V2S_PRODUCTO_CORPORA	S
11	style.classification.parentId			TBL_V2S_CLCO_CORP	CODI_CLAC_PADR		TBL_V2S_PRODUCTO_CORPORA	N
12	style.classification.type		Clacom				TBL_V2S_PRODUCTO_CORPORA	N
13	style.brand.id			TBL_V2S_MARCA_PROPIA	NB_ID_MARCA_PROPIA		TBL_V2S_PRODUCTO_CORPORA	N

Nota. Adaptado del proyecto.

Suscripción al catálogo de productos

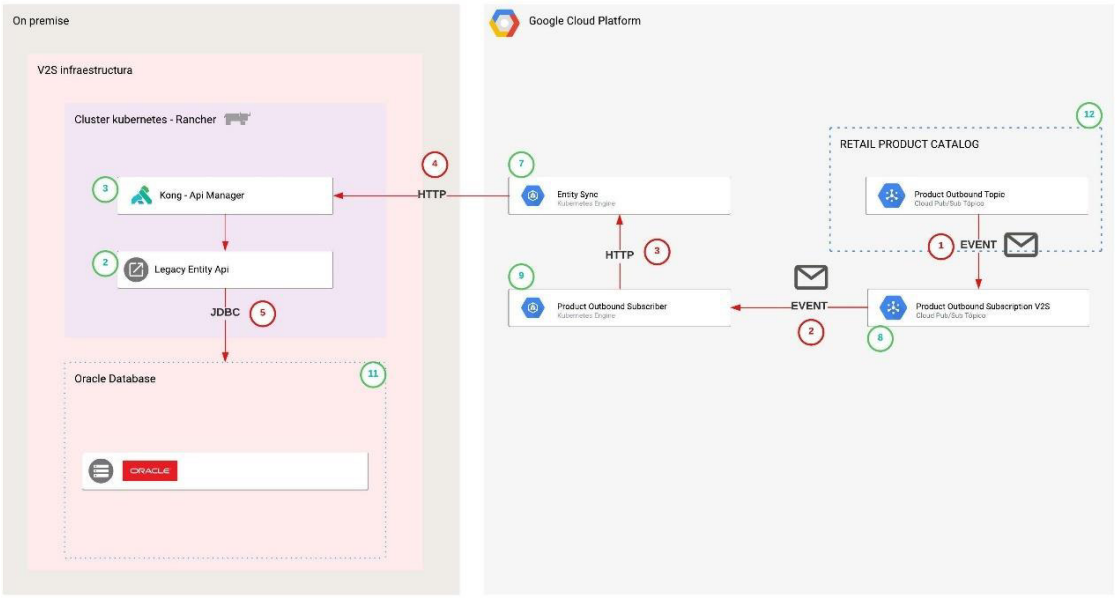
El flujo de suscripción al catálogo de productos se describe en los pasos enumerados a continuación. Lo descrito se muestra en la siguiente figura.

1. Una vez procesado el evento en RPC, este llega al product outbound subscription V2S, procedente del product outbound topic.
2. Una vez recibido el evento es enviado al product outbound subscriber, quien validara si la solicitud fue exitosa, o hubo algún fallo.
3. En base a la determinación de éxito o error en la integración, llamará mediante una solicitud HTTP al entity sync a su operación de éxito o error según sea el caso.

4. El entity sync bajo la misma lógica anterior llamará a las operaciones de éxito o error del legacy entity api.
5. Finalmente, el entity api registrará el resultado de la integración en la base de datos de la plataforma de V2S, en la tabla de solicitudes de notificación al catálogo. Además de ello si la operación fue exitosa actualizara información como marca y clasificación comercial también en base de datos y si la notificación fue de creación y el producto a integrar no poseía un sku en V2S (identificador único de producto), el catálogo genera uno y este es actualizado en este punto del proceso en V2S.

La Figura 37 resume de forma gráfica lo descrito en los pasos anteriormente mencionados.

Figura 37.
Suscripción a RPC.



Nota. Elaboración propia.

3.2.5.5 Mejoras sobre la plataforma V2S.

Para poder integrar la información de productos en el catálogo de productos, era necesario que cada vez que se creara o modificara un producto (o información relacionada) se realice una solicitud HTTP al microservicio legacy publisher api para dar

inicio a la integración. Los sistemas de la plataforma V2S impactados fueron los relacionados a la administración comercial, estos sistemas y la justificación de su impacto se muestra en la Tabla 12.

Tabla 12.

Descripción de los módulos de V2S.

Sistema	Motivo de impacto
Administración de productos	Encargado de administrar la información de los productos importados. Procesos de creación y modificación de productos masivos e individual. Procesos de homologación y des homologación de productos.
Campañas	Proceso puede crear y modificar información productos importados.
Administración de proveedores	Proceso de des homologación de productos

Nota. Adaptado del proyecto.

Para lograr la notificación al legacy publisher api, se creó una librería de nombre notify pim hecha en Java que encapsulaba operaciones de utilidad para realizar la solicitud HTTP. Las operaciones que expone esta librería son mostradas en la Figura 38.

Figura 38.

Contrato de la librería de notificación.

```
public interface PimService {

    public ResponsePimDTO saveProdPim(RequestPimDTO request) throws V2sException;

    public List<ResponsePimDTO> retryNotifyPim(RequestPimDTO request) throws V2sException;

    public List<ResponsePimDTO> retryAllNotifyPim() throws V2sException;

    public RequestPimDTO getActionPim(RequestPimDTO entidad);
}
```

Nota. Adaptado del proyecto.

Para la construcción de la librería fue necesario identificar que productos debían integrarse con el catálogo, además de validar a nivel general si la integración debería estar activa y finalmente saber el estado de la integración de cada producto. Para los dos primeros puntos, la solución fue mantener dos flags a nivel de base de datos, uno a nivel del producto y otro a nivel general que controlasen la integración. La Tabla 13 resume que productos se integraran en base a los valores de estos flags.

Tabla 13.

Flags de integración RPC.

Flag general	Flag producto	Resultado
T	T	Integra
T	F	No integra
F	T	No integra
F	F	No integra

Nota. Elaboración propia.

Para el tercer punto, se construyó la tabla de notificación catálogo tbl_v2s_notify_catalog a nivel de base de datos, cuyas columnas son mostradas en la Figura 39.

Figura 39.

Diccionario de la tabla tbl_v2s_notify_catalog.

Column Name	#	Tipo	No Nulo	Comentario
NB_ID_NOTIFY_CATALOG	1	NUMBER(18,0)	[v]	PK
NB_ID_ENTIDAD	2	NUMBER(18,0)	[ ]	ID DE LA ENTIDAD A SINCRONIZAR EN LA INTEGRACIÓN
VC_ID_EVENTO	3	VARCHAR2(50)	[ ]	ID EVENTO GENERADO EN LA INTEGRACIÓN
NB_HTTP_STATUS	4	NUMBER(18,0)	[ ]	ESTADO DE RESPUESTA HTTP DE LA ULTIMA ACTUALIZACIÓN
VC_RESPONSE	5	VARCHAR2(500)	[ ]	MENSAJE DE RESPUESTA DE LA ULTIMA ACTUALIZACIÓN
VC_USUARIO	6	VARCHAR2(50)	[ ]	USUARIO GENERADOR DE LA CREACION/MODIFICACIÓN
DT_FECHA_CREACION	7	DATE	[ ]	FECHA DE CREACION DE LA SOLICITUD
DT_FECHA_MODIFICACION	8	DATE	[ ]	FECHA DE MODIFICACION DE LA SOLICITUD
VC_ACCION	9	VARCHAR2(3)	[ ]	TIPO ACCION REALIZADA CREACIÓN O MODIFICACIÓN
VC_DOMINIO	10	VARCHAR2(20)	[ ]	DOMINIO DE LA ENTIDAD EN LA NOTIFICACION
NB_ESTADO	11	NUMBER(22,0)	[ ]	ESTADO SOLICITUD DE INTEGRACIÓN

Nota. Adaptado del proyecto.

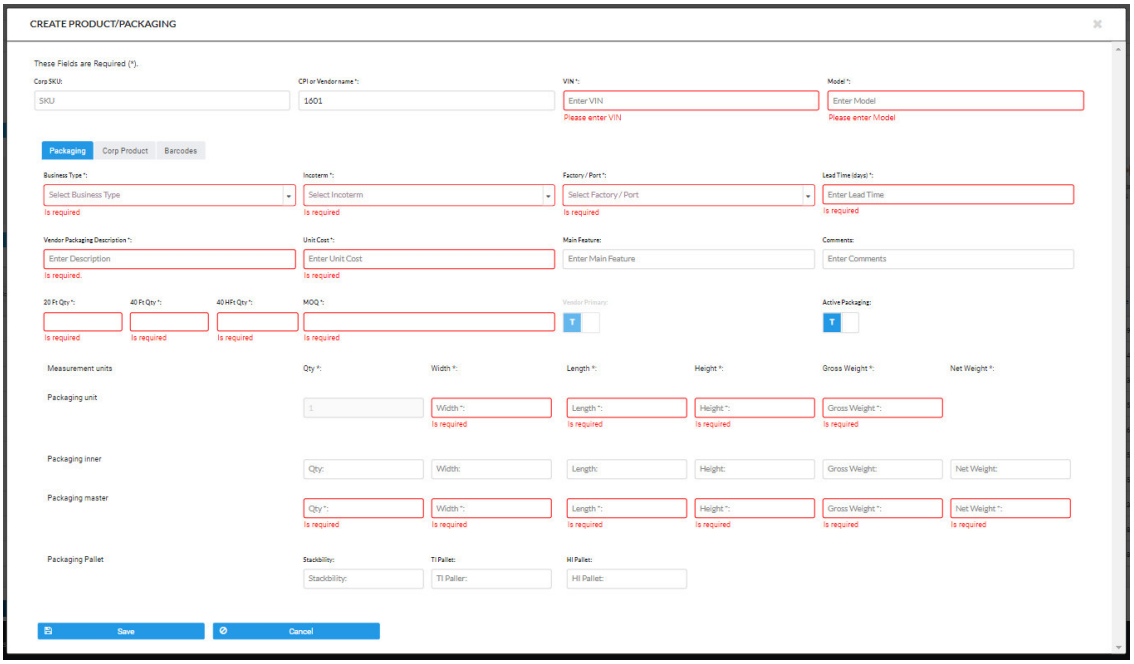
En base a la identificación de los sistemas impactados y la construcción de la librería, restaba identificar cuáles eran los procesos de creación y modificación de la

información de productos para hacer uso de la librería y dar inicio a la integración. Estos procesos fueron categorizados por sistema impactado y por la cantidad de datos que involucraba (individual y masivos). A continuación, se listan algunos ejemplos de procesos de creación y modificación de forma individual y masiva para diferentes sistemas.

La Figura 40 muestra una pantalla de creación de producto y empaque de forma individual en el sistema de administración de productos.

Figura 40.

Editar producto individual de V2S.

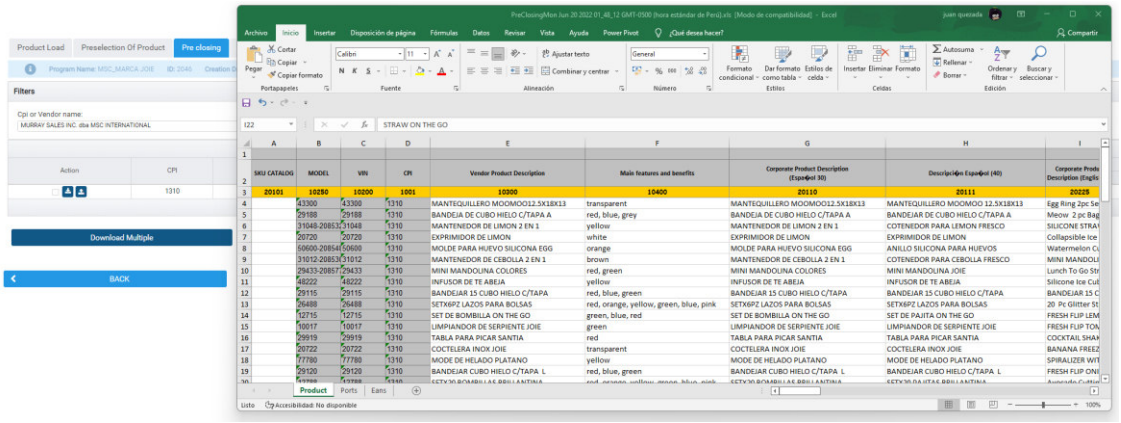


Nota. Adaptado del proyecto.

La Figura 41 muestra un archivo Excel para la creación o modificación masiva de productos en el sistema de campañas.

Figura 41.

Editar producto masivo de V2S.



Nota. Adaptado del proyecto.

A modo de enriquecer la integración desde V2S, se hicieron modificaciones enfocadas a agregar y estandarizar información adicional a la que ya se administraba con relación al producto, como el tiempo de caducidad, sensor de seguridad, unidades logísticas del empaque y producto armado, unidad de medida, factores de conversión, tipos de código de barra, estado de empaque y proveedor activo, agrupación de productos, clasificación comercial, pallets, entre otras.

Finalmente, era necesario que la integración tenga un visor del estado de la solicitud en la plataforma de V2S en donde pueda ser consultado por los usuarios de perfil UAM CORP, para ello se hizo uso de la nueva tabla de notificación catálogo. La solicitud maneja 4 estados que se muestran a continuación.

Tabla 14.

Estados de la solicitud de notificación.

Estado	Descripción	Registrado por
No iniciada	Ocurrió un error al momento de iniciar la integración	Librería notify pim
En proceso	El proceso de integración está en curso	Librería notify pim
Procesada	Se integraron los cambios correctamente en el catálogo de productos	Legacy entity api

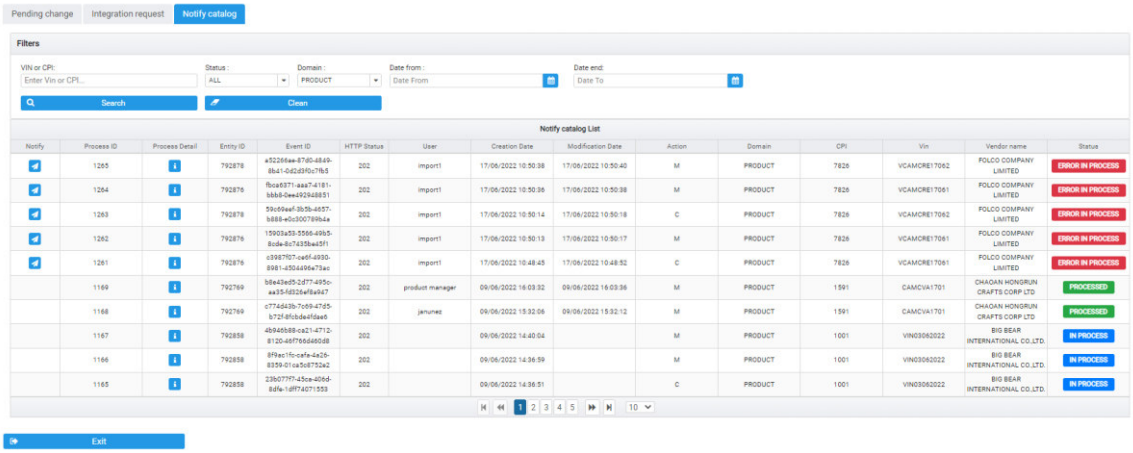
Error en proceso	Ocurrió un error al momento de procesar la solicitud en el catálogo de productos	Legacy entity api

Nota. Elaboración propia.

El visor de solicitud contempla en su mayoría los campos de la tabla de notificación de catalogo con inclusión del campo VIN, que representa un código único de empaquetado de productos y el campo CPI que representa un identificador único del proveedor. A demás de ello una opción para reintentar notificar al catálogo nuevamente en caso de ocurrencia de error. Lo descrito anteriormente, se muestra en la Figura 42.

Figura 42.

Visor de solicitudes de integración V2S - RPC.



Nota. Adaptado del proyecto.

3.2.5.6 Certificación

En esta etapa se ejecutaron las pruebas de la integración y la obtención de resultados de las mismas. Además, se describe el punto de partida para estas pruebas, que inicia una vez culminadas las pruebas funcionales ejecutadas en cada sprint.

Pruebas funcionales

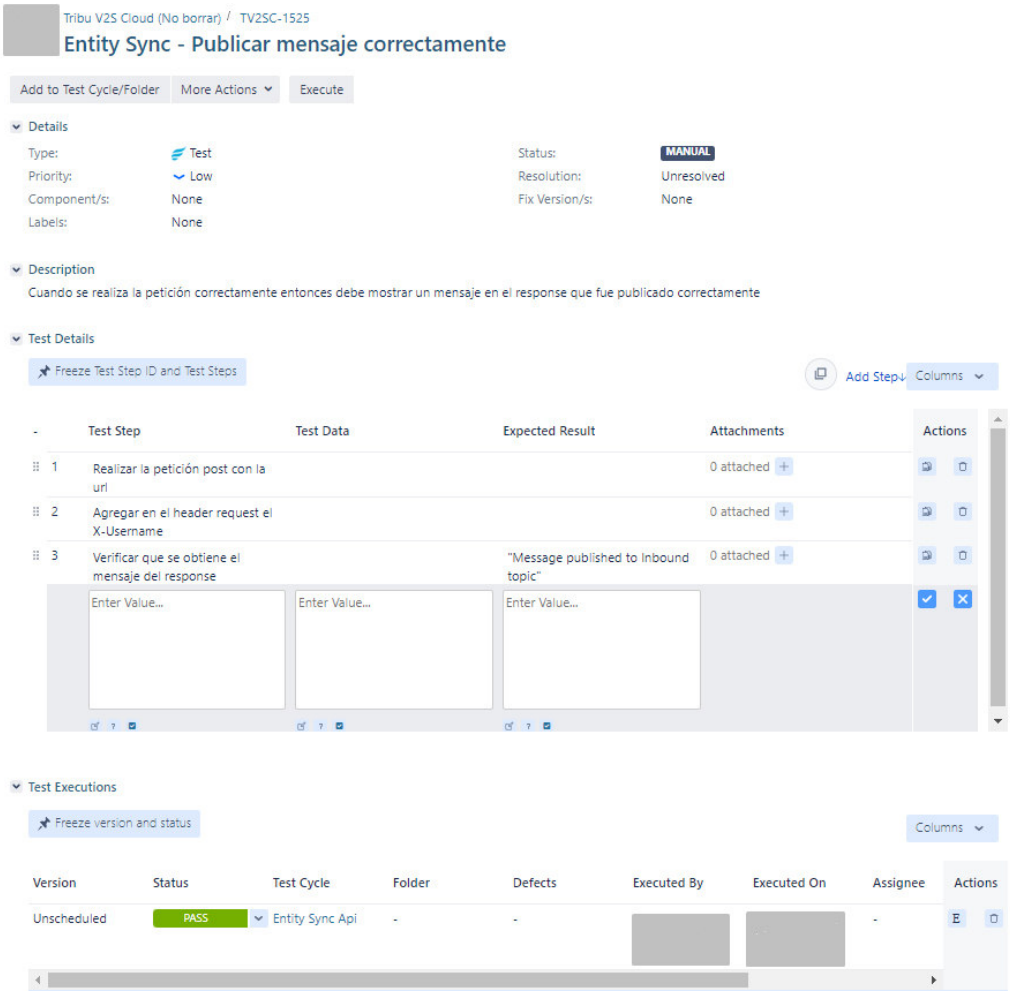
En los sprint 1 y 2 se definió la forma de trabajar en las pruebas funcionales, las cuales se administrarían a través de zephyr en conjunto con jira. Por regla se definió que cada historia de usuario tenga por lo menos una tarea en zephyr (otro tipo de ticket

registrado en jira como tarea, subtarea o bug no tenían la obligación de tener una tarea en zephyr). Durante los sprint 3 y 13 conforme se implementaban las historias se ejecutaban los casos de pruebas. La

Figura 43 muestra la ejecución exitosa de una tarea en zephyr.

Figura 43.

Tarea de zephir para pruebas funcionales.



Nota. Adaptado del proyecto.

El entregable de las ejecuciones de pruebas funcionales que daba el visto bueno para continuar con las pruebas mencionadas a continuación puede verse en el anexo 7.

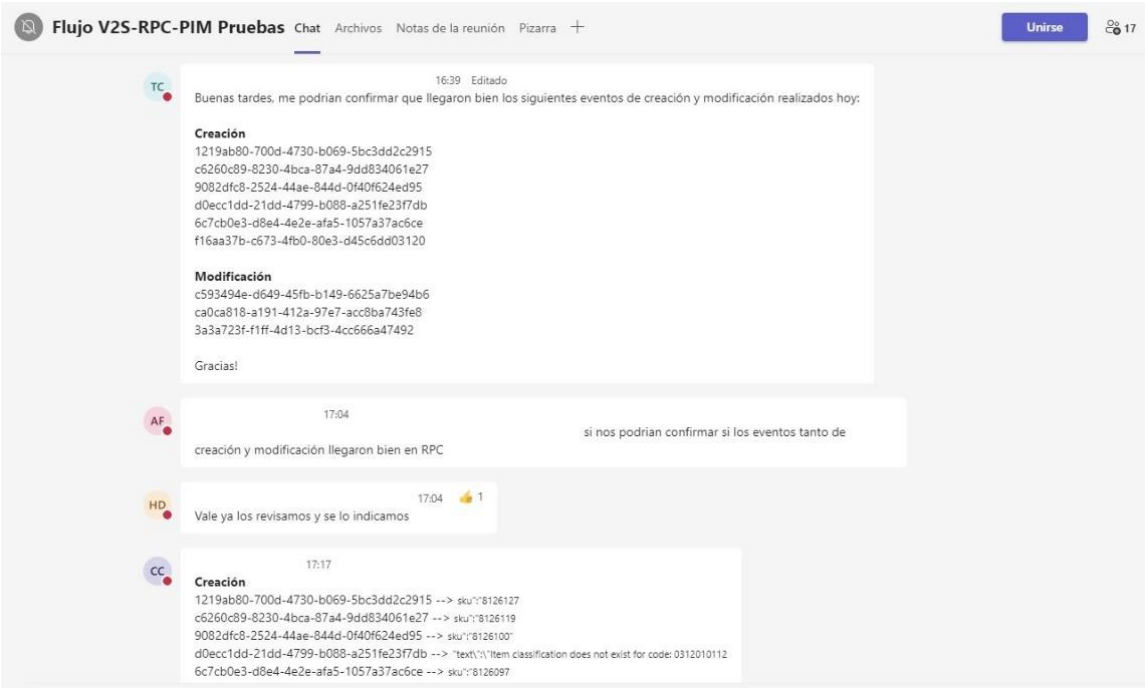
Pruebas de integración

Las pruebas de integración expuestas en este punto buscaban certificar el funcionamiento de la integración en base a escenarios propuestos por el equipo de primera línea (encargado de la plataforma V2S) en conjunto con el equipo del catálogo del producto, con la posibilidad a sumarse a equipos de PIM y ODBMS interesados en probar flujos más amplios. Para poder validar los flujos era necesario que el equipo de primera línea indicará los siguientes datos pertenecientes a los contratos de evento y api.

- EventId: Identificador del evento
- Sku: identificador del producto. (opcional)

En ocasiones especiales, podía agendarse una reunión, pero por lo general los equipos se vinculaban a un canal de comunicación común en Microsoft Teams como se muestra en la Figura 44.

Figura 44.
Coordinación de pruebas de integración.



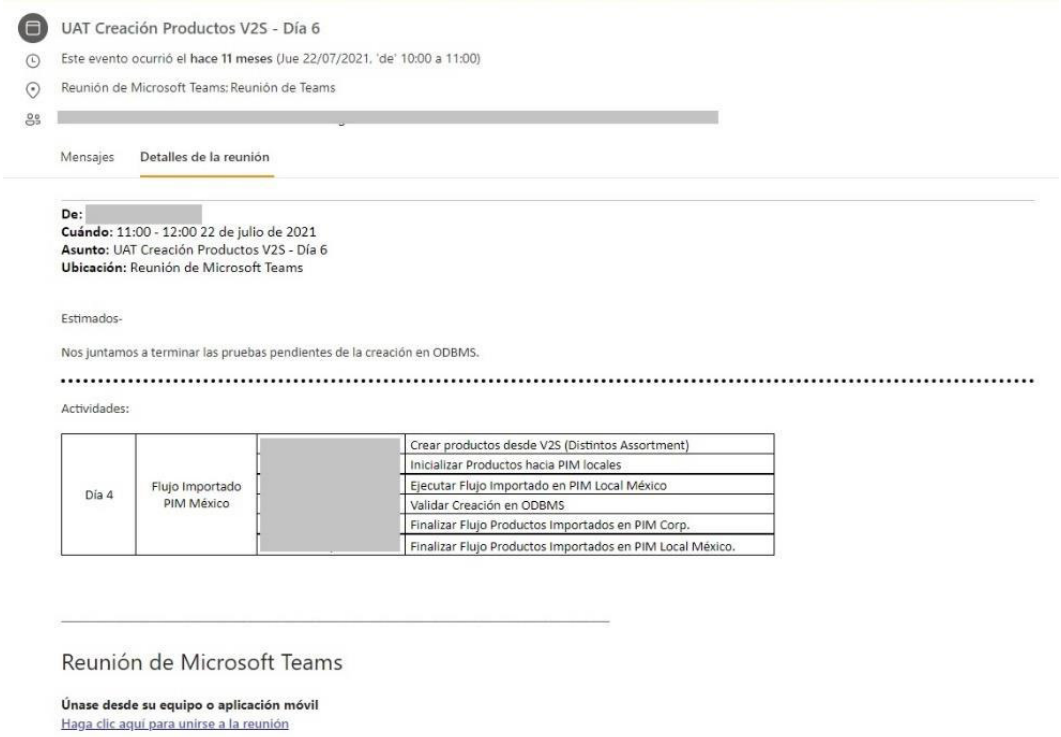
Nota. Adaptado del proyecto.

Estas pruebas involucraban por lo general al equipo técnico sin involucrar necesariamente a los usuarios y los resultados eran administrados por cada equipo de forma independiente. La aprobación de los flujos de estas pruebas representaba el inicio de las pruebas UAT.

Pruebas UAT

Las pruebas UAT eran agendadas en conjunto con los usuarios interesados, en este caso las pruebas UAT involucraban no solo a los usuarios del área comercial corporativa, sino a usuarios como los del área de procesos comerciales que formaban parte del flujo de integración global. Recordar que la plataforma de importaciones se integra con el catálogo de productos impulsados por la necesidad de poder integrarse con PIM, en la Figura 45 se muestra cómo se agendaban las reuniones y los procesos de negocio a abarcar en la reunión.

Figura 45.
Coordinación de pruebas UAT.

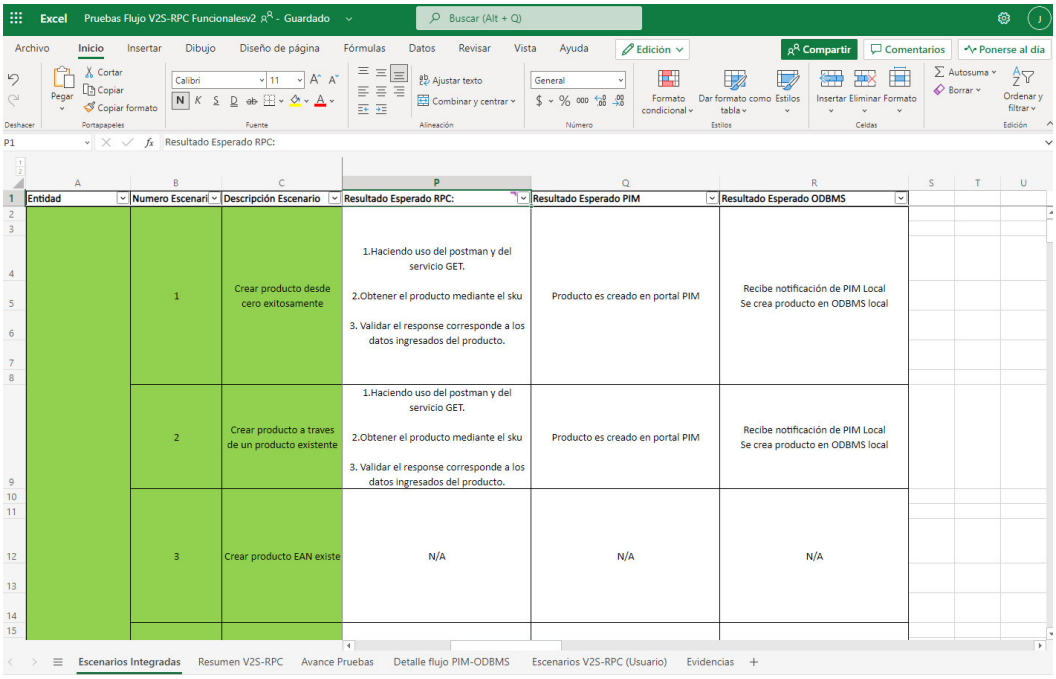


Nota. Adaptado del proyecto.

Los resultados de los casos de pruebas definidos por los usuarios eran gestionados mediante una hoja de cálculo como se evidencia en Figura 46, en donde se observan los resultados satisfactorios.

Figura 46.

Resultados de las pruebas UAT.



Entidad	Numero Escenari	Descripción Escenario	Resultado Esperado RPC:	Resultado Esperado PIM	Resultado Esperado ODBMS
	1	Crear producto desde cero exitosamente	1.Haciendo uso del postman y del servicio GET. 2.Obtener el producto mediante el sku 3. Validar el response corresponde a los datos ingresados del producto.	Producto es creado en portal PIM	Recibe notificación de PIM Local Se crea producto en ODBMS local
	2	Crear producto a traves de un producto existente	1.Haciendo uso del postman y del servicio GET. 2.Obtener el producto mediante el sku 3. Validar el response corresponde a los datos ingresados del producto.	Producto es creado en portal PIM	Recibe notificación de PIM Local Se crea producto en ODBMS local
	3	Crear producto EAN existe	N/A	N/A	N/A

Nota. Adaptado del proyecto.

3.2.5.7 Mantenimiento

En cuanto a las actividades de mantenimiento, la más resaltante es el cambio de versión de los contratos el cual pasaron a nivel de api a la versión 2.0 y a nivel de evento a la 3.0. Este punto como se indicó queda resumido pues esta fuera del alcance del proyecto.

3.3 Evaluación

3.3.1 Evaluación económica

Los costos del proyecto al tratarse de información sensible para la organización serán estimados. No se incluyen costos fijos relacionados a equipos informáticos, energía, internet, oficina, entre otros debido a que se dieron en un contexto de trabajo remoto, por

lo cual eran cubiertos por cada miembro que participo en el proyecto y para tal efecto se asume están incluidos en los pagos hechos a cada miembro del equipo.

Para determinar el costo aproximado se tendrán en cuenta principalmente los costos asociados al talento humano y a la infraestructura en términos de dólares (debido a que el equipo era de múltiples países y no podría expresarse en una moneda local) y tiempo en unidad de meses. La Tabla 15 y la Tabla 16 muestran los costos mencionados anteriormente.

Tabla 15.

Costos administrativos.

Rol	**Duración (mes)**	**Costo (USD)**	**Total (USD)**
Product Owner	8	3,500.00	28,000.00
Scrum Master	8	2,500.00	20,000.00
Business Analyst	5	3,000.00	15,000.00
QA Engineer 1	8	2,000.00	16,000.00
QA Engineer 2	8	2,000.00	16,000.00
Technical Lead	8	3,500.00	28,000.00
Software Engineer 1	8	3,000.00	24,000.00
Software Engineer 2	8	2,500.00	20,000.00
Software Engineer 3	3	2,500.00	7,500.00
Total			174,500.00 USD

Nota. Elaboración propia.

Tabla 16.

Costos tecnológicos.

Infraestructura	**Duración (mes)**	**Costo (USD)**	**Total (USD)**
Kubernetes Engine	6	18.00	108.00
Pub/Sub	6	5.00	30.00
Secret Manager	6	6.00	36.00
Logging Service	6	15.00	90.00
Rancher	6	50.00	300.00
Total			564.00 USD

Nota. Elaboración propia.

En total los gastos administrativos y tecnológicos suman la cantidad de 175,064.00 USD para los 8 meses que duró aproximadamente el proyecto.

3.3.2 Beneficios obtenidos

La integración con el catálogo de productos, ayuda al área comercial de importaciones a administrar la información de los productos en base a los estándares corporativos de la organización. Otro punto importante, es que permite que la información de los productos pueda permanecer también en un repositorio centralizado, en donde otras capacidades puedan consultar la información que necesiten, evitando que la plataforma de importaciones se convierta en un silo de información.

La arquitectura de la integración definida, ha servido como base para diseñar otras arquitecturas de integración relacionadas a otras entidades de negocio importantes como lo son las órdenes de compra.

El principal beneficio obtenido para este proyecto a partir de la integración con el catálogo fue habilitar este como intermediario de integraciones, desacoplando la necesidad de generar futuras integraciones entre sistemas ad hoc y conciliaciones manuales de la información de productos entre diferentes áreas.

Con la integración, las conciliaciones manuales necesarias entre el área comercial corporativa y el área comercial de operaciones se redujeron en un 100%. En cuanto a las integraciones entre sistemas ad hoc, las últimas integraciones que intercambian información de productos que se han tenido fueron con sistemas de información de equipos de Colombia y Brasil durando este último alrededor de 2 años, mientras que con el catálogo se espera desacoplar estas integraciones reduciendo el tiempo.

Finalmente, la Tabla 17 muestra un resumen de los beneficios obtenidos con la integración con el catálogo de productos.

Tabla 17.

Resumen comparativo de beneficios de la integración.

Beneficio	**Antes**	**Después**
Validación de la definición corporativa de producto.	No existía.	A través de las reglas de creación y actualización de productos.
Escalabilidad de la información de productos con otras áreas de negocio.	No existía.	A través de la base de datos centralizada de RPC.
Rango de conciliaciones de información a través de llamadas o correos	Alto	Nulo.
Tiempo esperado de futuras integraciones de información de productos.	Entre 18 - 24 meses	8 meses a menos. (usando RPC como intermediario)

Nota. Elaboración propia.

CAPÍTULO IV - REFLEXIÓN CRÍTICA DE LA EXPERIENCIA

En la integración, el autor del presente trabajo realizo las siguientes reflexiones sobre el proyecto.

- El proyecto represento un reto, desde el lado tecnológico debido a que muchos integrantes del equipo, incluyendo al autor, trabajaban con un ecosistema de microservicios y nube por primera vez y desde el lado de negocio porque se iteraron múltiples veces los contratos de la integración con el catálogo de productos hasta llegar a una versión final.
- La necesidad de contar con prácticas de integración y despliegue continuo fue fundamental, durante el proyecto fue inevitable hacer comparaciones entre el tiempo de pase a un ambiente de pruebas entre los componentes de la integración y los proyectos de la plataforma legada V2S, pues estas últimas dependían enteramente del factor humano y de la disponibilidad de tiempo de los ejecutores del pase.
- El uso de scrum como marco de trabajo era conocido por el autor de forma amplia, al igual que el equipo resaltando que la comunicación es un pilar. Aun así, en un inicio se evidenciaron problemas de comunicación dado que parte del equipo se estaba conociendo y la rotación de los miembros del equipo obligaban a tener nuevos miembros con el periodo de adaptación que esto involucra, lo cual con el pasar del tiempo fue mejorando.
- El equipo estaba encargado de atender las necesidades cotidianas del negocio, por lo que en muchas ocasiones tenía varios frentes los cuales debía atender, esto generaba que durante el proyecto miembros del equipo tuviesen que atender necesidades fuera de las actividades de la integración lo que dificultaba un horizonte temporal previsible del proyecto.
- En algún momento se barajaron alternativas como el uso de ETL para llevar la información de la plataforma legada V2S al catálogo de productos (RPC), pero esto podría generar desfaces de horas, días o semanas, y no se quería un periodo de tiempo tan largo entre la sincronización de ambas plataformas.

CAPÍTULO V - CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

- Se logró diseñar e implementar un esquema de integración de aplicaciones basado en microservicios robusto y fácil de mantener permitiendo establecer un canal de comunicación entre la plataforma de importaciones y el catálogo de productos corporativo a fin de aprovechar los beneficios que ofrece este último permitiendo mantener una definición corporativa de producto, la escalabilidad de esta información con otros procesos macro y futuras integraciones. Esto queda evidenciado en el punto 3.2.5.2 en donde se diseña la arquitectura de integración y en el punto 3.2.5.4. en donde se implementa.
- El aprovisionamiento de la infraestructura para dar soporte a la integración se logró mediante una combinación de herramientas de servicios de mensajería (Pub/Sub), administración de contenedores (Rancher y GKE), entre otras basadas en entornos de nube y on premise, en conjunto con un esquema de integración y despliegue continúo a través de Gitlab CI/CD permitiendo que los componentes de la integración sean escalables, desacoplados y de rápida puesta en marcha tal como se observa en el punto 3.2.5.3. Esto se logró en conjunto con el equipo del proyecto y equipos transversales de infraestructura y seguridad.
- Los flujos de creación y modificación de productos en los sistemas de campañas, productos y proveedores de la plataforma de importaciones (sistema legado) se modificaron para agregar nuevos datos enriqueciendo la información de productos y poder iniciar el proceso de integración, además de crear nuevas funcionalidades relacionadas, entre ellas la más importante como el visor de solicitudes de integración. Esto se puede ver evidenciado en el punto 3.2.5.5. Además de lo mencionado, fue necesario realizar capacitaciones a los usuarios en conjunto con una marcha blanca para apoyarlos en las dudas que tuviesen con la operación de los sistemas.
- Se realizaron las pruebas de la integración entre la plataforma de importaciones y el catálogo de productos, lo que brindo la conformidad del área técnica y del área usuaria de la integración permitiendo certificar la implementación tal como se observa en el punto 3.2.5.6.

Las conclusiones demuestran el cumplimiento de los objetivos específicos, lo que permitió cumplir con el objetivo general de integrar la información de productos importados en una empresa retail usando microservicios de forma exitosa con los beneficios mencionados en el punto 3.3.2.

5.2 Recomendaciones

- Para la integración, la información de los empaques de los productos se tomó como una entidad independiente a ser notificada (por ejemplo, si se crea un nuevo empaque a un producto, se notifica al catálogo solo la información del empaque, además se notifica la información del producto) pero esto representa un error a nivel conceptual dado que un empaque existe solo si existe un producto, por lo que no debería actuar de forma independiente, sobre todo teniendo en cuenta que la entidad de producto ya informa lo relacionado con respecto a su empaque por lo que debería eliminarse dicho comportamiento.
- Los procesos de creación o modificación de un producto en la plataforma de importaciones no están centralizados, lo que incurre en un difícil mantenimiento de estos procesos como se evidencio al tener que gatillar las solicitudes de integración hacia el catálogo en cada uno de estos procesos. Se recomienda centralizar esta lógica ya que es única.
- El estado de la solicitud actualmente cambia al inicio y al final del proceso de integración, lo cual no permite una trazabilidad más detallada de la solicitud, para ello sería bueno generar un mecanismo para actualizar el estado en etapas intermedias a fin de saber en qué parte del flujo esta la solicitud de integración.
- Actualmente existen otras integraciones de información de productos entre sistemas ad hoc que involucran la plataforma de importaciones, sería recomendable a futuro evaluar que se hagan por medio del catálogo de productos.

5.3 Fuentes de Información

Ambler, S. (2019). *Choose Your WoW ! A Discipline Agile Delivery Handbook for Optimizing Your Way of Working WOW*. Disciplined Agile, Inc.

ATLASSIAN. (2022). Continuous integration vs. delivery vs. deployment. Retrieved

May 5, 2022, from https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment

Bustillos, H. (2020). Integración continua e implementación continua en GitLab. Retrieved June 23, 2022, from https://www.encora.com/es/blog/implementing-continuous-integration-and-continuous-deployment-on-gitlab website: https://www.encora.com/es/blog/implementing-continuous-integration-and-continuous-deployment-on-gitlab

Chapbell, E. (2022). What Are Agile Scrum Artifacts? | ClickUp Blog. Retrieved June 22, 2022, from ClickUp Blog website: https://clickup.com/blog/scrum-artifacts/

Defossé, N., López, R. A., Gómez, M. E., Konstantinoff, P., Wahler, S., Castro, L., & Harris, G. (2017). Implementación de Middleware Publicador / Subscriptor para Aplicaciones Web de Monitoreo. *In XIX Workshop de Investigadores En Ciencias de La Computación*, 181–185.

Djogic, E., Ribic, S., & Donko, D. (2018). Monolithic to microservices redesign of event driven integration platform. *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2018 - Proceedings*, 1411–1414. https://doi.org/10.23919/MIPRO.2018.8400254

Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). Microservices: Yesterday, today, and tomorrow. *Present and Ulterior Software Engineering*, 195–216. https://doi.org/10.1007/978-3-319-67425-4_12/COVER/

Fowler, M. (2006). Continuous Integration. Retrieved May 5, 2022, from https://martinfowler.com/articles/continuousIntegration.html

GARTNER. (2021). Magic Quadrant for Cloud Infrastructure and Platform Services. Retrieved May 17, 2022, from Gartner website: https://www.gartner.com/doc/reprints?id=1-271SYZF2&ct=210802&st=sb

Geewax, J. (2018). *Google Cloud Platform in Action*.

Gholami, M. F., Daneshgar, F., Beydoun, G., & Rabhi, F. (2017). Challenges in migrating legacy software systems to the cloud — an empirical study. *Information Systems*, *67*, 100–113. https://doi.org/10.1016/j.is.2017.03.008

IBM. (2021). What is a REST API? | IBM. Retrieved June 23, 2022, from https://www.ibm.com/cloud/learn/rest-apis

Lukša, M. (2020). *Kubernetes in Action*.

MICROSOFT. (2022a). Describe CI/CD - Learn | Microsoft Docs. Retrieved May 6, 2022, from https://docs.microsoft.com/en-us/learn/modules/implement-ci-cd-azure-devops/2-describe-ci-cd

MICROSOFT. (2022b). Prácticas recomendadas de diseño de API web - Centro de arquitectura de Azure | Documentos de Microsoft. Retrieved June 10, 2022, from https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design

MICROSOFT. (2022c). What is Docker? Retrieved June 22, 2022, from https://docs.microsoft.com/en-us/dotnet/architecture/microservices/container-docker-introduction/docker-defined

PerúRetail. (2018). ¿Qué es retail? definiciones del comercio minorista | Perú Retail. Retrieved June 30, 2022, from https://www.peru-retail.com/que-es-retail/

RED HAT. (2018). ¿Qué es el Cloud Computing? Retrieved June 22, 2022, from https://www.redhat.com/es/topics/cloud

RED HAT. (2020a). ¿Qué es Kubernetes? Retrieved June 22, 2022, from https://www.redhat.com/es/topics/containers/what-is-kubernetes

RED HAT. (2020b). ¿Qué es una API de REST? Retrieved June 10, 2022, from https://www.redhat.com/es/topics/api/what-is-a-rest-api

Richards, C. (2018). *Microservices patterns: with examples in Java* (Vol. 2018).

Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide The Definitive Guide to Scrum: The Rules of the Game*.

SCRUM STUDY. (2017). *A Guide to the SCRUM BODY OF KNOWLEDGE (SBOK™GUIDE)*.

Srinivas, M., Ramakrishna, G., Rajasekhara Rao, K., & Suresh Babu, E. (2016). Analysis of legacy system in software application development: A comparative survey. *International Journal of Electrical and Computer Engineering*, *6*(1), 292–297. https://doi.org/10.11591/ijece.v6i1.8367

Strobl, S., Bernhart, M., & Grechenig, T. (2020). Towards a Topology for Legacy System Migration. *Proceedings - 2020 IEEE/ACM 42nd International Conference on Software Engineering Workshops, ICSEW 2020*, 586–594. https://doi.org/10.1145/3387940.3391476

Sutherland, J., Ph, D., & Scrum, C. (2007). *The Scrum Papers : Nuts , Bolts , and Origins of an Agile Process*.

Thönes, J. (2015). Microservices. *IEEE Software*, *32*(1).

https://doi.org/10.1109/MS.2015.11

5.4 Glosario de términos

Angular. Framework de javascript para desarrolla aplicaciones SPA.

Backend. Parte no visible encargada de ejecutar las diversas operaciones en un servidor en algún sistema de información.

CPI. Código de proveedor interno, es un identificador único.

DevOps. Cultura de trabajo que busca acortar la brecha entre el desarrollo y la operación mediante un conjunto de prácticas.

DevSecOps. Integración de la seguridad en la cultura DevOps.

Framework. Conjunto de conceptos, practicas, herramientas estándar utilizados para agilizar determinada actividad.

Frontend. Parte visible que permite interacción con el usuario en algún sistema de información.

GCP. Siglas de Google Cloud Platform la cual es la infraestructura de nube de Google.

HTTP. Siglas de Hypertext Transfer Protocol, permite el acceso a recursos de internet.

Kanban. Marco de trabajo ágil en donde se gestiona el trabajo conforme llega.

MDM. Siglas de Master Data Management, sistema de información responsable de la gestión de información maestra de productos.

Nube. También denominado cloud, referencia el uso de recursos informáticos remotos provistos a través de internet.

ODBMS. Siglas de Operational Data Base Merchandising System, el cual es un ERP encargado de procesos de compra y venta de productos, generación de órdenes de compra y control de inventario.

On premise. Hace referencia a infraestructura tecnológica administradas físicamente en instalaciones de la organización que la usa.

PIM. Siglas de Product Information Management, sistema de información responsable de la gestión de información maestra de productos orientada hacia la venta.

Primefaces. Librería de componentes web para java.

Retail. Término para definir un sector económico relacionado al comercio o venta al por menor de forma masiva.

RPC. Siglas de Retail Product Catalog, representa la plataforma del catálogo de productos corporativo de la organización.

SKU. Siglas de Stock Keeping Unit, identificador único del producto.

SOA. Siglas de Service Oriented Architecture, estilo de arquitectura que busca definir servicios reutilizables comunicados mediante la exposición de contratos de servicio a través de protocolos estándar.

SOAP. Siglas de Simple Object Access Protocol, el cual define como dos servicios pueden comunicarse mediante el intercambio de datos en formato XML.

SPA. Siglas de Single Page Applications, aplicación de una sola página en donde los componentes se renderizan dentro de ella.

Squad. Equipo multidisciplinario con autonomía para organizarse.

SRE. Siglas de System Reliability Engineer, representa un rol dentro del mundo de TI, en algunos casos mal llamados DevOps.

Stakeholder. Representación de un grupo de individuos que tienen algún interés o impacto en un proyecto.

UAT. Siglas de User Acceptance Testing, hace referencia a las pruebas de usuario final realizadas por el usuario que usara el producto software.

V2S. Siglas de Vendor to Store, que es como se llama la plataforma de importaciones legada del negocio.

VEV. Siglas de Venta en Verde, el cual es un sistema de generación de órdenes de compra automáticas.

VIN. Código único del empaquetado del producto de un proveedor.

Websockets. Tecnología que proporciona un canal de comunicación bidireccional entre el cliente y servidor.

Zephyr. Tecnología que proporciona un canal de comunicación bidireccional entre el cliente y servidor.

ANEXOS

Anexo 1. Contrato del evento

Atributo	Tipo	Descripción
eventId	UUID	Identificador de evento.
eventType	String	Tipo de evento, usando verbos en pasado.
entityId	String	Identificador de la entidad a la que se hace referencia.
entityType	String	Tipo de entidad.
timestamp	Unix epoch	Unix epoch que indica la ocurrencia del evento en el negocio.
datetime	ISO8601 Datetime	Fecha legible que indica la ocurrencia del evento en el negocio.
versión	String	La versión de la estructura del evento.
country	ISO3166-1 alpha-2 String	Ubicación física del negocio que origino el evento.
commerce	String	Nombre del comercio como se le conoce.
channel	String	Canal de distribución que origino el evento.
domain	String	Dominio de negocio.
capability	String	Dominio de la capacidad.
mimeType	String	Tipo o subtipo de Media.
sourceEntityId	String	Identificador único de la entidad en el sistema origen.
source	String	Sistema responsable de generar el mensaje.
tenantId	String	Identificador que corresponde a la combinación comercio país.
data	String	Payload del evento (o vacío en su defecto). Representación en formato String de la carga útil de acuerdo al mimeType.

Anexo 2. Contrato de producto.

Product ∨ {

sku
string
example: 6315974

Stock-keeping unit

barcode* > [...]

style* > [...]

primaryVendor* > [...]

productType* > [...]

productStatus* > [...]

createSourceSystem* > [...]

startingPrice* > [...]

packaging* > [...]

productGrouping > [...]

discontinuedProduct
boolean
example: false

Discountinued Product. Default false

legacySkus > [...]

longDescription*
string
example: HERVID.E RECCO RHE-6168 (D)

Product long description

businessType
string
example: Retail

Business type.

- Crossborder
- Retail
- Franchise

Enum:

> Array [3]

taxType > [...]

countryOfManufacture
string
example: CL

Country Of Manufacture. ISO3166-1 alpha-2 formatted country name

fabricationMaterial
string
example: 100% steel

Product materiality

mainFeature
string
example: STEEL FRAME WITH HAMMER SILVER FINISH AND RED FABRIC

Main Feature

securitySensor
boolean
example: true

Security sensor

perishableProduct > {...}

requiredArt
boolean
example: true

Required Art. Default false

season
string
example: S/T

Season

program
string
example: XMASS

Xmass, Back to School

size
string
example: M

Product size

color > {...}

labelType
string
example: Ticket

Label

buyerCode
string
example: 499523

Buyer code (Este es corp o local?)

markForDeletion
boolean
example: false

Determines whether the record is marked for deletion. Default false

sourcingType
string
example: National

Sourcing Type

- National
- Imported

Enum:

> Array [2]

dangerousGoods > [...]

qtyUOM*
string
example: UNIT
items: OrderedMap { "type": "string", "enum": List ["UNIT", "DM3", "M2", "KG", "M"] }

Quantity Unit Of Measure of hoe the product is sale.

- UNIT
- DM3
- M2
- KG
- M

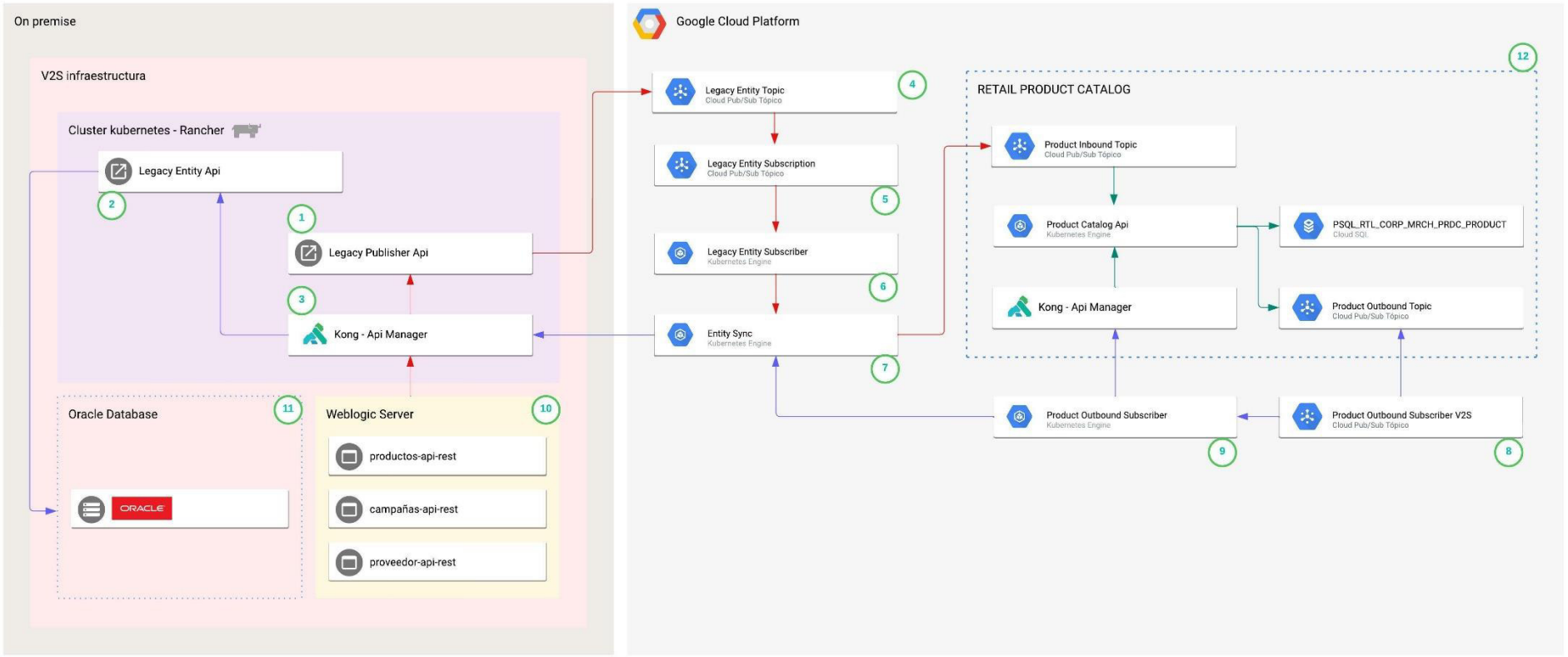
technicalSpecifications > {...}

storageConditions > {...}

updateSourceSystem > [...]

additionalAttributes > [...]

Anexo 3. Entregable de diseño de arquitectura de alto nivel.



Anexo 4. Entregable de responsabilidades de componentes de arquitectura

Elemento	Tipo de recurso	Descripción
Legacy publisher api (1)	Microservicio Publicador Java	Microservicio encargado de publicar un mensaje en el tópico legado. La estructura del evento es la misma que la definida en RPC.
Legacy entity api (2)	Microservicio Api Java	Microservicio encargado la información necesaria desde el legado para la integración
Rancher kong (3)	Administrador de apis	Encargado de exponer el legacy entity api para que pueda ser consumido por otros microservicios.
Legacy entity topic (4)	Pub/Sub tópico	Tópico encargado de recibir los mensajes de los eventos que nacen desde el legado.
Legacy entity subscription (5)	Pub/Sub suscripción	Suscripción que recibe los mensajes del legacy entity topic.
Legacy subscriber api (6)	Microservicio Suscriptor Java	Microservicio encargado de recibir los mensajes de la suscripción del legado.
Entity sync api (7)	Microservicio Publicador Java	Microservicio encargado de sincronizar los mensajes de los eventos entre el legado y GCP.
Outbound v2s subscription (8)	Pub/Sub suscripción	Suscripción encargada de recibir los mensajes del outbound topic de RPC.
Outbound v2s subscriber (9)	Microservicio Suscriptor Java	Microservicio encargado de recibir los mensajes de la suscripción outbound v2s
V2S Platform (10)	Plataforma legada	Plataforma legada, compuesta por tres sistemas de interés (productos, campañas y proveedores)
Legacy Data base (11)	Base de datos Oracle	Base de datos de la plataforma legada V2S
RPC Platform (12)	Plataforma de Catálogo de Productos	Representación simplificada de lo que es RPC

Anexo 5. Tareas de ejecución de pipelines

Grupo	Tarea	Objetivo
Calidad de código	Linter	Análisis de código en busca de errores sintácticos o de estilos definidos
	Unit Test	Automatización pruebas unitarias, entrega un reporte de cobertura de código a través de la herramienta Jacoco que es usado por Sonarqube.
	Sonarqube	Análisis de código estático, evalúa duplicidad de código, cobertura de código probado y código potencialmente mal escrito.
Seguridad	Detect Secrets	Detección de credenciales escritas en el código fuente de forma explícita. Resultado se muestra en formato json.
	Dependency Check	Detección de vulnerabilidades de las dependencias usadas. Resultado se muestra a través de un reporte.
	Clair	Escaneo de vulnerabilidades en la configuración de los contenedores Docker.
	Veracode	Encargado única y exclusivamente de la detección de vulnerabilidades de seguridad en el código estático.
Construcción y despliegue	Build Java	Compilación y empaquetado a través de Maven sin ejecutar los test.
	Registry	Generación de una imagen Docker del servicio y almacenamiento en un repositorio de imágenes.
	Deploy	Descarga la imagen de Docker generada y la despliega en los ambientes on premise o nube según sea el caso.

Anexo 6. Entregable de pruebas funcionales

N°	Titulo	Ticket
1	Packgaging-Crear Packgaging correctamente	TV2SC-1564 - Packgaging-Crear Packgaging correctamente **MANUAL**
2	Packgaging - Validar tiempo de respuesta	TV2SC-1566 - Packgaging - Validar tiempo de respuesta **MANUAL**
3	Packgaging- Error Crear Packgaging sin sku	TV2SC-1567 - Packgaging- Error Crear Packgaging sin sku **MANUAL**
4	Packgaging- Error Crear Packgaging sin vin	TV2SC-1568 - Packgaging- Error Crear Packgaging sin vin **MANUAL**
5	Packgaging-Verificar el api cumple con el esquema	TV2SC-1419 - Verificar la api cumple con el esquema **MANUAL**
6	Sku-Verificar error sin sku	TV2SC-1520 - Sku-Verificar error sin sku **MANUAL**
7	Sku - Validar tiempo de respuesta	TV2SC-1519 - Sku - Validar tiempo de respuesta **MANUAL**
8	Sku - Validar headers del response	TV2SC-1518 - Sku - Validar headers del response **MANUAL**
9	Sku-Verificar error con sku incorrecto	TV2SC-1517 - Sku-Verificar error con sku incorrecto **MANUAL**
10	Sku - Verificar el api cumple con el esquema	TV2SC-1516 - Sku - Verificar la api cumple con el esquema **MANUAL**
11	Sku-Obtener Producto correctamente	TV2SC-1515 - Sku-Obtener Producto correctamente **MANUAL**
12	Publisher-Crear Vendor correctamente	TV2SC-1443 - Publisher-Crear Vendor correctamente **MANUAL**
13	Publisher-Crear Vendor sin autorización	TV2SC-1444 - Publisher-Crear Vendor sin autorización **MANUAL**
14	Publisher-Crear Vendor sin header X-Username	TV2SC-1445 - Publisher-Crear Vendor sin header X-Username **MANUAL**
15	Publisher- Error Crear Vendor	TV2SC-1446 - Publisher- Error Crear Vendor **MANUAL**
16	Publisher - Validar tiempo de respuesta	TV2SC-1447 - Publisher - Validar tiempo de respuesta **MANUAL**
17	Publisher - Validar headers del response	TV2SC-1448 - Publisher - Validar headers del response **MANUAL**
18	Entity - Verificar el api cumple con el esquema	TV2SC-1484 - Entity - Verificar la api cumple con el esquema **MANUAL**
19	Entity-Obtener Producto correctamente	TV2SC-1481 - Entity-Obtener Producto correctamente **MANUAL**
20	Entity - Validar tiempo de respuesta	TV2SC-1483 - Entity - Validar tiempo de respuesta **MANUAL**
21	Entity- Error Obtener Producto	TV2SC-1482 - Entity- Error Obtener Producto **MANUAL**
22	Entity - Validar headers del response	TV2SC-1506 - Entity - Validar headers del response **MANUAL**

23	Entity-Verificar error sin productID	TV2SC-1508 - Entity-Verificar error sin productID **MANUAL**
24	Entity-Verificar error con productID incorrecto	TV2SC-1507 - Entity-Verificar error con productID incorrecto **MANUAL**
25	Entity-Obtener Producto sin autorización	TV2SC-1509 - Entity-Obtener Producto sin autorización **MANUAL**
26	Entity Sync - Validar que los type del response son los correctos	TV2SC-1420 - Validar que los type del response son los correctos **MANUAL**
27	Entity Sync - Verificar el api cumple con el esquema	TV2SC-1419 - Verificar el api cumple con el esquema **MANUAL**
28	Entity Sync - Validar tiempos de respuesta	TV2SC-1416 - Validar tiempos de respuesta **MANUAL**
29	Entity Sync - Verificar los mensajes de estado de respuesta	TV2SC-1413 - Verificar los mensajes de estado de respuesta **MANUAL**
30	Entity Sync -Verificar código de respuesta de solicitud	TV2SC-1412 - Verificar código de respuesta de solicitud **MANUAL**
31	Entity Sync - Publicar mensaje correctamente	TV2SC-1525 - Entity Sync - Publicar mensaje correctamente **MANUAL**
32	Entity Sync - Error al publicar mensaje	TV2SC-1526 - Entity Sync - Error al publicar mensaje **MANUAL**
33	Entity Sync - Realizar petición PATCH	TV2SC-1527 - Entity Sync - Realizar petición PATCH **MANUAL**
34	Publisher-Crear Vendor correctamente	TV2SC-1443 - Publisher-Crear Vendor correctamente **MANUAL**
35	Publisher-Crear Vendor sin autorización	TV2SC-1444 - Publisher-Crear Vendor sin autorización **MANUAL**
36	Publisher-Crear Vendor sin header X-Username	TV2SC-1445 - Publisher-Crear Vendor sin header X-Username **MANUAL**
37	Publisher- Error Crear Vendor	TV2SC-1446 - Publisher- Error Crear Vendor **MANUAL**
38	Publisher - Validar tiempo de respuesta	TV2SC-1447 - Publisher - Validar tiempo de respuesta **MANUAL**
39	Publisher - Validar headers del response	TV2SC-1448 - Publisher - Validar headers del response **MANUAL**