



Universidad Nacional Mayor de San Marcos

Universidad del Perú. Decana de América

Facultad de Ingeniería de Sistemas e Informática

Escuela Profesional de Ingeniería de Software

**Implementación de un sistema de gestión de
información de productos para optimizar la
administración del catálogo de productos en una
entidad E-commerce**

TRABAJO DE SUFICIENCIA PROFESIONAL

Para optar el Título Profesional de Ingeniero de Software

AUTOR

Carlos Iván POCLÍN MEZA

ASESOR

Mg. Joel Fernando MACHADO VICENTE

Lima, Perú

2025



Reconocimiento - No Comercial - Compartir Igual - Sin restricciones adicionales

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

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

Referencia bibliográfica

Poclín, C. (2025). *Implementación de un sistema de gestión de información de productos para optimizar la administración del catálogo de productos en una entidad E-commerce*. [Trabajo de suficiencia profesional de pregrado, Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería de Sistemas e Informática, Escuela Profesional de Ingeniería de Software]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	Carlos Iván Poclín Meza
Tipo de documento de identidad	DNI
Número de documento de identidad	72558335
URL de ORCID	https://orcid.org/0009-0007-8775-9528
Datos de asesor	
Nombres y apellidos	Joel Fernando Machado Vicente
Tipo de documento de identidad	DNI
Número de documento de identidad	40476778
URL de ORCID	https://orcid.org/0000-0002-4056-0438
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	María Elena Ruiz Rivera
Tipo de documento	DNI
Número de documento de identidad	08960021
Miembro del jurado 1	
Nombres y apellidos	Arturo Alejandro Bartra More
Tipo de documento	DNI
Número de documento de identidad	40233946
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	2024
URL de disciplinas OCDE	2.02.04 -- Ingeniería de sistemas y comunicaciones https://purl.org/pe-repo/ocde/ford#2.02.04



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

Acta Virtual de Sustentación
del Trabajo de Suficiencia Profesional

Siendo las 20:00 horas del día 14 de enero del año 2025, se reunieron virtualmente los docentes designados como Miembros de Jurado del Trabajo de Suficiencia Profesional, presidido por la Mg. Ruiz Rivera María Elena (Presidente), Ing. Bartra More Arturo Alejandro (Miembro) y el Mg. Machado Vicente Joel Fernando (Miembro Asesor), usando la plataforma Meet (<https://meet.google.com/jtg-ystx-bsz>), para la sustentación virtual del Trabajo de Suficiencia Profesional intitulado: **“IMPLEMENTACIÓN DE UN SISTEMA DE GESTIÓN DE INFORMACIÓN DE PRODUCTOS PARA OPTIMIZAR LA ADMINISTRACIÓN DEL CATÁLOGO DE PRODUCTOS EN UNA ENTIDAD E-COMMERCE”**, por el Bachiller **Poclín Meza Carlos Iván**; para obtener el Título Profesional de Ingeniero de Software.

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

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

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

A continuación la Presidente de Jurados la Mg. Ruiz Rivera María Elena, declara al Bachiller **Ingeniero de Software**.

Siendo las 20:50 horas, se levantó la sesión.

Presidente

Mg. Ruiz Rivera María Elena

Miembro

Ing. Bartra More Arturo Alejandro

Miembro Asesor

Mg. Machado Vicente Joel Fernando

CERTIFICADO DE SIMILITUD

Yo, MACHADO VICENTE JOEL FERNANDO en mi condición de asesor acreditado con la Resolución Decanal N° 001059-2024-D-FISI/UNMSM y Oficio N° 0018-PTTSP-FISI-2024 del Trabajo de Suficiencia Profesional cuyo título es Implementación de un sistema de gestión de información de productos para optimizar la administración del catálogo de productos en una entidad E-commerce, presentado por el bachiller POCLÍN MEZA CARLOS IVÁN para optar el título de Ingeniero de Software CERTIFICO que se ha cumplido con lo establecido en la Directiva de Originalidad y de Similitud de Trabajos Académicos, de Investigación y Producción Intelectual. Según la revisión, análisis y evaluación mediante la plataforma PIDS, el documento evaluado cuenta con el 9% de similitud, nivel **PERMITIDO** para continuar con los trámites correspondientes y para su **publicación en el repositorio institucional**.

Se emite el presente certificado en cumplimiento de lo establecido en las normas vigentes, como uno de los requisitos para la obtención del título.

Lima, 8 de enero de 2025



Firma del Asesor

Nombres y Apellidos del asesor: MACHADO VICENTE JOEL FERNANDO

DNI 40476778

DEDICATORIA

A mis padres, a mi hermano, y a mis abuelos por su apoyo incondicional y por ser la inspiración detrás de cada paso que doy.

A mi enamorada por su paciencia, comprensión y por motivarme a alcanzar mis metas con amor y determinación.

AGRADECIMIENTOS

A mi familia por su amor y apoyo incondicional.

A mi enamorada por su ánimo constante y por creer en mí en cada momento.

A mi asesor por su guía y paciencia que fueron fundamentales para realizar con éxito el presente trabajo.

A todos ustedes, mi gratitud infinita.

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

**Implementación de un sistema de gestión de información de productos para optimizar
la administración del catálogo de productos en una entidad E-commerce**

Autor: Poclín Meza, Carlos Iván
Asesor: Machado Vicente, Joel Fernando
Título: Trabajo de Suficiencia Profesional para optar el Título Profesional de
Ingeniero de Software
Fecha: Diciembre de 2024

RESUMEN

El presente trabajo de suficiencia profesional describe la implementación de un Sistema de Gestión de Información de Productos, o PIM, en una Entidad E-commerce. Anteriormente, la información de los productos estaba almacenada en una Plataforma de Terceros, o Plataforma Externa, lo cual generaba desafíos en términos de la propiedad u *ownership* de los datos, y limitaba la flexibilidad en la estructura de la información, dificultando así la implementación de nuevas funcionalidades y la integración con otros servicios, tales como: *checkout*, inventario, *delivery* y procesamiento de órdenes.

En el transcurso del año se realizan distintas campañas promocionales, y es necesario la actualización masiva del catálogo de productos, principalmente, los precios. Con este fin, se utilizaba un archivo en formato “.csv” que contenía la información de todos los productos, el cual era cargado en la Plataforma Externa. Este proceso presentaba riesgos considerables, tales como errores en los precios, que no solo afectarían la reputación de la empresa, sino que se podrían traducir en grandes pérdidas económicas.

Para resolver la problemática identificada, se implementó un Sistema PIM basado en una infraestructura *Serverless* utilizando servicios de AWS, con el objetivo de centralizar y gestionar el catálogo de productos de manera eficiente. Esta solución garantiza la integridad de la información y mejora la flexibilidad, facilitando la adición de nuevas funcionalidades, así como la integración con otros servicios.

Se integró un Sistema de Gestión de Contenido, o CMS, el cual brinda una interfaz de usuario intuitiva que reduce la posibilidad de errores durante la actualización de la información. Se creó un módulo para la automatización de las actualizaciones masivas, optimizando el proceso, además de mitigar el riesgo de pérdidas económicas por errores humanos.

Palabras clave: E-commerce, PIM, Serverless, CMS, actualización masiva.

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

**Implementation of a product information management system to optimize the product
catalog administration in an E-commerce entity**

Author: Poclín Meza, Carlos Iván
Advisor: Machado Vicente, Joel Fernando
Title: Professional Sufficiency Work to opt for the Professional Title of Software Engineer
Date: December 2024

ABSTRACT

This work describes the implementation of a Product Information Management System, or PIM, in an E-commerce Entity. Previously, the product information was stored in a Third-Party Platform, or External Platform, which generated challenges in terms of data ownership, and limited the flexibility in the information structure, making it difficult to implement new features and the integration with other services, such as: checkout, inventory, delivery and order processing.

During the year, different promotional campaigns are carried out, and it is necessary to massively update the product catalog, mainly the prices. For this purpose, a file in “.csv” format was used, containing the information of all products, which was uploaded to the External Platform. This process presented considerable risks, such as pricing errors, which would not only affect the reputation of the company, but could also result in large economic losses.

To solve the identified problem, a PIM system was implemented based on a Serverless infrastructure using AWS services, in order to centralize and manage the product catalog efficiently. This solution guarantees the integrity of the information and improves flexibility, facilitating the addition of new features, as well as the integration with other services.

A Content Management System, or CMS, was integrated, which provides an intuitive user interface that reduces the possibility of errors during the updating of information. A module for the automation of massive updates was created, optimizing the process and mitigating the risk of economic losses due to human errors.

Keywords: E-commerce, PIM, Serverless, CMS, massive updates.

TABLA DE CONTENIDO

RESUMEN	v
ABSTRACT	vii
TABLA DE CONTENIDO	ix
ÍNDICE DE FIGURAS	xi
ÍNDICE DE TABLAS	xii
INTRODUCCIÓN	1
CAPÍTULO I – TRAYECTORIA PROFESIONAL	3
1.1 Presentación profesional	3
1.2 Experiencia profesional	3
1.3 Formación académica	6
CAPÍTULO II – CONTEXTO EN EL QUE SE DESARROLLA LA EXPERIENCIA	9
2.1 Empresa – actividad que realiza	9
2.2 Visión	9
2.3 Misión	9
2.4 Organización de la empresa	10
2.5 Área, cargo y funciones desempeñadas	11
2.6 Experiencia profesional realizada en la organización	12
CAPÍTULO III – ACTIVIDADES DESARROLLADAS	14
3.1 Situación problemática	14
3.1.1 Definición del problema	17
3.2 Solución	18
3.2.1 Objetivos	18

3.2.2	Alcance	19
3.2.3	Etapas y metodología.....	20
3.2.4	Fundamentos utilizados	24
3.2.5	Implementación de las áreas, procesos, sistemas y buenas prácticas	39
3.3	Evaluación	77
3.3.1	Evaluación económica / Evaluación costo – beneficio	77
3.3.2	Interpretación del VAN y del TIR.....	80
CAPÍTULO IV – REFLEXIÓN CRÍTICA DE LA EXPERIENCIA.....		82
CAPÍTULO V – CONCLUSIONES Y RECOMENDACIONES		84
5.1	Conclusiones.....	84
5.2	Recomendaciones	85
5.3	Fuentes de información	86
5.4	Glosario	87
ANEXOS		90

ÍNDICE DE FIGURAS

Figura 1 Organización de la empresa	10
Figura 2 Equipo de desarrollo de software.....	11
Figura 3 Flujo Scrum de un Sprint	23
Figura 4 Cronograma del proyecto.....	39
Figura 5 Resultado de retrospectiva	42
Figura 6 Dashboard del proyecto	43
Figura 7 Flujo de trabajo con git	45
Figura 8 Flujo anterior para gestionar el catálogo de productos	47
Figura 9 Proceso anterior para acceder al catálogo de productos	47
Figura 10 Nuevo flujo para gestionar el catálogo de productos.....	51
Figura 11 Nuevo proceso para acceder al catálogo de productos.....	52
Figura 12 Arquitectura del Sistema PIM	53
Figura 13 Estructura de un pipeline.....	56
Figura 14 Ejemplo de option	58
Figura 15 Ejemplo de asset container.....	58
Figura 16 Ejemplo de variant	59
Figura 17 Ejemplo de product	60
Figura 18 Estructura de la petición POST	63
Figura 19 Lógica de autorización de eventos	66
Figura 20 Lógica de validación de eventos	67
Figura 21 Lógica de procesamiento de eventos	68
Figura 22 Lógica para acceder al catálogo de productos	69
Figura 23 Configuración de un release.....	70
Figura 24 Script para la migración de los datos	71
Figura 25 Actualización del Back - End.....	72
Figura 26 Integración de alarmas	73
Figura 27 Webhook del ambiente de producción	76
Figura 28 Feature flag encendido	76

ÍNDICE DE TABLAS

Tabla 1	Experiencia profesional	3
Tabla 2	Formación académica	7
Tabla 3	Estudios y certificaciones en idiomas	7
Tabla 4	Cursos y certificaciones complementarias	8
Tabla 5	Estructura del archivo en formato ".csv"	15
Tabla 6	Herramientas tecnológicas	19
Tabla 7	Equipo de desarrollo	40
Tabla 8	Scrum aplicado al proyecto.....	41
Tabla 9	Requisitos de un merge request	44
Tabla 10	Estructura de una historia de usuario	45
Tabla 11	Product backlog.....	48
Tabla 12	Clase option	49
Tabla 13	Clase asset container	49
Tabla 14	Clase variant.....	50
Tabla 15	Clase product	50
Tabla 16	Estados de las entidades en el CMS.....	61
Tabla 17	Configuración de los webhooks.....	62
Tabla 18	Configuración de los servicios de AWS.....	64
Tabla 19	Resumen de la migración del catálogo de productos.....	74
Tabla 20	Escenarios de prueba.....	74
Tabla 21	Costos estimados de la plataforma externa	77
Tabla 22	Costos estimados de los servicios de AWS	78
Tabla 23	Costo total estimado del Sistema PIM	78
Tabla 24	Reserva de contingencia anual estimada.....	79
Tabla 25	Beneficios del Sistema PIM.....	79
Tabla 26	VAN y TIR	80

INTRODUCCIÓN

La crisis del COVID-19 aceleró la transformación digital en diversos sectores, siendo el comercio electrónico uno de los más afectados. Este cambio obligó a las empresas a adaptarse rápidamente al E-commerce, que pasó a ser su principal canal de ventas. En este nuevo contexto, la gestión eficiente del catálogo de productos se convirtió en una necesidad crítica para mantener la competitividad y garantizar la calidad del servicio.

Muchas empresas, entre las cuales se encuentra incluida la Entidad E-commerce mencionada en el presente informe, adoptaron plataformas externas para simplificar la administración de la información de los productos. Sin embargo, a medida que el catálogo de productos crecía, la integridad de los datos se volvía más difícil de mantener. Además, la dependencia de plataformas de terceros introducía limitaciones en la estructura de los datos, dificultando la incorporación de nuevas funcionalidades y la integración con otros servicios, tales como: *checkout*, inventario, *delivery* y procesamiento de órdenes, reduciendo de esta manera la capacidad de adaptarse a los cambios del mercado.

Durante el año, las distintas campañas promocionales generan la necesidad de realizar actualizaciones masivas de los productos, principalmente los precios. Estas actualizaciones se solían realizar mediante la carga de un archivo en formato ".csv" en la Plataforma Externa, un proceso que, aunque era común, conllevaba un alto riesgo de que se presenten errores. Estas equivocaciones no solo podrían perjudicar la reputación de la empresa, sino que también provocarían pérdidas económicas significativas. Un claro ejemplo es mencionado por Blayzer Digital (2022), donde se explica que, en octubre del 2021, una gran cantidad de productos, principalmente muebles cuyos precios son de cientos de dólares, fueron vendidos por menos de \$1.00 en la plataforma de comercio electrónico de Walmart, afectando económicamente a la empresa.

Este trabajo de suficiencia profesional describe la implementación de un Sistema de Gestión de Información de Productos (PIM) basado en una infraestructura *Serverless*

utilizando los servicios de AWS con el objetivo de resolver los problemas identificados. Esta solución permitió:

- Tener el *ownership* y la gestión eficiente del catálogo de productos, lo cual facilitó la implementación de nuevas funcionalidades y la integración con otros servicios.
- Tener una interfaz de usuario intuitiva, lo cual redujo la posibilidad de que se presenten errores durante la actualización de la información.
- Automatizar las actualizaciones masivas, lo cual optimizó el proceso y mitigó el riesgo de pérdidas económicas por errores humanos.

En el CAPÍTULO I – TRAYECTORIA PROFESIONAL, se detallará la experiencia laboral y la formación académica del autor.

En el CAPÍTULO II – CONTEXTO EN EL QUE SE DESARROLLA LA EXPERIENCIA, se describirá la empresa Consultora, así como al cliente que viene a ser una Entidad E-commerce, se brindará información respecto a la visión, misión y organización de la empresa, así como las responsabilidades y experiencia del autor.

En el CAPÍTULO III – ACTIVIDADES DESARROLLADAS, se presentará la situación problemática, los objetivos, el alcance, la metodología y las etapas del proyecto, así como una evaluación de la solución.

En el CAPÍTULO IV – REFLEXIÓN CRÍTICA DE LA EXPERIENCIA, se presentará una reflexión crítica de la experiencia obtenida en la realización del presente proyecto.

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

Nota: Por motivo de acuerdos de confidencialidad, el nombre de la empresa Consultora y el cliente Entidad E-commerce fueron anonimizados en el presente informe, así como también el nombre de la Plataforma Externa o Plataforma de Terceros.

CAPÍTULO I – TRAYECTORIA PROFESIONAL

1.1 Presentación profesional

El autor del presente trabajo de suficiencia profesional es Bachiller en Ingeniería de Software por la Universidad Nacional Mayor de San Marcos. Cuenta con más de seis años de experiencia en el diseño, desarrollo y mantenimiento de soluciones de software de calidad. Se ha desempeñado como desarrollador en compañías nacionales e internacionales. Actualmente, brinda servicios a una empresa Consultora en el puesto de Senior Full Stack Developer. Es un apasionado de la tecnología y constantemente crea pruebas de concepto innovadoras para mejorar los productos de software.

1.2 Experiencia profesional

A continuación, en la Tabla 1, se detalla la experiencia profesional del autor, donde podemos resaltar que se desempeña laboralmente en empresas extranjeras desde el año 2022.

Tabla 1
Experiencia profesional

Periodo	Empresa
Agosto 2022 – Presente	Consultora Cliente: Entidad E-commerce Cargo: Senior Full Stack Developer Ubicación: USA, Remoto Funciones: - Diseñar, desarrollar, y mantener nuevas funcionalidades para la aplicación web de la E-commerce, así como sistemas de software que dan soporte a la lógica del negocio. - Identificar las causas de las incidencias e implementar las correcciones necesarias.

-
- Realizar pruebas de concepto para conocer la viabilidad de nuevas funcionalidades.
 - Gestionar y liderar pases a producción utilizando AWS CodePipeline.
 - Monitorear los sistemas de software utilizando AWS CloudWatch.

Tecnologías: Jest, Express.js, JavaScript, Node.js, CMS, TypeScript, React.js, AWS

Enero 2022 – junio 2022 **Clipboard Health**

Cargo: Software Developer

Ubicación: USA, Remoto

Funciones:

- Implementar una funcionalidad de búsqueda de alto rendimiento utilizando MongoDB Atlas.
- Migrar los datos de MongoDB a PostgreSQL utilizando Prisma, logrando optimizar los REST API.
- Implementar y optimizar GraphQL API.
- Implementar un paquete de código para la administración de logs, el cual es usado por todos los equipos de ingeniería.
- Implementar servicios basados en una arquitectura orientada a eventos utilizando Google Cloud Pub/Sub.
- Monitorear los sistemas de software con Datadog.

Tecnologías: Jenkins, Jest, GraphQL, NestJS, MongoDB, PostgreSQL, Node.js, TypeScript, REST API, Datadog, React.js

Agosto 2021 – **B89**

diciembre 2021 **Cargo:** Software Developer

Ubicación: Lima, Perú

Funciones:

-
- Diseñar, crear, y mantener nuevas funcionalidades en una arquitectura orientada a eventos construida con Amazon MSK (Apache Kafka).
 - Implementar microservicios de alto rendimiento basados en gRPC utilizando Node.js, TypeScript, Moleculer, y TiDB.
 - Optimizar procesos core usando Redis.
 - Implementar un paquete de código que se encarga de la configuración de distintos servicios de AWS utilizando AWS SDK.
 - Gestionar los pases a producción usando Jenkins.

Tecnologías: AWS, Jenkins, Jest, gRPC, Redis, Node.js, TypeScript, Apache Kafka

Febrero 2020 – Julio 2021

Quantum Talent

Cargo: Software Developer

Ubicación: Lima, Perú

Funciones:

- Diseñar e implementar una funcionalidad de comunicación en tiempo real usando Serverless, API Gateway WebSockets, Twilio, y SendGrid.
- Diseñar e implementar una arquitectura orientada a eventos para todos los procesos de la compañía utilizando AWS EventBridge.
- Diseñar, desarrollar y mantener REST API escalables.
- Implementar una aplicación web siguiendo el modelo de Atomic Design usando styled-components y Storybook.
- Implementar pruebas End-to-End con Cypress.

Tecnologías: AWS, Elasticsearch, Cypress, NestJS, MongoDB, Kubernetes, Serverless Framework, TypeScript, React.js, Gitlab CI/CD

Marzo 2019 – Julio 2019	Quantum Talent Cargo: Software Developer Ubicación: Lima, Perú Funciones: - Diseñar, desarrollar, y mantener REST API escalables. - Implementar una aplicación web que consiste en 7 micro frontends. Tecnologías: AWS, JavaScript, MongoDB, Kubernetes, Node.js, REST API, React.js, Gitlab CI/CD
Diciembre 2018 – febrero 2019	Prodequa Cargo: Software Developer Ubicación: Lima, Perú Funciones: - Diseñar, desarrollar, y mantener REST API escalables. - Desarrollar y mantener aplicaciones web con sus respectivos CMS. Tecnologías: JavaScript, Koa.js, MongoDB, Node.js, Vue.js, REST API, DigitalOcean
Agosto 2018 – octubre 2018	Huawei Technologies Cargo: Practicante de Aplicaciones y Software Ubicación: Lima, Perú Funciones: - Gestionar y monitorear la Base de Datos de la empresa. Tecnologías: Linux, Oracle

Nota. Fuente: Elaboración propia.

1.3 Formación académica

En la Tabla 2, se presenta la formación académica del autor, donde podemos destacar que realizó un intercambio académico internacional en México en el año 2019.

Tabla 2
Formación académica

Periodo	Institución
2015 – 2020	Universidad Nacional Mayor de San Marcos Grado: Bachiller de la Escuela Profesional de Ingeniería de Software – Facultad de Ingeniería de Sistemas e Informática Ubicación: Lima, Perú
2019	Universidad Nacional Autónoma de México Grado: Intercambio Académico Internacional en Ciencias de la Computación Ubicación: Ciudad de México, México

Nota. Fuente: Elaboración propia.

En la Tabla 3, se listan los estudios y certificaciones en idiomas realizados por el autor, el cual cuenta con el nivel B2 en inglés.

Tabla 3
Estudios y certificaciones en idiomas

Periodo	Institución
2017	University of Cambridge Idioma: Inglés Grado: First Certificate in English, nivel B2 Ubicación: Lima, Perú
2014 – 2016	Asociación Cultural Peruano Británica (Británico) Idioma: Inglés Grado: Avanzado Ubicación: Lima, Perú

Nota. Fuente: Elaboración propia.

En la Tabla 4, se presentan los cursos y certificaciones complementarias del autor, donde se puede destacar las pasantías internacionales que realizó en Estados Unidos y China en el año 2018.

Tabla 4
Cursos y certificaciones complementarias

Periodo	Institución
2020	Certiprof Grado: Scrum Foundation Professional Certificate (SFPC) Ubicación: USA, Remoto
2018	Huawei Grado: Seeds for the Future Program Certificate of Honor, por haber ganado el programa “Seeds for the Future, Perú, 2018” Ubicación: Shenzhen, China
2018	Beijing Language and Culture University Grado: Chinese Language & Culture Experience, por haber ganado el programa “Seeds for the Future, Perú, 2018” Ubicación: Pekín, China
2018	Harvard University y MIT Grado: Student Leaders for the 21st Century Universidad Nacional Mayor de San Marcos, por haber obtenido el primer puesto de la Escuela Profesional de Ingeniería de Software en la UNMSM Ubicación: Boston, USA

Nota. Fuente: Elaboración propia.

CAPÍTULO II – CONTEXTO EN EL QUE SE DESARROLLA LA EXPERIENCIA

2.1 Empresa – actividad que realiza

La empresa en la cual el autor de este trabajo de suficiencia profesional realizó sus labores, y fue parte del equipo encargado de la implementación del proyecto que es objeto de estudio, es una Consultora que ofrece servicios como gestión de proyectos, diseño y desarrollo de software, así como automatización de pruebas. Sus oficinas principales se encuentran en Estados Unidos y México, lo cual permite el trabajo de manera distribuida con equipos en diferentes zonas horarias, facilitando la colaboración eficaz y eficiente.

La empresa sigue una metodología ágil, trabajando en iteraciones cortas llamadas sprints. Su enfoque se basa en desarrollar y desplegar funcionalidades de manera incremental, permitiendo la retroalimentación constante de los clientes, asegurando que las soluciones se ajusten a las necesidades específicas de cada proyecto y cliente. Actualmente cuenta con más de 300 colaboradores de Latinoamérica y Estados Unidos, los cuales son expertos en tecnologías como TypeScript, JavaScript, Go, AWS, entre otros.

2.2 Visión

Ser un líder en el desarrollo de software, proporcionando talento especializado para satisfacer las necesidades de proyectos de diferentes tamaños. Se enfocan en una gestión ágil y en el desarrollo incremental, lo que les permite adaptarse rápidamente a los cambios y maximizar el retorno de inversión para sus clientes.

2.3 Misión

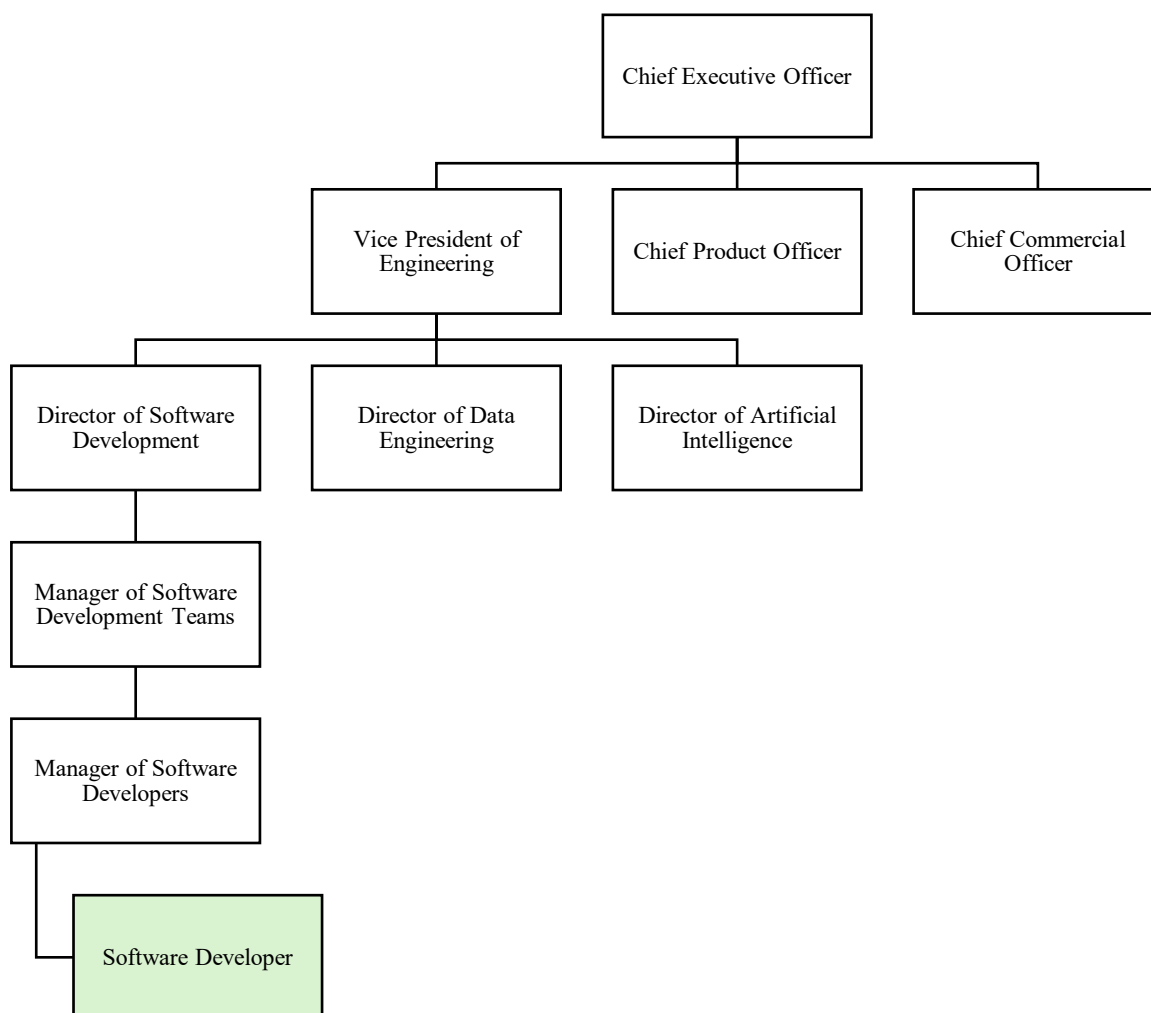
Desarrollar soluciones de software elegantes que generen resultados poderosos, priorizando la colaboración estrecha con sus clientes para entender sus objetivos y estrategias. Su enfoque se basa en integrar completamente a los desarrolladores en la cultura

y el estilo de trabajo de las empresas con las que colaboran, lo que permite una comunicación eficiente y una entrega rápida de resultados.

2.4 Organización de la empresa

En la Figura 1 se presenta el organigrama referencial de la Consultora. El cual está simplificado para tener visibilidad de la posición en la cual el autor del presente trabajo desempeñó sus labores.

Figura 1
Organización de la empresa



Nota. Fuente: Elaboración propia.

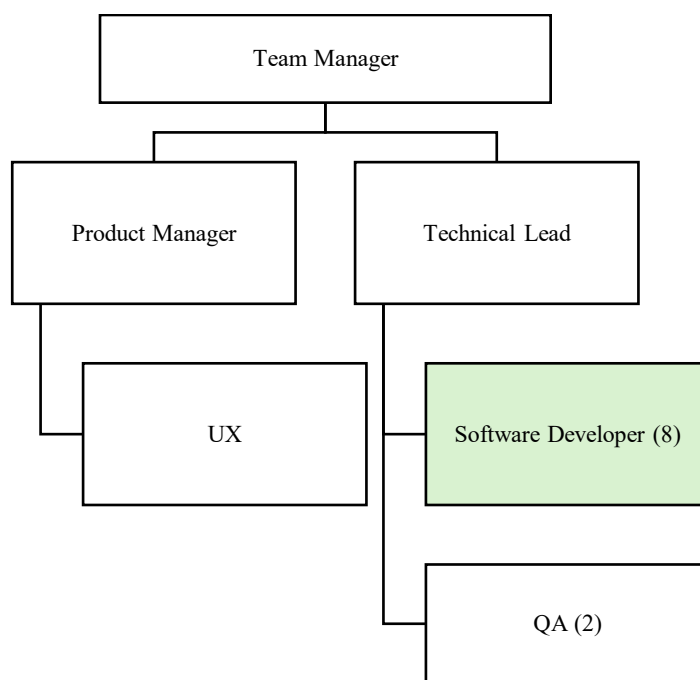
2.5 Área, cargo y funciones desempeñadas

Si bien se viene señalando en los puntos anteriores que el autor del presente informe brinda sus servicios a una Consultora, se debe indicar que sus actividades están completamente enfocadas en el cliente, el cual es una Entidad E-commerce dedicada a la venta de diversos productos electrónicos, siendo el comercio electrónico su principal canal de ventas.

El autor pertenece al equipo de desarrollo de software de la Entidad E-commerce, donde se desempeña en el puesto de Senior Full Stack Developer desde el año 2022, y es allí donde participó en el diseño e implementación del proyecto mencionado en este trabajo de suficiencia profesional.

En la Figura 2 se presenta la estructura del equipo de desarrollo de software de la Entidad E-commerce.

Figura 2
Equipo de desarrollo de software



Nota. Fuente: Elaboración propia.

A continuación, se listan las funciones desempeñadas:

- Diseñar e implementar nuevas funcionalidades de los sistemas de software de la Entidad E-commerce. Así como dar mantenimiento a los procesos existentes.
- Dar seguimiento a las incidencias, identificar las causas, y actualizar el código para solucionarlas.
- Implementar paquetes de código que son utilizados por los distintos sistemas de la Entidad E-commerce con el fin de estandarizar los procesos, tales como, logs y manejo de errores.
- Mitigar la deuda técnica mediante el aumento de la cobertura del código aplicando pruebas unitarias y funcionales.
- Analizar los procesos existentes e identificar posibilidades de mejora mediante la implementación de soluciones de software.
- Crear pruebas de concepto para analizar la viabilidad de nuevas funcionalidades, así como la integración con soluciones de terceros.
- Gestionar y liderar el proceso de pase a producción de los sistemas de software de la Entidad E-commerce.
- Realizar revisiones de código.

2.6 Experiencia profesional realizada en la organización

El autor ha participado en distintos proyectos como miembro del equipo de desarrollo de software de la Entidad E-commerce, entre los cuales se encuentran:

- Implementar un proceso que automatiza la configuración de la aplicación web de la E-commerce.
- Diseñar, implementar, y dar mantenimiento a un Sistema de Gestión de Información de Productos (PIM).
- Crear y ejecutar scripts para la migración de los productos teniendo en cuenta la consistencia de la información.
- Optimizar el motor de búsqueda de los productos de la E-commerce, mediante la implementación de Amazon Kendra.

- Integrar el catálogo de productos de la E-commerce con Click2Cart (SmartCommerce).
- Crear un widget que verifica la compatibilidad de los productos de la E-commerce con distintos modelos de vehículos, mediante la integración con CarMD.
- Trabajar en conjunto con otros equipos de la Entidad E-commerce para la integración de servicios tales como: inventario, checkout, delivery, y procesamiento de órdenes.

CAPÍTULO III – ACTIVIDADES DESARROLLADAS

3.1 Situación problemática

En la Entidad E-commerce en la cual el autor implementó la solución presentada en este informe, la información de los productos estaba almacenada en una Plataforma Externa, que además definía la estructura de los datos. Esto generaba dificultades respecto al desarrollo de nuevas funcionalidades, ya que existía poca flexibilidad debido a la dependencia que se tenía con la Plataforma de Terceros. Además, al no tener el *ownership* de la información, la integración con otros servicios tales como inventario, *checkout*, *delivery*, y procesamiento de órdenes, era innecesariamente compleja.

Aparte de ello, durante el año se tienen campañas promocionales que implican la actualización masiva del catálogo de productos, principalmente los precios. Esto es realizado por dos miembros del equipo llamados content authors o autores de contenido, quienes utilizaban un archivo en formato “.csv” que contenía la información de todos los productos, realizaban los cambios manualmente, y a altas horas de la noche anterior al inicio de las campañas subían el archivo a la Plataforma Externa, para así aplicar las actualizaciones a una gran cantidad de productos, por no decir todos. Este proceso presentaba un gran riesgo, ya que estaba propenso a equivocaciones humanas debido a la falta de una interfaz intuitiva que reduzca la posibilidad de que se presenten errores. Además, la visualización de información errónea en la E-commerce, especialmente los precios, no solo afectaría la reputación de la empresa, sino que podría generar pérdidas económicas significativas. Un ejemplo de esto es mencionado por Blayzer Digital (2022), donde se explica que, en enero del 2022, el precio de carteras de lujo en la tienda online de Marc Jacobs bajó de \$195.00 a \$0.00 más \$10.00 de impuestos y envío, lo cual generó que muchos compradores adquirieran el producto a este precio, afectando económicamente a la empresa.

Antes de realizar la implementación del presente proyecto, el catálogo de productos de la empresa contaba con 677 productos, pero, cada producto puede tener variaciones por color, fuente de energía, y si se vende en bundles o packs de dos o más productos, lo cual

hacía un total de 1324 variaciones, es decir, los autores de contenido tenían que realizar cambios manualmente de la información de hasta 1324 variaciones en un archivo en formato “.csv”, luego subían el archivo a la Plataforma Externa, y finalmente la información actualizada era visualizada en la E-commerce.

El archivo en formato “.csv” utilizado para actualizar la información de los productos en la Plataforma Externa tenía 59 columnas, de las cuales solo eran utilizadas 18, en la Tabla 5 se presentan solo las columnas más importantes, y también se incluye las limitaciones de los campos.

Tabla 5
Estructura del archivo en formato ".csv"

Columna	Detalle
Slug	- Descripción: Es el identificador de un producto - Tipo de dato: String - Validaciones: Requerido, hasta 64 caracteres
Title	- Descripción: Es el título o nombre de un producto - Tipo de dato: String - Validaciones: Requerido, hasta 256 caracteres
Body	- Descripción: Es la descripción de un producto, acepta contenido HTML - Tipo de dato: String - Limitaciones: Se aplica la misma descripción a todas las variaciones del producto
Tags	- Descripción: Es un conjunto de etiquetas que ayudan a personalizar los productos - Tipo de dato: Arreglo de String - Limitaciones: Se aplican los mismos tags a todas las variaciones del producto
Image Src	- Descripción: Es el URL de la imagen de un producto, la cual se encuentra en el CDN de la Plataforma Externa

	- Tipo de dato: String - Limitaciones: Solo acepta una imagen por producto
Variant Title	- Descripción: Es el título o nombre de una variación - Tipo de dato: String - Validaciones: Requerido, hasta 256 caracteres
Variant SKU	- Descripción: Es el identificador de una variación - Tipo de dato: String - Validaciones: Requerido, único, hasta 64 caracteres
Variant Price	- Descripción: Es el precio de una variación - Tipo de dato: Number - Validaciones: Requerido
Variant Compare At Price	- Descripción: Este valor se usa en caso la variación esté en oferta - Tipo de dato: Number
Variant Image	- Descripción: Es el URL de la imagen de una variación, la cual se encuentra en el CDN de la Plataforma Externa - Tipo de dato: String - Limitaciones: Solo acepta una imagen por variación

Nota. Fuente: Elaboración propia.

Tomando en cuenta el escenario anteriormente presentado, se implementó un Sistema de Gestión de Información de Productos (PIM) basado en una infraestructura Serverless con el objetivo de solucionar los problemas identificados, haciendo énfasis en los siguientes aspectos clave:

- La centralización y gestión del catálogo de productos garantizando la integridad de los datos.
- El control total sobre la información, facilitando la implementación de nuevas funcionalidades y la integración con otros servicios.
- La reducción de la posibilidad de errores en la actualización de la información mediante la integración con un Sistema de Gestión de Contenido (CMS) que brinda una interfaz intuitiva.

- La optimización de las actualizaciones masivas mediante la automatización del proceso, lo cual también mitiga el riesgo de pérdidas económicas por errores humanos.

3.1.1 Definición del problema

El problema principal consistía en que la información de los productos estaba almacenada en la Plataforma Externa, por lo cual existía una dependencia directa respecto al acceso y estructura de los datos. Esto agregaba complejidad a la implementación de nuevas funcionalidades y la integración con otros servicios. Además, el proceso de actualización de la información estaba propensa a equivocaciones que podrían afectar la reputación de la empresa y generar pérdidas económicas significativas.

3.1.1.1 Problema general.

Tomando en cuenta la problemática identificada anteriormente, se formuló el problema principal en la siguiente pregunta:

¿Cómo podemos tener control total sobre los datos del catálogo de productos, garantizando la integridad de la información?

3.1.1.2 Problemas específicos.

- ¿Cómo podemos facilitar la implementación de nuevas funcionalidades y la integración con otros servicios que utilizan la información de los productos?
- ¿Cómo podemos reducir el riesgo de que se presenten errores humanos en el proceso de actualización de la información de los productos?
- ¿Cómo podemos optimizar la actualización masiva de la información de los productos para mitigar el riesgo a equivocaciones?
- ¿Cómo podemos realizar la migración del catálogo de productos desde la Plataforma Externa al Sistema PIM garantizando la integridad de los datos?

3.2 Solución

Para solucionar la problemática identificada, se realizó la implementación de un Sistema de Gestión de Información de Productos (PIM) basado en una infraestructura Serverless que permitió la centralización y gestión del catálogo de productos garantizando la integridad de los datos. Esto facilitó la implementación de nuevas funcionalidades y la integración con otros servicios. Además, la integración con un Sistema de Gestión de Contenido (CMS), el cual brinda una interfaz intuitiva, contribuyó a reducir el riesgo de que se presenten equivocaciones en la actualización de la información. Por último, el desarrollo de un módulo para automatizar las actualizaciones masivas optimizó el proceso, mitigando el riesgo a errores.

3.2.1 Objetivos

3.2.1.1 Objetivo general.

Implementar un Sistema de Gestión de Información de Productos (PIM) basado en una infraestructura Serverless en una Entidad E-commerce.

3.2.1.2 Objetivos específicos.

- Exponer el catálogo de productos mediante la implementación de REST API.
- Integrar un Sistema de Gestión de Contenido (CMS) que brinda una interfaz intuitiva.
- Implementar un módulo que automatiza las actualizaciones masivas.
- Crear un script para la migración de los productos, el cual consistió en tres etapas: extracción, transformación, y publicación de los datos.

3.2.2 Alcance

3.2.2.1 Alcance del proyecto.

El alcance del proyecto contempló lo siguiente:

- Implementar un Sistema de Gestión de Información de Productos (PIM) basado en una infraestructura Serverless.
- Implementar REST API que permitan el acceso al catálogo de productos.
- Realizar la integración con un Sistema de Gestión de Contenido (CMS).
- Implementar un módulo para automatizar las actualizaciones masivas de información.
- Migrar la información del catálogo de productos desde la Plataforma Externa al nuevo Sistema PIM.

3.2.2.2 Alcance organizacional.

El alcance organizacional comprende al equipo de desarrollo de la Entidad E-commerce.

3.2.2.3 Alcance tecnológico.

A continuación, en la Tabla 6 se listan las herramientas tecnológicas utilizadas, donde se puede observar que se trabajó con diversos servicios de AWS.

Tabla 6
Herramientas tecnológicas

Lenguajes de programación y librerías	Node.js, TypeScript, React, Jest, Cypress
Plataforma y servicios en la nube	Amazon Web Services (AWS) - Para la implementación del sistema: API Gateway, Secrets Manager, SQS, Lambda, SNS, S3, ElastiCache, SDK

	- Para el monitoreo: CloudWatch, Chatbot - Para la integración y despliegue continuos (CI/CD): CodePipeline - Para la construcción y mantenimiento de la infraestructura: CDK
Sistema de Gestión de Contenido (CMS)	Admin API, Delivery API, Preview API, Custom Apps, Webhooks
Caché	Redis
Plataforma Externa	GraphQL Delivery API
Otros	Git, Husky

Nota. Fuente: Elaboración propia.

3.2.3 Etapas y metodología

3.2.3.1 Etapas.

El proyecto consistió en las siguientes etapas:

Iniciación

- Definición del cronograma del proyecto
- Formación del equipo
- Creación del Dashboard y Product Backlog del proyecto

Planificación

- Extracción y validación de los requerimientos
- Creación de las Historias de Usuario

Diseño

- Definición de la estructura de datos
- Diseño y validación de la arquitectura de la solución

Aprovisionamiento de la infraestructura

- Configuración del proceso de CI/CD
- Configuración del CMS, esto incluye la estructura de datos
- Construcción de la infraestructura del sistema

Desarrollo

- Desarrollo del código del sistema
- Desarrollo de scripts para la migración de los datos
- Desarrollo de REST API
- Implementar pruebas unitarias
- Ejecución de pruebas funcionales

Pre – producción

- Configuración de dashboards y alertas
- Migración del catálogo de productos
- Ejecución de pruebas funcionales

Despliegue

- Gestión y ejecución del pase a producción

Mantenimiento y monitoreo

- Verificación del correcto funcionamiento del sistema
- Revisión y resolución de incidencias
- Implementación de mejoras al sistema

3.2.3.2 Metodología.

Para desarrollar el presente proyecto se utilizó una metodología ágil, siguiendo los lineamientos del flujo de trabajo de la Entidad E-commerce, el cual se basa en su mayoría en el marco de trabajo Scrum.

Scrum

De acuerdo con SCRUMstudy™ (2022), Scrum es un marco de trabajo ágil utilizado para la gestión de proyectos de desarrollo de productos y servicios. Está enfocado en dar resultados rápidos e incrementales en ciclos cortos llamados Sprints, mediante la transparencia en la comunicación entre los miembros del equipo y la creación de un ambiente de responsabilidad colectiva y progreso continuo.

Sprint

SCRUMstudy™ (2022) menciona que un Sprint es un ciclo de trabajo corto que generalmente suele durar dos semanas. Durante este período, el equipo se enfoca en desarrollar un incremento del producto.

Historia de Usuario

Según SCRUMstudy™ (2022), una Historia de Usuario se utiliza para definir los requerimientos en Scrum, suele tener una breve descripción de una funcionalidad basándose en el punto de vista del usuario. Su formato es el siguiente: “Como <usuario>, quiero <acción>, para <beneficio>”.

Artefactos de Scrum

Tal como menciona SCRUMstudy™ (2022), los artefactos de Scrum son los siguientes:

- Product Backlog: Es la lista priorizada de las Historias de Usuario a desarrollar.
- Sprint Backlog: Es la lista de las Historias de Usuario a desarrollar en un Sprint.
- Incremento: Es el producto u avance completado durante un Sprint.

Ceremonias de un Sprint

De acuerdo con SCRUMstudy™ (2022), las ceremonias que se desarrollan durante un Sprint son las siguientes:

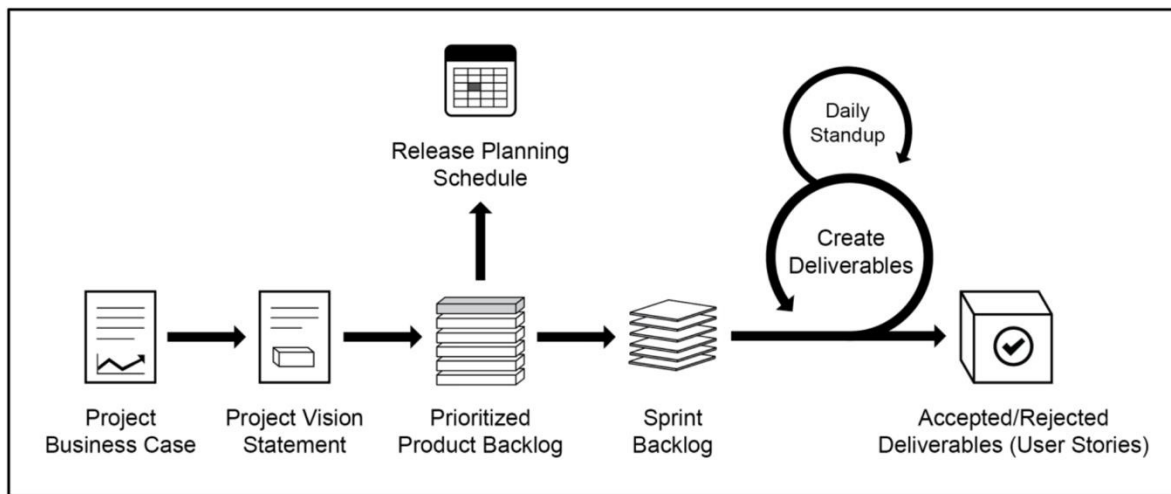
- Sprint planning o planeamiento de un Sprint. En esta reunión se seleccionan las Historias de Usuarios que serán trabajadas durante el Sprint.
- Daily meeting o reunión diaria. Reunión diaria para revisar el progreso del Sprint.

- Grooming o refinamiento. En esta reunión se preparan las Historias de Usuario.
- Retrospectiva. Aquí se hace una reflexión respecto al trabajo desarrollado durante el Sprint y cómo se podría mejorar en el siguiente.

Según SCRUMstudy™ (2022), primero se analizan los procesos o casos de negocio con el objetivo de crear las Historias de Usuario que alimentan el Product Backlog, posteriormente en la planificación se deciden que tareas se desarrollarán durante el Sprint mediante la creación del Sprint Backlog. Aparte del desarrollo de las tareas, durante el Sprint también se da lugar a los eventos o ceremonias como las reuniones diarias, para al final entregar los productos definidos en las Historias de Usuario.

En la Figura 3 se muestra el Flujo Scrum de un Sprint, el cual ha sido explicado anteriormente.

Figura 3
Flujo Scrum de un Sprint



Nota. Fuente: SCRUMstudy™ (2022).

3.2.4 Fundamentos utilizados

Comercio electrónico o E-commerce

De acuerdo con Forbes (2024), el comercio electrónico, conocido comúnmente como E-commerce, se define de manera sencilla como el proceso de comprar y vender productos o servicios en línea. Este proceso abarca una amplia gama de actividades comerciales, desde gigantes como Amazon hasta pequeños emprendedores que venden productos artesanales a través de redes sociales.

Forbes (2024) menciona que las transacciones de E-commerce pueden realizarse mediante aplicaciones web y móviles, correos electrónicos, redes sociales, entre otros. Al igual que en los negocios tradicionales, no existe un enfoque único que se adapte a todas las empresas; la clave está en alinear las estrategias de comercio electrónico con las necesidades tanto del cliente como de la organización.

Según Forbes (2024), una de las principales ventajas del E-commerce frente a los negocios físicos tradicionales es que elimina muchas barreras de entrada. Esto permite que pequeñas y medianas empresas, e incluso emprendedores individuales, puedan acceder a mercados globales, superando limitaciones como la ubicación geográfica o los costos iniciales elevados.

Sistema de gestión de información de productos o PIM

De acuerdo con Akeneo (2024), un sistema de gestión de información de productos, o PIM, es una solución tecnológica diseñada para que las empresas gestionen, centralicen y distribuyan la información de sus productos de manera eficiente a través de múltiples canales de venta y marketing. Su objetivo principal es garantizar que los datos de los productos sean precisos, consistentes y estén actualizados, independientemente de dónde los consumidores interactúen con ellos: aplicaciones web o móviles, catálogos impresos, entre otros.

Según Akeneo (2024), las características principales de un sistema PIM son:

- Centralización de datos: Proporciona una plataforma única para almacenar y gestionar toda la información de los productos.
- Mejoramiento de datos: Provee herramientas para enriquecer la calidad y presentación de la información.
- Capacidad de distribución: Facilita la sincronización de los datos en diversos canales.
- Gestión de contenidos localizados: Brinda soporte para múltiples regiones e idiomas.
- Colaboración optimizada: Mejora la comunicación entre equipos.
- Automatización: Simplifica tareas repetitivas, como la categorización de productos.

Akeneo (2024) menciona que un sistema PIM es un recurso indispensable para empresas que buscan escalar la gestión del catálogo de productos y mantener una ventaja competitiva en un entorno comercial cada vez más digital.

Sistema de gestión de contenido o CMS

IBM (2024) menciona que un sistema de gestión de contenido, o CMS, es una herramienta de software diseñada para facilitar la creación, gestión, almacenamiento y modificación de contenido digital. Este sistema integral ofrece una plataforma centralizada para almacenar y administrar diversos tipos de contenido, como texto, imágenes, videos, y estructuras de datos complejas, todo ello a través de una interfaz intuitiva que puede personalizarse según las necesidades de la organización.

De acuerdo con IBM (2024), las ventajas de un CMS son las siguientes:

- Control sobre el contenido: Permite a las empresas gestionar de forma autónoma todo su contenido, garantizando coherencia y calidad.
- Flexibilidad y adaptabilidad: La personalización del CMS lo hace adecuado para organizaciones de distintos tamaños y sectores.
- Interfaz intuitiva: Facilita la gestión del contenido sin requerir conocimientos avanzados.

REST API

De acuerdo con AWS.Amazon (2024a), una REST API es una interfaz que permite a dos sistemas informáticos intercambiar información de manera segura a través de internet. Este tipo de API es fundamental para aplicaciones empresariales, ya que facilita la comunicación entre sistemas internos y de terceros para llevar a cabo cualquier tipo de tarea.

Según AWS.Amazon (2024a), los beneficios de las REST API son:

- Escalabilidad. Las REST API optimizan las interacciones cliente – servidor al ser stateless, lo que significa que el servidor no necesita almacenar información de solicitudes pasadas.
- Flexibilidad. La separación completa entre cliente y servidor permite que cada componente evolucione de manera independiente.
- Independencia tecnológica. Las REST API son independientes de los lenguajes de programación o plataformas utilizados.

AWS.Amazon (2024a) menciona que algunos de los métodos de las REST API son:

- GET. Este método se utiliza para acceder a recursos en el servidor a través de una URL específica.
- POST. El método POST sirve para enviar datos al servidor, generalmente para crear nuevos recursos.

GraphQL API

De acuerdo con The GraphQL Foundation (2024), GraphQL es un lenguaje de consultas para API y un entorno de ejecución diseñado para satisfacer esas consultas utilizando los datos existentes de una aplicación. Su objetivo principal es proporcionar una descripción clara y completa de los datos disponibles en la API, permitir que los clientes soliciten exactamente lo que necesitan, sin más ni menos, y facilitar la evolución de las API a lo largo del tiempo.

Según The GraphQL Foundation (2024), las características de GraphQL son:

- Consultas precisas. Los clientes envían consultas específicas para obtener exactamente los datos requeridos, lo que evita tanto el exceso como la carencia de información.
- Eficiencia en la obtención de datos. A diferencia de las REST API, que requieren múltiples solicitudes a diferentes End Points para obtener datos relacionados, GraphQL permite acceder a toda la información necesaria en una sola solicitud.
- Organización basada en tipos y campos. Las GraphQL API están estructuradas en términos de tipos y campos, en lugar de End Points.
- Validación y errores claros. GraphQL utiliza su sistema de tipos para garantizar que las solicitudes sean válidas y ofrecer mensajes de errores claros y útiles.
- Evolución y flexibilidad. Gracias a su diseño adaptable, GraphQL facilita la evolución de las API sin afectar a los clientes existentes, permitiendo incorporar nuevos campos o tipos sin romper la funcionalidad previa.

Serverless

Según AWS.Amazon (2024b), Serverless es un modelo de desarrollo de aplicaciones en el cual se construyen y despliegan aplicaciones utilizando infraestructuras de servidores administradas por terceros, generalmente proveedores de nube o cloud. Aunque las aplicaciones requieren servidores para ejecutarse, en el modelo Serverless el proveedor de nube se encarga de gestionar tareas rutinarias como la provisión, escalado y mantenimiento de la infraestructura.

AWS.Amazon (2024b) menciona que los beneficios de Serverless son los siguientes:

- Aumento en la productividad de los desarrolladores. Al no tener que configurar ni mantener servidores, los equipos de desarrollo pueden enfocarse en la creación de aplicaciones.
- Escalabilidad eficiente. Las aplicaciones Serverless escalan automáticamente en función de la demanda, desde cero hasta niveles máximos de uso. Los desarrolladores no necesitan preocuparse por el escalado al escribir el código.
- Reducción de costos. Solo se paga por los recursos utilizados, como CPU y memoria durante la ejecución del código.

Jest

De acuerdo con The OpenJS Foundation (2024), Jest es un marco de pruebas para JavaScript que se destaca por su simplicidad y eficiencia.

The OpenJS Foundation (2024) menciona que las principales ventajas de Jest son:

- Rápido y seguro. Jest ejecuta las pruebas en paralelo de manera confiable gracias a que asegura que cada prueba tenga un estado global único. Además, prioriza las pruebas que han fallado anteriormente.
- Cobertura. Jest puede recopilar información sobre la cobertura de código de todo el proyecto, incluyendo archivos no probados.
- Simulación o mocking. Jest facilita la simulación de objetos fuera del alcance de las pruebas.

Amazon Web Services o AWS

De acuerdo con AWS.Amazon (2024c), Amazon Web Services, o AWS, es reconocido como uno de los ecosistemas en la nube más completos y adoptados globalmente, ofreciendo una amplia gama de servicios diseñados para satisfacer diversas necesidades tecnológicas.

AWS.Amazon (2024c) menciona que las principales características de AWS son:

- Funcionalidad y diversidad de servicios. AWS proporciona más de 200 servicios completamente funcionales que abarcan desde tecnologías de infraestructura (como cómputo, almacenamiento y bases de datos) hasta tecnologías emergentes (como aprendizaje automático, inteligencia artificial, análisis de datos e internet de las cosas).
- Comunidad y ecosistema. AWS alberga la comunidad más grande y dinámica de clientes y socios a nivel global, incluidos startups, grandes empresas y organismos gubernamentales.
- Seguridad y cumplimiento. AWS se posiciona como uno de los entornos de computación en la nube más seguros, diseñado para cumplir con los requisitos más exigentes de seguridad de industrias como la banca y el sector militar.

- Innovación acelerada. AWS ha sido pionero en tecnologías como Serverless, con el lanzamiento de AWS Lambda en 2014.
- Experiencia operativa. Con más de 17 años de experiencia entregando servicios en la nube, AWS se destaca por su madurez y fiabilidad operativa, lo que lo convierte en una opción confiable para aplicaciones críticas.

API Gateway

AWS.Amazon (2024d) menciona que Amazon API Gateway es un servicio totalmente administrado, diseñado para facilitar la creación, publicación, mantenimiento, monitoreo y seguridad de API a cualquier escala. Este servicio actúa como una puerta de enlace para que las aplicaciones accedan a datos, lógica empresarial o funcionalidades desde los servicios Back – End.

De acuerdo con AWS.Amazon (2024d), algunas características de API Gateway son:

- Compatibilidad con distintos tipos de API. REST API y WebSocket API.
- Escalabilidad. Adapta automáticamente su capacidad para manejar cargas variables.
- Rentabilidad. Su modelo basado en el consumo minimiza los costos para aplicaciones con demandas impredecibles.
- Flexibilidad. Compatible con múltiples arquitecturas, incluyendo Serverless, como AWS Lambda, y servicios Back – End externos.

AWS Secrets Manager

De acuerdo con AWS.Amazon (2024e), AWS Secrets Manager es un servicio que permite gestionar de manera segura contraseñas, claves, certificados, entre otros, que las aplicaciones y servicios necesitan para acceder a recursos protegidos.

Algunos beneficios mencionados por AWS.Amazon (2024e) son:

- Cifrado seguro de los valores almacenados.
- Gestión del acceso a los valores almacenados.
- Rotación automática de los valores almacenados.

- Replicación de los valores almacenados para facilitar la recuperación en caso ocurra alguna incidencia.

Amazon Simple Notification Service o SNS

De acuerdo con AWS.Amazon (2024f), Amazon Simple Notification Service, o SNS, es un servicio completamente administrado que facilita la publicación, el envío y la administración de notificaciones en arquitecturas distribuidas. Es ampliamente utilizado para enviar mensajes entre aplicaciones (A2A) y de aplicaciones a personas (A2P). SNS opera mediante temas o tópicos (topics) que actúan como intermediarios para distribuir mensajes a múltiples suscriptores.

Según AWS.Amazon (2024f), las principales características de SNS son:

- Compatibilidad con múltiples protocolos: Soporta el envío de mensajes mediante HTTP/S, correos electrónicos, SMS, dispositivos móviles (push notifications), Amazon SQS y AWS Lambda, entre otros.
- Escalabilidad y alta disponibilidad: Diseñado para manejar grandes volúmenes de tráfico y asegurar que los mensajes se entreguen de manera confiable.
- Mensajería FIFO y estándar: Permite elegir entre mensajes ordenados y sin duplicación (FIFO) o entregas más rápidas donde el orden no es crítico (estándar).
- Filtrado de mensajes: Los suscriptores pueden definir políticas de filtrado para recibir solo los mensajes relevantes.

Amazon Simple Queue Service o SQS

De acuerdo con AWS.Amazon (2024g), Amazon Simple Queue Service, o SQS, es un servicio de mensajería completamente administrado que permite el desacoplamiento en aplicaciones distribuidas, microservicios y otros sistemas. Con SQS, los desarrolladores pueden enviar, recibir, almacenar y administrar mensajes de manera confiable entre componentes de software, sin preocuparse por perder los mensajes o la disponibilidad de los servicios.

AWS.Amazon (2024g) menciona que algunas características de SQS son:

- Colas estándar. Ofrecen una alta capacidad de procesamiento con un modelo de entrega de "al menos una vez".
- Colas FIFO. Garantizan que los mensajes se "entreguen exactamente una vez" y el procesamiento en el mismo orden en que se envían los mensajes.
- Soporta un alto número de mensajes por segundo y ofrece redundancia almacenando los mensajes en varias zonas de disponibilidad de AWS.
- Ofrece un "Visibility Timeout" configurable, que oculta temporalmente los mensajes de otros consumidores mientras están siendo procesados.
- Dead Letter Queue o DLQ. Es una cola especial donde se almacenan los mensajes fallidos, permitiendo a los desarrolladores analizar los problemas sin perder los datos originales.

AWS Lambda

De acuerdo con AWS.Amazon (2024h), Lambda es un servicio de computación Serverless que permite ejecutar código sin necesidad de aprovisionar o gestionar servidores. Diseñado para simplificar el desarrollo y la escalabilidad, Lambda puede integrarse con más de 200 servicios de AWS y aplicaciones externas, lo que lo convierte en una solución versátil para diversas necesidades.

Según AWS.Amazon (2024h), las características clave de Lambda son:

- Escalabilidad automática. Lambda ajusta automáticamente la capacidad de ejecución en respuesta al tráfico, manejando desde unas pocas solicitudes por hora hasta miles por segundo.
- Basado en eventos. Se puede disparar por eventos de otros servicios de AWS, como S3, DynamoDB o API Gateway, o por eventos personalizados.
- Modelo de precios por uso. Solo se paga por la duración de la ejecución y la cantidad de invocaciones, lo que resulta eficiente para cargas de trabajo variables.
- Compatibilidad con múltiples lenguajes. Lambda admite lenguajes como Python, Node.js, Java, Go y más, permitiendo flexibilidad en el desarrollo.
- Almacenamiento efímero. Cada función tiene almacenamiento temporal (entre 512 MB y 10 GB), útil para datos que solo se necesitan durante la ejecución de la función.

Amazon S3

De acuerdo con AWS.Amazon (2024i), Amazon Simple Storage Service, o Amazon S3, es un servicio de almacenamiento de objetos reconocido por su escalabilidad, disponibilidad de datos, seguridad y rendimiento.

AWS.Amazon (2024i) menciona que S3 está diseñado para tener una durabilidad de 99.999999999%, lo cual permite almacenar y proteger cantidades ilimitadas de datos, siendo adecuado para múltiples casos de uso como data lakes, aplicaciones nativas en la nube, sitios web, respaldos, archivos, y aprendizaje automático. Además, Amazon S3 respalda a millones de clientes a nivel mundial, garantizando almacenamiento confiable y flexible para satisfacer diversas necesidades empresariales y tecnológicas.

AWS ElastiCache para Redis

De acuerdo con AWS.Amazon (2024j), Amazon ElastiCache para Redis es un servicio totalmente administrado que facilita la gestión del caché de aplicaciones y el procesamiento de datos en tiempo real. Su diseño permite mejorar el rendimiento al reducir el tiempo necesario para acceder a los datos en comparación con bases de datos tradicionales, lo que ayuda a las aplicaciones a escalar de manera eficiente.

Algunas características mencionadas por AWS.Amazon (2024j) son:

- Totalmente gestionado. AWS se encarga de las tareas administrativas como el monitoreo y copias de seguridad, lo que permite que los desarrolladores se concentren en el desarrollo de las aplicaciones.
- Escalabilidad y rendimiento. Soporta escalado automático para adaptarse a la demanda de la aplicación.
- Estructuras de datos avanzadas. Redis admite tipos de datos como cadenas, hashes, listas, entre otros, lo que permite almacenar y gestionar datos de manera diversa.

AWS SDK

Según AWS.Amazon (2024k), la librería de AWS SDK para JavaScript facilita la integración de aplicaciones con los servicios de AWS usando JavaScript en diversos entornos

como Node.js. Proporciona bibliotecas que manejan la autenticación, errores y serialización de datos. También es compatible con TypeScript, lo cual optimiza el tamaño de las aplicaciones mediante una arquitectura modular, y soporta la mayoría de los servicios de AWS.

AWS CDK

Según AWS.Amazon (2024l), AWS Cloud Development Kit, o CDK, es una herramienta de desarrollo de infraestructura como código que permite a los desarrolladores definir recursos de infraestructura en lenguajes de programación como TypeScript, JavaScript, Python y Java. Facilita la creación y gestión de recursos de AWS mediante plantillas reutilizables, lo que mejora la productividad y flexibilidad. Con CDK, se puede aprovechar la infraestructura de AWS sin escribir código complejo, permitiendo automatizar la configuración y el despliegue de servicios en la nube.

Amazon CloudWatch

De acuerdo con AWS.Amazon (2024m), CloudWatch es un servicio que proporciona monitoreo y trazabilidad para recursos y aplicaciones de AWS. Permite recopilar, rastrear, y monitorear métricas y logs, establecer alarmas y visualizar la información a través de dashboards. Con CloudWatch, se puede obtener visibilidad en tiempo real sobre el rendimiento y el estado de las aplicaciones y recursos, lo que facilita la toma de decisiones rápidas y la resolución de problemas.

Según AWS.Amazon (2024m), algunos beneficios son:

- Monitoreo continuo. Gestiona métricas y logs para aplicaciones y servicios.
- Alarmas. Configura alertas basadas en métricas específicas.
- Análisis de logs. Facilita la búsqueda y el análisis de logs.
- Dashboards. Visualiza el rendimiento con dashboards personalizables.

AWS Chatbot

De acuerdo con AWS.Amazon (2024n), AWS Chatbot es una herramienta interactiva que facilita el monitoreo y la gestión de recursos de AWS directamente desde canales de

comunicación como Slack. Permite recibir alertas, ejecutar comandos para obtener información, invocar funciones de AWS Lambda, entre otros, lo que mejora la colaboración y la capacidad de respuesta ante cualquier incidencia en tiempo real.

Webhook

De acuerdo con Red Hat (2024), un Webhook es una comunicación ligera y basada en eventos que envía datos automáticamente entre aplicaciones a través de HTTP. Se activa por eventos específicos y automatiza la comunicación entre interfaces de programación de aplicaciones (API). Los webhooks eliminan la necesidad de hacer consultas continuas, son rápidos de configurar, automatizan la transferencia de datos y son ideales para cargas pequeñas. Esta tecnología permite realizar acciones automatizadas cuando ocurre un evento, agilizando los procesos y optimizando los recursos.

Git

De acuerdo con Software Freedom Conservancy (2024), Git es un sistema de control de versiones (VCS) utilizado mayormente para gestionar las versiones del código de diversas soluciones de software.

Según Software Freedom Conservancy (2024), el flujo básico de Git es el siguiente:

- Modificar el código.
- Seleccionar los cambios que se quieren agregar al siguiente commit.
- Crear un commit para guardar permanentemente los cambios.

Jira

De acuerdo con Atlassian (2024), Jira es una herramienta de gestión de proyectos y seguimiento de incidencias ampliamente utilizada, especialmente en equipos que aplican metodologías ágiles basadas en Scrum y Kanban.

Según Atlassian (2024), algunos beneficios de Jira son:

- Planificación y organización eficiente. Permite descomponer proyectos grandes en tareas más pequeñas, crear hitos, mapear dependencias y mantener todo organizado.

- Seguimiento en tiempo real. Con Jira se puede visualizar el progreso de los proyectos a través de diferentes vistas como listas, tableros y backlogs, permitiendo un seguimiento detallado de cada tarea.
- Personalización y flexibilidad. Jira se adapta a las necesidades específicas de los equipos mediante integraciones con otras herramientas (Slack, Google Drive, Figma, etc.), además de ofrecer plantillas personalizables para procesos y tareas.
- Optimización mediante informes e información en tiempo real. Proporciona visibilidad detallada sobre el progreso del proyecto, permite la creación de informes personalizados y facilita la identificación de riesgos.
- Facilita la colaboración. Jira es ideal para equipos de cualquier tamaño y favorece la colaboración mediante la integración con otras herramientas y la centralización de la información en un solo lugar.

CI/CD

De acuerdo con Red Hat (2022), CI/CD son las siglas de Integración Continua (CI) y Distribución Continua (CD), cuyo principal objetivo es agilizar y mejorar el ciclo de vida del desarrollo de software. El uso de CI/CD permite a las empresas reducir errores y fallas de código sin interrumpir el desarrollo constante y las actualizaciones de software.

Según Red Hat (2022), los principales componentes de CI/CD son:

- Integración Continua (CI). Se refiere a la práctica de integrar cambios de código de manera frecuente y automática en un repositorio compartido, lo que facilita la detección temprana de errores.
- Distribución Continua (CD). Es un proceso de dos etapas. Primero, los cambios de código se integran y prueban, y luego se distribuyen.

AWS CodePipeline

Según AWS.Amazon (2024o), AWS CodePipeline es un servicio de integración y entrega continua (CI/CD) que automatiza el proceso de construcción, prueba e implementación de aplicaciones. Facilita el flujo de trabajo al permitir que el código se

nueva de manera automática a través de diversas etapas, como la extracción del código fuente, la compilación, las pruebas y finalmente la implementación.

AWS.Amazon (2024o) menciona que los beneficios de CodePipeline son:

- Automatización del flujo de trabajo. Permite automatizar las fases de desarrollo, lo que mejora la velocidad y la fiabilidad de los despliegues de software.
- Integración continua. Al integrar el código de manera continua y automatizada, facilita la incorporación frecuente de cambios y reduce el riesgo de errores.
- Escalabilidad. Al ser un servicio gestionado, CodePipeline puede escalar según las necesidades del proyecto, lo que lo hace adecuado tanto para pequeñas como grandes aplicaciones.
- Integración con otros servicios de AWS. Se integra perfectamente con herramientas como AWS CodeCommit, AWS CodeBuild, AWS CodeDeploy y otros servicios de AWS, lo que mejora la cohesión en el entorno de desarrollo.
- Ahorro de tiempo. Al automatizar tareas repetitivas, reduce significativamente el tiempo entre las fases de desarrollo, permitiendo lanzamientos de aplicaciones más rápidos y frecuentes.

Slack

Slack Technologies (2024) menciona que Slack es una herramienta de colaboración y mensajería diseñada para facilitar la comunicación dentro de equipos de trabajo. A través de canales dedicados, Slack permite la interacción instantánea entre los miembros del equipo, lo que mejora la eficiencia y el flujo de trabajo. Además, soporta mensajes directos, llamadas de audio y video, y permite compartir documentos, imágenes y otros archivos de manera rápida.

Según Slack Technologies (2024), los beneficios de Slack son:

- Comunicación en tiempo real. Permite conversaciones instantáneas, eliminando la espera de correos electrónicos o reuniones presenciales.
- Acceso fácil a la información. Todos los mensajes y archivos compartidos en Slack son fácilmente accesibles.

- Soporte para equipos remotos. Slack es especialmente útil para equipos distribuidos, permitiendo la colaboración efectiva a distancia.

Product management system: A bridge between the customer and company's business model and strategy for niche companies in oligopoly markets

Para que las empresas puedan prosperar y crecer, necesitan ofrecer productos únicos o procesos internos más eficientes en comparación con sus competidores. La digitalización se presenta como una herramienta clave que puede potenciar su modelo de negocio y estrategia a través de la transformación digital. Este estudio analiza los beneficios de implementar un sistema de gestión de productos digitalizado e integrado con una interfaz en línea para usuarios. Basándose en estudios de caso de tres empresas suecas en los sectores de telecomunicaciones y distribución de energía, se exploraron las ventajas de estos sistemas para mejorar la satisfacción del cliente, la eficiencia operativa y la reducción de costos. Además, se destaca cómo estas plataformas digitales permiten a las empresas responder de manera ágil a las demandas del mercado y a la competencia, consolidándose como un puente estratégico entre los clientes y las operaciones internas de la empresa. (Berggren, 2021)

Los resultados indican que un sistema de gestión de productos digitalizado e integrado es crucial para apoyar las estrategias de negocio de las empresas. Dichos sistemas no solo incrementan la satisfacción del cliente mediante la mejora de la experiencia digital, sino que también optimizan los procesos internos, reduciendo costos y permitiendo a los empleados centrarse en actividades de mayor valor. Además, esta digitalización impulsa la capacidad de las empresas para adaptarse dinámicamente a las necesidades cambiantes de sus clientes y al entorno competitivo, fortaleciendo su modelo de negocio y estrategia a largo plazo. (Berggren, 2021)

Implementación de un sistema de información para la gestión de unidades de manipulación en almacenes de grandes dimensiones

La gestión eficiente de grandes almacenes depende de tres funciones principales: la recepción, el almacenamiento y el despacho de los productos. Para mejorar estos procesos, este trabajo presenta el diseño e implementación de un sistema de información que

automatiza la asignación de espacios para productos y facilita la selección de unidades para su despacho. Mediante algoritmos metaheurísticos y herramientas tecnológicas, se desarrolló un sistema web para monitorear el inventario y una aplicación móvil que permite registrar las entradas y salidas de los productos. Este enfoque busca optimizar la gestión operativa, reducir costos asociados y minimizar errores humanos en el manejo de inventarios, promoviendo la eficiencia en almacenes de grandes dimensiones. (Canasa Mayta, 2022)

La implementación de sistemas de información en almacenes de grandes dimensiones ofrece una solución efectiva a los problemas derivados de la gestión manual de inventarios. Este estudio demostró que el uso de algoritmos para la asignación de espacios y la automatización del registro de productos no solo reduce la complejidad operativa, sino que también mejora la precisión y el control del inventario. Sin embargo, los resultados también señalan la necesidad de considerar factores como la capacitación del personal y la integración tecnológica en los procesos existentes para maximizar los beneficios. (Canasa Mayta, 2022)

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

Esta sección está organizada de acuerdo con el punto 3.2.3.1 donde se describen las etapas del proyecto.

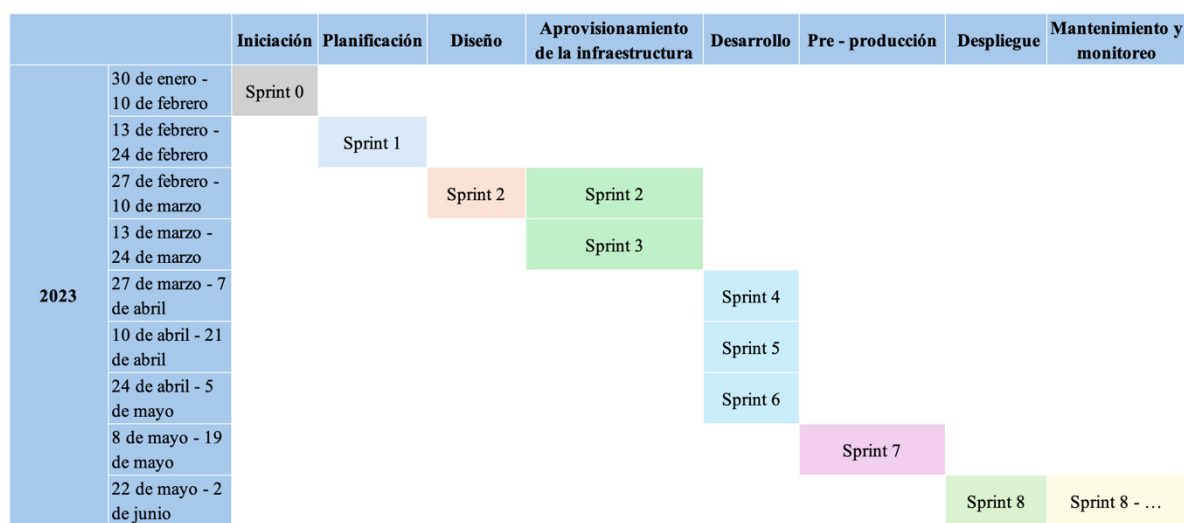
3.2.5.1 Iniciación.

Cronograma del proyecto

Una de las primeras actividades realizadas fue la definición del cronograma del proyecto, el cual se presenta en la Figura 4.

Figura 4

Cronograma del proyecto



Nota. Fuente: Adaptado del proyecto.

Equipo de desarrollo

También se formó el equipo que estuvo encargado de la implementación del proyecto, donde el autor tuvo el rol de desarrollador de software. En la Tabla 7 se visualiza los miembros del equipo.

Tabla 7
Equipo de desarrollo

Rol	Descripción	Cantidad de miembros
Product Manager	Es el encargado de la gestión y ejecución del proyecto, desde el punto de vista del producto.	1
Diseñador UX	Es el encargado de diseñar la experiencia del usuario, tomando en cuenta el flujo de trabajo y los procesos.	1
Content Author	Es el encargado de gestionar la información de la aplicación web de la E-commerce, así como también el catálogo de productos.	2
Technical Lead	Es el encargado de guiar técnicamente al equipo de desarrollo, así como también de la gestión y ejecución del proyecto, desde el punto de vista técnico.	1
Software Developer	Es el encargado del diseño, desarrollo, y mantenimiento de los distintos sistemas de la E-commerce.	3
Tester o QA	Es el encargado de asegurar la calidad de los sistemas mediante la ejecución de pruebas automatizadas.	1

Nota. Fuente: Elaboración propia.

Metodología aplicada al proyecto

Tal como se menciona en el punto 3.2.3.2, se utilizó una metodología ágil basada en el marco de trabajo Scrum para el desarrollo del proyecto, a continuación, en la Tabla 8 se presenta un resumen de Scrum aplicado al proyecto.

Tabla 8
Scrum aplicado al proyecto

Cantidad de sprints	9 sprints
Duración de cada sprint	2 semanas
Cantidad de miembros del equipo	9 miembros

Nota. Fuente: Elaboración propia.

Cada Sprint tuvo una duración de dos semanas, y durante el transcurso se desarrollaron las siguientes ceremonias o eventos.

Sprint planning o planeamiento

Esta reunión se daba antes de dar inicio a un sprint, se revisaba la capacidad de cada miembro del equipo y se asignaban las tareas que se iban a trabajar durante el sprint. Se creaba el backlog del sprint.

Daily meeting o reunión diaria

Esta reunión se daba diariamente, tenía una duración máxima de 15 minutos, aquí cada miembro del equipo explicaba sus progresos del día anterior, mencionaba qué actividades iba a realizar en el día, y comentaba si tenía algún blocker o dificultad en la realización de sus tareas.

Grooming o refinamiento

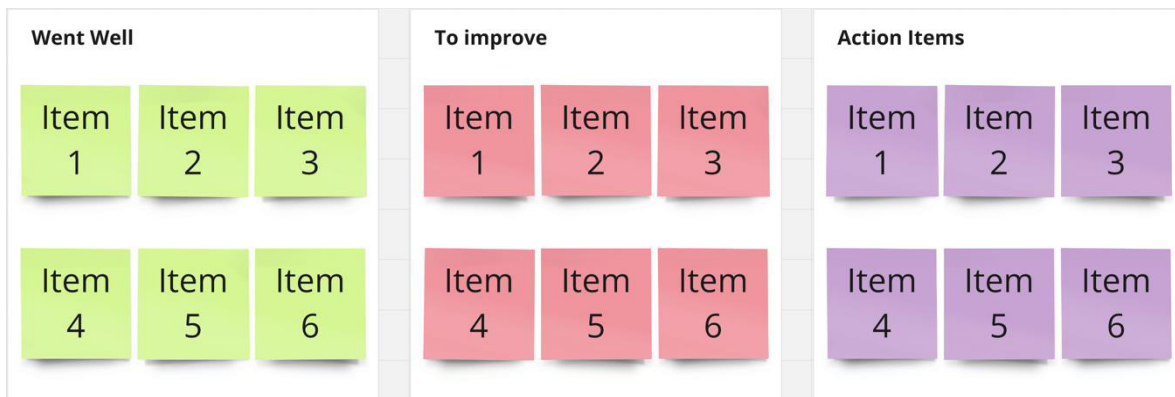
Esta reunión se daba cada dos semanas, es decir una vez por sprint, aquí se presentaban nuevas historias de usuario, se hacía el análisis técnico de cada una y se definía el alcance, así como los criterios de aceptación. Además, se realizaba la estimación basándose en el esfuerzo que tomaría el desarrollo, para la asignación de los puntos a cada historia de usuario se utilizó la escala Fibonacci, siendo tres puntos la cantidad estándar, si una historia de usuario era estimada con mayor puntaje, se creaban subtareas para optimizar el esfuerzo y el trabajo que tomaría implementarla. Las historias de usuario refinadas eran agregadas al backlog del proyecto.

Retrospectivas

Esta reunión se realizaba al finalizar cada sprint, aquí cada miembro del equipo compartía su experiencia durante el sprint, tomando en cuenta lo que se había hecho bien, lo que se podría mejorar, y si era necesario tomar una acción específica para optimizar el flujo de trabajo del equipo. Además, se analizaba los resultados de acciones específicas anteriores con el fin de integrarlas permanentemente o descartarlas. La Figura 5 presenta un ejemplo del resultado al finalizar este evento.

Figura 5

Resultado de retrospectiva



Nota. Fuente: Adaptado del proyecto.

Product backlog y Dashboard

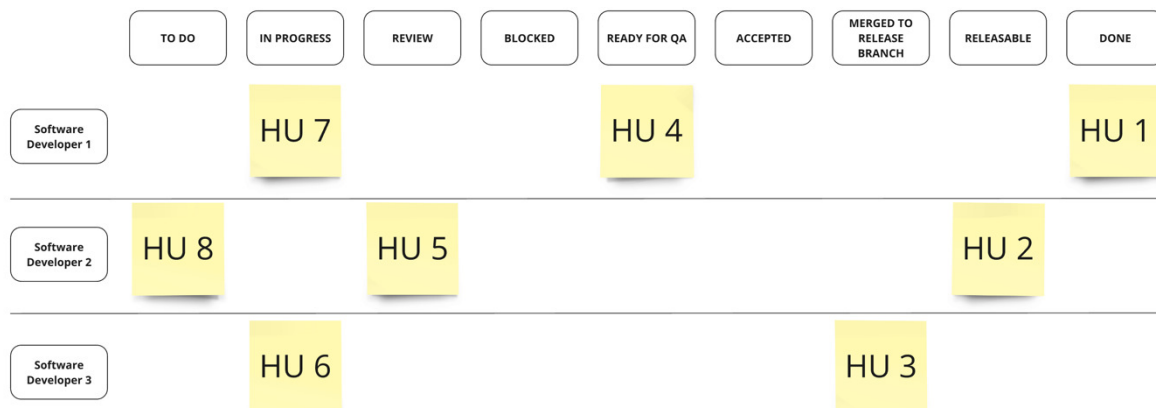
En esta etapa también se realizó la configuración del Product Backlog, en el cual se fueron agregando las historias de usuario del proyecto, y el Dashboard del proyecto. Con este fin, se utilizó la herramienta Jira. El Dashboard del proyecto consistió en los siguientes estados:

- **TO DO:** La Historia de Usuario está a la espera de ser iniciada.
- **IN PROGRESS:** La Historia de Usuario está en desarrollo. Aquí también se incluyen las Historias de Usuario que no fueron validadas correctamente y necesitan de un **REWORK**.
- **REVIEW:** Se ha finalizado el desarrollo y se envió para revisión de código o code review.

- **BLOCKED:** La Historia de Usuario depende de otra(s) o existe algún blocker o dificultad para el desarrollo de esta.
- **READY FOR QA:** El code review ha sido aprobado y el código ha sido enviado al ambiente de desarrollo o dev para que sea validado por el QA, aquí también se incluyen las Historias de Usuario que requieren ser validadas por el diseñador UX.
- **ACCEPTED:** La Historia de Usuario ha sido validada y aceptada.
- **MERGED TO RELEASE BRANCH:** La Historia de Usuario ha sido enviada al ambiente de Pre – producción o staging, donde se realizará otra validación.
- **RELEASABLE:** La Historia de Usuario ha sido validada en el ambiente de Pre – producción y está lista para ser desplegada en el ambiente de Producción.
- **DONE:** La Historia de Usuario ha sido desplegada en Producción correctamente.

En la Figura 6 podemos visualizar el Dashboard del proyecto, el cual ha sido descrito anteriormente.

Figura 6
Dashboard del proyecto



Nota. Fuente: Adaptado del proyecto.

Flujo de trabajo

Para realizar la implementación del Sistema PIM se siguió el flujo de trabajo estándar de la Entidad E-commerce, para lo cual, se utilizó Git con el objetivo de gestionar las versiones del código, y se trabajó con una plataforma privada para almacenar los repositorios

o paquetes de código. Cada repositorio de código tenía los mismos procesos y configuraciones que se mencionan a continuación.

Se tenía una rama o branch principal llamada main, y para el desarrollo de una nueva tarea, historia de usuario, o fix, se creaba una nueva rama a partir de la principal utilizando el ID del ticket en Jira, en la cual se realizaban los cambios necesarios. Se tenía configurado la herramienta Husky que era utilizada al momento de crear los commits, la cual se encargaba de aplicar las pruebas unitarias automáticamente, y si estas eran exitosas, el commit era creado. Posteriormente, se creaba un pull o merge request que era enviado para ser revisado. A continuación, en la Tabla 9, se detallan los requisitos necesarios que debía cumplir cada merge request.

Tabla 9
Requisitos de un merge request

Dry run build	Era un proceso automático que intentaba realizar el build del código con el objetivo de detectar algún error. Resultado requerido: - El proceso debía ser exitoso
Coverage	Era un proceso automático que realizaba las pruebas unitarias del código. Resultado requerido: - Todas las pruebas debían pasar correctamente - Se debía cubrir como mínimo el 90% de las líneas de código nuevas o modificadas
Security code scanner	Era un proceso automático que analizaba el código con el objetivo de encontrar vulnerabilidades. Resultado requerido: - No se debía encontrar vulnerabilidades
Code review	Era un proceso manual en el cual los otros desarrolladores o el líder técnico revisaban los cambios del código. Resultado requerido:

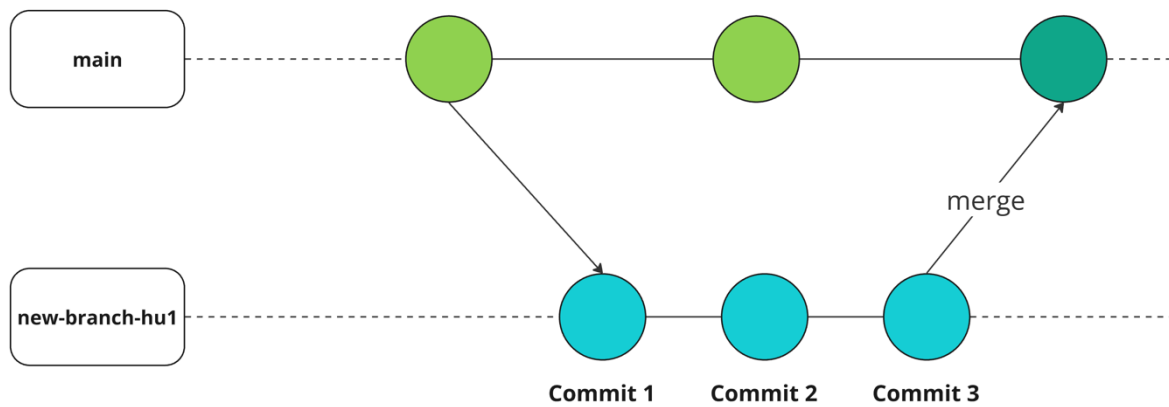
- Tener un mínimo de dos aprobaciones o approvals

Nota. Fuente: Adaptado del proyecto.

Una vez el merge request cumplía todos los requisitos, se procedía a realizar el merge y enviar los cambios del código a la rama principal. Este proceso se puede visualizar en la Figura 7.

Figura 7

Flujo de trabajo con git



Nota. Fuente: Elaboración propia.

3.2.5.2 Planificación.

En esta etapa se realizó la extracción de los requerimientos y se crearon las Historias de Usuario, las cuales fueron agregadas al Product Backlog del proyecto. Se utilizó la plataforma Jira.

A continuación, en la Tabla 10 se presenta la estructura de una Historia de Usuario.

Tabla 10

Estructura de una historia de usuario

ID	ID de la historia
Summary	Nombre de la historia

Reporter	Creador de la historia
Description	Descripción de la historia, con el formato: Como <usuario>, quiero <acción>, para <beneficio>
Fix Version/s	Indica en qué versión del sistema se ha desplegado la historia
Story Points	Estimación de la historia, se utilizó Fibonacci, siendo tres el puntaje estándar
Priority	Prioridad de la historia: Blocker, Critical, Major, Normal, Minor
Status	Indica el estado de la historia
Acceptance Criteria	Requisitos que se deben cumplir para validar la historia
Sub Tasks	En caso una historia sea compleja, se crean subtareas para optimizar el esfuerzo y el trabajo
Assignee	Responsable de la historia
Sprint	Indica en qué sprint se desarrolló la historia

Nota. Fuente: Adaptado del proyecto.

Para definir los requerimientos del sistema, se realizaron reuniones y se mantuvo comunicación constante con los Content Authors o autores de contenido, quienes son los encargados del proceso de gestión del catálogo de productos. Anteriormente, para crear, actualizar, o archivar un producto, utilizaban un archivo en formato “.csv”, cuya estructura se detalla en la Tabla 5. Ellos hacían sus modificaciones que podían ser desde uno hasta cientos de productos y posteriormente, a altas horas de la noche, cargaban el archivo en la Plataforma Externa para publicar sus cambios.

Además, los autores de contenido también son los responsables del contenido de la aplicación web de la E-commerce, es decir, ellos utilizan un Sistema de Gestión de Contenido o CMS para modificar los textos, imágenes, colores, entre otros elementos.

En conclusión, los autores de contenido realizaban dos procesos diferentes en dos plataformas distintas para mantener actualizada la aplicación web de la E-commerce:

- Gestión del catálogo de productos en la Plataforma Externa
- Gestión del contenido en el CMS

La Figura 8 muestra el proceso anterior que se utilizaba para gestionar el catálogo de productos.

Figura 8

Flujo anterior para gestionar el catálogo de productos

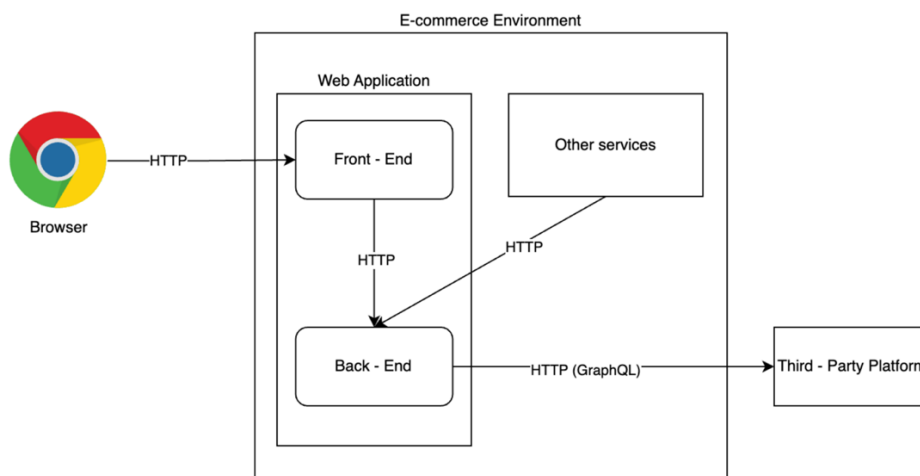


Nota. Fuente: Elaboración propia.

Adicionalmente, al tener el catálogo de productos almacenado en la Plataforma Externa, se tenía una capa de complejidad al momento de querer acceder a esta información, en la Figura 9 se puede visualizar que con el objetivo de mostrar la información en la aplicación web, se tenía que solicitar estos datos a la Plataforma Externa. Y en el caso de otros servicios tales como checkout, delivery, inventario, y procesamiento de órdenes que también necesitan esta información, la complejidad era mayor, ya que por motivos de responsabilidad u ownership, solamente la aplicación web tenía acceso a la Plataforma Externa.

Figura 9

Proceso anterior para acceder al catálogo de productos



Nota. Fuente: Elaboración propia.

En base a las conclusiones de la comunicación con los autores de contenido y el análisis del proceso, se crearon las Historias de Usuario que fueron agregadas al Product Backlog, el cual es presentado en la Tabla 11.

Tabla 11
Product backlog

Etapas	ID	Summary	Sprint
Diseño	HU_01	Definir la estructura de datos	2
	HU_02	Definir la arquitectura del sistema	2
Aprovisionamiento de la infraestructura	HU_03	Configurar el proceso de CI/CD	2
	HU_04	Configurar el CMS	3
	HU_05	Implementar la infraestructura del sistema	3
Desarrollo	HU_06	Lógica de autorización de eventos	4
	HU_07	Lógica de validación de eventos	4
	HU_08	Lógica de procesamiento de eventos	4
	HU_09	Exponer el catálogo de productos	5
	HU_10	Automatizar las actualizaciones masivas	5, 6
	HU_11	Script para la migración de los datos	5, 6
	HU_12	Actualizar el Back – End de la aplicación web	6
Pre – producción	HU_13	Configurar alertas y dashboards	7
	HU_14	Migrar el catálogo de productos	7
	HU_15	Certificar el funcionamiento del sistema	7
Despliegue	HU_16	Desplegar el sistema	8

Nota. Fuente: Adaptado del proyecto.

3.2.5.3 Diseño.

HU_01 Definir la estructura de datos

A continuación, se presenta la estructura de las clases que fueron definidas tomando en cuenta las limitaciones presentadas en la Tabla 5.

En la Tabla 12 se puede visualizar la estructura de la clase Option.

Tabla 12
Clase option

ID	option
Nombre	Option
Descripción	Las options se utilizan para configurar los variants o variaciones.
Atributos	Value: Es el valor de una option. String. Requerido, único.
	Type: Es el tipo de una option. String. Requerido, solo existen los siguientes tipos: Bundle, Color, Power Source.

Nota. Fuente: Adaptado del proyecto.

En la Tabla 13 se presenta la clase Asset Container.

Tabla 13
Clase asset container

ID	assetContainer
Nombre	Asset Container
Descripción	Se utiliza para configurar las imágenes y videos.
Atributos	Title: Es el título de la imagen o video. String. Requerido.
	Alt: Es el atributo alt de la imagen o video. String. Requerido.
	Image: Es el URL de la imagen, la cual se encuentra almacenada en el CDN del CMS. String.
	Video: Es el URL del video, el cual se encuentra almacenado en el CDN del CMS. String.
	Youtube ID: Es el ID de un video de Youtube. String.

Nota. Fuente: Adaptado del proyecto.

A continuación, en la Tabla 14 se muestra la clase Variant.

Tabla 14
Clase variant

ID	variant
Nombre	Variant
Descripción	Es el variant o variación de un producto.
Atributos	Title: Es el título o nombre de una variación. String. Requerido.
	SKU: Es el identificador de una variación. String. Requerido, único, solo acepta mayúsculas y números, debe tener 10 caracteres.
	Description: Es la descripción de una variación, acepta contenido HTML. String.
	Options: Aquí se configuran las options de una variación. Arreglo de Option. Requerido, se pueden configurar hasta tres options, pero de tipos diferentes.
	Assets: Se configuran las imágenes o videos de una variación. Arreglo de Asset Container.
	Price: El precio de una variación. Number. Requerido, debe ser mayor o igual a cero.
	Comparable Price: Se utiliza cuando la variación está en oferta. Number. Debe ser mayor o igual a cero.
	Out of Stock: Indica si hay o no hay stock de una variación. Boolean. Requerido, por defecto es false.
	Tags: Conjunto de etiquetas que ayudan a personalizar una variación. Arreglo de String.

Nota. Fuente: Adaptado del proyecto.

Para finalizar, la Tabla 15 presenta la clase Product.

Tabla 15
Clase product

ID	product
Nombre	Product

Descripción	Corresponde a un producto.
Atributos	Title: Es el título o nombre de un producto. String. Requerido.
	ID: Es el identificador de un producto. String. Requerido, único, solo acepta minúsculas, números, y guiones.
	Description: Es la descripción de un producto, acepta contenido HTML. String.
	Tags: Conjunto de etiquetas que ayudan a personalizar un producto. Arreglo de String.
	Variants: Aquí se configuran las variaciones del producto. Arreglo de Variant. Requerido.
	Assets: Aquí se configuran las imágenes o videos del producto. Arreglo de Asset Container.

Nota. Fuente: Adaptado del proyecto.

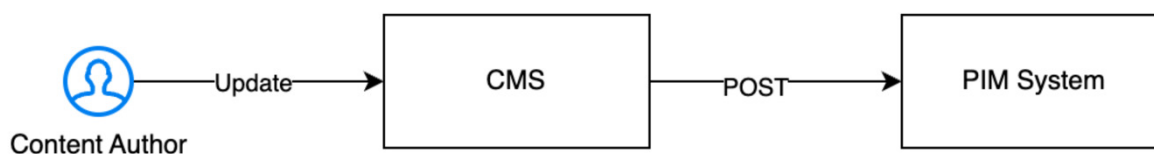
Las estructuras de los datos mencionados anteriormente se utilizaron para configurar el CMS, lo cual se detalla en la HU_04.

HU_02 Definir la arquitectura del sistema

A continuación, en la Figura 10 se presenta el nuevo flujo para la gestión del catálogo de productos, donde se observa que los autores de contenido realizan las actualizaciones en el CMS, y posteriormente, este se comunica con el Sistema PIM para procesar los cambios, lo cual optimiza el proceso en comparación al flujo anterior mostrado en la Figura 8.

Figura 10

Nuevo flujo para gestionar el catálogo de productos



Nota. Fuente: Elaboración propia.

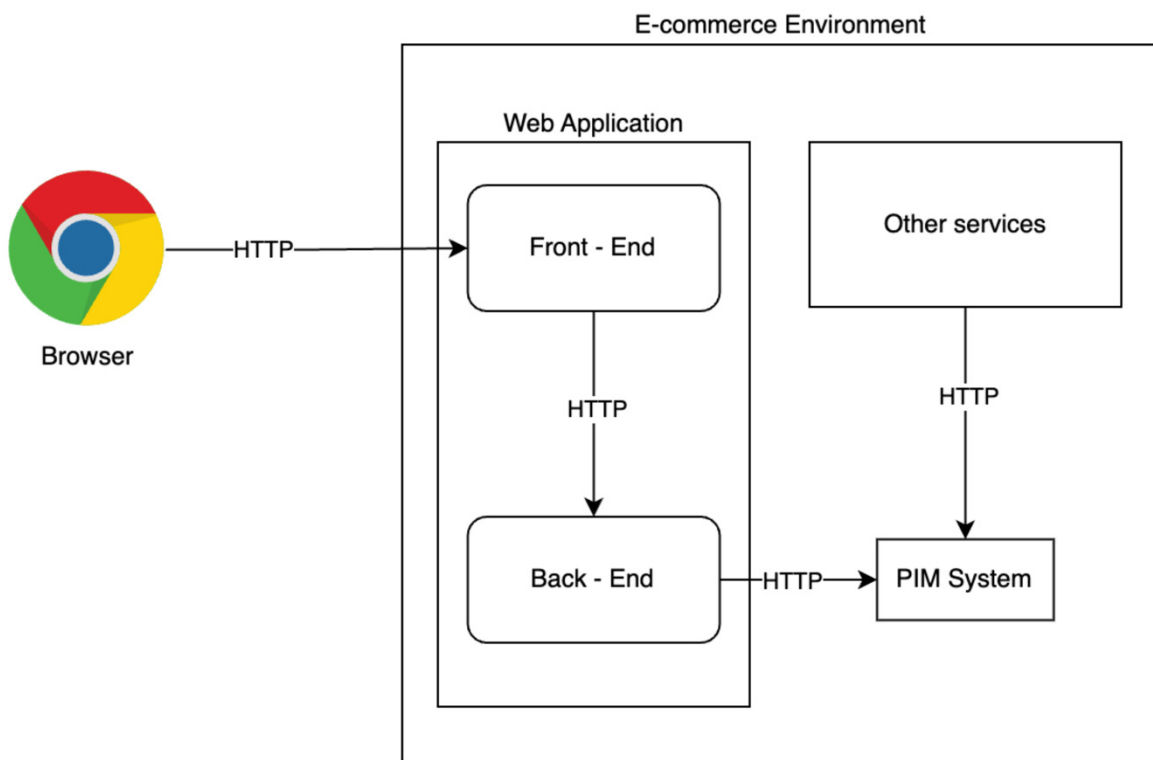
La implementación del Sistema PIM permitió que los autores de contenido utilicen solamente el CMS para mantener actualizada la aplicación web de la E-commerce:

- El catálogo de productos
- El contenido de la aplicación web

La Figura 11 presenta el nuevo proceso para acceder a la información del catálogo de productos, donde se puede ver que la aplicación web y otros servicios que requieren de estos datos ahora se comunican directamente con el Sistema PIM, lo cual eliminó la complejidad que existía en comparación con el proceso anterior presentado en la Figura 9.

Figura 11

Nuevo proceso para acceder al catálogo de productos

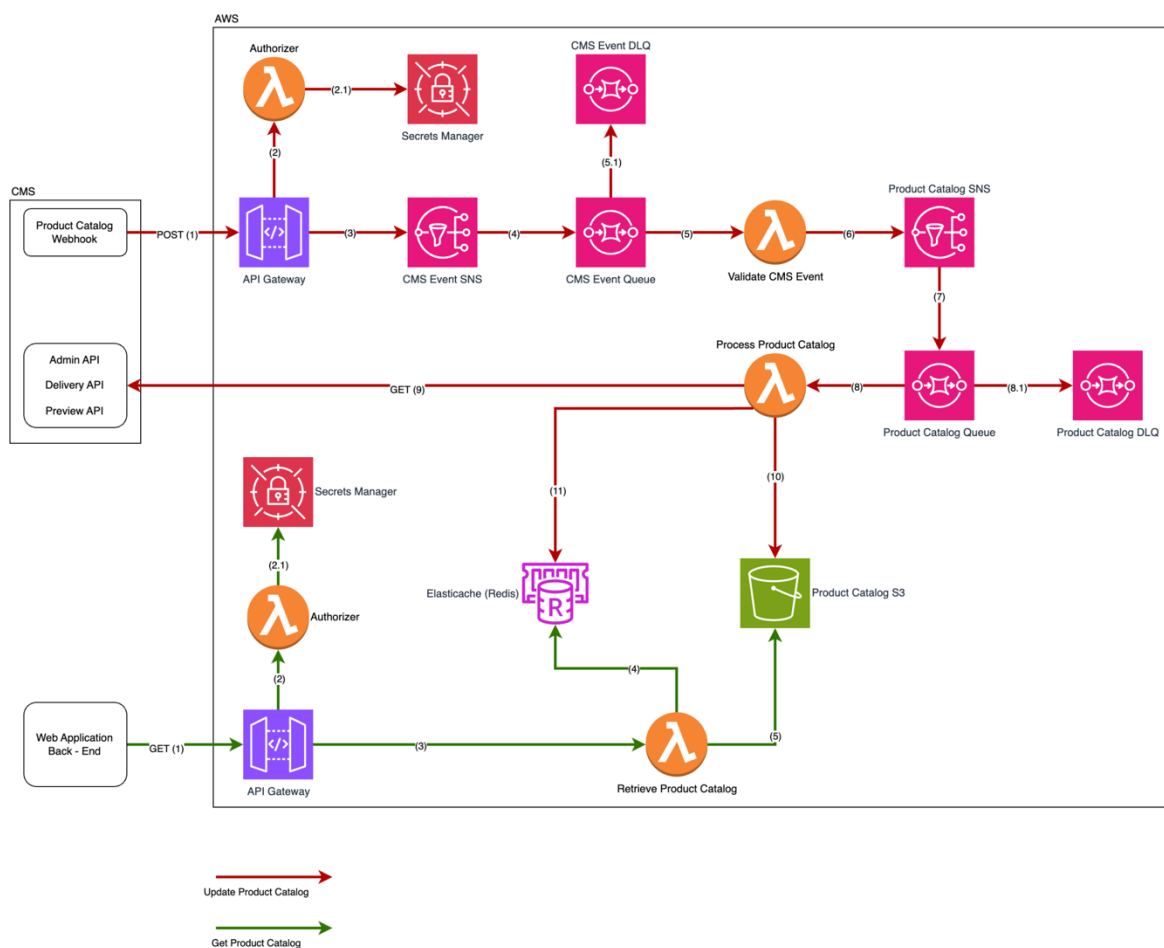


Nota. Fuente: Elaboración propia.

Para diseñar la arquitectura del Sistema PIM, se utilizaron distintos servicios de AWS, lo cual permitió lograr una alta escalabilidad y desacoplamiento, así como también una alta

tolerancia a fallos y recuperación de errores. Además, esto optimizó los costos al tratarse de una infraestructura Serverless. La Figura 12 presenta la arquitectura del Sistema PIM.

Figura 12
Arquitectura del Sistema PIM



Nota. Fuente: Adaptado del proyecto.

La arquitectura del Sistema PIM tiene dos flujos o procesos:

- Actualizar el catálogo de productos
- Obtener la información del catálogo de productos

Actualizar el catálogo de productos

El flujo inicia cuando los autores de contenido actualizan la información del catálogo de productos en el CMS. Por cada producto o variación actualizada, se ha configurado un

webhook que emite una petición o request POST al Sistema PIM, el cual lo recibe mediante API Gateway (1).

Para garantizar que los eventos recibidos son legítimos, se ha configurado un Lambda que se encarga de gestionar los accesos al procesamiento del evento (2), mediante el análisis de un token que es validado con una llave o key almacenada en Secrets Manager (2.1). Si la validación es correcta, se retorna una respuesta exitosa al CMS haciéndole conocer que se ha recibido el evento, y se inicia el procesamiento de manera asíncrona.

El evento es publicado en un SNS topic “cms-event-topic” (3), al cual se ha suscrito una cola SQS donde se reciben todos los eventos enviados por el CMS (4). Los eventos de la cola son procesados por un Lambda que se encarga de validar la integridad de los datos (5), para lo cual se ha configurado el número máximo de reintentos a 10, es decir, si el evento falla en ser validado 10 veces, será enviado a una cola de eventos fallidos o DLQ (5.1) para su posterior revisión.

Si se ha confirmado que los datos del evento son válidos, este es publicado en un SNS topic “product-catalog-topic” (6), para el cual se ha suscrito una cola SQS donde se reciben todos los eventos validados (7). Estos eventos son procesados por un Lambda (8) que se encarga de construir la estructura de datos del producto mediante la comunicación con los API del CMS (9), una vez se tiene la información, esta es almacenada en formato JSON en un Bucket en S3 (10), el cual nos permite acceder a las diferentes versiones de un producto. Además, se actualiza la información en Elasticache (Redis) (11). En caso el evento no sea procesado después de 10 intentos, será enviado a otra cola de eventos fallidos o DLQ (8.1) para que sea revisado.

Obtener la información del catálogo de productos

Este flujo inicia cuando la aplicación web de la E-commerce u otros servicios que requieren la información del catálogo de productos realizan una petición o request GET al Sistema PIM el cual lo recibe mediante API Gateway (1). De la misma manera que el flujo anterior, se tiene un Lambda para gestionar los accesos a la información (2) y (2.1).

En caso la autorización sea correcta, otro Lambda se encarga de obtener la información del catálogo de productos (3), ya sea de Elasticache (Redis) (4) o de S3 (5), para luego retornar los datos en la respuesta de la petición.

Esta arquitectura combina servicios de AWS altamente escalables para gestionar tanto la interacción en tiempo real, obtener el catálogo de productos, como el procesamiento asíncrono para actualizar la información.

- **API Gateway.** Proporciona una interfaz altamente escalable para que tanto el CMS, la aplicación web de la E-commerce, u otros servicios interactúen con el Sistema PIM mediante peticiones o requests HTTP/REST. Además, garantiza la seguridad al configurar Lambdas que se encargan de gestionar los accesos al procesamiento de las peticiones.
- **Secrets Manager.** Permite el almacenamiento seguro de contraseñas, llaves o keys, que luego pueden ser utilizados para validar el acceso a distintos recursos, en este caso, a los procesos del Sistema PIM.
- **SNS.** Permite distribuir los eventos rápida y eficientemente a varios suscriptores, en este caso a las colas SQS.
- **SQS.** Asegura que los eventos se almacenen en una cola, garantizando el procesamiento asíncrono.
- **Lambda.** Al ser Serverless, permite el escalamiento automático según la cantidad de eventos de las colas SQS, lo cual garantiza que los eventos siempre serán procesados.
- **DLQ.** Nos garantiza que los eventos fallidos no se pierdan, y puedan ser revisados posteriormente, así como también reprocesados. Esto mejora la tolerancia a fallos del Sistema PIM.
- **Elasticache (Redis).** Garantiza un acceso rápido al catálogo de productos, optimizando los tiempos de respuesta.
- **S3.** Proporciona un almacenamiento duradero y rentable, además que permite el acceso a las versiones anteriores de cada producto.

3.2.5.4 Aprovisionamiento de la infraestructura.

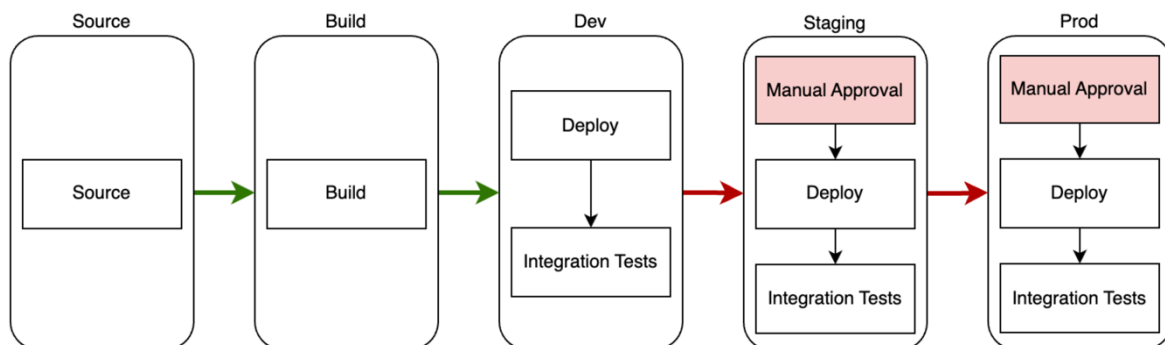
Para la implementación del Sistema PIM se configuraron tres ambientes o environments:

- Desarrollo o dev. Este ambiente se utilizó para probar las funcionalidades del Sistema PIM.
- Pre – producción o staging. Este ambiente se utiliza para validar los cambios realizados por los autores de contenido antes de ser publicados en el ambiente de producción.
- Producción o prod. Este es el ambiente utilizado para gestionar el catálogo de productos, cada cambio es visualizado en la aplicación web de la E-commerce.

HU_03 Configurar el proceso de CI/CD

Cada repositorio de código tenía su propio pipeline para garantizar la integración y despliegue continuos, para lo cual se utilizó un servicio de AWS llamado CodePipeline. La Figura 13 presenta las etapas o stages de cada pipeline, así como también las acciones o actions de cada stage.

Figura 13
Estructura de un pipeline



Nota. Fuente: Adaptado del proyecto.

Tal como se presenta en la Figura 7, cada repositorio de código tenía una rama principal llamada main, y para añadir los cambios de cada Historia de Usuario se lo hacía

mediante un merge request. Cada vez que se añadían cambios a la rama main, el pipeline empezaba a correr automáticamente, siendo Source la primera etapa o Stage.

Una vez se recibían los nuevos cambios, el siguiente paso era la etapa Build, donde se procesaba el código actualizado para obtener los ejecutables que estaban listos a ser desplegados.

La siguiente etapa consistía en desplegar los cambios en el ambiente de desarrollo o dev, una vez los cambios eran desplegados, se aplicaban pruebas funcionales y de integración para validar el correcto funcionamiento del sistema.

En la Tabla 10 se menciona que cada historia de usuario tenía asignado un Fix Version, este atributo indicaba en que versión del sistema se han desplegado los cambios correspondientes a la historia. Para la siguiente etapa de pre – producción o staging, se asignaban los Fix Versions, y se habilitaba la transición manualmente, de esta manera se aseguraba que solamente se desplieguen los cambios que estaban listos. Además, para iniciar esta etapa, se tenía que aprobar manualmente, posteriormente los cambios eran desplegados y se aplicaban las pruebas correspondientes. Al finalizar, se tenía que deshabilitar la transición.

Finalmente, se tenía la etapa de producción o prod, de manera similar a la etapa anterior, se tenía que habilitar la transición y aprobar el despliegue manualmente, una vez los cambios eran desplegados, se aplicaban las pruebas correspondientes. Al finalizar, se tenía que deshabilitar la transición.

HU_04 Configurar el CMS

Se configuraron los modelos de datos en el CMS, para lo cual se utilizaron las estructuras definidas en la HU_01.

A continuación, en las Figuras 14, 15, 16, y 17 se presentan ejemplos de entidades correspondientes a Option, Asset Container, Variant, y Product.

Figura 14
Ejemplo de option

Value (required)

Green

5 characters

Maximum 256 characters

Type (required)

Color



Nota. Fuente: Adaptado del proyecto.

Figura 15
Ejemplo de asset container

Title (required)

Green Variant Image

19 characters

Maximum 256 characters

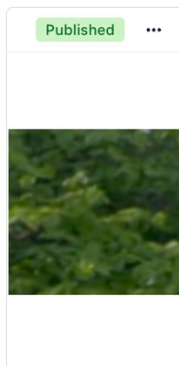
Alt (required)

Green Variant Image

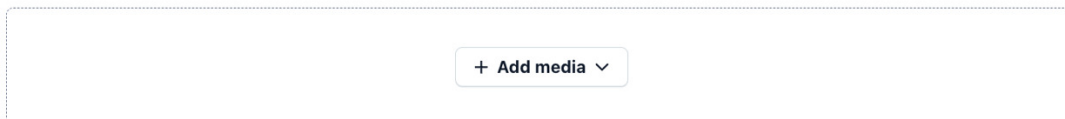
19 characters

Maximum 256 characters

Image



Video



Youtube ID

0 characters

Maximum 256 characters

Nota. Fuente: Adaptado del proyecto.

Figura 16
Ejemplo de variant

Title (required)

New Product 1 - Green

21 characters Maximum 256 characters

SKU (required)

GRNVRNT123

10 characters Maximum 256 characters

Description

Editor Preview

H B I " : : : Insert media ✕

This is a variant description

Format your text like a pro with the [markdown cheatsheet](#).

29 characters Maximum 50000 characters

Options (required)

Option Published ...

Green

+ Add content ▾

Assets

Asset Container Published ...

Green Variant Image

+ Add content ▾

Price (required)

99.99

Comparable Price

Out of Stock (required)

☐ Yes ☒ No [Clear](#)

Tags

Type the value and hit enter

tag_4 ✕ tag_5 ✕

Nota. Fuente: Adaptado del proyecto.

Figura 17
Ejemplo de product

Title (required)

New Product 1

13 characters

Maximum 256 characters

ID (required)

new-product-1

13 characters

Maximum 256 characters

Description

Editor

Preview

H

B

I

”

≡

≡

🔗

⋮

Insert media

✕

This is a new product description

Format your text like a pro with the [markdown cheatsheet](#).

33 characters

Maximum 50000 characters

Tags

Type the value and hit enter

tag_1

✕

tag_2

✕

tag_3

✕

Variants (required)

Variant

Published

⋮

⋮

New Product 1 - Green

+ Add content

▼

Assets

Asset Container

Published

⋮

⋮

New Product 1 Image

+ Add content

▼

Nota. Fuente: Adaptado del proyecto.

Cada entidad en el CMS podía tener los siguientes estados mostrados en la Tabla 16.

Tabla 16
Estados de las entidades en el CMS

Estado	Descripción	Se puede acceder con	Posibles futuros estados
Draft	La entidad no ha sido publicada.	- Admin API - Preview API	- Published - Archived
Changed	La entidad ha sido publicada, pero ha sido modificada.	- Admin API - Preview API - Delivery API (solo se obtendrán los últimos cambios publicados)	- Draft - Published
Published	La entidad ha sido publicada y no ha sido modificada.	- Admin API - Preview API - Delivery API	- Draft - Changed
Archived	La entidad ha sido archivada.	- Admin API	- Draft

Nota. Fuente: Adaptado del proyecto.

Para acceder a la información del catálogo de productos, el CMS tenía tres API:

- Admin API. Lectura y escritura de los datos. Se utilizó en el script para la migración de la información en la HU_11.
- Preview API. Solo lectura de los datos. Se tenía acceso a los cambios publicados y no publicados. Se utilizó en los ambientes de pre – producción y desarrollo.
- Delivery API. Solo lectura de los datos. Solamente se tenía acceso a los cambios publicados. Se utilizó en el ambiente de producción.

En el CMS también se configuraron dos ambientes o environments:

- Producción o prod. Los autores de contenido utilizan este ambiente para la gestión del catálogo de productos.
- Desarrollo o dev. Este ambiente se utilizó para probar las funcionalidades del Sistema PIM.

Además, se crearon y configuraron los webhooks, los cuales son detallados en la Tabla 17.

Tabla 17
Configuración de los webhooks

product-catalog-prod	- Método: POST - Eventos: publish, unpublish - Clases: product, variant - Ambiente de CMS: prod - Ambiente de Sistema PIM: prod - CMS API: Delivery API
product-catalog-staging	- Método: POST - Eventos: save, autosave, archive, unarchive, publish, unpublish - Clases: product, variant - Ambiente de CMS: prod - Ambiente de Sistema PIM: staging - CMS API: Preview API
product-catalog-dev	- Método: POST - Eventos: save, autosave, archive, unarchive, publish, unpublish - Clases: product, variant - Ambiente de CMS: dev - Ambiente de Sistema PIM: dev - CMS API: Preview API

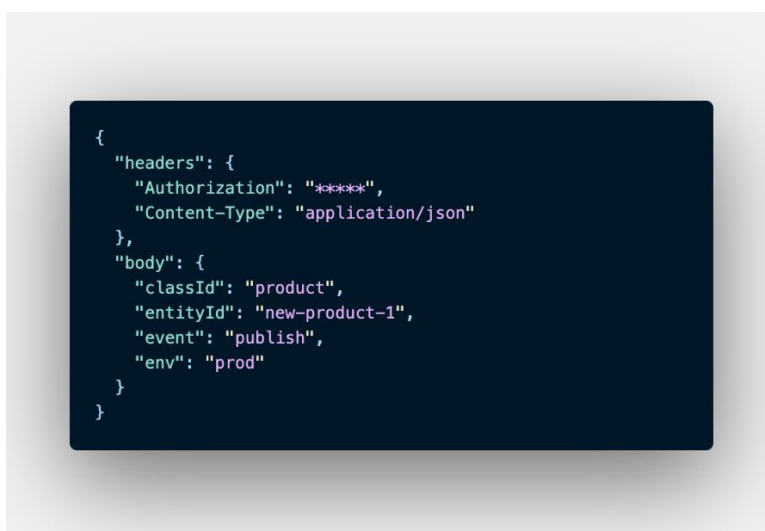
Nota. Fuente: Elaboración propia.

Como se observa en la Tabla 17, los webhooks correspondientes a los ambientes de producción y pre – producción procesan los eventos del ambiente de producción del CMS, la diferencia consiste en que el webhook de producción solamente procesa los eventos de publicación y despublicación de entidades, en cambio, el webhook de pre – producción también procesa cuando hay algún cambio que no ha sido publicado, esto permitió la validación de los cambios antes de ser publicados en producción.

A continuación, en la Figura 18 se presenta la estructura de la petición o request POST que se hace al Sistema PIM mediante los webhooks.

Figura 18

Estructura de la petición POST



```
{
  "headers": {
    "Authorization": "*****",
    "Content-Type": "application/json"
  },
  "body": {
    "classId": "product",
    "entityId": "new-product-1",
    "event": "publish",
    "env": "prod"
  }
}
```

Nota. Fuente: Adaptado del proyecto.

Cada petición o request POST tiene el header de Authorization, en el cual se envía un token que es validado en el Sistema PIM. En el body se envía el classId lo cual indica si el evento corresponde a un producto o a una variación, también se envía el entityId lo cual indica a que entidad corresponde el evento. También se envía el event o evento, así como el ambiente o env.

HU_05 Implementar la infraestructura del sistema

Para implementar la infraestructura del Sistema PIM se utilizó la librería de AWS CDK, la cual permitió construir la infraestructura mediante código, de esta manera, en caso sea necesario algún cambio, solamente se debe modificar el código.

A continuación, en la Tabla 18 se presenta la configuración de los servicios de AWS utilizados en la infraestructura del Sistema PIM.

Tabla 18

Configuración de los servicios de AWS

Servicio	Detalle
AWS Secrets Manager	- Se utilizó para almacenar una llave o key que sirve para validar las peticiones o requests recibidas por el Sistema PIM. - El nombre de la llave fue ProductCatalogSecret.
AWS API Gateway	- Se utilizó para configurar el REST API para el Sistema PIM. - POST: /product-catalog, recibe las peticiones del CMS. - GET: /product-catalog/:productId, se utiliza para obtener la información de los productos. - Se configuró un Lambda authorizer que se encarga de validar las peticiones.
AWS SNS	- Se utilizó para distribuir los eventos a través del Sistema PIM. - “cms-event-topic” para los eventos enviados por el CMS. - “product-catalog-topic” para los eventos validados y que serán procesados por el Sistema PIM.
AWS SQS	- Se utilizó para recibir los eventos enviados a través de los temas o topics de SNS y almacenarlos en una cola.

	- Visibility Timeout: 30 segundos, cuando un evento está siendo procesado, se hace invisible en la cola por 30 segundos. - Max Receive Count: 10, el número máximo de reintentos es de 10, luego los eventos son enviados a un DLQ. - También se configuraron DLQs para que los eventos fallidos puedan ser revisados y reprocesados, se retenían los eventos hasta 14 días, Retention Period: 14 días.
AWS Lambda	- Se utilizaron diferentes Lambdas, los cuales fueron utilizados para validar las peticiones y eventos, así como también procesar y obtener el catálogo de productos. - Los Lambdas utilizados para procesar el catálogo de productos podían procesar hasta 10 eventos, Batch Size: 10, y tenían una ventana de 10 segundos para iniciar el procesamiento, Max Batching Window: 10 segundos.
AWS S3	- Se utilizó para almacenar la información del catálogo de productos. - Se habilitó el versionado, es decir, se tiene acceso a las versiones anteriores de cada producto, Versioned: true.
AWS ElastiCache	- Se utilizó para configurar Redis como almacenamiento de memoria rápida o caché, lo cual optimiza los tiempos de respuesta al obtener la información del catálogo de productos.

Nota. Fuente: Elaboración propia.

3.2.5.5 Desarrollo.

HU_06 Lógica de autorización de eventos

Para implementar el código del Lambda encargado de autorizar el procesamiento de los eventos o peticiones se utilizó TypeScript y la librería de AWS Software Development Kit (SDK), además cada porción de código tiene sus respectivas pruebas unitarias o unit tests

para lo cual se utilizó Jest. En este Lambda, se analiza el parámetro o header Authorization enviado en el evento, el cual contiene una clave o token que es validada por la llave o key almacenada en AWS Secrets Manager. En caso la validación sea correcta, se da acceso al procesamiento del evento, o si no, este es denegado o rechazado. A continuación, en la Figura 19, se presenta un fragmento del código descrito.

Figura 19

Lógica de autorización de eventos

```
export const handler = async (
  event: APIGatewayAuthorizerEvent
): Promise<APIGatewayAuthorizerResult> => {
  const authorizationToken = event.headers?.Authorization;

  if (!authorizationToken) throw new Error('No authorization token');

  try {
    const secret = (
      await secretsManager.getSecretValue({ SecretId: SECRET_NAME }).promise()
    ).SecretString;

    const isValid = await validate(authorizationToken, secret);

    if (isValid) {
      return generatePolicy('user', 'Allow', event.methodArn);
    } else {
      return generatePolicy('user', 'Deny', event.methodArn);
    }
  } catch (error) {
    throw new Error(`Authorization error: ${error.message}`);
  }
};
```

Nota. Fuente: Adaptado del proyecto.

HU_07 Lógica de validación de eventos

De igual manera que el Lambda anterior, para la implementación del código encargado de validar el contenido o el body de los eventos, se utilizó TypeScript y AWS SDK, además se utilizó Jest para las pruebas unitarias. Este Lambda analiza cada evento y valida que la información esté completa y correcta, tal cual se presentó en la Figura 18. En caso la

validación sea positiva, se publica el evento en un tema o topic de SNS para que sea procesado, o si no, después de un máximo de 10 reintentos, será enviado a su respectivo DLQ. En la Figura 20 se puede visualizar un fragmento del código correspondiente a este Lambda.

Figura 20

Lógica de validación de eventos

```
export const handler: SQSHandler = async (event: SQSEvent) => {
  for (const record of event.Records) {
    const cmsEvent: CMSEvent = JSON.parse(record.body);

    const isValid = validateCMSEvent(cmsEvent);

    if (!isValid) {
      throw new Error('Invalid message structure');
    }

    const params = {
      Message: JSON.stringify(cmsEvent),
      TopicArn: TOPIC_ARN,
    };

    try {
      await sns.publish(params).promise();
    } catch (error) {
      throw error;
    }
  }
};
```

Nota. Fuente: Adaptado del proyecto.

HU_08 Lógica de procesamiento de eventos

Este Lambda se encarga de construir la información actualizada de los productos, para la implementación se utilizó TypeScript, AWS SDK, y los API del CMS, Delivery o Preview dependiendo del ambiente o env, así como también Jest para las pruebas unitarias. Primero

se construye el objeto que contiene la información del producto mediante peticiones utilizando los API del CMS, una vez el objeto está listo, se procede a guardarlo en un Bucket en S3 en formato JSON, además se actualiza el caché en ElastiCache (Redis), cada registro en el caché tiene un tiempo de vida o ttl de una hora, es decir, después de una hora el registro es eliminado. En caso haya algún error durante el procesamiento del evento, se reintentará hasta un máximo de 10 veces, y después será enviado a su respectivo DLQ para su revisión. La Figura 21 presenta un fragmento del código descrito.

Figura 21

Lógica de procesamiento de eventos

```
export const handler: SQSHandler = async (event: SQSEvent) => {
  for (const record of event.Records) {
    const cmsEvent: CMSEvent = JSON.parse(record.body);

    try {
      const product: Product = await buildProduct(cmsEvent.entityId);

      const s3Key = `product-catalog/${product.id}.json`;

      await s3
        .putObject({
          Bucket: BUCKET_NAME,
          Key: s3Key,
          Body: JSON.stringify(product),
          ContentType: 'application/json',
        })
        .promise();

      await redisClient.set(product.id, JSON.stringify(product));
    } catch (error) {
      throw error;
    }
  }
};
```

Nota. Fuente: Adaptado del proyecto.

HU_09 Exponer el catálogo de productos

Para acceder a la información del catálogo de productos se implementó un Lambda, para lo cual se utilizó TypeScript, AWS SDK, y Jest para las pruebas unitarias. Con el objetivo de obtener los datos de un producto, se requiere su identificador o ID, el cual debe ser enviado como un parámetro en la petición. Primero se realiza una búsqueda en el caché, en caso exista la información, se la retorna en la respuesta de la petición. En cambio, si no existe en el caché, se realiza la búsqueda en el Bucket de S3 y se actualiza el caché para futuras búsquedas. Cada registro en el caché tiene un tiempo de vida o ttl de una hora, es decir, luego de una hora el registro es eliminado. A continuación, la Figura 22 muestra un fragmento de la lógica descrita.

Figura 22

Lógica para acceder al catálogo de productos

```
export const handler: APIGatewayProxyHandler = async (event) => {
  const productId = event.pathParameters?.productId;

  if (!productId) {
    return { statusCode: 400, body: 'ProductId is required' };
  }

  try {
    const cachedData = await redisClient.get(productId);

    if (cachedData) {
      return { statusCode: 200, body: cachedData };
    }

    const s3Key = `product-catalog/${productId}.json`;

    const productData = (
      await s3.getObject({ Bucket: BUCKET_NAME, Key: s3Key }).promise()
    ).Body!.toString();

    await redisClient.set(productId, productData);

    return { statusCode: 200, body: productData };
  } catch (error) {
    if (error.code === 'NoSuchKey') {
      return { statusCode: 404, body: 'Product not found' };
    }

    throw new Error(`Error fetching product ${productId}: ${error.message}`);
  }
};
```

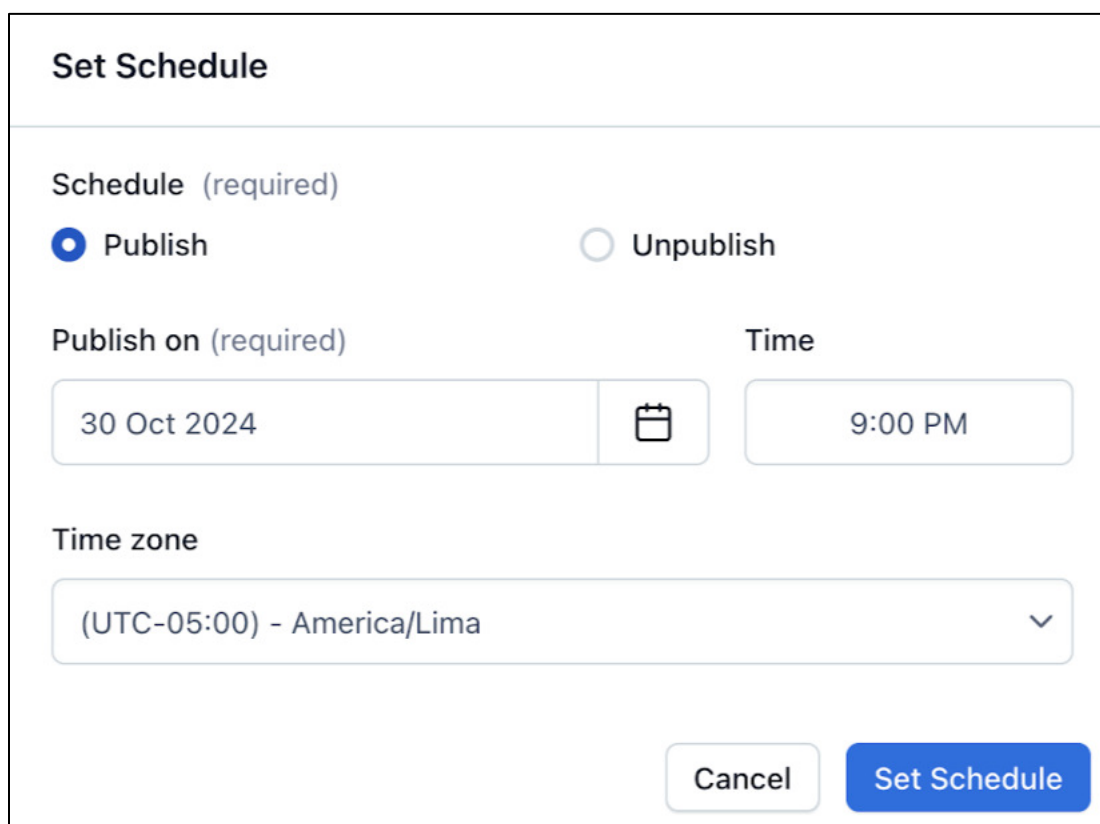
Nota. Fuente: Adaptado del proyecto.

HU_10 Automatizar las actualizaciones masivas

Con el objetivo de optimizar el proceso de actualizaciones masivas del catálogo de productos, se implementó un módulo el cual fue integrado con el CMS mediante la configuración de una aplicación personalizada o custom app. Para esto se utilizó React. Este módulo permite que los autores de contenido configuren releases, a los cuales pueden agregar entidades, ya sea productos o variaciones, para que sean publicadas automáticamente y así se actualice la información de todas las entidades correspondientes. En la Figura 23 se puede visualizar que un release puede ser configurado para publicar o despublicar entidades, así como también, se puede configurar la fecha y hora en la cual se quiere aplicar estos cambios de forma automática.

Figura 23

Configuración de un release



The image shows a web form titled "Set Schedule". It contains the following elements:

- Schedule (required)**: Two radio buttons, "Publish" (selected) and "Unpublish".
- Publish on (required)**: A date input field showing "30 Oct 2024" with a calendar icon to its right.
- Time**: A time input field showing "9:00 PM".
- Time zone**: A dropdown menu showing "(UTC-05:00) - America/Lima" with a downward arrow.
- Buttons**: "Cancel" and "Set Schedule" (highlighted in blue).

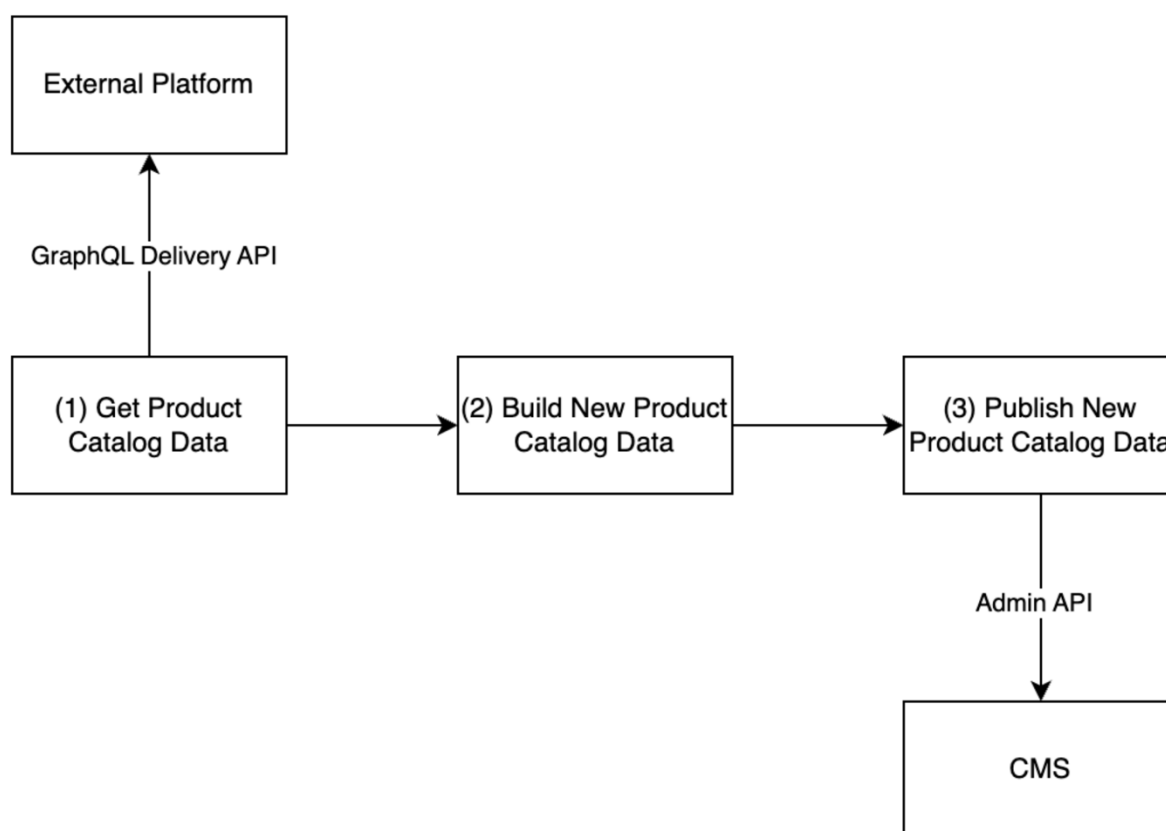
Nota. Fuente: Adaptado del proyecto.

HU_11 Script para la migración de los datos

Para la migración del catálogo de productos se implementó un script que consiste en tres etapas, para lo cual se utilizó TypeScript y Node.js. La primera etapa se encarga de extraer los datos de la plataforma externa mediante el uso del GraphQL Delivery API, el cual permitió obtener la última versión publicada del catálogo de productos. La segunda etapa procesa la información y la transforma basándose en el nuevo modelo de datos definido en la HU_01. Por último, la tercera etapa se encarga de publicar el catálogo de productos en el CMS, para lo cual se utilizó el Admin API. La Figura 24 muestra el flujo del script implementado.

Figura 24

Script para la migración de los datos



Nota. Fuente: Elaboración propia.

HU_12 Actualizar el Back – End de la aplicación web

Debido a que la información del catálogo de productos ahora corresponde al Sistema PIM, fue necesario actualizar el código para acceder a los datos de los productos en el Back – End de la aplicación web de la E-commerce, para lo cual se implementó un Feature Flag, el cual funciona como un interruptor, en caso esté encendido o habilitado, se obtienen los datos del Sistema PIM, o si está apagado o deshabilitado, se obtiene la información de la plataforma externa, esto permitió realizar el despliegue del Sistema PIM de manera segura. La Figura 25 presenta el fragmento de código descrito.

Figura 25

Actualización del Back - End

```
async function getProductCatalogData(  
  isPIMSystemEnabled: boolean  
) : Promise<Product> {  
  try {  
    if (isPIMSystemEnabled) {  
      return await getPIMData();  
    } else {  
      return await getDataFromExternalPlatform();  
    }  
  } catch (error) {  
    throw new Error('Error fetching product catalog data');  
  }  
}
```

Nota. Fuente: Adaptado del proyecto.

3.2.5.6 Pre – producción.

HU_13 Configurar alertas y dashboards

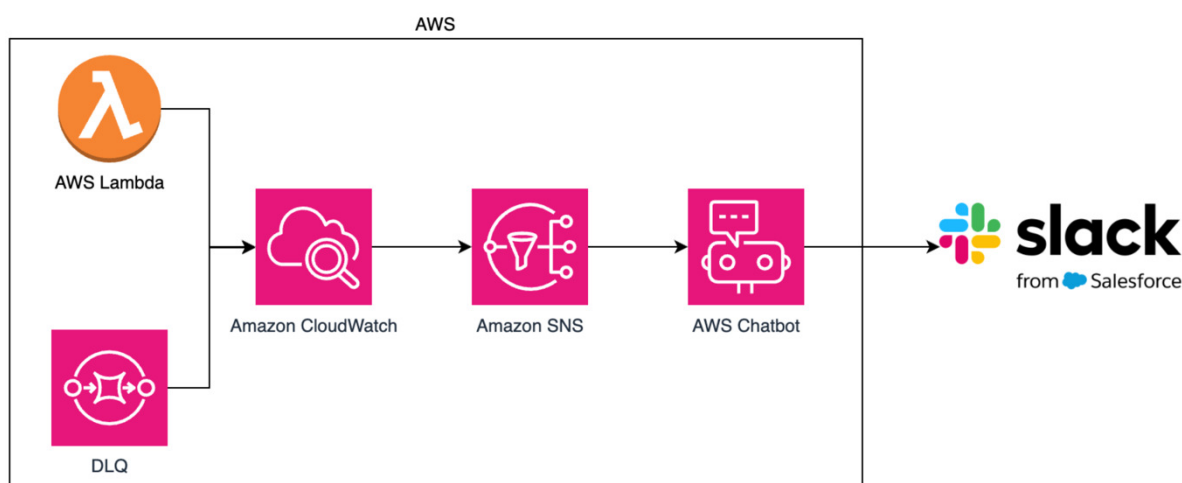
Con el objetivo de facilitar las actividades de monitoreo y asegurar el correcto funcionamiento del Sistema PIM, se configuraron alertas y gráficos o dashboards para lo cual

se utilizó AWS CloudWatch, AWS SNS, AWS Chatbot, y Slack. Esta integración consistió en dos tipos:

- DLQ: Como se mencionó anteriormente, cuando un evento no es procesado correctamente luego de un número máximo de 10 reintentos, este es enviado a su respectiva cola de mensajes fallidos o DLQ. En AWS CloudWatch, se configuraron alarmas para que cada vez que un evento sea recibido por una DLQ, se publique un mensaje en un tema o topic de AWS SNS, el cual es procesado por el servicio de AWS Chatbot, para así publicar el mensaje en un canal o channel de Slack. De esta manera, se tiene visibilidad cuando un evento no se ha procesado correctamente, y se lo puede revisar en la respectiva DLQ.
- Lambda: Cada Lambda implementado genera logs que se pueden revisar en AWS CloudWatch, de esta manera se tiene visibilidad para asegurar el correcto funcionamiento del Sistema PIM. Se configuraron alarmas para que cuando se supere una cierta cantidad de logs de advertencia o warning, así como logs de errores, se publique un mensaje en un tema o topic de SNS, el cual es procesado por AWS Chatbot, y finalmente publicado en un canal de Slack.

A continuación, la Figura 26 presenta la integración descrita.

Figura 26
Integración de alarmas



Nota. Fuente: Adaptado del proyecto.

HU_14 Migrar el catálogo de productos

Para realizar la migración de la información de los productos desde la plataforma externa hacia el Sistema PIM, se ejecutó el script implementado en la HU_11. Para lo cual se utilizó Node.js. La Tabla 19 presenta un resumen de los datos migrados.

Tabla 19

Resumen de la migración del catálogo de productos

Plataforma externa	CMS	Bucket de S3
- 677 imágenes de productos	- 2001 entidades de asset container	- 677 archivos o files en formato JSON correspondientes a los productos
- 1324 imágenes de variaciones		
- 1324 variaciones	- 1324 entidades de variaciones	
- 677 productos	- 677 entidades de productos	

Nota. Fuente: Elaboración propia.

HU_15 Certificar el funcionamiento del sistema

Para validar el correcto funcionamiento del Sistema PIM, se aplicaron pruebas funcionales o End – to – End, las cuales fueron implementadas utilizando Cypress. Se crearon y validaron 60 escenarios de prueba, a continuación, la Tabla 20 presenta los escenarios de prueba más importantes.

Tabla 20

Escenarios de prueba

Escenario de prueba	QA Status
El autor de contenido puede crear un producto con sus respectivas variaciones, imágenes, y videos.	Passed

El autor de contenido puede crear una variación con sus respectivas opciones, imágenes, y videos, y agregar la variación a su respectivo producto.	Passed
El autor de contenido puede actualizar la información de un producto y los cambios son reflejados automáticamente en el ambiente de pre – producción.	Passed
El autor de contenido puede actualizar la información de una variación y los cambios son reflejados automáticamente en el ambiente de pre – producción.	Passed
El autor de contenido puede publicar la información de un producto o variación y los cambios son reflejados en el ambiente de producción.	Passed
El autor de contenido puede despublicar la información de un producto o variación y los cambios son reflejados en el ambiente de producción.	Passed
El autor de contenido puede crear releases, agregar entidades de productos o variaciones, y programar la fecha y la hora para su publicación o despublicación automática.	Passed
Las entidades de productos o variaciones correspondientes a un release son publicados o despublicados automáticamente según la fecha y hora definidas, y los cambios son reflejados en el ambiente de producción.	Passed

Nota. Fuente: Adaptado del proyecto.

3.2.5.7 Despliegue.

HU_16 Desplegar el sistema

Para desplegar el Sistema PIM en el ambiente de producción se realizaron tres pasos, siendo el primero el despliegue del código, para lo cual se habilitó y aprobó la transición tal cual se describe en la HU_03.

El segundo paso consistió en habilitar o activar el webhook correspondiente al ambiente de producción en el CMS para que se inicien las peticiones o requests POST al Sistema PIM, a continuación, en la Figura 27 se puede visualizar el estado del webhook.

Figura 27
Webhook del ambiente de producción

Webhook: product-catalog-prod Active

Webhook settings Activity log

Webhook called at	Call result
Mon, 07 Oct 2024 at 8:11:54 PM	● HTTP 200
Mon, 07 Oct 2024 at 8:06:12 PM	● HTTP 200
Mon, 07 Oct 2024 at 7:56:53 PM	● HTTP 200
Mon, 07 Oct 2024 at 7:51:12 PM	● HTTP 200
Mon, 07 Oct 2024 at 7:41:53 PM	● HTTP 200
Mon, 07 Oct 2024 at 7:36:13 PM	● HTTP 200
Mon, 07 Oct 2024 at 7:26:54 PM	● HTTP 200
Mon, 07 Oct 2024 at 7:21:13 PM	● HTTP 200
Mon, 07 Oct 2024 at 7:11:54 PM	● HTTP 200

Nota. Fuente: Adaptado del proyecto.

El tercer paso consistió en activar el Feature Flag mencionado en la HU_12 para que la aplicación web de la E-commerce obtenga la información del catálogo de productos desde el Sistema PIM. A continuación, la Figura 28 presenta la configuración del Feature Flag.

Figura 28
Feature flag encendido

```
{
  "isPIMSystemEnabled": true
}
```

Nota. Fuente: Adaptado del proyecto.

3.2.5.8 Mantenimiento y monitoreo.

Una vez el Sistema PIM fue desplegado en el ambiente de producción, se iniciaron las actividades de mantenimiento y monitoreo, para lo cual se utilizaron las alarmas o alertas y los gráficos o dashboards configurados en la HU_13. Al tener configurados los logs en AWS CloudWatch, se tiene visibilidad en caso ocurra algún incidente, y las colas de eventos fallidos o DLQs permiten la revisión de cada evento y su posible reprocesamiento en caso sea necesario.

3.3 Evaluación

3.3.1 Evaluación económica / Evaluación costo – beneficio

Al tratarse de información sensible para la Entidad E-commerce, los costos del proyecto son estimados. No se incluyen gastos relacionados a equipos informáticos, internet, energía, entre otros, debido a que el proyecto fue implementado bajo la modalidad de trabajo remoto, por lo tanto, estos fueron cubiertos en los pagos realizados a cada miembro del equipo. La moneda utilizada para realizar esta evaluación es el dólar o USD.

La Tabla 21 muestra los costos relacionados a la Plataforma Externa, la cual fue reemplazada por el Sistema PIM.

Tabla 21
Costos estimados de la plataforma externa

	Costo mensual (USD)	Costo anual (USD)
Suscripción (Personalizada + Soporte TIER 5)	6,000.00	72,000.00
Costos variables	2,000.00	24,000.00
Total	8,000.00	96,000.00

Nota. Fuente: Elaboración propia.

La Tabla 22 presenta los costos estimados de los servicios de AWS que fueron utilizados en la implementación del Sistema PIM.

Tabla 22

Costos estimados de los servicios de AWS

Servicio	Costo mensual (USD)	Costo anual (USD)
Secrets Manager	60.80	729.60
API Gateway	42.00	504.00
SNS	2.00	24.00
SQS	1.60	19.20
Lambda	419.14	5,029.68
S3	2.30	27.60
ElastiCache	30.00	360.00
Total	557.84	6,694.08

Nota. Fuente: Elaboración propia.

En la Tabla 23 se puede visualizar el costo total estimado de la implementación del Sistema PIM, en el cual se consideran los pagos de los miembros de equipo, los recursos tecnológicos y el tiempo en meses.

Tabla 23

Costo total estimado del Sistema PIM

	Costo mensual (USD)	Cantidad	Tiempo (meses)	Costo total (USD)
Product Manager	7,000.00	1	4.5	31,500.00
Diseñador UX	4,000.00	1	4.5	18,000.00
Content Author	2,500.00	2	4.5	22,500.00
Technical Lead	8,000.00	1	4.5	36,000.00
Software Developer	6,000.00	3	4.5	81,000.00
Tester o QA	4,000.00	1	4.5	18,000.00
Servicios de AWS	557.84		4.5	2,510.28

Plataforma externa	8,000.00	4.5	36,000.00
Total			245,510.28

Nota. Fuente: Elaboración propia.

La implementación del Sistema PIM permitió la reducción de la reserva de contingencia anual en un 25%, la cual está destinada a mitigar los riesgos relacionados a la aplicación web de la E-commerce, entre los cuales se encuentra la posibilidad de que se presenten errores en la actualización de los precios, pero en un grado de ocurrencia mucho menor. La Tabla 24 presenta la reserva de contingencia.

Tabla 24

Reserva de contingencia anual estimada

Cantidad anterior (USD)	Cantidad actual (USD)	Cantidad ahorrada (USD)
1,000,000.00	750,000.00	250,000.00

Nota. Fuente: Elaboración propia.

A continuación, la Tabla 25 presenta los beneficios obtenidos con la implementación del Sistema PIM.

Tabla 25

Beneficios del Sistema PIM

	Antes	Después
Estructura de datos e información del catálogo de productos.	Dependencia de la plataforma externa.	Control total.
Implementación de funcionalidades e integración con servicios que requieren la información del catálogo de productos.	Complejidad innecesaria.	Mayor flexibilidad y acceso directo a la información.
Actualización del catálogo de productos.	Archivo en formato “.csv”.	Interfaz intuitiva brindada por el CMS.

Actualización masiva.	Proceso manual.	Proceso automatizado.
Riesgo de que se presenten errores en la actualización del catálogo de productos.	Alto.	Casi nulo.

Nota. Fuente: Elaboración propia.

3.3.2 Interpretación del VAN y del TIR

Para calcular el Valor Actual Neto (VAN) y la Tasa Interna de Retorno (TIR), se utilizaron los valores presentados en el punto 3.3.1.

Los costos de la plataforma externa presentados en la Tabla 21 son considerados como flujo de caja positivo ya que fueron reemplazados por el Sistema PIM. Los costos de la infraestructura del Sistema PIM presentados en la Tabla 22 son considerados como flujo de caja negativo. La inversión inicial corresponde al monto presentado en la Tabla 23. Por último, la reducción de la reserva de contingencia presentada en la Tabla 24 aumentó el flujo de caja positivo.

A continuación, la Tabla 26 muestra el flujo de caja de los cuatro años posteriores a la implementación del Sistema PIM y los valores correspondientes al VAN y TIR, para lo cual se utilizó una tasa de descuento del 12%.

Tabla 26
VAN y TIR

Inversión inicial	\$ 245,510.28
Efectivo Neto – Año 1	\$ 339,305.92
Efectivo Neto – Año 2	\$ 339,305.92
Efectivo Neto – Año 3	\$ 339,305.92
Efectivo Neto – Año 4	\$ 339,305.92
Tasa de descuento	12%
TIR	134%

VAN	\$ 785,080.33
-----	---------------

Nota. Fuente: Elaboración propia.

La inversión inicial del proyecto tuvo un costo de \$ 245,510.28. Se evaluó la viabilidad del proyecto con una tasa de descuento del 12% obteniendo un VAN de \$ 785,080.33 y un TIR de 134% demostrando que la implementación del Sistema PIM sí es rentable.

CAPÍTULO IV – REFLEXIÓN CRÍTICA DE LA EXPERIENCIA

En este capítulo se mencionan las reflexiones del autor respecto al proyecto presentado en este informe.

El autor del presente trabajo participó en cada una de las etapas del proyecto, y en la actualidad sigue realizando actividades de mantenimiento y monitoreo en caso sean necesarias. Durante este tiempo ha desarrollado y consolidado sus habilidades técnicas en diversas herramientas, incluyendo los servicios de AWS utilizados en el proyecto, los cuales tienen una alta demanda en el mercado actual. Asimismo, al tratarse de una modalidad de trabajo remoto, fue fundamental el fortalecimiento de habilidades blandas como el trabajo en equipo, la autoorganización, la gestión de conflictos, y la resiliencia.

Se logró la implementación del Sistema PIM siguiendo los lineamientos y flujos de trabajo de la Entidad E-commerce. Además, se gestionaron con éxito los procesos requeridos por la Entidad E-commerce tales como el despliegue o pase a producción.

Uno de los retos que fueron superados consistió en obtener los accesos y permisos requeridos para las cuentas de AWS dedicadas a los ambientes de Pre – producción y Producción, pero al tener acceso a la cuenta de desarrollo, esto no afectó la planificación inicial del proyecto. Una vez los accesos fueron dados, el despliegue en cada uno de estos ambientes se realizó con normalidad.

El uso de la metodología ágil basada en el marco de trabajo Scrum permitió que la comunicación entre los miembros del equipo sea eficaz, logrando así la resolución eficiente de dudas y preguntas que fueron surgiendo a lo largo del proyecto. Además, al involucrar a los autores de contenido, que en otras palabras son los usuarios finales del Sistema PIM, los requerimientos definidos estuvieron enfocados en resolver las necesidades y optimizar los procesos correspondientes a la gestión del catálogo de productos.

La experiencia y el compromiso de los miembros del equipo fueron fundamentales para lograr los objetivos trazados. La autoorganización del equipo contribuyó a la superación de adversidades tales como las ausencias por temas de salud.

Al tratarse de una infraestructura Serverless, el Sistema PIM brinda una gran capacidad de escalabilidad, lo cual permitió automatizar el proceso de actualizaciones masivas del catálogo de productos. Además, la implementación de un almacenamiento de memoria rápida o caché optimizó los tiempos de respuesta al momento de acceder a la información.

Por último, otro reto superado fue la rotación de algunos miembros del equipo, para lo cual fue importante la transferencia de conocimiento eficiente y eficaz, a nivel técnico, y los procesos que involucran la gestión del catálogo de productos.

CAPÍTULO V – CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

- Se implementó un Sistema de Gestión de Información de Productos, o PIM, basado en una infraestructura Serverless en una Entidad E-commerce, lo cual se detalla en el capítulo 3, obteniendo los beneficios mencionados en la Tabla 25.
- Se diseñó la estructura de datos del catálogo de productos y la arquitectura del sistema, lo cual se observa en el punto 3.2.5.3. Este diseño facilitó la construcción de nuevas funcionalidades al aumentar la flexibilidad y la capacidad de adaptarse al cambio. Por otro lado, al tener una infraestructura Serverless, el Sistema PIM es altamente escalable.
- La información del catálogo de productos fue expuesta mediante la implementación de REST API, lo cual se presenta en el punto 3.2.5.5. Estos datos fueron expuestos de manera segura mediante la configuración de un Lambda que se encarga de gestionar los accesos a la información. Además, la implementación de un almacenamiento de memoria rápida o caché optimizó los tiempos de respuesta. Esto permitió la construcción de nuevas funcionalidades y la integración con servicios que requieren de esta información.
- Se integró un Sistema de Gestión de Contenido, o CMS, el cual brinda una interfaz de usuario intuitiva, evidenciado en el punto 3.2.5.4. Esto contribuyó a reducir el riesgo de que se presenten errores en la actualización del catálogo de productos.
- Se realizó la implementación de un módulo para la automatización de las actualizaciones masivas, lo cual se observa en el punto 3.2.5.5. Esto permitió la configuración de releases, los cuales se encargan de publicar o despublicar los cambios del catálogo de productos en una fecha y hora previamente definidas. De esta manera, se redujo el riesgo de que se presenten errores en la actualización del catálogo de productos, principalmente los precios.
- Se creó un script para la migración del catálogo de productos desde la Plataforma Externa hacia el Sistema PIM, el cual tuvo tres etapas: extracción, transformación, y

publicación, evidenciado en el punto 3.2.5.5. De esta manera se garantizó la integridad de los datos.

- El Sistema PIM fue desplegado con éxito en el ambiente de Producción, para lo cual se implementó un Feature Flag que permitió encender el Sistema PIM de manera segura, lo cual se observa en el punto 3.2.5.7.
- La configuración de alarmas, gráficos, y colas de eventos fallidos facilitaron las actividades de monitoreo del Sistema PIM, además de permitir la revisión y reprocesamiento de eventos en caso sea necesario, lo cual se observa en el punto 3.2.5.6.

5.2 Recomendaciones

- Configurar los permisos de los autores de contenido en el Sistema de Gestión de Contenido, o CMS, para que tengan acceso solamente a la lectura y actualización de la información. Y que la capacidad de publicar o despublicar entidades sea únicamente del módulo de actualizaciones implementado para la gestión de releases. Esto con la finalidad de tener un proceso más seguro y automatizado.
- Las colas de eventos SQS garantizan que los mensajes sean procesados al menos una vez, se recomienda la implementación de una lógica para evitar el procesamiento de eventos duplicados, con el fin de reducir los costos.
- El almacenamiento del catálogo de productos en S3 en archivos en formato JSON facilita el desarrollo de nuevas funcionalidades, por lo cual, se recomienda evaluar nuevas herramientas tales como la implementación de un motor de búsqueda inteligente con Amazon Kendra.

5.3 Fuentes de información

- Akeneo. (2024). What is a PIM and Why Do You Need One? <https://www.akeneo.com/what-is-a-pim/>
- Atlassian. (2024). Jira. <https://www.atlassian.com/software/jira>
- AWS.Amazon. (2024a). What is a RESTful API? <https://aws.amazon.com/es/what-is/restful-api/>
- AWS.Amazon. (2024b). What is Serverless Computing? <https://aws.amazon.com/what-is/serverless-computing/>
- AWS.Amazon. (2024c). Cloud computing with AWS. <https://aws.amazon.com/es/what-is-aws/>
- AWS.Amazon. (2024d). Amazon API Gateway. <https://aws.amazon.com/api-gateway/>
- AWS.Amazon. (2024e). AWS Secrets Manager. <https://aws.amazon.com/secrets-manager/>
- AWS.Amazon. (2024f). Amazon Simple Notification Service. <https://aws.amazon.com/sns/>
- AWS.Amazon. (2024g). Amazon Simple Queue Service. <https://aws.amazon.com/sqs/>
- AWS.Amazon. (2024h). Serverless Computing - AWS Lambda. <https://aws.amazon.com/pm/lambda/>
- AWS.Amazon. (2024i). Cloud Object Storage - Amazon S3. <https://aws.amazon.com/pm/serv-s3/>
- AWS.Amazon. (2024j). Amazon ElastiCache for Valkey and for Redis OSS. <https://aws.amazon.com/elasticache/redis/>
- AWS.Amazon. (2024k). AWS SDK for JavaScript. <https://aws.amazon.com/sdk-for-javascript/>
- AWS.Amazon. (2024l). AWS Cloud Development Kit. <https://aws.amazon.com/cdk/>
- AWS.Amazon. (2024m). Amazon CloudWatch. <https://aws.amazon.com/cloudwatch/>
- AWS.Amazon. (2024n). AWS Chatbot. <https://aws.amazon.com/chatbot/>
- AWS.Amazon. (2024o). AWS CodePipeline. <https://aws.amazon.com/codepipeline/>
- Berggren, C. (2021). *Product management system: A bridge between the customer and company's business model and strategy for niche companies in oligopoly markets* [Master of science thesis, KTH Royal Institute of Technology]. <https://www.diva-portal.org/smash/get/diva2:1588888/FULLTEXT01.pdf>

- Blayzer Digital. (2022). How To Avoid Ecommerce Pricing Glitches. <https://www.blayzer.com/how-to-avoid-ecommerce-pricing-glitches/>
- Canasa Mayta, P. R. (2022). *Implementación de un sistema de información para la gestión de unidades de manipulación en almacenes de grandes dimensiones* [Tesis de pregrado, Pontificia Universidad Católica del Perú]. https://tesis.pucp.edu.pe/repositorio/bitstream/handle/20.500.12404/23555/CANASA_MAYTA_PAUL_RODRIGO_IMPLEMENTACION_SISTEMA_INFORMACION.pdf
- Forbes. (2024). What Is E-Commerce?: A Comprehensive Introduction. <https://www.forbes.com/advisor/business/what-is-ecommerce/>
- IBM. (2024). What is a content management system (CMS)? <https://www.ibm.com/topics/content-management-system>
- Red Hat. (2022). La integración y la distribución continuas (CI/CD). <https://www.redhat.com/es/topics/devops/what-is-ci-cd>
- Red Hat. (2024). What is a webhook? <https://www.redhat.com/en/topics/automation/what-is-a-webhook>
- SCRUMstudy™. (2022). *A Guide to the Scrum Body of Knowledge (SBOK® Guide)* – Fourth edition
- Slack Technologies. (2024). What is Slack? <https://slack.com/help/articles/115004071768-What-is-Slack>
- Software Freedom Conservancy. (2024). Getting Started - What is Git? <https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F>
- The GraphQL Foundation. (2024). Introduction to GraphQL. <https://graphql.org/learn/>
- The OpenJS Foundation. (2024). Jest. <https://jestjs.io/>

5.4 Glosario

- **Archivo en formato “.csv”:** Archivo utilizado anteriormente para actualizar el catálogo de productos en la Plataforma Externa.
- **Autor de contenido:** También llamado content author, es el usuario final del Sistema PIM, se encarga de mantener actualizado el catálogo de productos.

- **AWS:** Amazon Web Services, es una plataforma que ofrece diversos servicios en la nube.
- **Checkout:** Proceso final de compra en una aplicación web o E-commerce, el cliente confirma y realiza el pago respectivo.
- **CMS:** Sistema de gestión de contenido.
- **Consultora:** Empresa a la cual brinda servicios el autor del presente informe.
- **Cypress:** Herramienta que automatiza las pruebas funcionales o End – to – End.
- **Delivery:** Proceso de entrega de un producto.
- **DLQ:** Cola de mensajes o eventos fallidos.
- **E-commerce:** Comercio electrónico que permite la compra y venta de productos o servicios a través de internet.
- **Entidad E-commerce:** Es la empresa cliente en la cual el autor del presente informe desempeñó sus labores.
- **Husky:** Herramienta para automatizar Git hooks, mejorando la calidad del código.
- **JSON:** Formato de datos estructurado utilizado para el intercambio de información entre sistemas y almacenar la información del catálogo de productos.
- **Node.js:** Entorno de ejecución de JavaScript.
- **Ownership:** Responsabilidad y compromiso de un individuo o equipo sobre una tarea o proyecto.
- **PIM:** Sistema de gestión de información de productos.
- **Plataforma de Terceros:** También llamada Plataforma Externa, es la plataforma utilizada anteriormente para gestionar el catálogo de productos.
- **React:** Librería de JavaScript utilizada para construir interfaces de usuario dinámicas y reutilizables.
- **Release:** Contiene entidades de productos o variaciones que serán publicados o despublicados según una fecha y hora previamente definidos.
- **S3:** Servicio de almacenamiento de AWS.
- **Serverless:** Modelo de computación en la nube donde no es necesario gestionar servidores, las funciones se ejecutan bajo demanda.
- **SNS:** Servicio de AWS para enviar notificaciones, mensajes, o eventos.
- **Software Developer:** Desarrollador de software.

- **SQS:** Servicio de AWS que brinda colas para el procesamiento de mensajes o eventos.
- **TypeScript:** Lenguaje de programación que extiende JavaScript con tipado para mayor robustez y mantenibilidad.

ANEXOS

Anexo A: Producto almacenado en el Bucket en S3

En la siguiente imagen se presenta la información del archivo “product-1.json”, el cual está almacenado en el Bucket en S3 en formato JSON.

```
{
  "title": "Product 1",
  "id": "product-1",
  "description": "This is a description for Product 1.",
  "tags": ["new"],
  "variants": [
    {
      "title": "Black",
      "sku": "PRDCT1BLCK",
      "description": "This is a description for Black variant.",
      "options": [
        {
          "value": "Black",
          "type": "Color"
        }
      ]
    },
    {
      "title": "White",
      "sku": "PRDCT1WHT",
      "description": "This is a description for White variant.",
      "options": [
        {
          "value": "White",
          "type": "Color"
        }
      ]
    }
  ],
  "assets": [
    {
      "title": "Product 1 - Black - Image 1",
      "alt": "Alt for Product 1 - Black - Image 1",
      "image": "https://example.com/images/black-headphones-front.jpg"
    },
    {
      "title": "Product 1 - Black - Image 2",
      "alt": "Alt for Product 1 - Black - Image 2",
      "image": "https://example.com/images/black-headphones-side.jpg"
    },
    {
      "title": "Product 1 - Image 1",
      "alt": "Alt for Product 1 - Image 1",
      "image": "https://example.com/images/promotional-banner.jpg"
    },
    {
      "title": "Product 1 - Video 1",
      "alt": "Alt for Product 1 - Video 1",
      "image": "https://example.com/images/promotional-banner.jpg",
      "youtubeId": "youtubeId12345"
    }
  ],
  "price": 199.99,
  "comparablePrice": 249.99,
  "outOfStock": false,
  "tags": ["best-seller"]
}
```