UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO

INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

LICENCIATURA EN CIENCIAS COMPUTACIONALES

TESIS

IMPLEMENTACIÓN DE CMMI Y SCRUM PARA EL ASEGURAMIENTO DE LA CALIDAD EN LA EMPRESA DE DESARROLLO DE SOFTWARE S.A.P.I.

Para obtener el título de

Licenciada en Ciencias Computacionales

PRESENTA

Judith Vianey Altamirano Jiménez

Directora

M.C.C Iliana Castillo Pérez

Codirectora

Dra. Verónica Martínez Lazcano

Comité tutorial

Dra. Verónica Martínez Lazcano (Presidente)
Mtra. Edrein Marcela Aguilar Ramírez (Secretaria)
M.C.C. Iliana Castillo Pérez (Vocal)
Dr. Edgar Olguín Guzmán (Suplente)

Pachuca de Soto, Hidalgo, México, noviembre 2023

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO
Instituto de Ciencias Básicas e Ingeniería
School of Engineering and Basic Sciences

Mineral de la Reforma, Hgo., a 28 de noviembre de 2023

Número de control: ICBI-D/1825/2023
Asunto: Autorización de impresión.

MTRA. OJUKY DEL ROCÍO ISLAS MALDONADO
DIRECTORA DE ADMINISTRACIÓN ESCOLAR DE LA UAEH

Con fundamento en lo dispuesto en el Título Tercero, Capítulo I, Artículo 18 Fracción IV; Título Quinto, Capítulo II, Capítulo V, Artículo 51 Fracción IX del Estatuto General de nuestra Institución, por este medio le comunico que el Jurado asignado a la Pasante de la Licenciatura en Ciencias Computacionales **Judith Vianey Altamirano Jiménez,** quien presenta el trabajo de titulación **"Implementación de CMMI y SCRUM para el aseguramiento de la calidad en la empresa de desarrollo de Software S.A.P.I."**, después de revisar el trabajo en reunión de Sinodales ha decidido autorizar la impresión del mismo, hechas las correcciones que fueron acordadas.

A continuación, firman de conformidad los integrantes del Jurado:

Presidente Dra. Verónica Martínez Lazcano

Secretario: M.C.C. Edrein Marcela Aguilar Ramírez

Vocal: M.C.C. Iliana Castillo Pérez

Suplente: Dr. Edgar Olguín Guzmán

Sin otro particular por el momento, reciba un cordial saludo.

Atentamente
"Amor, Orden y Progreso"

Dr. Otilio Arturo Acevedo Sandoval
Director del ICBI

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO
ICBI
DIRECCIÓN

OAAS/YCC

Ciudad del Conocimiento
Carretera Pachuca-Tulancingo km 4.5 Colonia
Carboneras, Mineral de la Reforma, Hidalgo
México, C.P. 42184
Teléfono: 771 71 720 00 ext. 2231 Fax 2109
direccion_icbi@uaeh.edu.mx

DEDICATORIA

A mis maestros de vida, mis padres, gracias por enseñarme a caminar por esta vida y guiarme, por los valores que me inculcaron, gracias a ustedes aprendí a luchar por mis sueños sin importar los obstáculos, a ser tenaz, y superar las caídas, gracias mamá por tu amor, tu paciencia, por siempre estar.

A mis compañeros de vida, de alegrías y de sonrisas, mis hermanos y a Yunaideli Hiromi quienes han compartido este viaje conmigo, creando grandes recuerdos y superando desafíos a su lado, gracias por coincidir en esta vida.

A mi profesora Iliana Castillo Pérez, le agradezco la confianza depositada en mí, su dedicación, disposición y tiempo invertido, porque con su sabiduría y orientación guio el camino para llevar a cabo este proyecto. Gracias por creer en mí y hacer esto posible.

A mis profesores, quienes han compartido sus conocimientos, gracias por guiar mi desarrollo académico y crecimiento profesional, ampliando mi sabiduría y enriqueciendo mis perspectivas significativamente.

A todos aquellos que han sido parte de este sueño, gracias por ayudarme a que este viaje fuera más ligero, gracias por su apoyo, paciencia, porque de una u otra forma han contribuido en mi crecimiento.

RESUMEN

En el dinámico y competitivo mundo del desarrollo de software, la calidad, la eficiencia y la satisfacción del cliente son imprescindibles por lo que esta tesis se centra en la búsqueda de soluciones innovadoras y efectivas para garantizar la calidad del software, destinada a abordar estos desafíos en el entorno empresarial. A lo largo de esta investigación y aplicación, se lograron avances significativos y se alcanzaron los objetivos propuestos.

La presente tesis comenzó con una exhaustiva investigación que abarca diversas áreas. Se exploraron conceptos fundamentales de calidad de software, se analizaron estándares reconocidos y se estudiaron los diversos ciclos de vida del software. Se profundizó en las metodologías ágiles, y se examinó el marco de mejora de procesos CMMI por sus siglas en inglés *Capability Maturity Model Integration* (Integración de los modelos de madurez de capacidades). De acuerdo con dicha investigación, se realizó un análisis exhaustivo de aquellas herramientas, metodologías y técnicas disponibles para garantizar la calidad del desarrollo de software, lo que proporcionó una base sólida para la selección y aplicación de las mejores prácticas en una empresa de desarrollo de software.

La empresa S.A.P.I. se convirtió en el escenario para la implementación de estos conceptos y metodologías en un esfuerzo por mejorar la calidad y eficiencia del desarrollo de software. Se llevó a cabo un plan de trabajo diseñado meticulosamente para lograr la integración y adaptación de los procesos de CMMI y SCRUM. Este plan abordó áreas críticas que incluyeron la evaluación de la situación actual de la empresa, mediante el método SCAMPI como una herramienta para medir la madurez de los procesos. Luego, se adaptaron las prácticas ágiles de SCRUM para cumplir con los rigurosos requisitos de CMMI. Se brindó capacitación y formación adecuada al equipo para garantizar una comprensión clara de los principios y requisitos de CMMI. Se definió un sólido plan de mejora que incluía metas claras y un cronograma realista para implementar mejoras en los procesos.

Esta tesis ha demostrado de manera concluyente que la combinación de CMMI y SCRUM es una estrategia efectiva para garantizar la calidad y eficiencia en el desarrollo de software.

Los resultados obtenidos en la empresa S.A.P.I. validan la importancia de adoptar enfoques integrales que aprovechen las mejores prácticas de la industria y las adapten a las necesidades específicas de la organización. Este trabajo no solo enriquece el conocimiento en la gestión de calidad de software, sino que también proporciona una guía práctica para otras organizaciones que buscan mejorar sus estándares de calidad y eficiencia en un mercado en constante evolución.

"Probar es la parte inevitable de cualquier esfuerzo responsable por asegurar la calidad de un sistema de software."

William Howden

ABSTRACT

In the dynamic and competitive world of software development, quality, efficiency, and customer satisfaction are paramount. Therefore, this thesis focuses on seeking innovative and effective solutions to ensure software quality, aimed at addressing these challenges in the business environment. Significant progress was made and the proposed objectives were achieved throughout this research and implementation.

This thesis commenced with an exhaustive investigation covering various areas. Fundamental concepts of software quality were explored, recognized standards were analyzed, and various software development life cycles were studied. Agile methodologies were delved into, and the CMMI process (for its acronym in English, *Capability Maturity Model Integration*), improvement framework was examined. Based on this research, a comprehensive analysis of tools, methodologies, and techniques available to ensure software development quality was conducted, providing a solid foundation for selecting and applying best practices in a software development company.

The company S.A.P.I. served as the stage for implementing these concepts and methodologies in an effort to enhance software development quality and efficiency. A meticulously designed work plan was executed to achieve the integration and adaptation of CMMI and SCRUM processes. This plan addressed critical areas, including the assessment of the company's current situation, using the SCAMPI method as a tool to measure process maturity. Subsequently, agile SCRUM practices were adapted to meet the rigorous requirements of CMMI. Adequate training and education were provided to the team to ensure a clear understanding of CMMI principles and requirements. A robust improvement plan was defined, incorporating clear goals and a realistic schedule for implementing process enhancements.

This thesis conclusively demonstrates that the combination of CMMI and SCRUM is an effective strategy to ensure software development quality and efficiency. The results obtained in the S.A.P.I. company validate the importance of adopting comprehensive approaches that leverage industry best practices and adapt them to the organization's specific needs. This work not only enriches knowledge in software quality management but

also provides a practical guide for other organizations seeking to improve their quality and efficiency standards in an ever-evolving market.

"Testing is an inevitable part of any responsible effort to ensure the quality of a software system."

William Howden

ÍNDICE GENERAL

ÍNDICE DE ANEXOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Introducción

En los últimos años ha cobrado relevancia la disciplina de las pruebas de software, muchas empresas han decidido establecer un área de control de su estructura con personal especializado en pruebas. Esto se debe, entre otros factores, al papel que está jugando el software en la operación diaria de las organizaciones y en la relación y servicios que se les dan a los usuarios finales (Gómez García, Sosa Hernández, Verona Marcos, & Delgado Dapena, 2023), (Serna, Martínez, & Tamayo, 2019).

En la actualidad se le está dando mucha importancia a la satisfacción del cliente debido a la voz que tienen sobre todo en las redes sociales, anteriormente si el cliente no podía realizar ciertas operaciones desde su domicilio, simplemente se resignaba a no hacerlo o lo postergaba, actualmente con las redes sociales, el cliente se da a escuchar afectando así la reputación de la empresa en cuestión, desafortunadamente las quejas tienen mayor eco que los comentarios positivos.

Para hacer que el usuario final advierta menos fallas y se mantenga contento con el servicio que recibe, se implementa un modelo de pruebas de software, esto a la empresa le conviene por que le ayuda a ganar nuevos clientes y no perder a los existentes (C., 2016).

Otro aspecto que ha impulsado a las pruebas de software es que la operación de las empresas cada vez depende más del buen funcionamiento de sus sistemas de cómputo, una falla puede detener su operación e incluso generar pérdidas millonarias.

Afortunadamente, en la actualidad existen diversas empresas que han tomado conciencia de los múltiples beneficios que trae consigo realizar pruebas durante el ciclo de vida del desarrollo de software, sin embargo, otras tantas empresas aún se ven reacias debido, sobre todo, a que esto representa un costo considerable y algunas otras lo ven como un servicio de valor agregado.

En este trabajo se presentan los resultados de una exhaustiva investigación que abarca diversas áreas: calidad de software, estándares reconocidos, los diversos ciclos de vida del software y las metodologías ágiles. De igual forma se examinó el marco de mejora de

procesos CMMI (*Capability Maturity Model Integration*). De acuerdo con los resultados de dicha investigación, se diseñó un plan de trabajo meticulosamente con el fin de lograr la integración y adaptación de los procesos de CMMI y SCRUM para garantizar la calidad del desarrollo de software, lo que proporcionó una base sólida para la selección y aplicación de las mejores prácticas en una empresa de desarrollo de software.

Antecedentes

La industria del desarrollo de software ha experimentado un crecimiento significativo en las últimas décadas, por lo que la competencia de este mercado es cada vez más intensa y las empresas están bajo una presión constante para proporcionar productos y servicios de software de alta calidad que cumplan con las expectativas de los clientes. La calidad del software juega un papel muy importante en la industria del software, ya que influye directamente en la satisfacción del cliente, la imagen de la empresa y su éxito en el mercado (Aizprua, Ortega, & Von Chong, 2019).

A lo largo de la historia el desarrollo de software se ha enfocado principalmente en la programación y la entrega en tiempo dentro del presupuesto establecido de los productos de software. Sin embargo, a medida que las aplicaciones de software se volvieron más complejas y su adopción se generalizó, surgió la necesidad de garantizar la calidad del software. Esto llevo a que autores como Watts S. Humphrey contribuyera en el desarrollo de modelos como el CMMI, para evaluar y mejorar la madurez de los procesos de desarrollo de software y Jeff Sutherland cocreador de SCRUM, uno de los marcos de trabajo ágiles ampliamente adoptados en la industria.

A pesar de los avances en el aseguramiento de la calidad del software, todavía existen desafíos por abordar. Uno de los problemas persistentes es la gestión eficiente de proyectos de desarrollo de software, especialmente en entornos altamente cambiantes. La evolución tecnológica constante y las demandas cambiantes de los clientes pueden dificultar la planificación y ejecución efectiva de proyectos. Además, la necesidad de asegurar la seguridad cibernética y la protección de datos ha agregado un nivel adicional de complejidad al desarrollo de software.

La implantación de un modelo de madurez y la metodología pueden brindar una solución para que las organizaciones mejoren en gran medida la calidad de sus productos mientras mantienen la flexibilidad necesaria para adaptarse a los cambios. Al combinar las mejores prácticas para la gestión de procesos para la entrega de software, las empresas pueden lograr un equilibrio entre la calidad y la velocidad de entrega. Esto es esencial para satisfacer las expectativas de los clientes y mantener una ventaja competitiva en el mercado en constante evolución.

Planteamiento del problema

La empresa S.A.P.I. enfocada en sus inicios principalmente al servicio del diseño, implementación y gestión de la infraestructura de soporte, con el fin de atender los requerimientos de la industria en sus diversos sectores, dividió su negocio en dos diferentes niveles dando inicio al desarrollo de aplicaciones digitales, sin embargo, a lo largo de su paso por la industria ha enfocado gran parte de sus recursos principalmente a proponer soluciones tecnológicas, en la automatización de información en áreas estratégicas y brindar mantenimiento a las aplicaciones implementadas.

Debido a la demanda de nuevas y diversas aplicaciones, ha llevado a que el equipo de desarrollo se enfoque principalmente en las tareas más urgentes con el fin de cumplir plazos, dejando de lado factores muy importantes como el control, verificación y la validación, durante el proceso de desarrollo de software, que beneficie y asegure la calidad de sus productos. Esto ha llevado a que el proceso del ciclo de vida del desarrollo de software se realice de acuerdo con la prioridad o urgencia que solicite el usuario.

También, en el área de desarrollo no se cuenta con estándares ni procesos establecidos para el desarrollo de aplicaciones, lo que ha llevado a que el ciclo de desarrollo se lleve conforme surgen los requerimientos del usuario, esto ha dado como resultado aplicaciones con errores, falta de documentación, datos históricos poco eficientes, limitada comunicación dentro del equipo de desarrollo y el reproceso del software.

Propuesta de solución

Considerando los aspectos antes expuestos en relación con la importancia de la calidad y la situación actual de la empresa S.A.P.I. En el presente trabajo se realizó la implementación de un marco de trabajo, metodología y procesos claros para el desarrollo de aplicaciones. Esto debe incluir pautas de codificación, metodologías de desarrollo ágil que permiten una entrega iterativa y continua de software de alta calidad, así como un ciclo de vida de desarrollo definido que permita elaborar un plan de trabajo, tomando como marco de referencia las herramientas y técnicas existentes.

Por consiguiente, este trabajo se centra en la investigación y análisis que permita elaborar un plan de trabajo que sea adecuado para mejorar los procesos dentro de la organización y que brinde una resolución a la naturaleza del problema planteado, esto tomando como marco de referencia las herramientas y técnicas existentes.

En este trabajo no solo se busca la aplicación de procesos de calidad en el desarrollo de software, sino también la calidad en toda la organización, para lo que se propuso la implementación del área de procesos de CMMI.

Dicha implementación se iniciará en el desarrollo del sistema K¡Vo!, para que posteriormente sirva para fomentar la implementación de los procesos de calidad en otros proyectos de software dentro de la misma organización o proyectos futuros con el fin de satisfacer las necesidades del usuario.

Justificación

Debido a las diversas fallas que se han presentado durante el desarrollo de software han surgido metodologías, técnicas y herramientas que ayudan a validar y verificar el desarrollo del software, lo que ha aportado en gran medida para la evolución de la tecnología digital. A consecuencia de esta iniciativa, diversas empresas han tomado conciencia de la necesidad de mejorar los procesos de desarrollo, demandando productos de calidad con mejores servicios a las empresas de Tecnologías de la Información (TI).

Para asegurar que se esté construyendo un producto de software de manera correcta de acuerdo con los requerimientos del usuario y ayudar a asegurar la calidad del producto final, es importante realizar un análisis de las mejores herramientas, metodologías y técnicas existentes, para definir cuál es la óptima para el sistema en desarrollo.

En el competitivo mundo del desarrollo de software, donde la calidad y la eficiencia son cruciales, esta tesis busca implantar los procesos de CMMI y la metodología SCRUM en la empresa de desarrollo S.A.P.I. Se definen objetivos específicos que van desde el diagnóstico inicial hasta la propuesta de un plan de trabajo para una evaluación a largo plazo, creando un camino estructurado hacia la mejora continua de la calidad del software. El propósito no solo es enriquecer el conocimiento en la gestión de calidad de software, sino también proporcionar una guía práctica para otras organizaciones. La combinación de CMMI y SCRUM se presenta como una estrategia para cumplir y superar las expectativas de los clientes en un mercado en constante evolución.

Objetivo general

Implementar un marco de trabajo y procesos durante el ciclo de vida del desarrollo de software que asegure la calidad en todo el proceso del proyecto, mediante el análisis y la aplicación de métodos, herramientas y técnicas estudiadas con el propósito de elevar el nivel de madurez de la empresa en la mejora de sus procesos y proyectos, y obtener una evaluación positiva según el modelo CMMI, garantizando el cumplimiento exitoso de los requerimientos del cliente y mejorando la calidad general de los productos y servicios de la empresa.

Objetivos específicos

1. Realizar un análisis de las herramientas, metodologías y técnicas especializadas que aseguran la calidad del software mediante la recopilación de estudios previos.
2. Realizar un diagnóstico de la situación actual de la empresa S.A.P.I. que permita identificar las debilidades y áreas de mejora.

3. Integrar los estándares y prácticas de CMMI en los procesos de desarrollo de la empresa, que se enfoquen en la gestión de calidad, la medición de procesos y la mejora continua.
4. Implementar SCRUM como marco de trabajo ágil para la entrega de software, promoviendo la colaboración, la entrega incremental y la adaptación constante a las necesidades del cliente.
5. Mejorar el nivel de madurez de la empresa con base en las prácticas propuestas con la intención de que alcance un nivel 5.

Alcances

La presente tesis se enfoca en CMMI y SCRUM en un proyecto específico de la empresa S.A.P.I. Siguiendo todo su ciclo de vida bajo estos dos marcos de trabajo y evaluando su impacto en términos de calidad y eficiencia, y recopilando datos relacionados con este proyecto en particular. Además de desarrollar un plan de trabajo que sirva como guía para cumplir con los estándares requeridos para la certificación de CMMI.

Limitaciones

Las principales limitaciones de este trabajo son las limitantes de tiempo para implementar todos los procesos de CMMI y el acceso a datos y documentación debido a que la empresa tiene restricciones en cuanto a la disponibilidad de datos internos y documentación, lo que podría limitar la capacidad de recopilar información detallada sobre proyectos pasados. Además de la resistencia al cambio en su forma de trabajar por parte de los miembros del equipo de desarrollo o la de la misma gerencia.

Estructura del documento

La presente tesis se sitúa en un contexto muy importante en la industria del desarrollo de software como es la calidad y eficiencia en la entrega de productos de software. La implementación de metodologías y la mejora de procesos sugieren ser fundamentales para garantizar la calidad y eficiencia.

Esta tesis tiene como objetivo principal explorar cómo la combinación de SCRUM y CMMI-DEV generan un cambio significativo en la calidad y eficiencia de los procesos en la empresa de desarrollo S.A.P.I.

Con el fin de lograr este objetivo, la tesis se estructura en varios capítulos. En cada capítulo se expone aspectos relacionados con la implementación de mejora de procesos para asegura la calidad y eficiencia en el desarrollo de software. A continuación, se presenta una descripción detallada de la estructura de este trabajo:

Capítulo 1: es el "Marco referencial", en este primer capítulo se brinda un contexto teórico y se establecen las bases que sustentan el enfoque de investigación propuesto en el presente trabajo. Aquí se explora en los conceptos y temas teóricos, profundizando en las metodologías ágiles, como SCRUM, y los modelos de mejora de procesos, como CMMI.

Capítulo 2: titulado "Implementación del modelo de SCRUM y CMMI a la empresa S.A.P.I.", en este capítulo se realiza un análisis minucioso de la situación de la empresa S.A.P.I. con respecto a los procesos que requiere CMMI. Para llevar a cabo dicho análisis se empleó el método SCAMPI para tener una visión precisa de la brecha actual de la organización. Además, en este capítulo, se analiza cómo se pueden combinar de manera efectiva los procesos de CMMI y las prácticas de SCRUM para definir un marco de trabajo en la empresa S.A.P.I.

Capítulo 3: en este capítulo se muestra el "Plan de acción propuesto" el cual representa un componente esencial de este trabajo. Aquí se detalla el plan de trabajo diseñado para implementar las prácticas SCRUM y los procesos de CMMI con el fin de guiar a la empresa S.A.P.I. Para obtener la certificación de CMMI al nivel 5. Este plan aborda aspectos clave, como la planificación de proyectos, la monitorización y control, la gestión de requisitos, la verificación y validación, la definición de procesos de la organización.

Asimismo, en el capítulo 3 se brinda un plan de auditoría y se realiza dicha auditoría para evaluar y verificar la conformidad y eficacia de los procesos y prácticas implementados en la empresa S.A.P.I con los estándares y requisitos definidos en el modelo CMMI.

Capítulo 1. Marco referencial

En este capítulo se presenta el Marco Teórico el cual busca introducir los conceptos y términos esenciales que servirán como base de comunicación y entendimiento en el trabajo que sigue.

La industria del desarrollo de software ha ido evolucionando de manera acelerada en las últimas décadas con relación a los cambios continuos que ha ido experimentando. A causa de estos cambios y a la demanda de entregas rápidas y eficientes, surgieron las metodologías ágiles para dar respuesta a la necesidad de adaptarse a los cambios constantes. Sin embargo, ha resultado desafiante para diversas organizaciones garantizar la calidad del software.

Con el fin de garantizar la calidad del software y el proceso de desarrollo del mismo, surgen organizaciones que buscan identificar, analizar y mejorar los procesos utilizados en la producción de software, en la sección Estado del Arte se describirán algunos trabajos realizados por diversos autores en este sentido.

1.1. Marco teórico

En esta sección se establece el conocimiento fundamental para facilitar la comprensión y el estudio de la información presentada en las próximas secciones. El enfoque del trabajo es determinar los modelos y metodologías que mejor se ajusten a las circunstancias y necesidades de la empresa, además de describir de manera general las temáticas que involucran el desarrollo de software.

1.1.1. Ingeniería de software

La Ingeniería de Software es un campo de estudio que se enfoca en la aplicación de herramientas, métodos y principios para la operación, mantenimiento y desarrollo del software de alta calidad de manera eficiente durante todas las fases de su ciclo de vida. Otro de los enfoques de la ingeniería de software es la gestión de proyectos de desarrollo de software, la gestión de configuración y calidad del software. El objetivo principal de la Ingeniería de Software es producir software que resulte eficiente, confiable y seguro, que

garantice el cumplimiento de los requerimientos del usuario y las limitaciones de tiempo y costo (Pressman, 2014).

La Ingeniería de Software comprende un enfoque metódico organizado y disciplinado para el desarrollo de software, haciendo uso de múltiples modelos y metodologías de desarrollo. Además, la ingeniería de software incluye la evaluación de mejora continua en los procesos de desarrollo de software, al igual que los conceptos y seguimiento de estándares y mejores prácticas para el desarrollo de software.

1.1.2. Ciclo de vida del desarrollo de Software

El ciclo de Vida del desarrollo de Software es el proceso que describe las etapas por las que pasa un software y proporciona una estructura para el desarrollo del software con el fin de prevenir y minimizar todas las fallas que se puedan presentar y así cumplir con los requisitos del cliente. En 1945, George Polya, propone la resolución de problemas de manera efectiva, entendiéndolo a profundidad y definiendo claramente lo que se busca resolver (análisis), para después plantear un plan que aborde el problema de forma estructurada y eficiente (Diseño). Una vez que se tiene un plan, llevar a cabo la solución planteada (Planteamiento) para después evaluar si los resultados obtenidos satisfacen los requerimientos del problema (pruebas) (Polya, 1945).

Las etapas del ciclo de vida pueden variar de acuerdo con el modelo de desarrollo que se utilice, pero generalmente abarca los siguientes procesos:

- **Análisis**

Es la primera y más importante de las etapas del ciclo de vida del desarrollo de software. Esta etapa se enfoca en identificar y comprender los problemas que se pretenden abordar en el desarrollo y se establecen los objetivos del software, así como las funcionalidades y características que debe tener el sistema. En esta etapa también se realiza la documentación de los requisitos del cliente para el software a desarrollar, se analiza si el desarrollo es factible, el ambiente del proyecto, las necesidades de mercado, las herramientas de desarrollo, la factibilidad y aceptación.

Es fundamental que el proceso de análisis se realice de manera detallada y minuciosa en virtud de que los resultados que se obtengan de esta etapa serán la base para la etapa de diseño, implementación y pruebas del software. Un análisis deficiente puede derivar en una serie de problemas en el funcionamiento y la calidad del software.

- **Planificación**

Esta fase se lleva a cabo después de la fase de análisis y resulta ser un proceso muy importante y crítico en el ciclo de vida del desarrollo de software debido a que en esta fase se determinan las metas, objetivos, recursos y presupuestos para el desarrollo del proyecto.

En la planificación también se identifican los requerimientos de los clientes, las restricciones, las tareas a realizar, las dependencias entre ellas, los hitos que permitan medir el avance del proyecto para establecer los plazos de entrega. Además, se establecen mecanismos de seguimiento y control para monitorear el progreso del proyecto.

Es importante que la planificación de un proyecto sea realista para identificar los posibles riesgos u obstáculos que se puedan presentar en el proceso de desarrollo con el fin de definir planes de contingencia que hagan frente a estos problemas que surjan y hacer los ajustes necesarios a medida que avance el proyecto.

Es fundamental que la planificación se comunique de manera clara al equipo involucrado en el proyecto, para que todos trabajen de acuerdo con los objetivos y plazos establecidos.

- **Diseño**

En la fase de diseño se elabora una descripción detallada de cómo se construirá el software, las tecnologías, herramientas a utilizar, la infraestructura con la que cuenta la organización y se especifica el comportamiento que tendrá el software a desarrollar. El principal objetivo del diseño es brindar una solución a los requerimientos establecidos en la fase de análisis.

Durante la etapa de diseño de software se llevan a cabo los siguientes procesos:

Diseño arquitectónico: se establece la estructura del software que incluye los módulos y las interfaces que lo componen.

Diseño detallado: se define la funcionalidad de componentes y la comunicación entre ellos, así como los algoritmos y estructura de datos necesarios.

Diseño de interfaz de usuario: se diseña la disposición de los elementos en pantalla y el flujo de interacción con el usuario.

Diseño de prueba: se definen los casos de prueba para verificar que el software cumpla de manera correcta con los requerimientos del usuario.

- **Implementación**

En la fase de implementación el equipo de desarrollo analiza los requerimientos del usuario para llevar a cabo la construcción mediante la codificación e integración de las diferentes partes del código, utilizando un lenguaje de programación específico y herramientas de desarrollo, se realiza la configuración de hardware y software para su ejecución y se realiza la integración y las pruebas unitarias para verificar que cada módulo o componente funcionen correctamente.

Cualquier error que se realice en este proceso puede afectar en la calidad del software, por lo que es importante realizar pruebas exhaustivas para garantizar que el software cumple con lo establecido y su correcto funcionamiento.

- **Pruebas**

La fase de pruebas en el desarrollo de software consiste en la ejecución de una serie de verificaciones en el software para comprobar su correcto funcionamiento y detectar posibles errores o fallas en su desempeño antes de su implementación en el entorno de producción. Además, el proceso de pruebas asegura que el software cumpla con los requerimientos establecidos por el usuario y que su funcionalidad corresponda a las especificaciones definidas.

Es importante que el proceso de pruebas se lleve a cabo de una manera exhaustiva y detallada para garantizar la calidad del producto final.

- **Despliegue**

El despliegue es la última etapa del ciclo de vida del desarrollo del software en la cual se libera la última copia del software, se realiza la instalación y configuración para poner en funcionamiento el software en el entorno del usuario.

Es importante garantizar el correcto funcionamiento del software en el ambiente de producción y proporcionar mantenimiento y soporte continuo después del despliegue, así como manuales de usuario.

1.1.3. Aseguramiento de la calidad

El aseguramiento de la calidad en el desarrollo de software ha sido un tema que han abordado diversos autores (Pressman, 2014), en diferentes modelos considerándolo como un enfoque en la planificación, gestión y aplicación de prácticas, técnicas de calidad efectivas en el desarrollo de software que garanticen que el software cumple con los estándares de calidad, con el objetivo de entregar productos que sean confiables y que resulten de alta calidad (Sommerville, 2015).

El principal objetivo del aseguramiento de la calidad es garantizar que el software cumple con los requisitos del usuario, que funcione correctamente, que se adapte y resulte fácil de mantener. También busca mejorar la productividad y eficiencia del desarrollo del software, minimizando los posibles riesgos y problemas que puedan surgir en la implementación.

El aseguramiento de la calidad en el desarrollo de software conlleva la aplicación de prácticas de revisión de código, pruebas de software, seguimiento de defectos y documentación adecuada, un análisis estático y control de versiones. Dichas actividades se deben llevar a cabo desde la etapa de planificación y diseño hasta el mantenimiento.

1.1.4. Estándares de calidad

Los estándares juegan un papel importante en el aseguramiento de la calidad del desarrollo de software, debido a que estos brindan un conjunto de normas, procesos y mejores prácticas

que garantizan que el software cumpla con los requisitos de calidad y la eficiencia del proceso de desarrollo.

Los estándares en el desarrollo de software proporcionan un marco de referencia para el proceso de desarrollo, la implementación y las pruebas del software, asegurando que se sigan prácticas adecuadas y se cumplan los criterios de calidad establecidos. Los estándares abarcan diversos aspectos dentro del desarrollo de software, como la gestión del proyecto, la documentación, la usabilidad y la fiabilidad (Sommerville, 2015).

Existen diferentes modelos y estándares que se clasifican según su alcance o ámbito de aplicación:

- ***Quality Management*** **(Normas enfocadas de forma global a la gestión de Calidad)**

ISO 9001 - Este estándar establece requisitos para la gestión de la calidad en una organización. Su enfoque principal es la satisfacción del cliente, mediante la mejora continua, el liderazgo, la participación del equipo y la toma de decisiones basada en evidencias.

- ***Quality Assurance*** **(Normas enfocadas al Aseguramiento de la Calidad)**

ISO/IEC 25000 (*SQuaRE*) - Es un estándar de Calidad del Software de Requisitos y evaluación que define un conjunto de características de calidad y métricas asociadas, también define procesos para la evaluación y certificación de productos de software.

- **Normas enfocadas a las pruebas de calidad**

IEEE 829 - También conocida como *Standard for Software Test Documentation*, este estándar establece los formatos del proceso de pruebas en el desarrollo de software, como es el plan de pruebas, casos de pruebas, informes de pruebas, asegurando la claridad en los entregables.

1.1.5. Modelos de calidad para el desarrollo de Software

A lo largo de la historia del desarrollo de software, se han presentado problemas como el retraso en los plazos de entrega, la falta de calidad en el software y la dificultad de gestionar

un proyecto, lo que llevó a buscar nuevas formas de trabajo más estructuradas y mejor organizadas. Como resultado surgieron diferentes metodologías (Pressman, 2020), para la gestión y desarrollo de proyectos que contienen diferente orden cronológico de las fases de desarrollo de software con el objetivo de:

1. Organizar el trabajo del equipo de desarrollo de manera estructurada para lograr mayor eficiencia y productividad.
2. Controlar el progreso del proyecto mediante un marco de trabajo para identificar y corregir posibles problemas o riesgos.
3. Brindar calidad del software haciendo uso de estándares y prácticas que garanticen el cumplimiento de los requerimientos del cliente.
4. Fomentar una comunicación efectiva entre el equipo involucrado en el desarrollo del proyecto y el cliente.

1.1.5.1. Modelo de cascada.

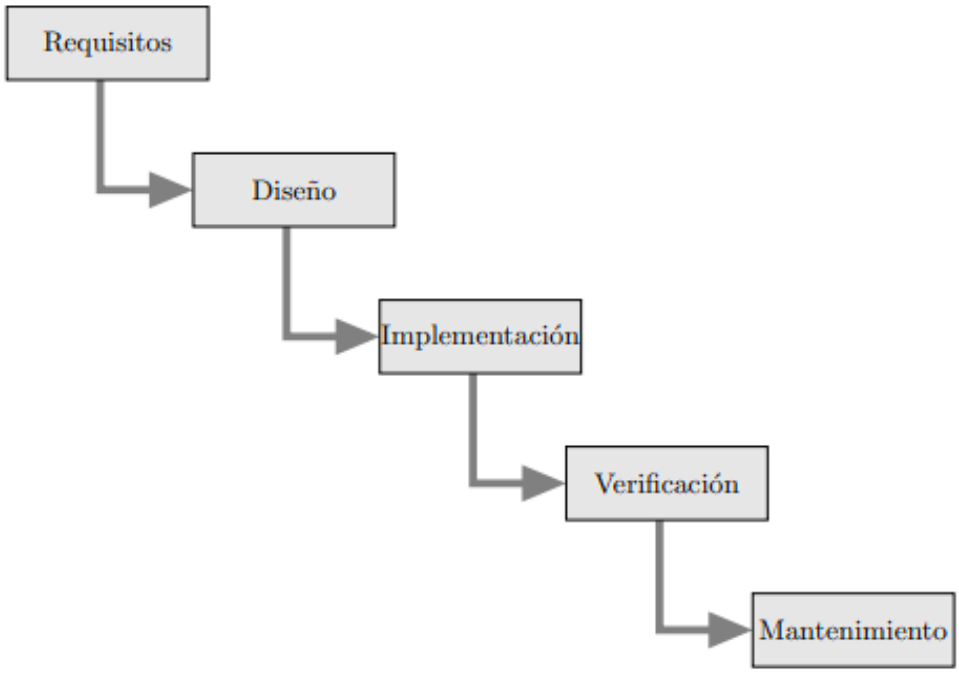
El modelo de cascada fue propuesto en 1970 por Winston W. Royce. Este modelo se enfoca en completar una fase antes de comenzar la siguiente. El proceso sigue una secuencia lineal, donde cada fase depende de la fase anterior y es seguida por la siguiente fase, con una retroalimentación limitada entre ellas y sin retroceder a una etapa anterior como se aprecia en la Figura 1. El modelo Cascada realiza la fase de pruebas después de la implementación (Royce, 1970).

Los principales beneficios del desarrollo cascada son:

1. *Planificación exhaustiva*: se realiza una planificación detallada al comienzo del proyecto definiendo todos los requisitos y etapas del desarrollo de software.
2. *Enfoque secuencial*: cada etapa del desarrollo se lleva a cabo de manera secuencial donde una fase debe completarse antes de pasar a la siguiente.
3. *Documentación extensa*: se presta especial atención a la documentación de requisitos, diseño, pruebas y otros aspectos del proyecto.

Figura 1

Modelo de cascada



1.1.5.2. Modelo Iterativo.

El modelo iterativo fue propuesto por Barry Boehm en 1980. Su enfoque iterativo se basa en el proceso de dividir el desarrollo de software en ciclos repetitivos de diseño, análisis, implementación y pruebas, permitiendo retroalimentación y adaptación continua a lo largo del proceso de desarrollo. Al final de cada iteración se produce una versión funcional del software (Boehm, 1981). Las etapas del modelo iterativo son las descritas en la Figura 2.

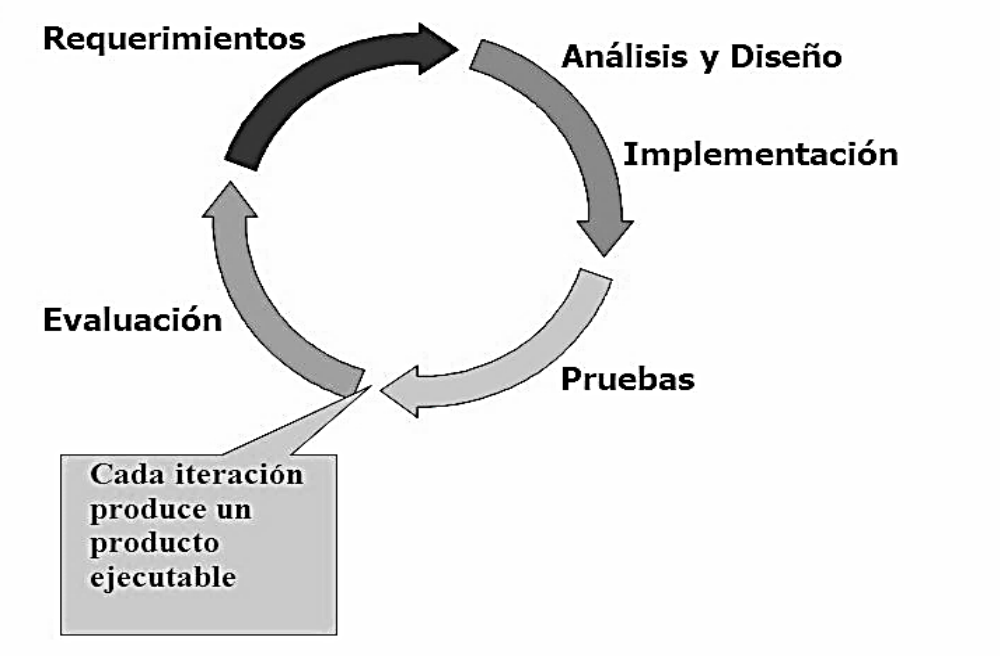
Algunos aspectos del modelo iterativo incluyen:

1. Ciclos de desarrollo con actividades en todas las fases del proceso, como la recopilación de requisitos, el diseño, la implementación y las pruebas.
2. Enfoque incremental permitiendo la entrega de incrementos funcionales en cada iteración.

3. Retroalimentación continua, que promueve una comunicación constante con el cliente, lo que facilita la detección temprana de problema y la realización de mejoras continuas.
4. Mejora continua, ya que cada iteración es una oportunidad para aprender y mejorar.

Figura 2

Modelo iterativo



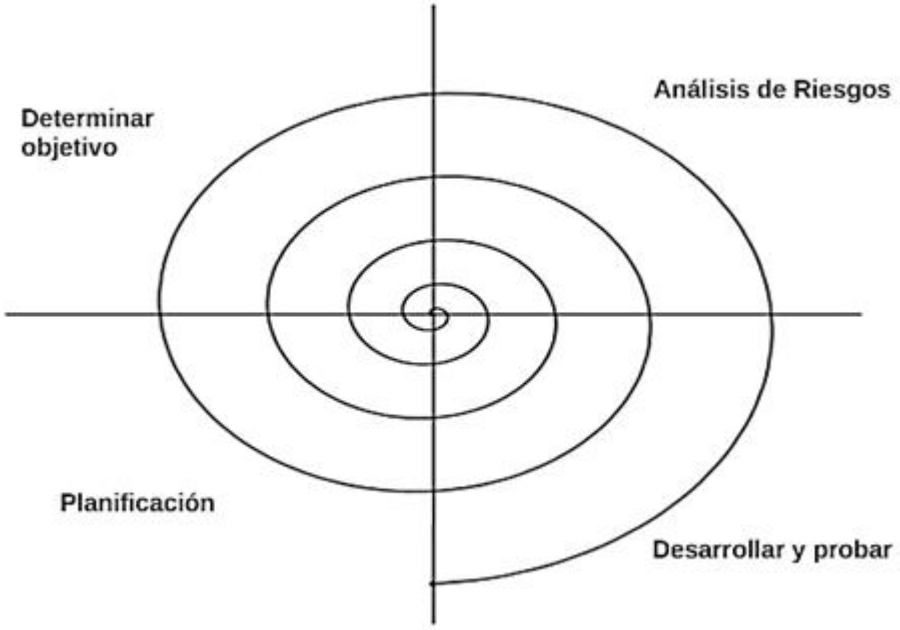
1.1.5.3. Modelo Espiral.

El modelo espiral fue propuesto en 1986 por Barry Boehm, donde combina elementos del modelo en cascada y el modelo iterativo. Este modelo se centra en identificar y gestionar riesgos, la flexibilidad y adaptabilidad, así como la entrega incremental del sistema basado en un ciclo repetitivo de planificación, análisis de riesgo, desarrollo y evaluación a lo largo del proyecto. Al final de cada ciclo, se realiza una revisión de los riesgos y se toma una decisión con base en la evaluación de dicha revisión, de si continuar con la siguiente iteración y se ajusta el plan (Boehm, 1986).

El modelo se representa como una espiral, donde cada ciclo de desarrollo se divide en cuatro principales actividades: planificación, análisis de riesgo, desarrollo y evaluación como se describe en la Figura 3. En cada ciclo, se identifican los objetivos y los riesgos del proyecto y se realiza una evaluación exhaustiva para determinar si el proyecto está por buen camino y si es necesario realizar ajustes.

Figura 3

Modelo Espiral



1.1.5.4. Modelo ágil.

El modelo ágil se desarrolló en la década de los 60s, en la compañía de IBM, donde se realizaron los primeros desarrollos iterativos. El enfoque ágil se centra en la entrega rápida e incremental de software funcional y a diferencia de los enfoques tradicionales no sigue una planificación lineal (Cockburn, 2001).

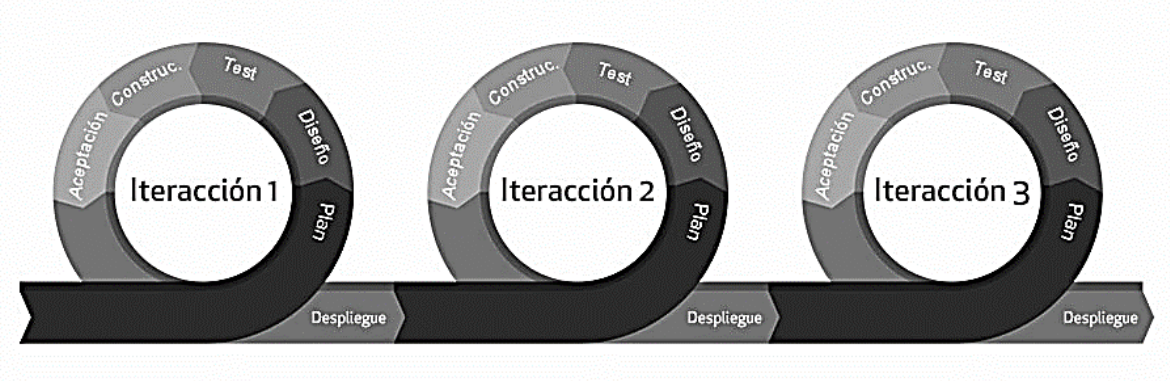
Una de las principales características del modelo ágil es su flexibilidad, adaptabilidad al cambio, su enfoque colaborativo ya que promueve la comunicación constante con el cliente

y los involucrados en el desarrollo del software, permite cambios y ajustes en función de los requisitos cambiantes del proyecto.

El modelo ágil, representado en la Figura 4, se basa en la entrega continua de software funcional en ciclos cortos, conocidos como iteraciones o *sprints*. Cada iteración resulta en una versión funcional del producto que puede ser evaluado y probado por los *stakeholders* (clientes).

Figura 4

Modelo ágil



1.1.5.5. Manifiesto ágil.

El Manifiesto Ágil fue redactado en 2001 por un grupo de ingenieros de software que basándose en su experiencia brindaron una respuesta a aquellas limitaciones y desafíos que se presentaban en las metodologías de desarrollo de software tradicionales, pues en la década de 1990, los enfoques de desarrollo de software basados en procesos rígidos y planificación detallada ya estaban mostrando dificultades para adaptarse a los cambios rápidos y constantes en los requisitos del cliente, así como para entregar software de alta calidad en plazos ajustados (Beck, y otros, 2001).

De acuerdo con los autores Beck y otros (2001), el manifiesto ágil establece cuatro valores fundamentales:

1. Individuos e interacciones sobre procesos y herramientas.
2. Software funcionando sobre documentación exhaustiva.

3. Colaboración con el cliente sobre negociación contractual.
4. Respuesta al cambio sobre seguir un plan.

El Manifiesto Ágil también establece 12 principios que guían la forma de trabajar en un proyecto ágil:

1. Priorizar la satisfacción del cliente mediante la entrega temprana y continua de software de valor.
2. Aceptar los cambios en los requisitos, incluso en etapas tardías del desarrollo.
3. Entregar software funcional en periodos cortos de tiempo entre entregas y de manera regular.
4. Fomentar la colaboración diaria entre los involucrados del proyecto.
5. Generar confianza entre el equipo, brindarles el entorno y los recursos necesarios para hacer bien su trabajo.
6. Las conversaciones directas son la forma más efectiva de transmitir información dentro de un equipo ágil.
7. Brindar mayor valor al software funcional que la documentación exhaustiva.
8. Mantener un ritmo constante y sostenible de trabajo a lo largo del proyecto.
9. Fomentar la atención continua a la calidad técnica y al buen diseño para mejorar la agilidad del equipo.
10. Buscar la simplicidad en los procesos y maximizar la cantidad de trabajo no realizado.
11. El equipo se puede auto organizarse y tomar decisiones sobre cómo realizar el trabajo.
12. Reflexionar sobre el trabajo realizado y buscar mejorar y ajustar los procesos de manera continua.

A partir de las ideas redactadas en el manifiesto ágil, se han desarrollado diversas metodologías ágiles como SCRUM, *Extreme Programming* (XP), Kanban, Lean, entre otras.

1.1.5.6. Kanban.

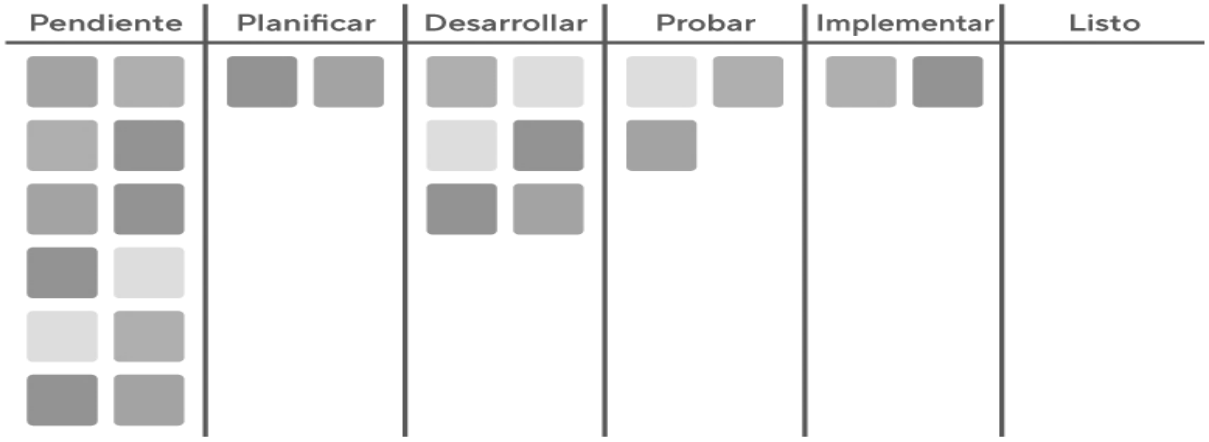
Kanban es una metodología de gestión visual utilizada para optimizar y mejorar la eficiencia en el flujo de trabajo. Se originó principalmente en Japón en la producción de Toyota y se ha adaptado al desarrollo de software. Pero fue en 2004 cuando David J. Anderson introdujo Kanban al desarrollo de software para presentar el flujo de trabajo y las tareas pendientes a su equipo de desarrollo en Microsoft (Anderson, 2010).

Kanban significa “Tarjeta o tablero visual”. El método Kanban es un enfoque que divide el trabajo en pequeñas partes representadas en tarjetas que se colocan en un tablero, como se muestra en la Figura 5. Estas tarjetas contienen información variable como requisitos, estimación temporal, recursos necesarios, entre otros detalles relevantes con el fin de visualizar el trabajo, las prioridades, los tiempos y los estados de las tareas.

Algunos autores no consideran Kanban como una metodología, sino más bien una herramienta complementaria que puede utilizarse en conjunto con otras metodologías como la Programación Extrema (XP) o SCRUM, ya que no proporciona un marco de trabajo completo y estructurado para la gestión de proyectos.

Figura 5

Tablero Kanban



1.1.6. La transformación ágil a través de SCRUM

En el desarrollo de software tradicional se han identificado diversos problemas entre los que se encuentra la rigidez en los procesos, la falta de transparencia, el *feedback* tardío, la escasa flexibilidad y la falta colaborativa, un flujo secuencial desde el análisis hasta las pruebas, una planificación predictiva, lo que dificulta adaptarse a los cambios rápidos y constantes en los requisitos del cliente. Dicha rigidez impide una respuesta ágil a las necesidades del negocio en sí (Cohn V. , 2010).

Además, esta estrategia retrasa el *feedback*, el aprendizaje y el retorno de inversión, ya que el software funcional no está disponible hasta las etapas finales del proyecto, lo que limita la transparencia y visibilidad sobre el progreso del proyecto. Esto dificulta la detección temprana de problemas tanto en el desarrollo como en la toma de decisiones.

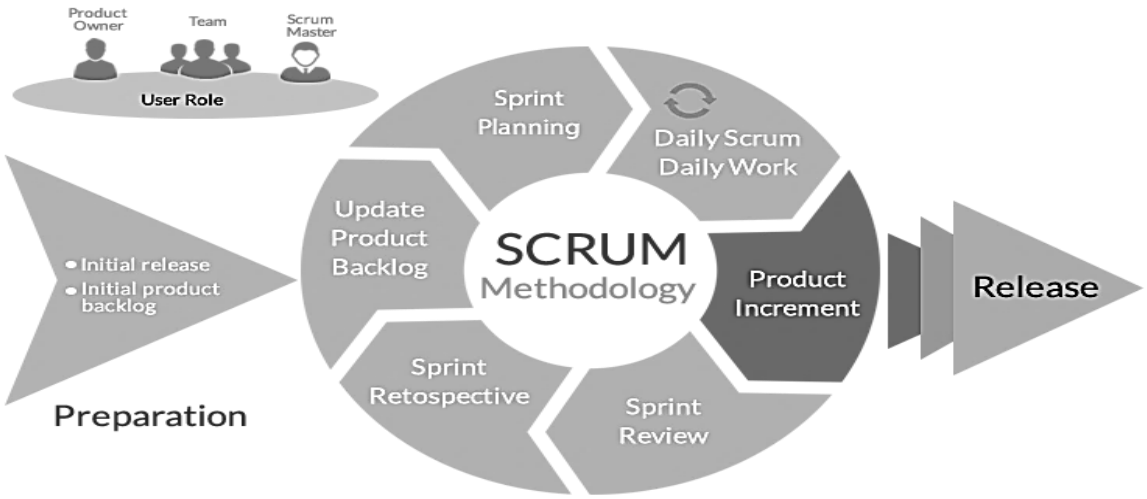
En respuesta a estos desafíos, se han desarrollado enfoques ágiles como SCRUM, que introduce conceptos simples, promueve la flexibilidad, la transparencia, el *feedback* temprano y la colaboración entre los miembros del equipo. Estos enfoques permiten un desarrollo más ágil y eficiente, centrado en ciclos cortos e iterativos de *feedback*, en la entrega temprana de valor al cliente y en la adaptación continua a medida que se van descubriendo nuevas necesidades y desafíos en el proceso de desarrollo de software. También permite un retorno de inversión más temprano y se reduce el riesgo asociado al desarrollo de software (Sutherland, 2017).

1.1.6.1 SCRUM.

SCRUM es una metodología ágil que sigue un ciclo de vida iterativo e incremental como se muestra en la Figura 6. Principalmente se basa en la colaboración efectiva y mejora continua. Se centra en la entrega rápida e incremental de software funcional y enfatiza la flexibilidad, adaptabilidad y colaboración entre el equipo de desarrollo y el cliente (Sutherland, 2017).

Figura 6

Ciclo de vida de SCRUM



1.1.6.2 Prácticas en SCRUM.

Las prácticas SCRUM están diseñadas para fomentar la transparencia, la inspección y la adaptación continua, lo que a su vez permite a los equipos responder de manera efectiva a las demandas cambiantes y a los requisitos del cliente (Sutherland, 2017). Las prácticas en las que se basa *SCRUM* para abordar los desafíos comunes de desarrollo son las siguientes:

- ***Burn-down****:* es una gráfica que permite medir el progreso del trabajo en un sprint de un proyecto, monitorear el trabajo pendiente y/o desviaciones en el progreso para abordarlo de manera proactiva

- ***Pre Game****:* es una etapa de preparación donde se establece la dirección del desarrollo. Se analizan los relatos de usuarios, se priorizan y se clarifican para formar el *Backlog*, que luego se utilizará en la planificación y ejecución de los *Sprints* en un enfoque ágil como SCRUM.

- ***Sprints***: son iteraciones de tiempo, generalmente de 2 a 4 semanas. Durante este tiempo el equipo de desarrollo trabaja para completar un conjunto de elementos del *Product Backlog* y entregar incremento de un software funcional al finalizar el *Sprint.*

En cada *sprint* se llevan a cabo las etapas de análisis de requisitos, diseño, implementación y pruebas.

- ***Product Backlog****:* es una lista en la que se priorizan los requisitos funcionales que deben realizarse en el desarrollo del software y los cambios que puedan surgir. Las tareas se priorizan en función de su importancia y valor para el cliente, estas se van actualizando a medida que se obtiene nueva información o se realizan cambios en los requisitos.

- ***Planning de Sprint***: al comienzo de cada *Sprint*, el equipo de desarrollo y el *Product Owner* se reúnen para seleccionar las historias de usuario del *Product Backlog* que se comprometen a completar durante el *Sprint* y estiman las tareas necesarias.

- ***Reuniones diarias***: conocidas como *Daily Stand-ups* o *Daily SCRUM* son reuniones diarias que se llevan a cabo durante el *Sprint*. En estas reuniones, cada miembro del equipo comparte brevemente su progreso, los obstáculos que enfrenta y la planificación para el siguiente día. Durante la reunión diaria, el equipo de desarrollo regularmente actualiza el tablero Kanban o cualquier otra herramienta de seguimiento visual que utilicen, lo que permite tener una representación visual del progreso para identificar las posibles desviaciones.

- ***Revisión de Sprint* (*Sprint Review*)**: es una reunión que se lleva a cabo al finalizar cada *Sprint* en la que el equipo de desarrollo presenta el trabajo realizado durante el *Sprint* al *Product Owner* y a otras partes interesadas. Se realiza una demostración del producto funcional para su revisión y se considera la retroalimentación del *Product Owner* y los S*takeholders* para mejorar el próximo *Sprint*. Durante la revisión del *Sprints* también se pueden identificar nuevos requerimientos o cambios en el *Product Backlog* con base en las necesidades del cliente.

- ***Retrospectiva de Sprint* (*Sprint Retrospective*)**: en la *Retrospectiva de Sprint* el equipo de desarrollo se reúne para discutir qué funcionó bien y qué no, qué se puede hacer para mejorar, con el objetivo de permitir al equipo *SCRUM* reflexionar sobre el *sprint* que ha finalizado e identificar prácticas, herramientas, comunicación y

colaboración de equipo que permitan mejoras continuas en el proceso de trabajo. Al final de la retrospectiva, el equipo acuerda las acciones de mejora que se implementarán en el próximo *Sprint*. Estas acciones se agregan al *Backlog del Sprint* o al *Product Backlog* y se priorizan según su importancia.

1.1.6.3. Roles en SCRUM.

El marco de trabajo *SCRUM* (Sutherland, 2017), se caracteriza por sus roles clave:

- ***Product Owner***: es el responsable de asegurar que se cumplan los objetivos del proyecto y de definir y priorizar los elementos del *Product Backlog*, representando los intereses del cliente. Esto implica identificar y capturar los requisitos del cliente, desglosando las historias de usuario.

- ***SCRUM Máster***: es el facilitador del equipo, facilita las reuniones, es responsable de garantizar que se sigan las prácticas y tiempos establecidos por *SCRUM*, identifica y elimina los obstáculos que se puedan presentar durante el desarrollo del producto.

- ***Equipo de Desarrollo***: son los encargados de desarrollar y entregar el incremento del producto funcional al final de cada *Sprint*. Es multidisciplinario ya que está compuesto por profesionales con diversas habilidades. Pueden incluir desarrolladores de software, diseñadores, *tester*, analistas, entre otros.

- **QA (*Quality Assurance*)**: aunque en SCRUM no esté considerado formalmente el rol de los *QA* o *tester*, este rol desempeña un papel importante en los equipos *SCRUM* ya que su principal responsabilidad es garantizar la calidad del software mediante la identificación y corrección de defectos, la ejecución de pruebas exhaustiva y la verificación de que el producto cumpla con los requisitos y estándares establecidos.

En la actualidad diversas organizaciones han considerado cada vez más la introducción de este rol a su equipo de trabajo debido a la creciente importancia de la calidad del software, la complejidad de los sistemas, la necesidad de prevención de problemas, el enfoque en la mejora continua y los cambios en la percepción del rol de *QA*.

1.1.7. Validación y verificación para calidad del desarrollo de software

La validación y verificación en el desarrollo son dos procesos de gran importancia para asegurar la calidad y conformidad de los requisitos del producto. A demás estos procesos reducen problemas en etapas avanzadas, mejoran la satisfacción del cliente al entregar un software confiable y eficiente, y aseguran que se cumplan estándares y regulaciones, lo que reduce riesgos y fortalece la reputación de la organización en la industria del desarrollo de software (William, 2008).

Verificación: se enfoca en evaluar si el software se está construyendo correctamente y con base a los requisitos establecidos por el cliente. Este proceso implica examinar el código, la documentación, realizar pruebas y otros elementos relevantes del proyecto diseñados para garantizar la calidad del desarrollo del software y del mismo producto final.

Validación: se centra en asegurar que el software que se está desarrollando satisfaga las necesidades del usuario final, asegurando que sea la solución adecuada para cumplir con los requisitos o problemas del cliente. Una validación del software se lleva mediante pruebas de usuario, pruebas de aceptación del cliente (UAT) y pruebas de sistema que garantizan el correcto funcionamiento del software en su entorno de uso real.

1.1.8. Pruebas para la calidad de Software

Las pruebas de software son un proceso importante en el desarrollo de aplicaciones para garantizar su calidad, el correcto funcionamiento e interacción con el usuario. En el equipo *SCRUM* son realizadas de manera continua por todo el equipo, con la participación y en colaboración con el *Tester* o *QA* para definir los casos de pruebas, crear escenarios de pruebas y la ejecución de pruebas necesarias (Cohn M. , 2009).

Las pruebas de software consisten en validar y verificar el software mediante la ejecución de diferentes pruebas y escenarios con el objetivo de reducir riesgos, detectar defectos, garantizar la calidad del software, verificar que se cumplan los requisitos y especificaciones establecidas, validar la usabilidad y experiencia del usuario, así como asegurar la estabilidad y confiabilidad del software.

1.1.8.1. Técnicas de pruebas.

Las técnicas de pruebas son diferentes enfoques o estrategias que se utilizan para identificar y diseñar casos de pruebas efectivos. Estas técnicas se clasifican en diferentes categorías según el enfoque seleccionado como se muestra en la Tabla 1.

Tabla 1

Técnica de pruebas

Pruebas de caja Negra	Pruebas de caja blanca	Pruebas basadas en experiencia
Funcionalidad	Estructura de código	Usabilidad
Interacción con el usuario	Camino de ejecución	Adecuación con el contexto
Manejo de Datos	Cobertura de código	Problemas de usabilidad
Flujo de control	Flujo de datos	Retroalimentación cuantitativa
Gestión de errores	Condiciones y bucles	Alineación con los objetivos del usuario
Seguridad	Manejo de excepciones	Escenarios de uso real
Entrada y salida del sistema	Rendimiento	Satisfacción del usuario

1.1.8.2. Pruebas de Caja negra.

Las pruebas de caja negra, o también conocidas como pruebas funcionales, permiten evaluar la entrada y salida del sistema sin conocer la estructura interna o el código fuente. Este tipo de prueba se centra en probar el software desde la percepción del usuario final. Las pruebas funcionales implican analizar las especificaciones para obtener los casos de pruebas necesarios (Lewis, 1999). Algunas de ellas se describen en los siguientes párrafos.

- **Pruebas de caso de uso**

En las pruebas de caso de uso se validan los escenarios de uso específico del software, siguiendo los pasos definidos en los casos de uso. Esto permite comprobar si el software se comporta según lo esperado por el usuario.

- **Pruebas de interfaz de usuario**

Se comprueba que los elementos de la interfaz funcionen correctamente, como botones, enlaces, formularios, entre otros, y se verifica la interacción del usuario con la interfaz.

- **Pruebas de entrada y salida**

Se valida la respuesta del software ante diferentes entradas de datos, verificando que las salidas sean correctas y coherentes.

- **Pruebas de integración**

En las pruebas de integración se comprueba el correcto funcionamiento de la interacción entre diferentes componentes o módulos del software. Estas pruebas evalúan si los distintos elementos del sistema trabajan de manera conjunta y se comunican adecuadamente.

- **Pruebas de regresión**

Se realizan pruebas para asegurarse de que las nuevas implementaciones o correcciones no hayan introducido errores en funcionalidades existentes.

1.1.8.3. Pruebas de Caja Blanca.

Las pruebas de caja blanca llamadas *pruebas estructurales* o *pruebas basadas en código*, se enfocan en examinar internamente la lógica y estructura del software. El principal objetivo de este tipo de pruebas es identificar errores en la implementación del software, con el fin de mejorar la calidad del software a través de validaciones exhaustivas del código (Offutt, 2008).

- **Pruebas unitarias**

Las pruebas unitarias se enfocan en verificar el correcto funcionamiento de las unidades individuales de código. Estas pruebas se realizan a nivel de funciones, métodos o clases y son fundamentales para asegurar la calidad y estabilidad del código durante el desarrollo del software.

- **Pruebas de flujo de control**

Se evalúa cómo fluye la ejecución del programa a través de las diferentes estructuras de control, como bucles, condiciones y ramificaciones.

1.1.8.4 Pruebas basadas en experiencia.

Las pruebas basadas en la experiencia son técnicas que se enfocan en el uso de la intuición, el conocimiento, el dominio y las experiencias pasadas del equipo de pruebas para diseñar casos de pruebas e identificar errores (Fewster, 2012).

- **Pruebas exploratorias**

También conocidas como *Testing exploratorio* se basan en explorar el software sin seguir un plan de pruebas predefinido. El *Tester* utiliza su conocimiento y experiencia. Este tipo de pruebas son útiles cuando hay limitación de tiempo y recursos para hacer pruebas más detalladas.

- **Pruebas de usabilidad**

Se centran en evaluar la facilidad de usuario y las expectativas de los usuarios finales para identificar posibles problemas de usabilidad.

1. *Preparación de las pruebas*: se configura el entorno y se preparan los datos necesarios.
2. *Ejecución de las pruebas*: se llevan a cabo las pruebas y se registran los resultados.
3. *Análisis de resultados*: se analizan los resultados para identificar errores o defectos, se documentan y se realiza un seguimiento del defecto.
4. *Retorno a las etapas de diseño y ejecución*: Si se detectan errores se regresa a las etapas anteriores.
5. *Cierre de las pruebas*: finalmente, se realiza un cierre de las pruebas con informes finales y lecciones aprendidas.

1.1.9. CMMI

CMMI fue desarrollado por el Instituto de Ingeniería de Software (SEI, por sus siglas en inglés, *Software Engineering Institute*), a partir del modelo CMM de mejora de procesos. El SEI propuso un marco de madurez de los procesos que permitiera evaluar a las empresas mediante modelos de buenas prácticas, métodos de entrenamiento y evaluación.

Si bien CMMI se originó en el contexto del desarrollo de software, ha evolucionado para abarcar diferentes áreas de negocio y sectores de la industria. Los modelos de CMMI se pueden aplicar en organizaciones que se dedican a la ingeniería de sistemas, servicios de Tecnologías de la Información (TI), fabricación, gestión de proyectos, entre otros (Chissis, Konrad, & Shrum, 2011). Los principales modelos CMMI se describen a continuación:

- **CMMI- DEV**: se centra en la mejora de procesos de desarrollo de software y sistemas, incluyendo actividades como análisis de requisitos, diseño, implementación, pruebas y gestión de configuración.
- **CMMI-ACQ**: se enfoca en los procesos de adquisición y gestión de proveedores, proporcionando pautas para la selección, contratación y gestión de proveedores externos.
- **CMMI-SVC**: se centra en la mejora de procesos relacionados con las organizaciones que prestan servicios, como la gestión de incidencias, la gestión de la capacidad, la gestión de configuración y la gestión de conocimiento.

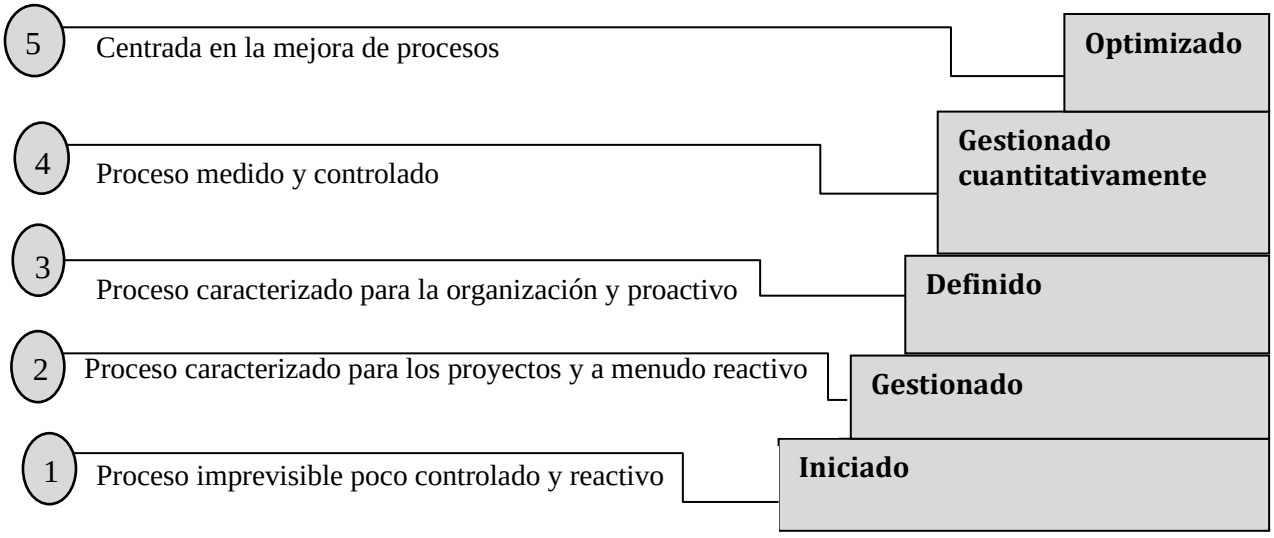
1.1.9.1. Niveles de Madurez de CMMI.

Los niveles de madurez de CMMI proporcionan un marco de referencia que permite a las organizaciones evaluar su madurez, la capacidad de gestionar los procesos y establecer metas para la mejora continua (Chissis, Konrad, & Shrum, 2011).

Los niveles de madurez se clasifican en virtud de la satisfacción de los objetivos genéricos y específicos de las áreas de procesos en el nivel correspondiente y el nivel anterior, como se muestra en la Figura 7.

Figura 7

Niveles de madurez CMMI



Nivel 1: depende de las habilidades y conocimientos del equipo, los procesos no son formales, con enfoque *ad hoc*, e imprescindibles.

Nivel 2: de establecen procesos básicos de gestión de proyectos. Se definen prácticas y se documentan los procedimientos necesarios para ejecutar los proyectos de manera controlada.

Nivel 3: las organizaciones tienen procesos definidos y estandarizados para la gestión del proyecto. Los proyectos tienen una gestión de proyectos que se adapta a los requisitos de éste. Se establecen métricas para medir el desempeño y la calidad.

Nivel 4: se establece una capacidad cuantitativa para medir y controlar los procesos. Se recopilan datos y se utilizan técnicas estadísticas para analizar el rendimiento de los procesos y la toma de decisiones.

Nivel 5: las organizaciones utilizan técnicas de mejora y optimización para identificar áreas de mejora, implementar cambios y evaluar el impacto de dichos cambios en el desempeño y la calidad.

1.1.9.2. Área de Procesos de CMMI.

Las áreas de procesos son un conjunto de prácticas esenciales para lograr una mejora de procesos en una organización. Dichas áreas están diseñadas para abordar aspectos fundamentales del desarrollo CMMI DEV, la prestación de servicios CMMI SVC y la adquisición de productos o servicios CMMI ACQ (Chissis, Konrad, & Shrum, 2011).

A continuación, se describen 18 de las áreas base que cubren los conceptos fundamentales para la mejora de procesos de las áreas de interés de CMMI-DEV, CMMI-SVC y CMMI-ACQ:

- ***Configuration Management* (CM):** establece y mantiene la integridad de los productos y activos de trabajo en todo el ciclo de vida del desarrollo de software.
- ***Decision Analysis and Resolution* (DAR)**: ayuda a la toma de decisiones basadas en análisis y datos con el fin de resolver problemas durante el desarrollo de software.
- ***Project Management* (PM)**: asegura que los proyectos se planifiquen, ejecuten y controlen utilizando prácticas de gestión establecidas y efectivas.
- ***Measurement and Analysis* (MA)**: establece métricas para evaluar la calidad, la eficiencia y efectividad del proceso y del producto de software.
- ***Organizational Process Definition* (OPD)**: define procesos que sean bien documentados y adecuados para alcanzar los objetivos de la empresa.
- ***Organizational Process Focus* (OPF)**: asegura que la organización esté comprometida con la mejora continua de sus procesos.
- ***Organizational Process Performance* (OPP)**: mide y analiza el rendimiento de los procesos organizacionales con el objetivo de mejorar continuamente su eficiencia.
- ***Organizational Training* (OT)**: se enfoca en la planificación y ejecución de programas de capacitación al personal a fin de mejorar sus habilidades respecto al proceso.
- ***Project Monitoring and Control* (PMC)**: da seguimiento a los proyectos para garantizar que se alcancen los objetivos establecidos.

- ***Project Planning*** **(PP)**: establece y mantiene un plan de proyecto que guíe la ejecución del trabajo y garantice el cumplimiento de los objetivos establecidos.
- ***Process and Product Quality Assurance*** **(PPQA)**: asegura que los procesos y productos del proyecto cumplan con los estándares y requisitos establecidos.
- ***Risk Managemen*****t (RSKM)**: identificar, analizar y abordar los riesgos que puedan afectar el éxito del proyecto.
- ***Product Integration*** **(PI)**: se enfoca en la interacción de los componentes y subsistemas del producto para crear un producto coherente y completo.
- ***Requirements Development*** **(RD)**: Se enfoca en el proceso de capturar, analizar y definir los requisitos del sistema que se va a desarrollar.
- ***Technical Solutions*** **(TS)**: selección de soluciones técnicas apropiadas para cumplir con los requisitos del sistema.
- ***Validation*** **(VAL)**: se enfoca en la evaluación y confirmación de que un producto o componente cumple con los requisitos establecidos antes de su entrega al cliente o usuario final.
- ***Verification*** **(VER)**: revisar los productos de trabajo y los resultados de los procesos para asegurarse de que se han cumplido los requisitos y se han seguido los estándares establecidos.

1.1.9.3. SCAMPI Método de evaluación basado en CMMI.

El método SCAMPI es utilizado para evaluar las prácticas y procesos implementados por una organización y determinar su grado de madurez en términos de las áreas clave de CMMI (Glazer, 2015).

De igual forma, el método SCAMPI basa su evaluación principalmente en las evidencias sobre el cumplimiento de las prácticas y procesos del modelo de CMMI. Dichas evidencias se pueden recopilar de documentación, artefactos resultantes de implementación y entrevistas con los roles encargados de implementar o usar los procesos.

Asimismo, el método presenta tres tipos de métodos de evaluación (*Class* A, *Class* B y *Class* C), que se aplican de acuerdo con el objetivo de la evaluación, como se muestra en la Tabla 2.

- **SCAMPI *Class* A**: es un tipo de evaluación más extenso y riguroso que se realiza para obtener una certificación oficial de CMMI. Requiere una revisión profunda de la documentación y la evidencia de implementación de los procesos.
- **SCAMPI *Class* B y *Class* C**: son tipos de evaluaciones más limitados en alcance que pueden ser utilizados para obtener una visión general de la madurez de la organización sin buscar una certificación formal.

Tabla 2

Comparativa de los métodos de Evaluación SCAMPI

Características	***Class* A**	***Class* B**	***Class* C**
Cantidad de evidencia objetiva requerida	Alta	Media	Baja
Calificación generada	Si	No	No
Recursos requeridos	Alta	Media	Baja
Tamaño del equipo evaluador	Grande	Mediano	Pequeño
Líder del equipo evaluador	Persona autorizada y acreditada.	Persona autorizada o con formación y experiencia.	Persona con formación y experiencia.

El modelo de referencia SCAMPI consta de tres fases:

- Planificación y preparación para la evaluación.
- Conducción de la evaluación.
- Reporte de resultados.

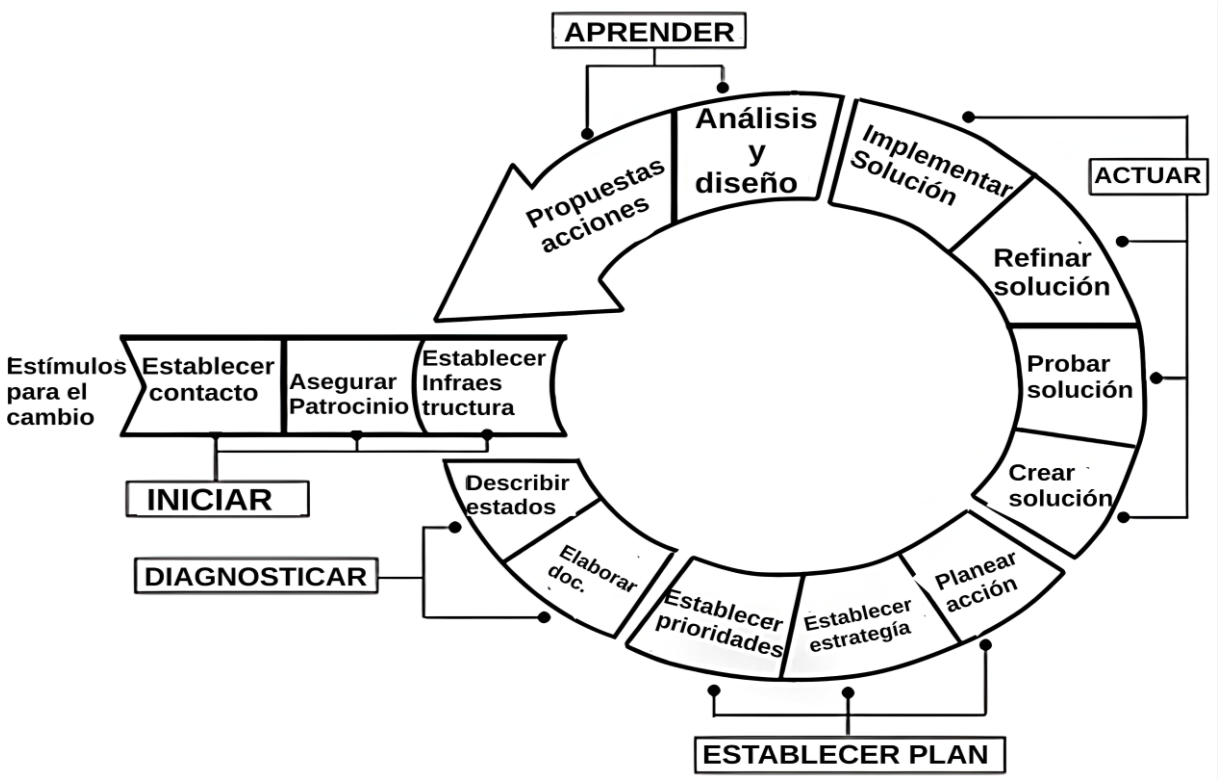
1.1.9.4. IDEAL.

A la hora de implementar una mejora de procesos como CMMI en una organización se puede presentar una gran brecha entre simplemente poseer el modelo y realmente aplicar todas sus mejores prácticas de manera efectiva. Debido a esto surge la necesidad de encontrar una estrategia que permita a una empresa de software implementar y cumplir con todas las características propuestas por el modelo CMMI, por lo que el modelo IDEAL se propone para cubrir dicha necesidad (SEI, 1997).

El modelo IDEAL, representado en la Figura 8, es una metodología de mejora organizacional que actúa como una guía para iniciar, planificar e implementar acciones que apunten a mejorar los procesos. Es una respuesta directa a la pregunta fundamental de cómo llevar los procesos de una organización a alinearse con las características y mejores prácticas del modelo CMMI.

Figura 8

Metodología IDEAL



Nota: el modelo IDEAL garantiza una implementación estructurada y efectiva de CMMI.

El modelo ideal se basa en 5 fases que se describen en la Tabla 3.

Tabla 3

Fases del modelo IDEAL

Letra inicial	Nombre de la fase	Descripción
I	Iniciar	Esta fase comienza con la aplicación de un SCAMPI informal, especialmente adaptado a la naturaleza y contexto de la empresa.
D	Diagnosticar	Identificar dónde está posicionada la Organización y a dónde quiere llegar.
E	Establecer	Planificar las acciones a ejecutar para alcanzar el estado deseado.
A	Actuar	Ejecutar el Plan.
L	Aprender	Aprender de la experiencia realizada y visualizar oportunidades de mejora.

Nota: el modelo IDEAL establece las acciones correctivas necesarias para abordar problemas y mejoras necesarias

1.2. Estado del Arte

En esta sección se describirán los trabajos realizados por otros autores, quienes han buscado identificar, analizar y mejorar los procesos utilizados en el desarrollo de software con el fin de garantizar la calidad del mismo y de su proceso de desarrollo.

En la creciente industria del desarrollo de software la calidad y la eficiencia a la hora de adaptarse son factores muy importantes para lograr el éxito de las organizaciones y el producto final. En la búsqueda de la mejora de la calidad de los productos y servicios del desarrollo de software, una adaptación rápida a los cambios continuos de los requisitos del cliente, así como la rápida entrega que brinde una mayor satisfacción del cliente, se han realizado estudios para cumplir con estos objetivos.

En estos estudios la implementación de las prácticas agiles como SCRUM ha sido relevante por su gran capacidad para aumentar la flexibilidad y la capacidad de respuesta ante los cambios. Al mismo tiempo, los modelos de mejora de procesos como CMMI han resultado un elemento clave para asegurar la calidad y eficiencia en el desarrollo de software.

En la literatura actual estos estudios ofrecen una visión de como esta metodología y proceso se integran y complementan entre sí, en las siguientes secciones se describirán algunos trabajos relacionados.

1.2.1. Guía ágil para el desarrollo de software en entidades públicas en Colombia empleando CMMI 2.0 y SCRUM

Dicho estudio, realizado por Calderón, Lascarro, Mattos y Mercado (2022), ofrece una guía orientada a las entidades públicas nacionales o territoriales en Colombia, que busca conservar la buenas prácticas y agilidad relacionando cada una de las actividades de CMMI 2.0 con SCRUM, los lineamientos de gobierno digital y las políticas nacionales de Colombia. El resultado de su investigación es una guía denominada GIDSEO que integra la calidad de software, la gestión de proyectos por medio de políticas nacionales y la metodología de desarrollo ágil SCRUM, para la cual se determinaron cuatro componentes: lineamientos, prácticas, roles y artefactos.

Entre las conclusiones de los autores, señalan haber encontrado evidencia de que el uso de SCRUM no cubre completamente con las necesidades o lineamientos en el desarrollo de software en relación con las normativas nacionales de Colombia, además, como parte de la comparación de SCRUM con el modelo CMMI versión 2.0, hallaron ausencias de productos de trabajo que abarquen ciertas prácticas dentro de CMMI en el nivel 3 (Calderón Jaraba, Lascarro Altamiranda, Mattos Badillo, & Mercado Cervantes, 2022).

1.2.2. Manual para la implementación de CMMI a nivel de capacidad 2 con SCRUM en el área de desarrollo de software

Otro ejemplo es el trabajo presentado por García López y Linares Valencia (2020), cuyo objetivo fue establecer el nivel de capacidad 2 de CMMI con SCRUM. En este manual se identifica la documentación que se requiere para el nivel 2 de CMMI, así como los productos esperados que pueden obtenerse a través de las diferentes actividades de SCRUM realizando una combinación de ambas prácticas y herramientas. Como resultado de la investigación, los autores obtuvieron una matriz de convergencia que les permite establecer las actividades de SCRUM que intervienen en las áreas de proceso para la obtención de los productos esperados

de CMMI, además de haber tomado en cuenta observaciones y recomendaciones realizadas por una empresa pública y una privada, considerando que la empresa tenga personal que cuente con conocimiento sobre el modelo de mejoras CMMI y SCRUM.

Entre las conclusiones de los autores, resaltan que el seguimiento del manual facilita la aplicación de CMMI en combinación con SCRUM, sin embargo, dejan claro que no es suficiente para llevar a una empresa a nivel de capacidad 2, ya que se requiere del compromiso de la alta dirección de la compañía y el involucramiento del área de negocio debido a las muchas decisiones clave que impactan en el desarrollo de los proyectos (García López & Linares Valencia, 2020).

1.2.3. Integración del modelo CMMI-DEV y el marco de trabajo SCRUM, en el proceso de desarrollo de software

Las problemáticas comunes en el desarrollo de software, como el incumplimiento de expectativas, la falta de procedimientos estandarizados para gestionar la calidad del software y los proyectos que no aportar valor a la entidad han sido motivo de estudio. En el trabajo presentado por Guevara Lora y Guerrero Vera (2020), se describe la integración del modelo CMMI-DEV versión 1.3 y SCRUM, en el proceso de desarrollo de software de la gestión de proyectos (calidad en el uso del software y calidad del proceso de desarrollo de software) bajo una propuesta de marco de trabajo. Esta investigación proporciona una base para comprender cómo CMMI-DEV y SCRUM pueden trabajar juntos para mejorar la calidad y la eficiencia en el desarrollo de software, brindando importancia de la gestión de proyectos en una implementación cuidadosa y personalizada.

Entre los resultados del trabajo presentado, las autoras señalan haber obtenido resultados satisfactorios al observar un incremento del 13% en cuanto a la mejora de la calidad en el uso del software, mientras que en la calidad del proceso de desarrollo de software obtuvieron un incremento del 42% en el nivel de satisfacción del equipo de desarrollo, aseverando que la calidad del software se mide a través de la calidad de sus procesos (Guerrero Vera & Guevara Lora, 2019).

La presente tesis no solo refiere una revisión exhaustiva de la literatura actual, sino que también presenta un detallado plan de trabajo que describe paso a paso las actividades necesarias para lograr con éxito la implementación de los procesos de CMMI en nivel 5 y la integración del marco de trabajo SCRUM en la empresa de desarrollo de software S.A.P.I. Además, se vale de dos metodologías complementarias: SCAMPI para llevar a cabo una evaluación de la situación inicial de la organización en relación con los procesos de CMMI, y la metodología de mejora organizacional IDEAL, que sirve de guía para la iniciación, planificación e implementación de acciones destinadas a la mejora de los procesos. Los resultados obtenidos también se respaldan con datos y evidencia real de mejoras en la calidad del software, la eficiencia operativa y la satisfacción del cliente. Esta tesis brinda una guía sólida para otras organizaciones que buscan alcanzar niveles superiores de madurez en la gestión de procesos de desarrollo de software a través de la combinación de CMMI y SCRUM.

Capítulo 2. Implementación del modelo SCRUM y CMMI a la empresa S.A.P.I.

En esta sección se describen las etapas clave para la implementación exitosa de CMMI-DEV dentro de la empresa S.A.P.I. para cumplir con sus objetivos y requerimientos, adicionalmente. De acuerdo con las necesidades del equipo de desarrollo se propuso un modelo de desarrollo de software basado en SCRUM con el fin de lograr un compromiso sólido por parte del equipo de trabajo, la entrega del producto en plazos cortos.

Para implementar la combinación de las prácticas SCRUM y los procesos de CMMI en la empresa se realizarán las siguientes fases alineadas al método IDEAL con el fin de llevar a cabo esta integración de manera exitosa.

1. **Inicio:** se inicia realizando una evaluación SCAMPI de los procesos y prácticas actuales en la empresa. Para establece una base sólida para el proyecto de implementación, asegurando que todos estén alineados en cuanto a objetivos
2. **Diagnosticar:** con los resultados obtenidos de la evaluación SCAMPI, se identificarán la brecha existente y las áreas en donde se requiere una mejora significativa para cumplir con el objetivo principal de la empresa.
3. **Establecer:** se planificará en conjunto con la organización en que proyecto se iniciará a realizar la implementación de las metodologías propuestas y establecerá las herramientas y recurso necesarios para una implementación exitosa. Además, se realizará una valoración de cómo se pueden combinar de manera sinérgica las prácticas de SCRUM con los procesos de CMMI. Además, se va a priorizar brindar una capacitación integral a los miembros del equipo SCRUM con el fin de que ejecuten sus tareas en la implementación de manera adecuada.
4. **Actuar**: basado en la evaluación inicial, crea un plan de trabajo detallado que describa las actividades específicas necesarias para mejorar los procesos y prácticas del equipo. Este plan incluirá, plazos, recursos asignados, así como algunos formatos.
5. **Aprender**: en el plan de trabajo, también se brindará una atención específica al proceso de monitoreo y control para identificar los problemas que puedan surgir el

proyecto seleccionado que promueve un enfoque de aprendizaje constante al analizar las acciones realizadas y las soluciones aplicadas. También se realiza un plan de auditorías internas con el fin de evaluar periódicamente el cumplimiento de los requisitos de CMMI, la efectividad de las practicas implementadas e identificar áreas de mejora adicionales y corregir desviaciones.

2.1. Evaluar situación actual de la empresa

Esta fase se enfoca en realizar una evaluación de calidad de la situación actual de la empresa S.A.P.I. que permita identificar la brecha que existe en relación con los procesos implementados actualmente en sus proyectos de desarrollo de software y los procesos propuestos por CMMI -DEV.

Para llevar a cabo esta evaluación se empleará el modelo de evaluación SCAMPI de clase A debido al nivel de certificación que requiere la organización. El método SCAMPI permitirá realizar una investigación de tipo cuantitativo, ya que la evaluación se realizará mediante la recolección de documentos de la organización, referente a sus procesos, entrevistas y un cuestionario.

- **Planificación y preparación de la evaluación**

Con base en lo mencionado por el método SCAMPI en la sección 1.1.9.3, primero se realizó una planificación y preparación de la evaluación que se describe en la Tabla 4. Es preciso mencionar que esta evaluación se realizó por un miembro del equipo evaluador y fue aplicada a los proyectos que se están desarrollando actualmente y documentos generados hasta el momento.

Tabla 4

Planificación y preparación para la evaluación

Proyecto	Actividad	Fecha
Keep Coding	Entrevista	15/05/2023
	Cuestionario	17/05/2023
	Recopilación de documentos	18/05/2023
APP OPN	Entrevista	25/05/2023
	Cuestionario	29/05/2023
	Recopilación de documentos	31/05/2023

- **Conducción de la evaluación**

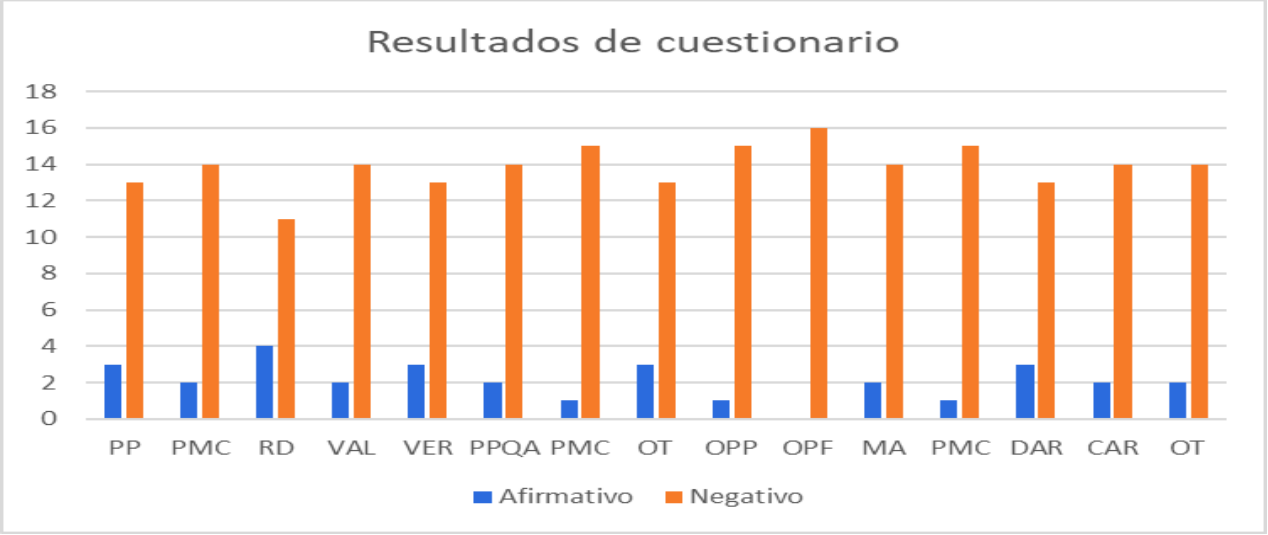
Posteriormente la evaluación se llevó a cabo a través de una entrevista dirigida al líder del proyecto y documentador, para obtener información detallada de los estándares, procesos y prácticas actuales en la organización (ver anexo A). Además, se administró a los 16 miembros del equipo de desarrollo un cuestionario en relación con el cumplimiento de los procesos propuestos por CMMI en la organización. En el cuestionario se debe contestar si se realiza lo que indica la pregunta o no.

- **Reporte de resultados**

En la Figura 9 se presentan los resultados agrupados por área del proceso, que se obtuvieron mediante la evaluación SCAMPI, realizada a los integrantes del equipo de desarrollo de la empresa, para evaluar el grado de alineación de la empresa con los requisitos de CMMI.

Figura 9

Resultados del cuestionario CMMI



Nota: la figura muestra el grado de alineación de los requerimientos de CMMI que presentó la empresa S.A.P.I.

2.1.1. Hallazgos de los resultados obtenidos

De acuerdo con la información obtenida mediante los instrumentos de recolección previamente descritos, se pudo observar que la mayoría de las preguntas fueron respondidas de manera negativa y que la organización presenta una documentación escasa de sus procesos, lo que indica que hay muchas mejoras que realizar en sus procesos para que puedan alcanzar el objetivo de certificarse a un nivel 5 en CMMI- DEV.

Con base en los resultados obtenidos y los objetivos de la organización, se determinó en conjunto con los líderes de proyectos de la empresa, aquellos procesos que son prioridad para cumplir con las normas de CMMI-DEV.

Aunque todas las áreas son importantes, se sabe que las áreas inferiores sirven como soporte para las áreas de procesos superiores. Por lo que se tomó en cuenta que los procesos Planificación de proyectos (PP), Monitorización y Control del proyecto (PMC) y Gestión de Requisitos (REQM) corresponden al Nivel 2 de madurez; mientras que los procesos de Desarrollo de requisitos (RD), Verificación (Validación) y Definición de procesos de la Organización (ODP) corresponden al nivel 3.

2.1.2. Herramientas utilizadas para la implementación de CMMI en SCRUM

Con el fin de poder gestionar y recolectar los datos que serán la base y soporte para el desarrollo de la presente tesis, se analizaron diversas herramientas que permitan administrar proyectos de software de tal manera que faciliten la planificación, seguimiento y colaboración del desarrollo del proyecto y que además cumplan con los diferentes procesos de CMMI.

A continuación, se describen las tareas que cubrirán algunas de las *herramientas para el desarrollo del proyecto*:

- **SharePoint:** es una herramienta de almacenamiento utilizada como gestor de almacenamiento de la documentación generada para la implementación de CMMI como: plan de proyecto, *Product Backlog*, el alcance del proyecto, las estimaciones del

proyecto, entre otros. Además de que ayuda a la gestión de versiones de documentos y facilita la colaboración del equipo SCRUM.

- **Microsoft Teams:** es una plataforma de comunicación que permitió formar equipos de trabajo colaborativo, realizar reuniones de video con personas de la organización o con los clientes desde cualquier punto, además de que ayudó a almacenar y editar simultáneamente archivos, video de reuniones o chat, por lo que se vuelve una herramienta adecuada para cumplir con los procesos de CMMI, ya que brinda diversos artefactos para la gestión de proyectos como las reuniones diarias (*Daily SCRUM*), los *Planning de Sprint*, *Sprint Review* y *Sprint Retrospective*.

- **Azure Devop Server:** la plataforma de Azure Devop Server brindó al equipo *SCRUM* diversas herramientas que permitieron planificar y gestionar el proyecto, realizar el desarrollo de software ágil, tener un control de versiones y cambios, además de permitir gestionar los requisitos, la calidad y riesgos mediante las pruebas del producto. Azure comprende el ciclo de vida de una aplicación lo que permitió aplicar varios procesos de CMMI.

- **Microsoft Word:** a través de esta herramienta se generaron los informes requeridos por los procesos de CMMI como: planes de proyecto, informes de seguimiento y documentación de procesos. Microsoft Word permitió controlar versiones y realizar seguimientos de cambios en los documentos.

2.2. Adaptar las prácticas ágiles SCRUM a los requisitos de CMMI

La combinación de CMMI y SCRUM es una práctica cada vez más común en la industria del desarrollo de software, especialmente en organizaciones que buscan beneficiarse de los enfoques ágiles y al mismo tiempo desean establecer un marco de mejora de procesos más formal y estructurado.

La combinación de CMMI y SCRUM se logra adaptando las prácticas de SCRUM para que se alineen con los objetivos de mejora de procesos establecidos por CMMI. Algunas de las formas en que se pueden combinar son las siguientes:

2.2.1. Planificación del proyecto (PP) con SCRUM

El principal propósito del área de procesos de *Planificación de proyectos* (PP) es establecer y mantener de manera ordenada todas las actividades dentro del proyecto a desarrollar. Al proceso PP lo definen catorce prácticas específicas que a su vez están agrupadas en tres metas concretas las cuales son:

- SG1: establecer de estimaciones.
- SG2: desarrollo de un plan de proyecto.
- SG3: compromiso con el plan.

Para entender qué prácticas SCRUM se alinean con los objetivos y proceso particulares de la sección PP del modelo CMMI, se llevó a cabo un análisis. En la Tabla 5 se plasmó la relación entre ambos enfoques.

Tabla 5

Planificación del proyecto (PP) con SCRUM

Metas Específicas CMMI	Prácticas específicas (PP)	SCRUM
SG1: establecimiento de estimaciones	SP 1.1: El alcance del proyecto.	El *sprint* y el *Product Backlog* estiman los recursos para del alcance del proyecto.
	SP 1.2: Estimar los atributos de los productos de trabajo y tareas	En el *Product Backlog* y cronograma de actividades identificadas.
	SP 1.3: definir las fases del ciclo de vida del proyecto.	Se considera en el propio ciclo de vida de SCRUM.
	SP 1.4: estimar el esfuerzo y el coste.	*Product Backlog* e Historias de usuario

Metas Específicas CMMI	Prácticas específicas (PP)	SCRUM
SG2: desarrollar plan de proyecto	SP 2.1: establecer el presupuesto y el calendario.	Se establecen los acontecimientos importantes para todo el proyecto (hitos) y las fechas de entregables en el *product Backlog*.
	SP 2.2: identificar los riesgos del proyecto.	Se analizan los riesgos y oportunidades que se puedan presentar en el transcurso de la elaboración del proyecto. Además, de identificar en las *Daily* SCRUM los impedimentos que surjan.
	SP 2.3: planificar la gestión de los datos.	No abordada.
	SP 2.4: planificar los recursos del proyecto.	Antes de iniciar el *Sprint* se analiza los recursos materiales y humanos a utilizar de acuerdo con las necesidades del proyecto y se asignan las tareas.
	SP 2.5: planificar el conocimiento y las habilidades necesarias.	Se analizan las tecnologías a utilizar y se realiza el plan de capacitación de acuerdo con la curva de aprendizaje que necesiten los perfiles que son asignados al proyecto.
	SP 2.6: planificar la participación de las partes interesadas.	Regularmente las partes interesadas participan en el *Sprint Planning* y el *Sprint Review*.
	SP 2.7: establecer el plan de proyecto.	Se integra el plan de proyecto con base en el análisis y actividades identificadas en el proceso de planeación (SP 1.2, SP 2.1, SP 2.2, SP 2.4, SP 2.5).
SG3: adquisición de compromisos para el Plan de proyectos	SP 3.1: revisar los planes que afectan al proyecto.	Se aborda en el *Sprint Planning* hasta el *Sprint Review*.
	SP 3.2: conciliar los niveles de trabajo y de recursos.	Se aborda en el *Sprint Planning*.
	SP 3.3: obtener el compromiso con el plan.	Se lleva a cabo mediante las *Daily* SCRUM.

2.2.2. Monitoreo y control (PMC) con SCRUM

La finalidad principal del proceso de *Monitoreo y control* es suministrar información del progreso del proyecto de tal forma que permita tomar decisiones de las medidas adaptativas cuando el proyecto se desvíe del plan inicial. Una desviación significativa que impida cumplir los objetivos propios del proyecto.

Con base en los atributos, las tareas, coste, esfuerzos, hitos y cronogramas establecidos en un plan del proyecto bien estructurado y documentado, se monitorizan las actividades, se conoce el estado del proyecto y se toman decisiones de las acciones correctivas.

El concepto de *Plan de proyecto* se emplea en todos estos procesos con el fin de señalar la estrategia integral destinada a gestionar el proyecto de manera efectiva.

En la Tabla 6 se muestra un análisis de la relación que se halló entre las prácticas específicas del área de procesos de monitoreo y control (PMC) con los componentes de SCRUM que las abordan.

Tabla 6

Monitoreo y control (PMC) con SCRUM

Metas Específicas CMMI	Prácticas específicas (PP)	SCRUM
SG1: supervisar el seguimiento del plan.	SP 1.1: monitorear los parámetros de planificación del proyecto.	Se monitorea durante el *Daily* SCRUM del Sprint y reunión de retrospectiva.
	SP 1.2: monitorear los compromisos.	Se monitorea durante el *Daily* SCRUM del Sprint y reunión de retrospectiva.
	SP 1.3: monitorear los riesgos del proyecto.	Se monitorea durante el *Daily* SCRUM del Sprint y reunión de retrospectiva.
	SP 1.4: monitorear la gestión de los datos.	No abordado.
	SP 1.5: monitorear la participación de las partes interesadas.	Se monitorea la participación en la reunión de retrospectiva.
	SP 1.6: llevar a cabo las revisiones del progreso.	Se lleva a cabo durante la reunión de revisión del Sprint, se revisa el progreso. Gráficas "*Burndown*" y "*burnup*".
	SP 1.7: llevar a cabo las revisiones de hitos.	Se lleva a cabo durante la reunión de revisión del Sprint.
SG2: gestionar acciones correctivas.	SP 2.1: analizar los problemas	Se analizan los problemas durante el *Daily* SCRUM del Sprint y reunión de retrospectiva.
	SP 2.2: llevar a cabo las acciones correctivas	Reunión de revisión de Sprint.
	SP 2.3: gestionar las acciones correctivas.	Se gestionan en la reunión de retrospectiva.

2.2.3. Gestión integración del proyecto (IPM) con SCRUM

El proceso de *Gestión Integrada del Proyecto* (IPM) se centra en establecer y gestionar un proyecto de manera efectiva. La gestión se puede llevar a cabo mediante la integración de información histórica de proyectos pasados para estimar y dibujar el nuevo plan de proyecto.

Durante el transcurso del proyecto, se recopilan y documentan experiencias y lecciones aprendidas, lo que contribuye a una mejora continua en futuros proyectos. Se adopta un enfoque proactivo en la gestión de la participación de los involucrados. En última instancia, el proceso IPM se dedica a resolver de manera eficaz cualquier problema de coordinación que pueda surgir entre las partes interesadas del proyecto.

En la Tabla7 se realiza un análisis entre cada práctica específica del área de procesos de "Gestión integración del proyecto" y las prácticas de SCRUM que las complementan. Esto tiene como objetivo vincular las actividades propuestas en el modelo de desarrollo con las sugeridas por CMMI.

Debido a la integración de la metodología SCRUM a la empresa se incorporó este proceso.

Tabla 7

Gestión integración del proyecto (IPM) con SCRUM

Metas Específicas CMMI	Prácticas específicas (PP)	SCRUM
SG1	SP 1.1: establecer el proceso definido del proyecto.	No abordado.
	SP 1.2: utilizar los activos de proceso de la organización para planificar las actividades del proyecto.	No abordado.
	SP 1.3: establecer el entorno de trabajo del proyecto.	Detalles del entorno de trabajo.
	SP 1.4: integrar los planes.	Se integran en el backlog.
	SP 1.5: gestionar el proyecto utilizando planes integrados.	Reunión de planificación del sprint.
	SP 1.6: establecer los equipos.	Roles de SCRUM.
	SP 1.7: contribuir a los activos de proceso de la organización.	Reunión de retrospectiva.

Metas Específicas CMMI	Prácticas específicas (PP)	SCRUM
SG2	SP 2.1: gestionar la involucración de las partes interesadas.	*Daily*, retrospectiva. *Sprint review.*
	SP 2.2: gestionar las dependencias.	*Daily*, retrospectiva. *Sprint review.*
	SP 2.3: resolver las cuestiones de coordinación.	*Daily*, retrospectiva. *Sprint review.*

2.2.4. Gestión de riesgos (RSKM) con SCRUM

El propósito de la *Gestión de riesgos* es anticiparse a posibles problemas antes de que afecten la producción, permitiendo planificar acciones para minimizar o mitigar estos riesgos. Es importante diferenciar entre problemas y riesgos: un problema ya ha ocurrido y causado daño, mientras que un riesgo es una posible situación que podría generar pérdida o daño. La gestión de riesgos debe abordar situaciones que amenacen el logro de objetivos críticos para el éxito del proyecto. Una efectiva gestión de riesgos es esencial para asegurar el progreso exitoso del proyecto.

En la Tabla 8 se describe un análisis de las prácticas de SCRUM que se relacionan con las áreas de procesos de gestión de requerimientos de CMMI-DEV.

Tabla 8

Gestión de riesgos (RSKM) con SCRUM

Metas Específicas CMMI	Prácticas específicas (PP)	SCRUM
SG1	SP 1.1: determinar las fuentes y las categorías de riesgos.	En reunión de inicio del proyecto y *Sprint planning*
	SP 1.2: definir los parámetros de riesgos	En reunión de inicio del proyecto y *Sprint planning*
	SP 1.3: establecer una estrategia de gestión de riesgos.	En reunión de inicio del proyecto y *Sprint planning*

Metas Específicas CMMI	Prácticas específicas (PP)	SCRUM
SG2	SP 2.1: identificar los riesgos.	En reunión de inicio del proyecto y *Sprint planning* y retrospectiva.
	SP 2.2: evaluar, clasificar y priorizar los riesgos.	En reunión de inicio del proyecto y *Sprint planning* y retrospectiva.
SG3	SP 3.1: desarrollar los planes de mitigación de riesgos.	En reunión de inicio del proyecto y *Sprint planning* y retrospectiva.
	SP 3.2: implementar los planes de mitigación de riesgos.	En reunión de inicio del proyecto y *Sprint planning* y retrospectiva.

2.2.5. Gestión de requerimientos (REQM) con SCRUM

La *Gestión de Requisitos* (REQM) en la implementación de un proyecto, se trata de identificar, asignar y monitorear los requisitos esenciales para alcanzar los objetivos del proyecto. Esto es fundamental, ya que una buena gestión de requisitos impacta positivamente en todo el proceso.

La idea clave es asegurarse de entender y documentar correctamente lo que el cliente necesita. Al hacerlo, se pueden evitar malentendidos y cambios inesperados a lo largo del proyecto. Además, al tener un control constante sobre los requisitos, se puede garantizar que el proyecto siga alineado con las expectativas del cliente y no se desvíe.

En definitiva, la *Gestión de Requisitos* es un paso crítico para asegurar que el proyecto avance de manera fluida y que se cumpla con lo que el cliente espera.

En la Tabla 9 se presentan las relaciones que se identificaron de cada una de las prácticas específicas del área de procesos de gestión de requisitos con los componentes *SCRUM*.

Tabla 9

Gestión de requerimientos (REQM) con SCRUM

Metas Específicas CMMI	Prácticas específicas (PP)	SCRUM
SG1	SP 1.1: comprender los requerimientos.	Historias de usuario. Sprint backlog.
	SP 1.2: obtener el compromiso sobre los requerimientos.	Reunión de planeación. *Product Backlog.* *Sprint backlog.*
	SP 1.3: gestionar los cambios a los requerimientos.	Reunión de planeación. Reunión de revisión del *sprint*.
	SP 1.4: mantener la trazabilidad bidireccional de los requerimientos	Historias de usuario.
	SP 1.5: asegurar el alineamiento entre el trabajo del proyecto y los requerimientos.	Reuniones de planeación del *sprint*.

2.2.6. Desarrollo de requerimientos (RD) con SCRUM

El proceso de *Desarrollo de Requisitos* (RD) implica la adquisición, análisis y definición de los requisitos provenientes del cliente, así como de los requisitos de producto o componentes del producto. Estos requisitos deben ser compatibles con las limitaciones impuestas por las decisiones de diseño. Este proceso se enfoca en tres metas específicas: la adquisición y desarrollo de los requisitos del cliente, que son el punto de partida para la creación; de los requisitos del producto y de los componentes del producto.

En la Tabla 10 se exponen las correspondencias entre las prácticas específicas del área de procesos y los componentes de *SCRUM* que las abarcan.

Tabla 10

Desarrollo de requerimientos (RD) con SCRUM

Metas Específicas CMMI	Prácticas específicas (PP)	SCRUM
SG1	SP 1.1: obtener las necesidades.	Se obtiene en las historias de usuario.
	SP 1.2: transformar las necesidades de las partes interesadas en requisitos de cliente.	Se abordan los requisitos en las historias de usuario y el *Product Backlog.*
SG2	SP 2.1: establecer los requisitos de producto y de componente de producto.	Se establecen en Historias de usuario clasificadas por módulo.
	SP 2.2: asignar los requisitos de componente de producto.	Se asignan en historias de usuario clasificadas por módulo.
	SP 2.3: identificar los requisitos de interfaz.	Se identifican en historias de usuario relacionadas con los requerimientos de interfaz.
SG3	SP 3.1: establecer los conceptos y los escenarios de operación.	Se establece en las historias de usuario.
	SP 3.2: establecer una definición de la funcionalidad y de los atributos de calidad requeridos.	Se establece en las historias de usuario.
	SP 3.3: analizar los requisitos.	Se analizan para las historias de usuario clasificadas por prioridad.
	SP 3.4: analizar los requisitos para conseguir un equilibrio.	Se analizan para las historias de usuario clasificadas por prioridad.
	SP 3.5: validar los requisitos.	Se validan en el *Product backlog.*

2.2.7. Solución técnica (TS) con SCRUM

El proceso de *Soluciones Técnicas* (TS) se centra en la elección, diseño e implementación de los requisitos. Estas soluciones pueden abarcar productos, componentes de productos, y otros elementos del ciclo de vida del proceso, ya sea de manera individual o mediante combinaciones estratégicas. Esta área de proceso también abarca servicios en adición a productos. En el proceso de solución técnica se determina la aproximación técnica para el

diseño y desarrollo del producto, se lleva a cabo el diseño e implementación, y se desarrolla el producto. Las prácticas específicas implican explorar diversas soluciones posibles, definir criterios para seleccionar la más adecuada, realizar el diseño del producto y sus componentes, y crear documentación técnica detallada. Además, se establece la necesidad de elegir entre adquirir, utilizar o desarrollar productos o componentes, desde cero o basándose en otros recursos.

Se realizó un análisis de la relación que existe entre las prácticas específicas del proceso de soluciones técnicas con los componentes SCRUM, el resultado se muestra en la Tabla 11.

Tabla 11

Solución técnica (TS) con SCRUM

Metas Específicas CMMI	Prácticas específicas (PP)	SCRUM
SG1	SP 1.1: desarrollar soluciones alternativas y los criterios de selección.	No abordado
	SP 1.2: seleccionar las soluciones de componentes de producto.	No abordado
SG2	SP 2.1: diseñar el producto o los componentes del producto.	Se elabora documento de arquitectura, en el *Product Backlog*, se aborda en reuniones de planeación de sprint.
	SP 2.2: establecer un paquete de datos técnicos.	Se solventa en las Historias de usuario, documento de arquitectura, casos de prueba.
	SP 2.3: diseñar las interfaces usando criterios.	En Historias de usuario y documento de arquitectura.
	SP 2.4: realizar los análisis sobre si hacer, comprar o reutilizar.	No abordado
SG3	SP 3.1: implementar el diseño.	Código fuente.
	SP 3.2: desarrollar la documentación de soporte del producto.	Manual de usuario. Manual del producto.

2.2.8 Integración del producto (PI) con SCRUM

El proceso de *Integración de Producto* (PI) se enfoca en construir el producto a partir de sus componentes, asegurando su funcionamiento correcto y entregándolo al cliente. Se establecen tres metas clave en PI. Primero, se prepara la integración, asegurando que los componentes sean compatibles y se lleva a cabo el proceso de ensamblaje. Se crea una estrategia de integración que abarca cómo se recibirán, ensamblarán y evaluarán los componentes que forman el producto. Antes de iniciar la integración, se establece un entorno adecuado que podría incluir equipos de prueba, simuladores y herramientas de medición.

Este proceso asegura que todos los componentes encajen correctamente y que el producto resultante cumpla con los estándares de calidad establecidos.

En la Tabla12 se describe la relación que existe entre las prácticas SCRUM y los procesos propuestos por CMMI-DEV.

Tabla 12

Integración del producto (PI) con SCRUM

Metas Específicas CMMI	Prácticas específicas (PP)	SCRUM
SG1	SP 1.1: establecer una estrategia de integración.	Se detalla la integración continua del producto.
	SP 1.2: establecer el entorno de integración del producto	Configuración del ambiente del producto. Detalles de la integración del producto.
	SP 1.3: establecer los procedimientos y los criterios de integración del producto.	Documentación de los procedimientos del desarrollo del producto.
SG2	SP 2.1: revisar la completitud de las descripciones de las interfaces	Pruebas de integración y desarrollo del producto.
	SP 2.2: gestionar las interfaces.	Pruebas de integración y desarrollo del producto.
SG3	SP 3.1: disponibilidad de componentes del producto para la integración.	Desarrollo del producto.
	SP 3.2: ensamblar los componentes de producto.	Desarrollo del producto.
	SP 3.3: evaluar los componentes de producto ensamblados.	Producto resultante del sprint.
	SP 3.4 empaquetar y entregar el producto o componente de producto.	Se entregan los compilados del producto.

2.2.9. Verificación (VER) con SCRUM

El proceso de *Verificación* (VER) tiene como objetivo asegurar que el proyecto o producto cumple con sus especificaciones. La verificación se lleva a cabo en diferentes etapas del ciclo de desarrollo, comenzando desde la verificación de los requisitos, pasando por la verificación de los productos intermedios y finalmente llegando a la verificación del producto completo. Esta estrategia escalonada de verificación a lo largo del ciclo de vida incrementa las probabilidades de éxito general del proyecto.

Se establecen tres metas clave en el proceso de verificación. En primer lugar, se realiza la preparación para la verificación, asegurando que todos los elementos necesarios estén en su lugar. Luego, se realiza una revisión exhaustiva de lo preparado anteriormente.

Finalmente, se efectúa la verificación en sí. Además de estos objetivos específicos, se establece un objetivo global para garantizar que el proceso de verificación se convierta en una práctica institucionalizada en la organización.

En la Tabla 13 se realiza un análisis entre cada práctica específica del área de procesos de *Verificación* y las prácticas de SCRUM que las complementan. Esto tiene como objetivo vincular las actividades propuestas en el modelo de desarrollo con las sugeridas por CMMI.

Tabla 13

Verificación (VER) con SCRUM

Metas Específicas CMMI	Prácticas específicas (PP)	SCRUM
SG1	SP 1.1: seleccionar los productos de trabajo para la verificación.	Se elabora un plan de pruebas.
	SP 1.2: establecer el entorno de verificación.	Se elabora un plan de pruebas.
	SP 1.3: establecer los procedimientos y los criterios de verificación.	Se elabora un plan de pruebas.

Metas Específicas CMMI	Prácticas específicas (PP)	SCRUM
SG2	SP 2.1: preparar las revisiones entre pares.	Se realiza una revisión par.
	SP 2.2: realizar las revisiones entre pares.	Reporte de revisión de pares.
	SP 2.3: analizar los datos de las revisiones entre pares.	Cierre de defectos.
SG3	SP 3.1: realizar la verificación.	Se realizan pruebas unitarias.
	SP 3.2: analizar los resultados de la verificación.	Análisis y prevención de defectos.

2.2.10. Validación (VAL) con SCRUM

El propósito del proceso de *Validación* (VAL) es demostrar que el producto o sus componentes cumplen con las intenciones de uso cuando están en su entorno real de aplicación. A diferencia de la verificación, que se enfoca en asegurar el cumplimiento de los requisitos, la validación se centra en confirmar que el producto es adecuado y efectivo para su uso previsto.

El proceso de Validación se inicia identificando los productos que necesitan ser validados. Esta selección se realiza basada en su relevancia para las necesidades del cliente final. Posteriormente, se eligen criterios y procedimientos de validación que garanticen que el producto cumple sus objetivos de uso de manera efectiva. Una vez completadas estas actividades preparatorias, se lleva a cabo la validación propiamente dicha.

En la Tabla 14 se muestra una relación entre las prácticas específicas del área de procesos de *Validación* y las prácticas de SCRUM, con el fin de vincular las actividades propuestas en el modelo de desarrollo con las sugeridas por CMMI.

Tabla 14

Validación (VAL) con SCRUM

Metas Específicas CMMI	Prácticas específicas (PP)	SCRUM
SG1	SP 1.1: seleccionar los productos para la validación.	Se elabora un plan de pruebas.
	SP 1.2: establecer el entorno de validación.	Configuración del ambiente de pruebas.
	SP 1.3: establecer los procedimientos y los criterios de validación.	Se elabora un plan de pruebas.
SG2	SP 2.1: realizar la validación.	Resultados de la ejecución del plan de pruebas.
	SP 2.2: analizar los resultados de la validación.	Análisis y prevención de defectos.

2.2.11. Formación y capacitación para iniciar plan de acción

La fase de *Capacitación* es esencial para preparar a los equipos para el éxito en la implementación de SCRUM y CMMI. Una capacitación efectiva asegurará que todos estén alineados con los objetivos y puedan contribuir de manera significativa a la mejora de los procesos y la calidad en el desarrollo de software.

Para llevarla a cabo se diseñó un programa de capacitación, como se muestra en la Tabla 15, adaptado a las necesidades específicas de la organización y los equipos.

Tabla 15

Diseño del programa de Capacitación

ID tema	Nombre del curso	Objetivo del curso	Temario del curso		Duración hrs.	Dirigido a:	Tipo	Imparte	Material necesarios
DCP 002	Procesos de CMMI e implementar organizacional	Proporcionar una guía que mejore las buenas prácticas en el desarrollo y proporcionar un marco de mejora de procesos	1	Capacitación de los procesos de CMMI a implementar en la organización:	1 hr.	Todo el personal de S.A.P.I.	Interno	Business *Analyst*	Equipo de cómputo Internet
			1.1	Proceso de Venta	1 hr.				
			1.2	Proceso de Análisis y diseño	1 hr.				
			1.3	Proceso de construcción	1 hr.				
			1.4	Proceso de Monitoreo y control	1 hr.				
			1.5	Proceso de Liberación					
			1.6	Proceso de Auditorias					

Capítulo 3. Plan de acción propuesto

En este capítulo, se presenta el plan de trabajo propuesto, que desempeñará un papel fundamental en la adopción exitosa de las prácticas SCRUM y los procesos de CMMI en el entorno de desarrollo de software de la organización S.A.P.I. Este plan de trabajo es esencial para guiar al equipo de desarrollo a lo largo de las diversas etapas del proceso de implementación, lo que asegurará una transición fluida y efectiva hacia un enfoque más ágil y maduro en la gestión de proyectos y desarrollo de software.

Para la implementación de los procesos de CMMI y la metodología SCRUM se consideró el proyecto K¡Vo! en su primera versión. Es una tienda en línea que se consideró por relevancia estratégica y nivel de complejidad.

Se mostrará la implementación de algunas tareas de documentación en el proyecto K¡Vo!, pero algunas otras por razones de privacidad solo se mostrará el formato o descripción general, que permita tener una idea de gran trascendencia de las actividades a realizar para cumplir con los procesos propuestos por CMMI- DEV.

3.1 Proceso de alcance

El objetivo del *Proceso de alcance* en relación con CMMI es mostrar cómo el modelo puede ser una solución valiosa para las organizaciones que buscan mejorar sus procesos y resultados en el desarrollo de software. Se trata de una etapa crucial para generar interés y compromiso por parte de las partes interesadas y potenciales clientes en la adopción exitosa de las prácticas de CMMI.

3.1.1. Análisis del cliente

Esta actividad se centra en el análisis y abordaje de los prospectos y contactos de posibles clientes a los que la fábrica de software desea ofrecer sus servicios. Aquí está un resumen de los aspectos clave:

- **Oferta de Servicios**: los servicios de la fábrica de software se presentan al cliente mediante una presentación.

- **Responsable:** el *Product Owner* es el encargado de esta actividad.
- **Entrada de Actividad:** la información de entrada proviene de una lista de prospectos, que es la base para identificar posibles oportunidades de negocio.
- **Actividad de Salida:** el resultado de esta actividad es la generación de prospectos, es decir, se identifican clientes potenciales interesados en los servicios ofrecidos.

3.1.2. Presentación comercial

Esta actividad se enfoca en la presentación de los servicios de la fábrica de software a posibles clientes. La presentación comercial sirve como punto de partida para esta actividad.

- **Entrada de Actividad:** el *Product Owner* se encargará de planificar una reunión con el cliente para el levantamiento de requerimientos. Esto implica recopilar información esencial para comprender las necesidades y expectativas del cliente.
- **Actividad de Salida**: como resultado de la presentación, se genera una minuta que recoge los aspectos discutidos. Además, se notificará al *SCRUM Master* la necesidad de recursos para el levantamiento de requerimientos.

3.1.3. Análisis de recursos disponibles

El *Análisis de recursos disponibles* implica que el *SCRUM Master* realice una evaluación de los recursos disponibles en la organización y asigne los recursos en función de los requisitos de la empresa y los proyectos planificados. Este análisis se realiza para asegurar que se asignen los recursos adecuados para llevar a cabo con éxito, el proceso de levantamiento de requerimientos.

- **Entrada de Actividad:** las principales entradas para llevar a cabo esta tarea es la minuta generada durante la reunión de presentación comercial, que proporciona información sobre los recursos requeridos, y cualquier notificación adicional de necesidades de recursos.

- **Actividad de Salida:** la salida principal de esta etapa es la asignación efectiva de los recursos necesarios. Esto debe incluir la designación de personal y otros recursos que garantizarán el éxito en el levantamiento de requerimientos.

3.1.4. Programar reunión

Una vez asignados los recursos, el *Product Owner* es responsable de programar la reunión entre el cliente y el equipo de desarrollo para recopilar información y definir los requerimientos del proyecto.

3.1.5. Levantamiento de requerimientos

El levantamiento de requerimientos es una actividad crucial para garantizar el éxito del proyecto, ya que proporciona una comprensión completa de lo que se debe construir y cómo debe funcionar el sistema o la aplicación.

Los responsables de esta actividad son el *Product Owner* y el *SCRUM Master*, quienes desempeñan roles esenciales en la gestión de requerimientos.

- **Actividad de Salida:** el resultado de esta actividad es la *Minuta de Levantamiento de Requerimientos*, que documenta de manera clara y concisa todos los aspectos discutidos en la reunión con base a formato de la minuta establecido (ver anexo B).

3.1.6. Product Backlog

En esta actividad, se debe de elabora una lista detallada de los requerimientos del cliente para el sistema o la aplicación, como se muestra en la Tabla 16. El responsable de esta tarea es el *Product Owner*, quien trabaja en colaboración con otros miembros del equipo según sea necesario. Esta lista de requerimientos se conoce como *Product Backlog*.

El *Product Backlog* debe incluir información esencial sobre los requerimientos del proyecto, incluyendo las siguientes secciones:

1. **Épicas**: descripciones generales de funcionalidades o características importantes del sistema.

2. **Historias de Usuario**: requerimientos específicos del cliente, generalmente escritos en forma de historias que describen una funcionalidad deseada desde la perspectiva del usuario.
3. **Criterios de Aceptación**: condiciones que deben cumplirse para que una historia de usuario se considere completa y aceptada.
4. **Refinamiento**: detalles adicionales o aclaraciones sobre las historias de usuario, si es necesario.

Tabla 16

Product Backlog del proyecto K¡Vo!

ID-Épica	Épica	ID- HU	Historia de Usuario	Criterios de Aceptación	Tipo	Prioridad	Tamaño SP	Sprint	Estado	Notas de Arquitectura
EPI-HS-APW-001	Página web	HU-HS-001	Yo como usuario quiero que el sistema tenga una pantalla principal con un menú por categorías para que la navegación sea ordenada	Cuando el usuario ingrese a la pantalla principal de la tienda encontrará: Menú de opciones: Colecciones Categorías Presentado. A cerca de: La marca. Sostenibilidad Reseñas. Carrito de compras. Carrusel de nueva colección. Atención al cliente. Información. Almacenista. Redes sociales.	Obligatoria	Alta	2	1	Atendido	
EPI-HS-APW-001	Página web	HU-HS-002	Yo como usuario quiero que el sistema cuente con un menú principal para visualizar y seleccionar un producto.	Cuando el usuario ingrese al sistema encontrará un menú principal (Comercio) con las opciones: COLECCIONES Nueva Colección de ánforas Oro rosa para repensar Colección Paraíso CATEGORÍAS Pendientes Collares Esposas Anillos	Obligatoria	Alta	2	1	Atendido	
EPI-HS-APW-001	Página web	HU-HS-003	Yo como usuario quiero que el Sistema muestre las COLECCIONES con elementos que hagan posible la elección de artículos disponibles.	Cuando el usuario ingrese a alguna de las colecciones visualizará: Barra de búsqueda por nombre (Campo de captura) Nombre de la colección (dato consulta). Breve descripción de la colección y producto. Imagen del producto, nombre del producto, calificación, costo unitario, cantidad de productos Vista rápida. Añadir a la cesta	Obligatoria	Alta		1	Atendido	

ID-Épica	Épica	ID-Historia de Usuario	Historia de Usuario	Criterios de Aceptación	Tipo	Prioridad	Tamaño SP	Sprint	Estado	Notas de Arquitectura
EPI-HS-APW-001	Página web	HU-HS-004	Yo como usuario quiero que el Sistema me muestre las CATEGORÍAS con elementos que hagan posible la elección de artículos disponibles.	Cuando el usuario ingrese a alguna de las colecciones visualizará: Barra de búsqueda por nombre (Campo de captura) Nombre de la categoría (dato consulta). Breve descripción del producto. Imagen del producto, nombre del producto, costo unitario, cantidad de productos Vista rápida. Añadir a la cesta	Obligatoria	Alta		1	Atendido	
EPI-HS-APW-001	Página web	HU-HS-005	Yo como usuario quiero que el sistema procese pagos electrónicos, para que pueda adquirir productos.	"Cuando el usuario realice el pago de sus productos, encontrará: Elegir método de pago (*select* de iconos) Ingresar los datos de cuenta o tarjeta (campos de captura) Correo electrónico (campo de captura) Nombre y apellido (Campo de captura) dirección (campo de captura)	Obligatoria	Alta		1	Atendido	
EPI-HS-PW-002	Página web	HU-HS-006	Yo como usuario quiero que el sistema muestra opiniones y calificaciones de los otros usuarios	"Cuando el usuario Seleccione el botón de "Reseñas" encontrará: Calificación del producto Reseña de los usuarios. Galería de fotos. *Like*. Esta función debe ofrecer los elementos que permitan al cliente tener una idea general de los productos publicados.	Obligatoria	Alta		1	Atendido	

3.1.7. Pantallas prototipo

En esta actividad, se crean bocetos o prototipos de las pantallas del sistema K¡Vo! que se va a construir, como se muestra en las Figura 10, 11 y 12. Estos prototipos ofrecen una visión de alto nivel de cómo se verán y funcionarán las pantallas, y se basan en los requerimientos recopilados previamente durante el levantamiento de requerimientos y las *Épicas* e *Historias de Usuario* del *Product Backlog*.

El responsable de esta actividad es el *Product Owner*, quien trabaja en colaboración con otros miembros del equipo según sea necesario.

Figura 10

Pantalla prototipo de Menú para el aplicativo K¡Vo!

Figura 11

Pantalla prototipo del módulo de productos

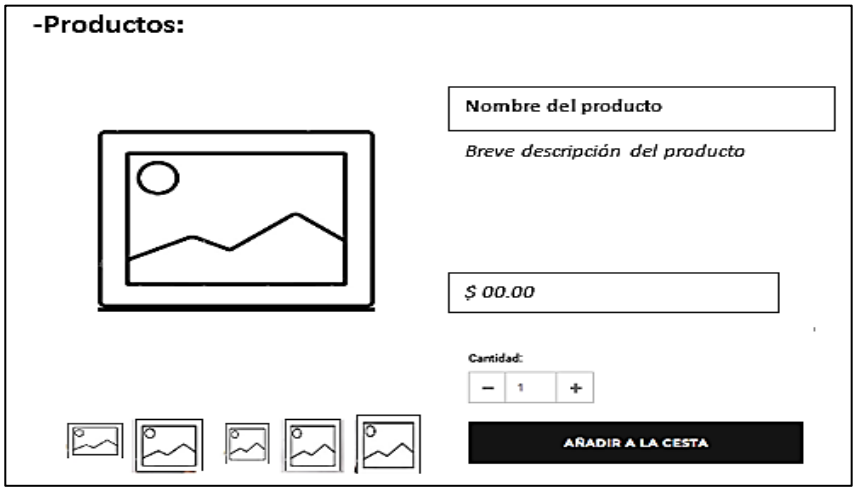
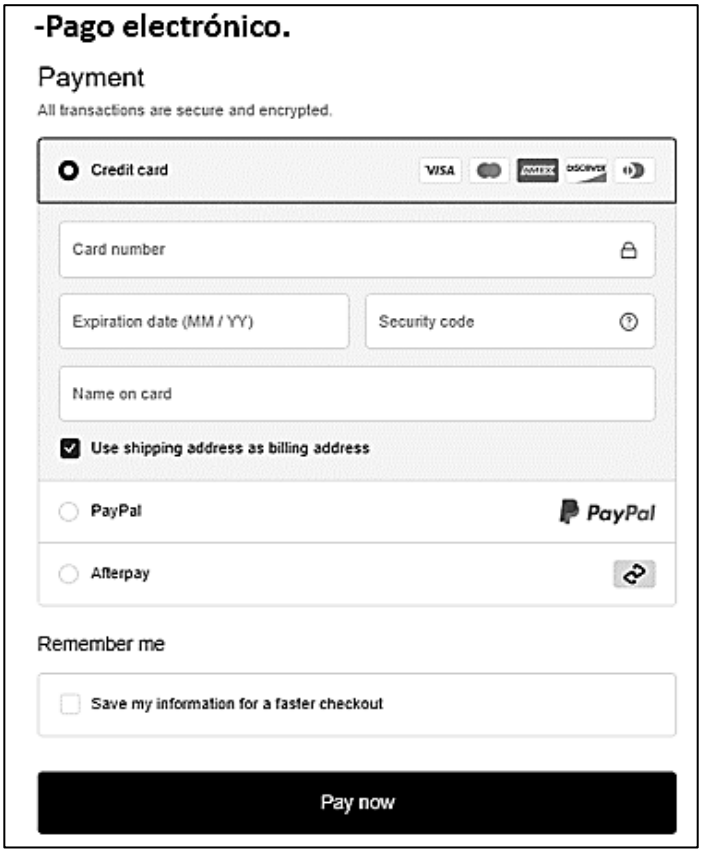


Figura 12

Pantalla prototipo del módulo de pagos electrónicos



3.1.8. Realizar estimación

Para esta actividad, el *SCRUM Master* trabaja en colaboración con el *Product Owner* para realizar estimaciones de tiempo para comprender cuánto trabajo se necesita para completar cada elemento del backlog y planificar los *Sprint* y entregas futuras.

Posteriormente, se actualizará el producto *backlog* con las estimaciones realizadas, como se muestra en la Tabla 17, para la toma de decisiones informadas en los *Sprint* futuros.

Tabla 17

Estimación de Product Backlog

Descripción	Valor	Criterios para realizar la estimación
Historias de Usuario	6	
Puntos de Historia (SP)	6	
Factor de productividad	8	
# Personas en el Equipo	5	
# de semanas por Sprint	2	
# de Sprint en el proyecto	2	
Costo por Hora Estándar	$299.33	

3.1.9. Realizar propuesta técnica

El *Product Owner* será el encargado de elaborar una propuesta técnica que servirá como base para la toma de decisiones posteriores en cuanto a la ejecución del proyecto. En la propuesta técnica se describirá cómo se abordarán y desarrollarán los requerimientos del proyecto. Incluye información detallada sobre la arquitectura, tecnologías, metodologías y recursos necesarios para llevar a cabo el proyecto de manera exitosa.

Esta actividad se debe realizar en colaboración con el *Product Owner* y otros miembros del equipo, como arquitectos de software y especialistas técnicos, para garantizar que la propuesta técnica sea realista y viable. Para esta actividad se propuso un formato (ver anexo C).

3.1.10. Revisar propuesta técnica

En esta actividad, el *SCRUM Master* verificará si la propuesta técnica elaborada anteriormente, cuenta con toda la información necesaria y los elementos requeridos para que pueda ser aprobada. La revisión se realiza para asegurarse de que la propuesta esté completa y cumpla con los estándares y requisitos establecidos por la organización para su implementación.

3.1.11. Generar propuesta comercial

Con la propuesta técnica aprobada el *Product Owner* se encargará de generar una propuesta comercial en la cual deberá detallar los aspectos económicos del proyecto, así como los costos estimados, los plazos de entrega y cualquier otro elemento relacionado con los aspectos financieros entre el equipo de desarrollo y el cliente, para definir los términos y condiciones bajo los cuales se llevará a cabo el proyecto (ver anexo D).

Una vez que se ha generado la propuesta comercial, se procede a realizar la presentación ante el cliente con el fin de discutir y aclarar cualquier aspecto de la propuesta y asegurarse de que ambas partes estén alineadas en cuanto a los términos y condiciones del proyecto.

3.1.12. Revisión de propuesta comercial

En esta etapa, se le brindará la oportunidad al cliente de realizar una revisión y validar la propuesta comercial presentada por el equipo de desarrollo. La revisión se realiza para asegurarse de que la información contenida en la propuesta comercial esté alineada con lo que se acordó en la propuesta técnica y que incluye toda la información necesaria para ser aprobada.

Si la propuesta comercial cumple con los requisitos y expectativas del cliente, se procede a aprobarla. En caso contrario, si hay discrepancias o se requieren cambios, se puede generar un *Fallo de Licitación*, lo que implica que la propuesta no es aceptada en su forma actual y se requieren modificaciones o negociaciones adicionales.

3.1.13. Seguimiento a la aprobación de la propuesta

Una vez que la propuesta comercial ha sido revisada y aprobada por el cliente, el *Product Owner* genera un registro que documenta esta aprobación para mantener un historial claro y organizado de las transacciones y acuerdos relacionados con la propuesta comercial. En el seguimiento se incluyen la fecha de aprobación, la firma del cliente u otras evidencias de aceptación, y cualquier otra información relevante.

3.1.14. Notificación de aceptación del proyecto

Con la propuesta comercial aprobada por el cliente, el *Product Owner* procederá a notificar al cliente y al equipo de desarrollo que el proyecto está listo para comenzar.

El correo de inicio del proyecto debe contener detalles clave, como la fecha de inicio prevista, los hitos iniciales, los roles y responsabilidades de ambas partes, y cualquier otra información relevante para el inicio del proyecto.

3.2. Proceso de planeación

En este proceso se brindó una hoja de ruta clara que guía a las organizaciones y equipos hacia el logro de sus metas, al tiempo que les permite adaptarse a cambios en el entorno y tomar decisiones informadas.

3.2.1. Revisión de cargas de trabajo

Para esta esta actividad el *SCRUM Master* del proyecto debe realizar una revisión exhaustiva de la carga de trabajo de los recursos disponibles, con el objetivo de evaluar las capacidades y la disponibilidad de los recursos con base en los requerimientos del proyecto, para determinar los recursos que serán asignados al mismo, con el fin de garantizar que el proyecto cuente con los recursos adecuados para llevar a cabo las tareas y alcanzar los objetivos.

El *Líder Técnico* colabora en este proceso proporcionando información sobre las habilidades técnicas y las necesidades del proyecto.

Como resultado de esta actividad, se generan dos salidas importantes, las cuales se describen en los siguientes párrafos.

1. **Asignación de Recursos**: se determina qué recursos específicos se asignarán al proyecto. Esto puede incluir roles, equipos o individuos que desempeñarán funciones clave en el proyecto.
2. **Matriz de Recursos**: se crea una matriz que documenta la asignación de recursos, lo que proporciona una visión general de quién está asignado a qué tarea o actividad dentro del proyecto como se muestra en la Tabla 18. Esto es fundamental para la planificación y la gestión de recursos a lo largo del ciclo de vida del proyecto.

Tabla 18

Matriz de recurso del proyecto K¡Vo!

Perfil	Actividad
SCRUM Master	Planear y coordinación de las diversas actividades involucradas para llevar a cabo el desarrollo, realizar supervisión continua para asegurarse que se estén cumpliendo según lo previsto las tareas, detectar desviaciones y buscar soluciones
Product Owner	Especificar los requerimientos funcionales, análisis y documentación.
QA	Asegurar la calidad del software mediante el diseño, ejecución y documentación de pruebas, Seguimiento de errores.
Desarrollador *Frontend*	Crear el diseño visual y la disposición de elementos en la página web, asegurando que sea atractiva y fácil de usar para los usuarios, escribir e integrar código para construir la página web
Desarrollador *Backend*	Diseñar la estructura de la base de datos, escribir código para implementar la lógica del negocio y conectar métodos de pago.

3.2.2. Realizar plan de recursos humanos

Con base en la *Matriz de recursos* el *SCRUM Master* realizará un *Plan de recursos Humanos*, en el que se documentará de manera detallada cómo se administrarán y coordinarán los

recursos humanos a lo largo del proyecto como se muestra en la Tabla 19. Incluye aspectos como:

1. **Asignación de Roles**: define quién asumirá cada rol en el proyecto y cuáles son sus responsabilidades.
2. **Estructura del Equipo**: describe cómo se organizará el equipo de proyecto, incluyendo cualquier subdivisión de equipos si es necesario.
3. **Desarrollo de Habilidades**: si es necesario, se pueden incluir planes de capacitación o desarrollo de habilidades para el equipo.
4. **Comunicación**: establece cómo se llevará a cabo la comunicación interna en el equipo y con partes interesadas externas.
5. **Coordinación**: define cómo se coordinarán las actividades y tareas entre los miembros del equipo.

Tabla 19

Plan de recurso Humanos para el Proyecto K¡Vo!

Recursos para el desarrollo del proyecto	
1 *SCRUM Master*	Encargado de las métricas y el avance del proyecto.
1 *Product Owner*	Tendrá la función principal de documentar los requerimientos del cliente.
2 Desarrolladores	Un desarrollador *Frontend* y un desarrollador *Backend* encargados de realizar e integrar el código para la base de datos y la interfaz del usuario.
1 *Tester QA*	Encargado de las pruebas de calidad para la liberación del producto final.

Nota: Para llevar a cabo el desarrollo del proyecto K¡Vo! se contemplan 5 recursos.

3.2.3. Realizar plan de recursos materiales

En esta actividad el *SCRUM Master* se lleva a cabo un análisis detallado de los recursos materiales necesarios para llevar a cabo el proyecto. Esto incluye cualquier equipo, herramientas, software, hardware u otros materiales necesarios, para ejecutar las tareas del proyecto de manera eficiente, se incluye información sobre su disponibilidad y costo estimado. Además, se generan *Solicitudes de recursos* que pueden ser utilizadas para adquirir o reservar los materiales necesarios.

En este proyecto no se solicitan recursos materiales debido a que está basado en la nube y no son necesarios recursos adicionales.

3.2.4. Realizar plan de capacitación

El *SCRUM Master* llevará a cabo un análisis de las tecnologías, herramientas o conocimientos específicos que serán necesarios para llevar a cabo el proyecto de manera efectiva. Esto se hace en función de los perfiles y roles asignados al proyecto, ya que diferentes miembros del equipo pueden requerir diferentes tipos de capacitación.

Esto implica evaluar el nivel de conocimiento actual de los miembros del equipo en relación con las tecnologías o herramientas que se utilizarán en el proyecto y determinar qué tipo de capacitación es necesaria.

El resultado principal de esta actividad es el *Plan de Capacitación*, como el que se muestra en la Tabla 20. Este plan detalla las necesidades de capacitación identificadas, para llevar acabo el desarrollo del proyecto K¡Vo! incluyendo los temas específicos a abordar, los miembros del equipo que requieren capacitación y el cronograma previsto para llevar a cabo esta capacitación.

Tabla 20

Plan de capacitación para el proyecto K¡Vo!

No. De tema	Nombre del curso	Objetivo del curso	Temario del curso		Duración hrs.	Dirigido a:	Tipo	Imparte	Tipo
PDC3	JavaScript	Introducción a los conceptos básicos de JS	3.1	¿Por qué JavaScript?	19 horas	Desarrollo	Externo	DO y CDT	Técnica
			3.2	¿Qué es *JavaScript* y para qué sirve?					
		Trabajar con objetos	3.3	Elementos de un Lenguaje de Programación: Variables, Funciones y Sintaxis					
			3.4	Qué es una variable en *JavaScript*					

No. De tema	Nombre del curso	Objetivo del curso	Temario del curso		Duración hrs.	Dirigido a:	Tipo	Imparte	Tipo
			3.5	Qué son las funciones en *JavaScript*					
			3.6	¿Qué es una función declarativa y una expresiva?					
			3.7	*Playground*: retorna el tipo					
			3.8	*Scope*					
			3.9	*Hoisting*					
			4	Coerción					
			4.1	Valores: *Truthy* y *Falsy*					
			4.2	Operadores: Asignación, Comparación y Aritméticos.					
			4.3	*Playground*: compara un número secreto					
			4.4	Condicionales: If, *Else, else if*					
			4.5	Switch					
			4.6	*Playground*: tienda de tecnología					
			4.7	¿Qué es un *array*?					
			4.8	*Playground*: detecta el elemento impostor de un *array*					
			4.9	*Loops: For y For...of*					

3.2.5. Identificación de hitos

El *SCRUM Master* lidera esta actividad y su objetivo es realizar un análisis detallado de las actividades que serán consideradas como acontecimientos importantes durante la ejecución del proyecto, como se muestra en la Tabla 21. Estos acontecimientos importantes se

denominan *Hitos*. Además, se identifican los entregables comprometidos para cada uno de estos hitos.

Esta actividad establecerá fechas límite para lograr objetivos específicos y garantizar que el avance de acuerdo con el cronograma previsto. Además, proporciona una base para medir el progreso y evaluar si el proyecto está cumpliendo con sus compromisos en términos de entregables.

Tabla 21

Entregables e Hitos del proyecto K¡Vo!

Productos y entregables	Fase	Quien elabora	Fecha real	Medio
Plan de administración del proyecto.	Planeación	SCRUM Master	12/06/2023	S.A.P.I.
Entregables parciales.	Ejecución	SCRUM Master	19/06/2023	S.A.P.I.
Carta de cierre del proyecto.	Cierre	SCRUM Master	04/08/2023	S.A.P.I.

3.2.6. Realizar el plan de riesgo

Para llevar a cabo esta tarea, el *SCRUM Master* utiliza la *Propuesta Técnica* como punto de partida para identificar los posibles riesgos y oportunidades que pueden surgir durante la ejecución del proyecto con el fin de anticipar posibles problemas y establecer estrategias para mitigar los riesgos o aprovechar las oportunidades. Una vez que se ha completado el análisis de riesgos y oportunidades, se documenta en el *Plan de Riesgos y Oportunidades* en donde se detallan las acciones específicas que se deben tomar en caso de que se materialicen los riesgos identificados, así como medidas preventivas para reducir la probabilidad de ocurrencia de estos riesgos. Estos riesgos pueden estar relacionados con aspectos técnicos, recursos, plazos, presupuesto, entre otros (ver anexo E).

3.2.7. Identificar adaptaciones del proyecto

El *SCRUM Master* lleva a cabo un análisis detallado de los procesos del proyecto y del *Plan de Riesgos y Oportunidades* con el objetivo de identificar posibles adaptaciones o cambios en los procesos, enfoques o estrategias planificadas originalmente, que puedan ser necesarias durante la ejecución del proyecto. Como resultado de esta actividad realizará una guía que detalle las adaptaciones específicas que se deben considerar y cómo implementarlas en caso de ser necesarias.

3.2.8. Realizar cronograma

En esta actividad, el *SCRUM Master* utiliza el *Product Backlog*, que contiene una lista de todas las tareas y actividades necesarias para completar el proyecto, junto con las estimaciones de tiempo para cada una de ellas. El objetivo es elaborar un cronograma detallado, como el que se observa en la Tabla 22, que indica cuándo se llevará a cabo estas tareas a lo largo del proyecto. El cronograma también puede incluir dependencias entre tareas, lo que significa que algunas tareas deben completarse antes de que otras puedan comenzar. Esto se tiene en cuenta al planificar el orden de ejecución de las actividades.

Tabla 22

Cronograma de construcción para el proyecto K¡Vo!

Actividad	**Junio**		**Julio**				**Agosto**
	S0	*S1*	*S2*	*S3*	*S4*	*S5*	*S6*
Análisis y diseño							
Planeación							
Construcción							
Seguimiento y Control							
Liberación							

3.2.9. Integración del plan de proyecto

Para esta actividad el *SCRUM Master* incluye documentos de los planes relacionados con los recursos humanos, los recursos materiales, la capacitación, los hitos y entregables, los riesgos y el cronograma en un único documento: *Plan de Proyecto*.

En el *Plan de Proyecto* se integra y sintetiza información crucial, como quién será responsable de qué tareas, cuáles son los recursos necesarios, cuáles son los hitos clave y entregables, cómo se gestionan los riesgos y cuál es la línea de tiempo del proyecto. Para esta actividad se estableció un formato (ver anexo F).

3.2.10. Revisar plan de proyecto

La actividad de *Revisar plan de proyecto* implica un análisis detallado del plan del proyecto para evaluar si se ajusta a lo comprometido y es viable. Este proceso es llevado a cabo por el *Gerente de Proyecto*, quien es responsable de asegurar que el plan esté alineado con los objetivos y requerimientos del proyecto. Una vez que se ha revisado y aprobado el plan de proyecto, se obtiene la aprobación necesaria para proceder con su implementación.

3.2.11. Realizar presentación Kick Off

Con base en el *Plan de proyecto* aprobado, el *SCRUM Master* va a generar una presentación que sirva como *Kick Off* para el proyecto. El *Kick Off* es una reunión inicial importante que marca el comienzo oficial del proyecto y que involucra a todas las partes interesadas clave.

En la presentación se abordan los puntos trabajados como:

- Introducción del proyecto.
- Equipos y roles.
- Cronograma.
- Recursos humanos y materiales.
- Riesgos y planes de mitigación.
- Canales de comunicación.

- Preguntas y aclaraciones del equipo de desarrollo.

3.2.12. Presentar Kick Off externo

Posteriormente, el *SCRUM Master* también realizará el *Kick off* del proyecto al cliente o las partes interesadas externas para asegurar que éstos estén al tanto de los detalles clave, los objetivos, el alcance y otros aspectos relevantes. Derivado de la reunión se elabora una minuta de la reunión como evidencia, en donde registra los puntos clave discutidos durante la presentación del *Kick Off*.

3.3. Proceso de análisis y diseño

Durante esta etapa, se busca garantizar que el software se construya de acuerdo con los requerimientos del cliente y se diseñan las soluciones técnicas que permitirán la implementación de un software de alta calidad y funcionalidad.

3.3.1. Realizar revisión par

En esta etapa, los analistas pares, que son miembros del equipo encargados de evaluar la calidad y precisión de la información técnica, revisan la información contenida en el *Product Backlog* y los prototipos establecidos en el alcance del proyecto para garantizar que la información contenida en estos sea precisa y suficiente para que los desarrolladores puedan comenzar a trabajar en la implementación del software.

Después de la revisión, se espera que el *Analista Par* proporcione retroalimentación y comentarios sobre la información técnica evaluada. Esto puede incluir correcciones, aclaraciones o sugerencias de mejora, como se puede apreciar en la Tabla 23.

Tabla 23

Revisión par del proyecto K¡Vo!

Verificación de PB Historias	HU-APPC-TEL-001	HU-APPC-TEL-002	HU-APPC-TEL-003	HU-APPC-TEL-004	HU-APPC-TEL-005	HU-APPC-TEL-006
Detalle	¿Cumple?	¿Cumple?	¿Cumple?	¿Cumple?	¿Cumple?	¿Cumple?
¿Es entendible para cualquier persona?	1	1	1	1	1	1
¿La Historia es menor a 10 puntos?	1	1	1	1	1	1
¿Esta utilizada la estructura de una historia de usuario?	1	1	0	1	1	1
¿Tiene especificada la descripción?	1	1	1	1	1	1
¿Incluye los criterios de aceptación y estos contemplan todo los necesario para asegurar que la historia va a ser construida en forma?	0	0	1	0	0	1
¿La historia tiene especificada el sprint en que se está trabajando?	1	1	1	1	1	1
¿La Historia esta priorizada?	1	1	1	1	1	1
¿La historia está relacionada con una épica?	1	1	1	1	1	1
Cantidad de cumplimiento	7	7	7	7	7	8
Cantidad de No cumplimiento	1	1	1	1	1	0
Porcentaje de verificación	85.5	87	87.5	88	89	100
Comentarios	Se solicita mayor especificación de los botones.	Indicar reglas de negocio.	Explicar criterios de aceptación			

3.3.2. Crear casos de prueba

La actividad *Crear Casos de Prueba* involucra que el equipo QA realice un *Análisis detallado de las Historias de Usuario* contenidas en el *Product Backlog* y las pantallas prototipo. A partir de este análisis, los QA crearán una *Matriz de casos de prueba* que permiten verificar que las funcionalidades del software se ajusten a los requerimientos y expectativas del cliente.

La matriz de pruebas incluirá una lista detallada de casos de prueba que se utilizarán para evaluar el software durante el proceso de pruebas (Ver anexo G). Estos casos de prueba son esenciales para garantizar la calidad y el funcionamiento correcto del producto final.

3.3.3. Generar matriz de trazabilidad

Para esta actividad en *Product Owner* creará una *Matriz de trazabilidad* en la que se plasme las actividades descritas en *Product Backlog*, las pantallas prototipo y la matriz de pruebas (ver anexo H). Conforme estas actividades se desarrollen el *Product Owner* realiza una actualización de matriz de trazabilidad indicando el estatus de la actividad para que se rastree y vincule elementos clave del proyecto.

3.3.4. Revisión de aprobación interna

El SCRUM Master analizará la información minuciosamente de *Product Backlog,* que contiene las descripciones detalladas de las Historias de Usuario y otros elementos relacionados con el proyecto, y verifica que todos los aspectos del trabajo a realizar estén correctamente definidos. Posterior al análisis envía un "Correo de Aprobación Interna", que confirma que el *Product Backlog* ha sido revisado y aprobado por el equipo de proyecto.

3.3.5. Revisión de aprobación externa

Para esta actividad se presentará al cliente las Historias de Usuario que se han desarrollado como parte del alcance del proyecto con el fin de que las revise y apruebe oficialmente, para que posteriormente se notifique al equipo de desarrollo que las Historias de usuario cumplen con lo requerido y cuentan con su aprobación.

3.4. Proceso de construcción

El *Proceso de Construcción* en el desarrollo de software es una etapa crucial en la que se lleva a cabo la creación real del producto de software. Esta fase sigue a la fase de Planificación y Análisis y precede a la fase de Pruebas e Implementación.

3.4.1. Realizar planeación de Sprint

Para la *Planeación de Sprint*, tendrán que participar todos los miembros del equipo SCRUM, incluyendo el *SCRUM Master* y el *Product Owner*. El *SCRUM Master* lidera esta actividad, para asegurar que se implementen reglas de SCRUM y de que se cumplan los objetivos del *Sprint*.

Este proceso implica discusiones, estimaciones y acuerdos entre los miembros del equipo para que seleccionen un conjunto *de Historias de Usuario* del *Product Backlog* que serán abordadas, las cuales definirán el objetivo del *Sprint* y el trabajo a realizar por el equipo de desarrollo, también se establecerán tiempos para su realización. El equipo SCRUM debe tener en cuenta la capacidad y las limitaciones del equipo al seleccionar las *Historias de Usuario*.

Esta actividad se repite al comienzo de cada *Sprint* para garantizar una entrega continua y enfocada en el valor del producto.

3.4.2. Realizar diseño de arquitectura

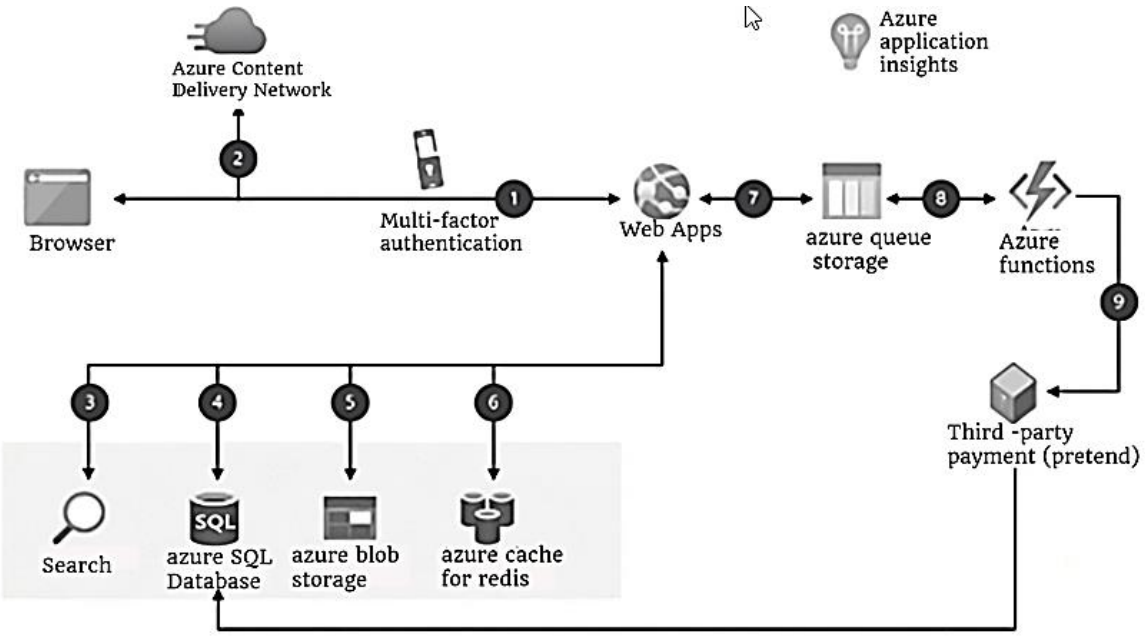
El arquitecto del software es el responsable de definir las propiedades externas visibles y la estructura general del sistema, los componentes y sus interacciones y lineamientos que rigen el diseño, por cada arquitectura se tomarán las siguientes estructuras:

- **Estructura de Módulos**: la cual nos servirá para establecer el funcionamiento interno del sistema.
- **Diagrama de componentes**: éste nos servirá para ingresar cada componente en un archivo físico del sistema o producto de cada proyecto. Aquí se deberá de identificar los componentes que se van a reutilizar, comprar, desarrollar desde cero y las interconexiones que se tendrán con las conexiones externas, aplicar un código de colores para un mejor entendimiento.
- **Diagrama de Clases**: este diagrama describe la estructura de un sistema mostrando las clases del sistema, sus atributos, operaciones y las relaciones entre los objetos.

- **Diagrama de Entidad Relación**: este diagrama es una herramienta para el modelo de datos, la cual facilita la representación de entidades de una base de datos. Se deberá de realizar una toma de decisiones en caso de que se tenga que identificar qué tipo de arquitectura, herramientas, componentes, librerías que pudieran ser utilizadas para el proyecto. En la Figura 13 se muestra la arquitectura propuesta.

Figura 13

Arquitectura propuesta para el proyecto K¡Vo!



3.4.3. Revisión de arquitectura

Con la arquitectura previamente definida, un *Revisor par de Arquitectura* con conocimientos técnicos y experiencia en arquitectura, realiza un análisis exhaustivo para evaluar si la arquitectura es técnicamente viable y cumpla con los requisitos del proyecto, documenta cualquier hallazgo o problema identificados durante el proceso de revisión en un *Checklist de RXP Arquitectura* y asigna una calificación.

3.4.4. Realizar el desarrollo

Durante este proceso, el equipo de desarrollo trabaja en conjunto para construir los módulos o componentes del software según lo especificado en la arquitectura y el análisis previos.

Esto implica escribir y probar el código necesario para implementar las funciones y características del producto.

Para esta actividad, también pueden colaborar con otros equipos, como el equipo de control de calidad, para garantizar la calidad del código.

Para el desarrollo podrá considerarse un entorno configurado de manera estándar para garantizar que el código se escriba de manera consistente y que se sigan procedimientos adecuados.

3.4.5. Realizar pruebas unitarias

Cada desarrollador prueba el código que ha escrito para asegurarse de que funcione correctamente y cumpla con los requisitos definidos. Esto implica ejecutar pruebas en unidades individuales de código para identificar posibles errores o problemas.

De acuerdo con los resultados obtenidos se generan informes que muestran si las pruebas unitarias se aprobaron con éxito o si se encontraron problemas que necesitan corrección.

3.4.6. Revisión por par código

Para esta actividad, la revisión del código es llevada a cabo por R*evisores pares de código*, es decir, desarrolladores o miembros del equipo de desarrollo que no fueron responsables de escribir el código original. Su función es evaluar de manera imparcial y objetiva la calidad del código para identificar fallas en la codificación, garantizar que se cumplan los estándares generales de codificación establecidos y registrar los hallazgos de la revisión en un *Checklist de RXP Código*.

La revisión por pares implica una colaboración estrecha entre los miembros del equipo de desarrollo. Los revisores ofrecen comentarios constructivos sobre el código revisado y trabajan en conjunto para mejorar su calidad.

3.4.7. Realizar integración

Para esta tarea el equipo de *Integración Continua/Entrega Continua* (CI/CD) se encarga de combinar en un ambiente de *Pre-Producción*, el código que ha sido desarrollado por diferentes integrantes del equipo de desarrollo, esto con base en las pautas específicas para realizar la integración de acuerdo con las tecnologías y herramientas utilizadas en el proyecto, con el fin de verificar que las diferentes partes del software funcionen correctamente cuando se ejecutan juntas.

Como actividad de salida se espera un *Código Unificado*, que es el resultado de la integración exitosa de las diferentes partes del código.

3.4.8. Desarrollar pruebas de integración

Con la salida del código unificado, el equipo de *Control de Calidad* (QA) se encarga de diseñar las *pruebas específicas* para probar las diferentes partes del software que interactúan entre sí cuando están integradas. Los QA también se encargarán de ejecutar y documentar dichas pruebas de integración en un *Checklist de Resultados de Pruebas*, con el fin de identificar y resolver problemas que puedan surgir durante esta fase crítica.

Estas pruebas tendrán que ser realizadas en un *Ambiente de Pruebas* específico.

En el *Checklist de Resultados de Pruebas*, se debe detallar los problemas encontrados, errores corregidos y funcionalidades que funcionan según lo previsto.

Los QA trabajarán estrechamente con los desarrolladores para comprender cómo las partes del software se integran y para resolver problemas identificados durante las pruebas.

3.4.9. Realizar integración ambiente productivo

En esta etapa el equipo CI/CD toma el *Código Unificado* que ha sido previamente probado, asegurando que la aplicación esté lista para su implementación: se implementa en un servidor o entorno productivo donde los usuarios finales puedan acceder y utilizar la aplicación.

El equipo de CI/CD colaborará con otros equipos, como el equipo de desarrollo y el equipo de operaciones, para garantizar una implementación exitosa.

3.4.10. Realizar pruebas operativas/sistema

Cuando el aplicativo ya se encuentra en el Server, el equipo QA será el responsable de ejecutar las pruebas funcionales que evalúen el sistema para identificar cualquier error, defecto o problema de funcionamiento.

De acuerdo con las pruebas realizadas, el equipo QA documentara los errores identificados, las mejoras realizadas y la validación de que el sistema cumple con los requisitos establecidos en un Checklist de Resultados de Pruebas, como se muestra en la Tabla 24.

Tabla 24

Pruebas operativas

Tipo de prueba: Pruebas operativas

Fecha de prueba: 04/08/23

Lugar de la prueba: Laboratorio de desarrollo S.A.P.I.

QA:

Descripción de actividades		Resultados /Inspección		
		Cumple	No Cumple	Comentarios
1	Ingreso a URL.	Cumple		
2	Ingreso a pantalla principal.	Cumple		
3	Menú en pantalla principal.	Cumple		
4	A cerca de, Sostenibilidad, atención al cliente, Información, Almacenista, Redes sociales.	Cumple		
5	Reseñas, Carrito de compras, Carrusel de nueva colección.	No cumple		No se muestran las reseñas
6	Productos clasificados por categoría y Colecciones.	Cumple		
7	Búsqueda por nombre del producto.	No cumple		No devuelve todas las búsquedas
8	Nombre y descripción del producto.	Cumple		
9	Imagen del producto.	Cumple		
10	Precio del producto.	Cumple		
11	Galería del producto.	Cumple		
12	Calificación del producto.	Cumple		
13	Métodos de pagos.	Cumple		
14	Pasarela de pagos.	Cumple		
15	Opiniones de los productos.	Cumple		
16	*Like* de productos.	Cumple		
17	Contador de *Like*.	No cumple		No se actualiza contador de pagos
18	Galería de productos adquiridos.	Cumple		

3.4.11. Desarrollar manuales

Esta actividad implica que el *Product Owner* elabore los manuales de usuario, administrativos u operativos que describan todas las funciones del sistema desde la perspectiva del usuario final. Estos manuales deben contener información detallada sobre cómo usar el sistema, incluyendo descripciones de funciones, capturas de pantalla, ejemplos y cualquier otra información relevante que pueda ayudar a los usuarios a operar el sistema de manera efectiva. La creación de estos manuales adicionales dependerá de las necesidades específicas del cliente.

Los desarrolladores pueden colaborar en esta actividad para proporcionar información técnica precisa sobre el funcionamiento del sistema.

3.5. Proceso de liberación

Con actividades realizadas previamente, se asegura que el producto esté listo y cumpla con los estándares de calidad antes de su lanzamiento y se prepara y entrega el producto a los usuarios finales o clientes.

3.5.1. Preparación de reunión

Antes de la liberación, el *SCRUM Master* realiza una solicitud de reunión de presentación con el cliente y prepara una *Carta de entrega parcial* o, en caso de ser la última entrega, se debe generar una *Carta de cierre*. Esta carta especifica qué partes o características del proyecto se presentarán y entregarán al cliente.

3.5.2. Preparar presentación

El equipo de desarrollo identifica las *Historias de usuario* concluidas y que fueron acordadas previamente en la planeación del *Sprint* para presentarlas al cliente para su presentación y validación. También prepara un *checklist de validación* y se encarga de verificar que el ambiente *pre-Productivo* esté operando sin problemas y contenga las historias mencionadas.

3.5.3. Realizar demo

Una vez preparada la presentación, en la reunión se muestra y explica al cliente y las partes interesadas, en el ambiente *pre- productivo* las funcionalidades y la interfaz de la aplicación desarrollada hasta el momento.

Conforme el cliente revisa los requerimientos y decide si el producto entregado cumple con sus expectativas, se registra en el *Checklist de Validación* si las funciones cumplen con lo requerido.

El *Product Owner* documenta en una "Minuta" los puntos abordados durante la reunión, así como los acuerdos, las no conformidades y/o mejoras que surjan.

Como resultado de esta actividad se espera obtener la *Carta de Entrega Parcial Firmada* por el cliente, aprobando el producto entregado.

3.5.4. Notificar puesta en producción

Para esta actividad, el *SCRUM Master* se encargará de coordinar y comunicar al cliente y equipo de desarrollo mediante un *Correo de Visto Bueno*, que las secciones del proyecto que se presentaron en la *demo* han sido revisadas y están listas para ser puestas en producción. Es una notificación formal de que el trabajo ha sido completado y que el cliente puede proceder con la implementación de estas secciones.

Cabe mencionar que la notificación de puesta en producción está sujeta a la aprobación y disposición del cliente.

3.6. Proceso de monitoreo y control

Con el fin de evaluar y medir el desempeño, la calidad y eficiencia del desarrollo del proyecto, se realizará una serie de actividades de *Monitoreo y control* que brinden las métricas que sean representativas para realizar un análisis objetivo mediante reuniones que garanticen que el proyecto se mantenga en camino hacia los objetivos establecidos y permita tomar decisiones preventivas que garanticen la calidad del producto.

3.6.1. Generar reporte de métricas

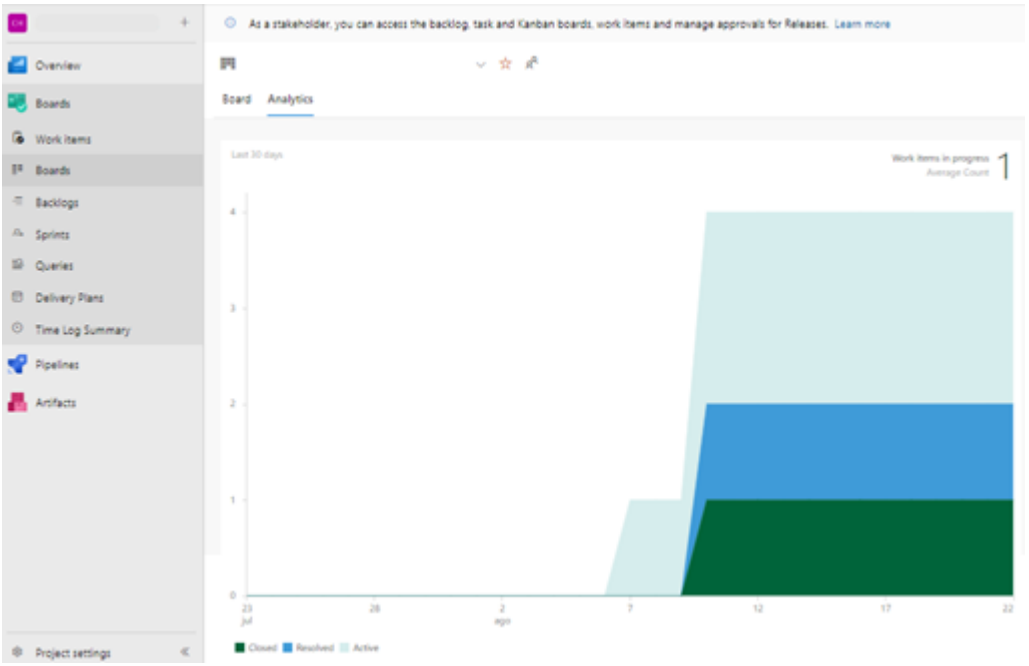
Durante el *Sprint*, el *SCRUM Master* recopilará los datos relevantes como es: la cantidad de *Historias de usuario* completadas, el tiempo dedicado a cada tarea, la calidad del código, la satisfacción del cliente o usuario, y cualquier métrica específica que sea importante para el proyecto.

Una vez recopilados los datos, se realiza un análisis en profundidad con base en el *Plan de proyecto* para comprender el rendimiento del equipo y el *Sprint*. Esto puede implicar el cálculo de métricas clave, como la velocidad del equipo, la tasa de defectos, la capacidad de entrega y otras métricas específicas del proyecto.

Con los datos analizados, el *SCRUM Master* también creará un informe de métricas que presenta de manera clara y concisa la información relevante. Este informe puede incluir gráficos, tablas y resúmenes que ayudarán a visualizar el rendimiento del *Sprint*. El reporte es el que se aprecia en la Figura 14.

Figura 14

Reporte de métricas generado desde Azure para el proyecto K¡Vo!



3.6.2. Realizar reunión de retrospectiva

El reporte de métricas se utiliza como base para la retrospectiva del *Sprint*. Durante esta reunión, el equipo se reúne para analizar los resultados del *Sprint* y se reflexiona cómo se trabajó.

El equipo de trabajo responde a las preguntas, ¿Qué hicimos bien? ¿Qué podemos mejorar? ¿Qué debemos dejar de hacer? Con base al formulario establecido (ver anexo I).

En esta actividad también, se exponen las métricas y se analizan, se documentan las acciones preventivas-correctivas a realizar para el siguiente arranque de trabajo y se genera un reporte de la reunión (ver anexo J).

3.6.3. Realizar monitoreo de riesgo u oportunidades

Durante esta actividad, el *SCRUM Master* lleva a cabo una revisión de los riesgos y oportunidades identificados en el proyecto. Prioriza aquellos con mayor exposición o impacto y actualiza en la *Matriz de Riesgos* las actividades planificadas para abordarlos. Además, verifica si se han detectado nuevos riesgos u oportunidades durante las reuniones o *Sprints*, y se los incorpora a la matriz. Para llevar a cabo esta actividad, se requiere la colaboración de múltiples roles, incluyendo desarrolladores, *tester* y analistas.

3.6.4. Gestión de acciones correctivas

En este paso el *SCRUM Master*, de acuerdo a la *Matriz de Riesgos actualizada* procederá a registrar las acciones que se deben llevar a cabo para mejorar el trabajo en el proyecto. Estas acciones se documentan en un *Plan de Acciones Correctivas*. Además, el *SCRUM Master* realizará una revisión para verificar si existen actividades pendientes de *Sprints* anteriores que requieran seguimiento.

3.7. Proceso de auditoría

Con esta actividad se busca evaluar y verificar la conformidad y eficacia de los procesos y prácticas implementados en la empresa S.A.P.I. con los estándares y requisitos definidos en el modelo CMMI.

3.7.1. Estrategia de auditoría

El equipo auditor es el encargado de crear la *Estrategia de auditoría* en la que define un plan detallado de auditoría, para abordar cómo y cuándo se llevarán a cabo las auditorías. Este documento debe incluir auditorías regulares programadas, así como auditorías específicas en momentos críticos del proyecto; especifica qué se auditará.

En la estrategia de auditoría también se incluirá auditorías de procesos y auditorías de productos, los niveles de gravedad de los hallazgos o problemas identificados durante las auditorías, cómo mejorarlas y los plazos en los que se deberán resolver para garantiza la mejora continua. Además, se establecerán las acciones correctivas y preventivas. La Tabla 25 muestra la estrategia de auditoría que se planteó.

Tabla 25

Estrategia de auditoría

Tipo de Auditoria	Nombre	Tiempo de Resolución	Responsable de Auditar	Escalamiento	Periodicidad
Procesos	Proceso de Alcance	3 días	Auditor	Dirección	Finalizando el proceso
Procesos	Proceso de Planeación	5 días	Auditor	Dirección	Mensual
Procesos	Proceso de Análisis y Diseño	3 días	Auditor	Líder del proyecto	Al finalizar sprint
Procesos	Proceso de Construcción	3 días	Auditor	Líder del proyecto	Al finalizar sprint
Procesos	Proceso de Liberación	5 días	Auditor	Líder del proyecto	Al finalizar sprint
Procesos	Proceso de Monitoreo y control	5 días	Auditor	Dirección	Al finalizar sprint
Procesos	Proceso de Auditoria	5 días	Auditor Externo	Dirección	Trimestral
Producto	Propuesta técnica	3 días	Auditor	Dirección	Finalizando el proceso
Producto	Plan de proyecto	5 días	Auditor	Dirección	Mensual
Producto	*Product Backlog*	3 días	Auditor	Líder del proyecto	Al finalizar sprint

Tipo de Auditoria	Nombre	Tiempo de Resolución	Responsable de Auditar	Escalamiento	Periodicidad
Producto	Matriz de casos de prueba	3 días	Auditor	Líder del proyecto	Al finalizar sprint
Producto	Arquitectura	3 días	Auditor	Líder del proyecto	Al finalizar sprint
Producto	Manual de Usuario	5 días	Auditor	Líder del proyecto	Al finalizar sprint
Producto	*Check list* de validación	5 días	Auditor	Líder del proyecto	Al finalizar sprint
Producto	Carta de liberación o cierre	3 días	Auditor	Líder del proyecto	Al finalizar sprint
Producto	Reportes de Retrospectiva	5 días	Auditor	Dirección	Al finalizar sprint
Producto	Plan de entrenamiento.	5 días	Auditor	Dirección	Trimestral
Producto	Estrategia de auditoria	5 días	Auditor Externo	Dirección	Trimestral
Producto	Plan de auditoría	5 días	Auditor Externo	Dirección	Trimestral

3.7.2. Plan de auditoría

En esta actividad el equipo auditor realiza un *Plan de auditoría* que defina cuándo se realizarán las auditorías y los involucrados en este proceso, de acuerdo con la estrategia de auditorías previamente establecida.

El plan de auditoría puede incluir auditorías regulares, auditorías específicas en momentos críticos del proyecto o auditorías aleatorias según necesite el equipo de trabajo.

Se deberá informar a todas las partes involucradas sobre las auditorías que se llevarán a cabo y cuáles procesos o productos serán auditados con el fin de garantizar la transparencia y la preparación adecuada de los equipos de trabajo. El plan de auditoría del proyecto K¡Vo! Se puede observar en la Tabla 26.

Tabla 26

Plan de auditoría para el proyecto K¡Vo!

PLAN DE AUDITORIA INTERNA DE CALIDAD					
OBJETIVO			**ALCANCE**		**Fecha inicio:**
Cumplimiento de los procesos de la organización			Proceso de auditoría		23/06/2023
					Fecha de Finalización: 13/07/2023
Métodos de auditoria			**Equipo auditor**		
Control de registros					
Fecha	**Tipo de Auditoria**	**Proceso /Actividad**	**Auditado**	**Responsable de auditar**	**Proyecto**
23/06/2023	Procesos	Proceso de Alcance	*Product Owner*	Auditor	K¡Vo!
27/07/2023	Procesos	Proceso de Planeación	*SCRUM Master*	Auditor	K¡Vo!
30/07/2023	Procesos	Proceso de Análisis y Diseño	*Product Owner*	Auditor	K¡Vo!
01/08/2023	Procesos	Proceso de Construcción	Arquitecto/*SCRUM Master*/Desarrollo	Auditor	K¡Vo!
12/08/2023	Procesos	Proceso de Liberación	*SCRUM Master*	Auditor	K¡Vo!
15/08/2023	Procesos	Proceso de Monitoreo y control	*SCRUM Master*	Auditor	K¡Vo!
19/08/2023	Procesos	Entrenamiento Organizacional	Líder de Capacitación	Auditor	K¡Vo!
23/06/2023	Producto	Propuesta técnica	Arquitecto	Auditor	K¡Vo!
27/06/2023	Producto	Plan de proyecto	*SCRUM Master*	Auditor	K¡Vo!
30/07/2023	Producto	*Product Backlog*	*Product Owner*	Auditor	K¡Vo!
30/07/2023	Producto	Matriz de casos de prueba	*Tester*	Auditor	K¡Vo!
01/08/2023	Producto	Arquitectura	Arquitecto	Auditor	K¡Vo!
12/08/2023	Producto	Manual de Usuario	*Product Owner*	Auditor	K¡Vo!
01/08/2023	Producto	*Check list* de validación	*Tester*	Auditor	K¡Vo!
19/08/2023	Producto	Plan de entrenamiento	Líder de capacitación	Auditor	K¡Vo!

3.7.3. Realizar informe de Auditoría

Durante la auditoría el equipo auditor lleva a cabo una revisión exhaustiva de la documentación generada para los procesos del proyecto y registra todas las desviaciones o incumplimientos que indiquen que la organización no está realizando alguna de las actividades propuestas para cumplir con los procesos de CMMI-Dev, analiza la causa raíz y su impacto en los procesos o productos.

El auditor deberá registra de manera detallada todos los hallazgos y áreas de mejora o no conformidades localizadas durante la auditoría para generar un *Informe de auditoría*. La descripción de los hallazgos encontrados puede incluir una descripción precisa, evidencia que respalde el hallazgo y cualquier impacto potencial en la calidad de los procesos o productos.

El informe de auditoría, pueden incluir recomendaciones para abordar los hallazgos identificados. Estas recomendaciones pueden ser acciones correctivas o preventivas destinadas para mejorar la calidad.

3.7.4. Notificación de resultados

El equipo auditor se encargará de notificar en un correo a los miembros del equipo de proyecto, gerentes, responsables de calidad u otras personas involucradas en el proyecto, los resultados obtenidos en la auditoría. En el correo electrónico, se adjunta el archivo generado durante la actividad anterior, donde se describen detalladamente todos los hallazgos.

3.7.5. Resolver las no conformidades

Una vez comprendidas las no conformidades, el equipo de la organización involucrado en el proyecto, trabajará para buscar soluciones efectivas para abordarlas y proceder a su implementación. Esto implica cambios en los procesos, la implementación de medidas correctivas o preventivas y la asignación de responsabilidades específicas para resolver cada no conformidad.

Después de que se han implementado las soluciones y se han resuelto las no conformidades, se debe notificar de manera formal, la solución de los hallazgos con el fin de que las partes interesadas estén informadas de que se han tomado medidas para abordar y corregir las no conformidades identificadas.

3.7.6. Seguimiento a las no conformidades

Cuando el equipo de trabajo involucrado en el proyecto notifique que ha resuelto las no conformidades, el auditor responsable de esta actividad verificará que las no conformidades identificadas previamente se hayan abordado y resuelto de acuerdo con las soluciones propuestas, lo que implica revisar la documentación, los registros y cualquier otra evidencia relevante para asegurarse de que las correcciones se hayan implementado de manera efectiva.

Si se confirma que las no conformidades se han abordado satisfactoriamente, se procede a actualizar el *Informe de auditoría*. Esta actualización refleja el estado actual de las no conformidades y documenta que se han resuelto de manera adecuada.

En caso de que se confirme que las no conformidades no se han resuelto, el auditor envía un correo electrónico de notificación de no resolución a las partes interesadas. Este correo electrónico debe incluir detalles sobre las no conformidades no resueltas, los hallazgos encontrados y cualquier información relevante.

3.7.7. Analizar los resultados

Cada mes el *Auditor* recopila los informes de auditoría correspondientes al período en cuestión, analiza los resultados de las auditorías de manera integral. Esto implica revisar los hallazgos, identificar tendencias o patrones y evaluar la efectividad de las acciones correctivas tomadas en respuesta a auditorías anteriores.

En función de los resultados y hallazgos de las auditorías, se identifican áreas que requieren mejoras o acciones correctivas adicionales y se genera una *Solicitud de mejoras*. Estas mejoras pueden estar relacionadas con procesos específicos, productos o prácticas.

Posteriormente el *Auditor* genera un *Reporte de métricos*, donde resumirá los resultados de las auditorías y proporcionará información cuantitativa sobre el desempeño de los procesos y productos. Estos métricos pueden incluir tasas de no conformidad y gráficas.

3.7.8. Presentación de resultados de auditoría

Con el *Reporte de métricas* el *Auditor* realiza una presentación a los involucrados del proyecto. Si los indicadores revelan que no se han alcanzado los objetivos o metas establecidos, se plantean acciones preventivas y correctivas. Estas acciones pueden ser propuestas para abordar las áreas que requieren mejoras y para evitar futuras no conformidades.

Se genera una Minuta de la presentación que resume los puntos clave discutidos durante la reunión de presentación, para documentar las decisiones tomadas y las acciones propuestas.

Resultados

Con el desarrollo del este proyecto la empresa S.A.P.I. Logró implementar con éxito los procesos de CMMI y la metodología SCRUM en su ciclo de vida de desarrollo de software en la empresa S.A.P.I. Como resultado de la implementación, se observó una serie de mejoras significativas de la calidad de los productos de software desarrollados por la empresa. Se redujeron los errores y se mejoró la eficiencia en el proceso de desarrollo y en la entrega de éste. Esto se tradujo en una mayor satisfacción del cliente y una reducción en los costos asociados con la corrección de errores posteriores a la entrega.

La implementación del marco de trabajo SCRUM permitió entregas más eficientes y a tiempo, al dividir los proyectos en iteraciones más cortas, lo que facilitó la gestión del tiempo y la entrega de software funcional de manera regular. Esto contribuyó a una mayor previsibilidad en los plazos de entrega. Además, las reuniones diarias de SCRUM promovieron una comunicación más efectiva, lo que resultó en una comprensión más clara de los objetivos y requisitos del proyecto y mejoró la colaboración dentro del equipo de desarrollo y los *stakeholders* del proyecto. La combinación de una mayor calidad del software, entregas más eficientes y una mayor capacidad de adaptación llevó a una mejora significativa en la satisfacción del cliente y a que experimentaran un software más confiable.

Por otra parte, la implantación de CMMI mejoró la documentación de los procesos y procedimientos de desarrollo de software, lo que facilitó la comprensión de los procesos y la capacitación de nuevos miembros del equipo, ayudo a reducir los reprocesos, lo que significa que se necesita menos tiempo y recursos para corregir problemas y defectos.

Después de la llevar a cabo el plan de acción propuesto, la empresa alcanzó un nivel de madurez más alto en la gestión de procesos de desarrollo de software, evaluado mediante el modelo CMMI. Esto indicó una mayor capacidad de la organización para gestionar y mejorar sus procesos de desarrollo de software de manera sistemática como se puede observar en la Figura 15.

Figura 15

Resultados de auditoría

Conclusiones

A través de la implementación realizada en la empresa de desarrollo de software S.A.P.I. Donde se ha buscado implementar con éxito los enfoques de CMMI y SCRUM con el objetivo último de obtener la certificación de nivel 5 en CMMI. A través de este proceso, se han logrado avances significativos y se han obtenido valiosos conocimientos y resultados en el camino hacia la excelencia en la gestión de procesos de desarrollo de software. Los resultados obtenidos a lo largo de este proceso han demostrado claramente que la combinación de estos enfoques no solo es efectiva en términos de cumplir con los rigurosos estándares de CMMI, sino que también conduce a una mejora sustancial en la calidad del software entregado, una mayor eficiencia operativa y una mayor satisfacción del cliente.

La implementación conjunta de CMMI y SCRUM en S.A.P.I. No solo ha sido un logro técnico, sino también una demostración del compromiso firme de la organización con la excelencia en la gestión de procesos de desarrollo de software. La empresa se ha comprometido hacia la mejora continua y la búsqueda constante de la calidad. Al alcanzar el nivel 5 de CMMI, S.A.P.I. Establece un estándar excepcional en la industria y demuestra su capacidad para gestionar proyectos de software de la manera más eficiente y efectiva posible. Este logro no solo beneficia a la organización en términos de competitividad y reputación en el mercado, sino que también establece un estándar para la industria en general.

Además de la mejora de la calidad, la implementación de SCRUM ha llevado a una mayor eficiencia operativa en S.A.P.I. Al dividir los proyectos en iteraciones más cortas y enfocarse en entregas incrementales, la empresa ha mejorado la gestión del tiempo y la planificación de proyectos. Las reuniones diarias de SCRUM han promovido una comunicación más efectiva, lo que ha llevado a una mayor comprensión de los objetivos y requisitos del proyecto tanto dentro del equipo de desarrollo como con las partes interesadas del proyecto (*stakeholders*).

En última instancia, esta tesis no solo representa un proyecto de investigación y aplicación exitoso, sino también una guía práctica para otras organizaciones que buscan alcanzar niveles superiores de madurez en la gestión de procesos de desarrollo de software. La implementación de CMMI y SCRUM es una estrategia sólida que puede adaptarse y personalizarse para satisfacer las necesidades específicas de cualquier empresa en la industria del desarrollo de software.

Glosario

Ciclo de vida: periodo que cubre desde la definición de especificaciones, la realización de modificaciones y nuevas versiones, hasta su disposición o substitución.

Software: es el conocimiento basado en instrucciones secuenciales y procesos de manejo de datos.

Stakeholders: son las partes interesadas, individuos, grupos, organizaciones, en un proyecto que pueden influir o verse afectados por las acciones o resultados de éste.

Hitos: eventos o puntos de referencia importantes dentro de un proyecto que marcan logros significativos o etapas clave en su desarrollo.

Feedback: es el proceso de brindar una evaluación en respuesta a una acción para mejorar las futuras acciones o resultados.

Referencias

Aizprua, S., Ortega, A., & Von Chong, L. (2019). Calidad del Software una Perspectiva Continua. *Centros: Revista Científica Universitaria, 8*(2), 120-134. Obtenido de https://uptv.up.ac.pa/index.php/centros/article/view/741

Anderson, D. J. (2010). *Kanban: Successful Evolutionary Change for Your Technology .* Blue Hole Press.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., . . . Thomas, D. (2001). *Manifiesto por el Desarrollo Ágil de Software*. Obtenido de http://agilemanifesto.org/iso/es/manifesto.html

Boehm, B. (1981). *Software Engineering Economics.* Prentice-Hall.

Boehm, B. (1986). A Spiral Model of Software Development and Enhancement. *ACM SIGSOFT Software Engineering Notes*, 14-24.

C., J. (2016). *Importancia de las pruebas de software*. Obtenido de Mundo Testing: https://mundotesting.com/importancia-las-pruebas-software/#:~:text=El%20implementar%20un%20modelo%20de,no%20perder%20a%20los%20existentes.

Calderón Jaraba, J., Lascarro Altamiranda, B., Mattos Badillo, C., & Mercado Cervantes, V. (2022). *Guía ágil para el desarrollo de software en entidades públicas en Colombia empleando CMMI 2.0 y SCRUM.* Tesis de especialidad, Universidad Simón Bolívar, Instituto de Posgrados, Barranquilla-Colombia. Obtenido de https://bonga.unisimon.edu.co/handle/20.500.12442/9429

Chissis, M., Konrad, M., & Shrum, S. (2011). *CMMI for Development: Guidelines for Process Integration and Product Improvement.* Addison-Wesley Professional.

Cockburn, A. (2001). *Agile Software Development: The Cooperative Game.* Addison-Wesley Professional.

Cohn, M. (2009). *Succeeding with Agile: Software Development Using SCRUM.* Addison-Wesley Professional.

Cohn, V. (2010). *Succeeding with Agile: Software Development Using SCRUM.* Pearson.

Fewster, D. G. (2012). *Experiences of Test Automation: Case Studies of Software Test Automation.* Addison-Wesley Professional.

García López, N., & Linares Valencia, J. (2020). *Manual para la implementación de CMMI a nivel de capacidad 2 con SCRUM en el área de desarrollo de software.* Tesis de maestría, Universidad Don Bosco, Facultad de Ingeniería, El Salvador. Obtenido de https://rd.udb.edu.sv/server/api/core/bitstreams/b3febfbe-95c7-4d43-a479-44d970bba104/content

Glazer, H. (2015). *CMMI Assessment Handbook: An Introduction to Appraising CMMI.* Addison-Wesley Professional.

Gómez García, A., Sosa Hernández, M., Verona Marcos, S., & Delgado Dapena, M. (2023). Buenas prácticas de la ingeniería de software: pruebas de software. *Revista Cubana de Transformación Digital, 4*(2), 205:1-13. Obtenido de https://rctd.uic.cu/rctd/article/view/205

Gómez, A., Juristo, N., Montes, C., & Pazos, J. (1997). *Ingeniería del conocimiento.* Madrid: R. Areces.

Guerrero Vera, G., & Guevara Lora, A. (2019). *Integración del modelo CMMI-DEV y el marco de trabajo SCRUM, en el proceso de desarrollo de software.* Tesis de maestría, Universidad Técnica del Norte, Instiituto de Posgrado, Ibarra-Ecuador. Obtenido de http://repositorio.utn.edu.ec/bitstream/123456789/9972/2/PG%20776%20TRABAJO%20GRADO.pdf

Lewis, W. E. (1999). *Software Testing and Continuous Quality Improvement.* Auerbach Publications.

Moorthy, V. (2018). *CMMI Implementation Guide: A Reference Book for Process Improvement and Appraisal.* CRC Press.

Offutt, P. A. (2008). *Introduction to Software Testing.* Cambridge University Press.

Polya, G. (1945). *How to Solve It: A New Aspect of Mathematical Method.* Princeton University Press.

Pressman, R. S. (2014). *Software Engineering: A Practitioner's Approach.* McGraw-Hill Education.

Pressman, R. S. (2020). *Software Engineering: A Practitioner's Approach* (Séptima ed.). McGRAW-HILL.

Royce, W. W. (1970). *Managing the Development of Large Software Systems.*

SEI. (1997). The IDEAL Model. *Software Engineering Institute (SEI)*, 23.

SEI. (2011). *Standard CMMI Appraisal Method for Process Improvement (SCAMPISM).* U.S: Carnegie Mellon University.

SEI. (2019). *History of Innovation at the SEI*. Obtenido de Software Engineering Institute: https://www.sei.cmu.edu/about/historyof-innovation-at-the-sei/index.cfm

SEI. (2019). *The IDEAL model.* US: Software Engineering Institute.

Serna, E., Martínez, R., & Tamayo, P. (2019). Una revisión a la realidad de la automatización de las pruebas del software. *Computación y Sistemas, 23*(1), 169-183. doi:https://doi.org/10.13053/cys-23-1-2782

Sommerville, I. (2015). *Software Engineering.* Pearson.

Sutherland, K. S. (2017). *The SCRUM Guide.* SCRUM.org.

William, E. (2008). *Software Testing and Continuous Quality Improvement.* CRC press.

Anexos.

Anexo A Encuesta realizada al equipo de desarrollo de la empresa S.A.P.I.

Encuesta sobre procesos que se realizan en el desarrollo de los proyectos de software de la empresa S.A.P.I.

Los datos personales e información son confidenciales y de uso exclusivo de la empresa y no podrá ser usado por terceros no autorizados.

*Obligatorios.

1.- Nombre y puesto del colaborador*.

Enfoque organizacional en el proceso (OPF)

2.- Determinar oportunidades de mejora de procesos. * (Marca solo un óvalo por fila.)

- ¿Los procesos de la organización están documentados?
- ¿Se evalúan los procesos de la organización?
- ¿Se documentan las mejoras a los procesos de la organización?
- 3.-Planificar e implementar actividades de mejora*
- ¿Los planes de acciones están documentados?
- ¿Se documenta de los planes de acción implementados?

3.- Desplegar los activos de procesos de la organización e incorporar experiencias*

- ¿Se aplican los procesos estándar de la organización?
- ¿Se ponen en práctica los procesos de la organización?
- ¿Los procesos implementados son monitoreados?
- ¿Se implementan mejoras a los procesos de la organización?

Definición organizacional del proceso (OPD)

4.- Establece activos de procesos* (Marca solo un óvalo por fila)

- ¿Están definidos los procesos estándar de la organización?
- ¿Son definidos los modelos de ciclos de vida?
- ¿Están definidos las guías y criterios para adoptar procesos?

- ¿El entorno de trabajo se rige por estándares?
- ¿Las reglas para el equipo se encuentran definidas las guías?

Entrenamiento organizacional (OT)

5.- Capacidad de entrenamiento organizacional*

- ¿Se han definido las necesidades estratégicas del entrenamiento del proyecto?
- ¿Existe un plan táctico de entrenamiento organizacional?
- ¿Existe un plan con las capacidades de entrenamiento de la organización?

6.- Proveer el entrenamiento necesario*

- ¿Se realiza entrenamiento en la organización?
- ¿Existen registros de los entrenamientos?
- ¿Se realiza evaluación del entrenamiento para verificar su eficacia?

Planificación del proyecto (PP)

7.- Establecer estimaciones*

- ¿Se estima y documenta el plan del proyecto?
- ¿Se estima y documenta los atributos de las tareas y productos del proyecto?
- ¿Se define y documenta el ciclo de vida del proyecto?
- ¿Se estima y documenta el esfuerzo y costo del proyecto?

8.- Desarrollar el plan de proyecto

- ¿Se realiza un cronograma y presupuesto del proyecto?
- ¿Se identifica y documenta los riesgos del proyecto?
- ¿Se realiza una planificación y documentación para la administración de los datos del proyecto?
- ¿Se realiza una planificación y documentación de los recursos necesarios para el proyecto?
- ¿Se realiza una planificación y documentación para la adquisición de conocimiento y habilidades?
- ¿Se realiza una planificación y documentación de la participación de los interesados en el proyecto?

- ¿Se realiza y documenta el plan del proyecto?

9.- Obtener el compromiso de los interesados acerca del plan del proyecto*

- ¿Se realiza una revisión de todos los planes que puedan afectar al proyecto?
- ¿Se realizan ajustes al plan del proyecto para reflejar los recursos estimados versus los disponibles?
- ¿Se documentan los compromisos con respecto al plan?

Monitoreo y control del proyecto (PMC)

10.- Monitorear el proyecto*

- ¿Se realiza un monitoreo de los parámetros de planificación del proyecto?
- ¿Se realiza un monitoreo de los compromisos del proyecto?
- ¿Se realiza un monitoreo de los riesgos del proyecto?
- ¿Se realiza un monitoreo de la administración de datos del proyecto?
- ¿Se realiza un monitoreo de la participación de los interesados del proyecto?
- ¿Se realizan revisiones del avance del proyecto?
- ¿Se realiza revisión del cumplimiento de los hitos del proyecto?

11.- Gestionar acciones correctivas*

- ¿Se realiza un análisis de los temas pendientes?
- ¿Se ejecutan y documentan las acciones correctivas?
- ¿Se hace una administración de las acciones correctivas?

Administración de acuerdos con proveedores (SAM)

12.- Establecer acuerdos con proveedores*

- ¿Existen criterios para determinar el tipo de adquisición?
- ¿Existen criterios para la selección de los proveedores?
- ¿Se realiza un documento de acuerdos con los proveedores?

13.- Satisfacer los acuerdos con los proveedores*

- ¿Se ejecutan los acuerdos con los proveedores?
- ¿Se acepta o rechaza el producto adquirido?
- ¿Se realiza una transición del producto?

Gestión integrada del proyecto (IPM)

14.- Utilizar el proceso definido del proyecto*

- ¿Se encuentra definido el proceso de ejecución de proyectos?
- ¿Se usan los activos de procesos para realizar la planificación del proyecto?
- ¿Se define el entorno de trabajo para el proyecto?
- ¿Se integran los planes del proyecto?
- ¿Se hace una gerencia del proyecto usando los planes integrados?
- ¿Se establecen y se formalizan los equipos del proyecto?
- ¿El proyecto contribuye con los activos de proceso de la organización?

15.- Coordinar y colaborar con los interesados*

- ¿Se realiza una administración de la participación de los interesados en el proyecto?
- ¿Se realiza una administración de las dependencias del proyecto?
- ¿Se resuelven los problemas de coordinación del proyecto?

Gestión de riesgo (RSKM)

16.- Preparar la gestión de riesgo*

- ¿Se definen las fuentes y categorías del riesgo?
- ¿Se definen los parámetros de riesgo?
- ¿Se define una estrategia para la gestión del riesgo?

17.- Identificar y analizar los riesgos*

- ¿Se realiza la identificación de los riesgos?
- ¿Se evalúa, categorizan y priorizan los riesgos?

18.- Mitigar riesgos*

- ¿Se desarrollan planes de mitigación de riesgo?
- ¿Se implementan los planes de mitigación de riesgo?

Administración de requerimientos (REQM)

19.- Gestionar los requerimientos*

- ¿Se comprende el significado de los requerimientos?

- ¿Se obtiene el compromiso de los interesados sobre los requerimientos?
- ¿Se administran los cambios a los requerimientos?
- ¿Se mantiene la trazabilidad bidireccional de los requerimientos?
- ¿Se identifican las inconsistencias entre los requerimientos y otros productos del proyecto?

Desarrollo de requerimientos (RD)

20.- Desarrollar los requerimientos del cliente*

- ¿Se realiza un entendimiento de las necesidades del cliente?
- ¿Se desarrollan los requerimientos del cliente?

21.- Desarrollar los requerimientos del producto.

- ¿Se definen los requerimientos del producto y sus componentes?
- ¿Se asignaron requerimientos a los componentes del producto?
- ¿Se identifican los requerimientos de interfaz?

22.- Analizar y validar los requerimientos*

- ¿Se realiza una definición del concepto, operación y los escenarios de los requerimientos?
- ¿Se realiza una definición de la funcionalidad requerida?
- ¿Se realiza un análisis de los requerimientos?
- ¿Se analiza los requerimientos con el fin de balancear las necesidades y restricciones?
- ¿Se validan los requerimientos?

23.- Solución técnica (TS)

- ¿Se documentan las soluciones alternativas y los criterios seleccionados?
- ¿Se definen los conceptos y escenarios de operación?
- ¿Se seleccionan las soluciones?

24.- Desarrollar el diseño

- ¿Se elabora la arquitectura del producto con sus componentes?
- ¿Se elabora un paquete técnico de datos?
- ¿Se realiza el diseño previo de las interfaces del producto?

25.- Implementar el diseño del producto.

- ¿Se implementa el diseño del producto?
- ¿Se elabora la documentación de soporte?

26.-Preparar la integración del producto (PI)

- ¿Se realiza una estrategia y se documentan?
- ¿Se define y documenta el ambiente de la integración?
- ¿Los criterios de la integración se establecen?

27.- Asegurar la compatibilidad de las interfaces

- ¿Se documentan la descripción de las interfaces?
- ¿se administran las interfaces?

28.- Ensamblar los componentes y entrega del producto.

- ¿Se verifica la disponibilidad de los componentes del producto para su integración?
- ¿Se evalúan los componentes integrados?
- ¿Se lleva a cabo la integración y entrega del producto?

Verificación (VE)

29.- Preparar la verificación.

- ¿Se establecen los artefactos para verificación?
- ¿Se llevan procedimientos y criterios de verificación?

30.- Realizar revisiones de pares.

- ¿Se planea y documenta la revisión entre pares?
- ¿Se realiza una revisión par?
- ¿Se realiza un análisis de los datos obtenidos de la revisión par?

31.- Verificar los artefactos seleccionados.

- ¿Se verifican los artefactos?
- ¿Se analizan los resultados de la verificación de artefactos?
- ¿Se identifican las acciones correctivas?

Validación (VA)

32.- Preparar Validación

- ¿Se lleva a cabo una selección de los a validar?
- ¿Se establece un ambiente de manera adecuada para realizar validaciones?
- ¿Se establecen procedimientos y criterios de validación?

33.- Validar el producto o sus componentes.

- ¿Se lleva a cabo una validación del producto o sus componentes?
- ¿Se analizan los resultados de la validación?

34.- Administración de la configuración.

- ¿Se identifican los ítems de configuración?
- ¿Se utiliza un sistema de gestión de configuración?
- ¿Se crean y liberan líneas base?

35.- Monitoreo y control de cambios?

- ¿Se lleva a cabo un monitoreo de los cambios solicitados?
- ¿Se tiene un control de las solicitudes de cambios?

36.- Establecer la integridad.

- ¿Se documenta la gestión de la configuración?
- ¿Se realiza el proceso de auditoría para las configuraciones?

37.- Evaluar procesos y artefactos.

- ¿Se realiza una evaluación objetiva de los procesos?
- ¿Se evalúan objetivamente los productos y servicios?

Anexo *B* Minuta de Levantamiento de requerimientos

Fecha:		Hora:	
Lugar de Reunión		Responsable	
Área		Proyecto	
Asunto			

Temas tratados				
NO	Tema/Acuerdo	Responsable	Estatus	Fecha Compromiso
1				
2				
3				
4				
5				
6				

Asistencia			
No	Nombre	Empresa y Puesto	Correo
1			
2			
3			

Anexo C Propuesta Técnica.

S.A.P.I

Contenido

Propuesta de solución.

En un mercado en constante evolución, una tienda de ropa en línea necesita destacar y brindar una experiencia excepcional a los clientes. Nuestra solución se enfoca en optimizar la plataforma de comercio electrónico y mejorar la experiencia de compra en línea para aumentar la satisfacción del cliente y las conversiones.

Nuestra solución se basa en tres pilares principales:

1. Plataforma de Comercio Electrónico Mejorada.

2. Experiencia de Compra Personalizada.

3. Mejora en el Proceso de Compra.

Beneficios:

Una plataforma más atractiva y fácil de usar aumentará la tasa de conversión de visitantes a compradores.

La experiencia de compra personalizada y la atención al cliente en línea aumentarán la lealtad de los clientes.

Una plataforma mejorada y una experiencia de compra excepcional mejorarán la percepción de su marca.

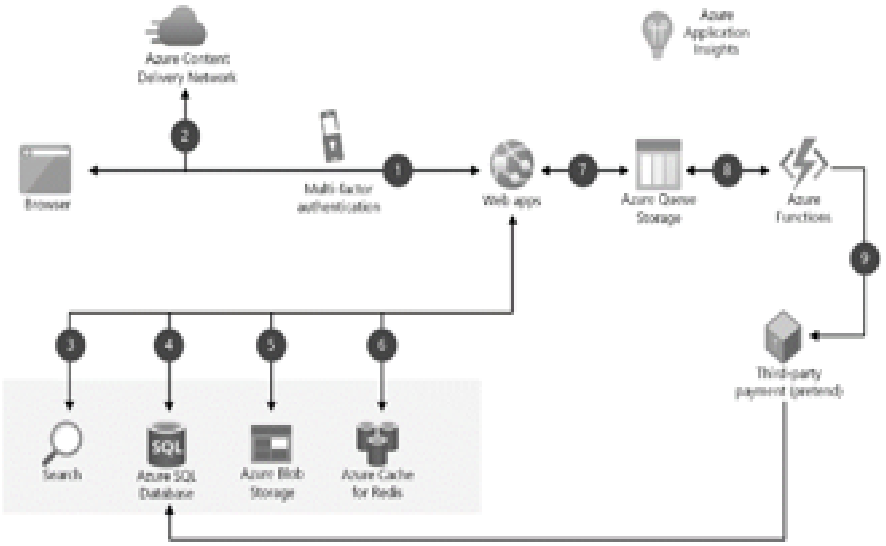
La plataforma de tienda en línea contara con las siguientes secciones:

Menú principal: Contara con las opciones de navegación para las que los usuarios ingresen a las diferentes secciones.

Colecciones: Muestra los productos clasificados por colección que se encuentran disponibles a la venta para su realizar una compra.

**Las pantallas presentadas son prototipos que se utilizan como apoyo para describir las funcionalidades de la tienda en línea "KiVo"*

Arquitectura propuesta.



Frontend de la aplicación web que los usuarios acceden a través del explorador será desarrollada en el lenguaje de JavaScript para mejorar la interactividad y la experiencia del usuario en las páginas web. JavaScript permitirá crear contenido que cambie dinámicamente en función de las acciones del usuario, mostrar mensajes emergentes, crear efectos visuales acceder y modificar el DOM (Document Object Model) de la página, interactuar con APIs externas y navegadores modernos, controlar eventos del usuario.

Cronograma de construcción

Anexo D Propuesta comercial

Contenido

Propuesta Comercial

1. Objetivo

. . . . . .

2. Propuesta de Solución (Propuesta Técnica).

Se coloca la propuesta de solución como es su:
Insertar Liga o hacer referencia a la propuesta Técnica.

- Arquitectura del Sistema
- Beneficios de la solución

3. Propuesta Económica.

Concepto	Cantidad	Monto
Total		

En caso de existir soporte adicional se menciona el costo

Concepto	Cantidad	Monto
Total		

4. Condiciones Comerciales

Leído que fue lo anterior y enteradas las partes del contenido, valor y alcance, lo firman al margen y al calce para su debida constancia en la Lugar a los DD días del mes de XXXXXX del año XXXX.

Firman el presente documento

Anexo E Plan de Riesgos

	IDENTIFICACIÓN DEL RIESGO							Evaluación de Riego %					Analisis de C	
No. De Riesgo	Tipo	Descripción	Consecuencia	Fecha de Identificación	Categoria del Riesgo	Causa Raiz	Restricción Afectada	Probabilidad	Impacto	Nivel de riesgo	Prioridad	Lista de Supervisión	Costo si ocurre el Riesgo	VME(Costo)
1	Oportunidad	Que no sea asignado el personal en tiempo y forma	No terminar la implementación en el tiempo establecido.	12/06/2022	Desempeño	Equipo de trabajo	Tiempo	10%	60%	6,00%	Baja	Si		$0,00
2	Riesgo													
3	Oportunidad													
4	Oportunidad													
5	Oportunidad									0,00%	Cerrado	No		$0,00
6	Oportunidad									0,00%	Cerrado	No		$0,00
													Reservas de Contingencia (Costos)	$0,00

Analisis de Costo y Tiempo			RESPUESTA A LOS RIESGOS						SEGUIMIENTO				
VME(Costo)	Tiempo si ocurre el Riesgo	VME(Tiempo)	Resposable	Estrategia	Plan de Estrategia de respuesta	Resultado esperado	Fecha de inicio planeada	Fecha final planeada	Estatus	Avance	Fecha de Inicio real	Fecha final real	Comentarios o seguimiento
$0,00		0		Compartir									
				Mitigar									
$0,00		0											
$0,00		0											
	Reservas de Contingencia (Tiempo)												
$0,00		0											

Anexo F Plan de proyecto

Contenido

1. Recursos Humanos

A. *Equipo Interno del proyecto*

Nombre	Rol	Responsabilidad	Correo Electrónico	Teléfonos
	Administrador de Proyecto	cumplir en tiempo y costo el proyecto asignado, además de hacer un correctomuso de los recursos.	ejemplo@XXXXX.net	xxx-xxx-xxxx
Desarrollador SR	Líder Tecnico	Experto técnico en el desarrollo del proyecto	ejemplo@XXXXX.net	xxx-xxx-xxxx
Desarrollador back	Desarrollador	Desarrollador Back	ejemplo@XXXXX.net	xxx-xxx-xxxx
Ingeniero Front	Desarrollador	Desarrollador Front	ejemplo@XXXXX.net	xxx-xxx-xxxx

B. Equipo del proyecto del cliente

Nombre	Rol	Responsabilidad	Correo Electrónico	Teléfonos
		Autorizar y dar el visto bueno a los entregables	xxxxx@xxxx.com	xxx-xxx-xxx

C. Plan de Asignación del Personal

- **Calendario de Recursos Humanos**

2. Plan de Capacitación

- **Necesidades de Capacitación**

- **Políticas de Seguridad**

políticas conforme al Reglamento del Cliente

3. Plan de Comunicaciones

A. Objetivo del Documento

Responder a las necesidades de información y comunicación entre los interesados como factor crítico de éxito para el proyecto.

B. Comunicación entre los Involucrados en el Proyecto

Necesidades de Comunicacion de los Involucrados en el Proyecto	Técnicas de Comunicacion a Utilizar
Reunión del equipo de trabajo	Se levantarán minutas y se firmarán (en caso de ser necesario).
Reunión de seguimiento con el cliente y/o Usuario	Se deberá mantener una comunicación directa y en caso de ser necesario; se levantarán minutas y se firmarán.
Correos	En caso de formalizar algún acuerdo.
Llamadas	Documentar por correo acuerdos telefónicos en caso de ser necesario.
Reuniones con el proveedor internos	Se levantarán minutas y se firmarán en caso de ser necesario.
Comunicación Verbal	Deberá sustentarse por medio de un correo en caso de ser necesario.
Proveedores del Cliente	Por llamada telefónica

C. *Matriz de Escalamiento*

Matriz	Nivel	Responsable
Funcional	Nivel 1	Desarrollador
	Nivel 2	Administrador de Proyectos
	Nivel 3	Gerentes o Directivos
Jerárquico	Nivel 1	Administrador de Proyectos
	Nivel 2	Jefe Directo del PM
	Nivel 3	Director

6. Cronograma del proyecto

Colocar la liga del cronograma

7. Matriz de Riesgos

Colocar la liga de la Matriz de Riesgos

8. Guías de Adaptación

Colocar la liga del cronograma

D. Matriz de Notificaciones

Notificaciones	Área	Responsable
Incidentes Graves del Proyecto Cambios Emergentes	Cliente	Cliente
		Sponsor
	Mitrol	Administrador de proyecto
		Jefe Directo del PM
		Director

4. Entregables e Hitos del proyecto

5. Plan de Recursos Materiales

Descripción	Tipo de Recurso	Fecha de Solicitud	Actividad o Hito asociado

Anexo G Matriz de caso de pruebas

Id	Sprint	Caso de Prueba	Descripción	Área Funcional / Sub proceso	Funcionalidad	Datos / Acciones de Entrada	Resultado Esperado	Requerimiento de Ambiente de Pruebas	Procedimientos especiales requeridos	Tipos de Pruebas
1	1	E1. CP01. Pantalla Principal.	Ingresar a la página y visualizar el menú	Página principal	-Menú principal. - Carrusel de nuevas colecciones.	1.- Ingresar a la URL. 2.- Verificar que se visualice: -Menú con las opciones: Colecciones, Categorías. -Presentado - A cerca de. - Carrito de compras - Carrusel de nueva colección. -Atención al cliente - Redes sociales.	El sistema debe mostrar un menú con las siguientes opciones: Colecciones, Categorías, Presentado, A cerca de, Carrito de compras, Carrusel de nueva colección, Atención al cliente y Redes sociales.	Conexión a la base de datos	La página principal de la tienda ha cargado correctamente.	Pruebas funcionales
2	1	E1. CP02 Navegación en el Menú par colecciones	Validar que el sistema permita navegar en el menú	Menú	Colecciones	1.- Seleccionar la opción del menú "Coleccione". 2.- Validar que de desplieguen las opciones -Colección de ánforas -Oro rosa para repensar -Colección Paraíso 3.- Verificar que el sistema redireccione a la sección de colecciones.	El sistema despliega las opciones, Colección de ánforas, oro rosa para repensar, Colección Paraíso y redirecciona a la seccionen de productos clasificados por colecciones seleccionada.	Conexión a la base de datos	Menú configurado de manera correcta	Pruebas funcionales

Id	Sprint	Caso de Prueba	Descripción	Área Funcional / Sub proceso	Funcionalidad	Datos / Acciones de Entrada	Resultado Esperado	Requerimiento de Ambiente de Pruebas	Procedimientos especiales requeridos	Tipos de Pruebas
3	1	E1.CP03 Navegación en el Menú par categorías.		Menú	Categorías	3.- Seleccionar la opción del menú "Categoría". 4.- Validar que de desplieguen las opciones: -Pendientes -Collares -pulsera. - Anillos 3.- Verificar que el sistema redireccione a la sección de colecciones.	El sistema despliega las opciones, Colección de ánforas, oro rosa para repensar, Colección Paraíso y redirecciona a la sección de productos clasificados por colecciones seleccionada.	Conexión a la base de datos	Productos clasificados de manera correcta por categoría y colección	Pruebas funcionales
4	1	E1.CP04 Buscar producto	Realizar la búsqueda de un producto por nombre, categoría o colección	Buscar	Buscar Nombre del producto Descripción Costo	1.- Capturar el nombre, categoría o colección de un producto. 2.- Verificar que el sistema muestre el resultado correcto. 3.- Seleccionar producto. 4.- Dar clic en el botón "Vista previa". 5.- validar que se muestre Nombre del producto Breve descripción del producto. Imagen del producto, costo unitario.	El sistema consulta la base de datos de acuerdo a los parámetros ingresados por el usuario y muestra resultados en la interfaz. El sistema muestra información y galería de imágenes del producto.	Conexión a la base de datos	Productos registrados en la base de datos	Pruebas funcionales

Id	Sprint	Caso de Prueba	Descripción	Área Funcional / Sub proceso	Funcionalidad	Datos / Acciones de Entrada	Resultado Esperado	Requerimiento de Ambiente de Pruebas	Procedimientos especiales requeridos	Tipos de Pruebas
5	1	Procesar pago	Seleccionar producto y procesar pago	Realizar pago	Procesar pago.	1.- Buscar producto. 2.- Seleccionar producto. 3.- Seleccionar cantidad de productos. 4.- Dar clic en el botón "Agregar producto". 5.-Dar clic en carrito de compras. 6.- Seleccionar el botón "Procesar pago" Seleccionar forma de pago. 6.- Ingresar correo. 7.- Ingresar nombre, apellido. 8.- Ingresar datos de la cuenta o tarjeta.	El sistema muestra los productos agregados en la "Cesta de compras" El sistema procesa el pago correctamente y muestra un mensaje de confirmación de compra. Se genera una transacción exitosa en el sistema de pago.	Conexión a la base de datos	Datos de la cuenta o tarjeta validos	Pruebas funcionales
6	1	Pago fallido	Pago fallido por información de tarjeta inválida.	Realizar pago	Procesar pago.	1.- Buscar producto. 2.- Seleccionar producto. 3.- Seleccionar cantidad de productos. 4.- Dar clic en el botón "Agregar producto". 5.- Seleccionar forma de pago. 6.- ingresa información de tarjeta inválida (número de tarjeta incorrecto, fecha de vencimiento expirada, código de seguridad incorrecto).	El sistema redirige al usuario al sitio de PayPal para completar el pago. Después de realizar el pago en PayPal, se muestra un mensaje de confirmación en el aplicativo.	Conexión a la base de datos	Datos de la cuenta o tarjeta no validos	Pruebas funcionales

Id	Sprint	Caso de Prueba	Descripción	Área Funcional / Sub proceso	Funcionalidad	Datos / Acciones de Entrada	Resultado Esperado	Requerimiento de Ambiente de Pruebas	Procedimientos especiales requeridos	Tipos de Pruebas
7	1	Visualización de Opiniones y Calificaciones de Usuarios	Verificar que el sistema muestre las opiniones y calificaciones sección de "Reseñas".	Reseña	Calificación del producto Reseña de los usuarios. Galería de fotos. *Like*.	1.- Ingresar a la url de la tienda. 2.- Navegar a la página del producto deseado. 3.- En la página del producto, identificar y hacer clic en el botón de "Reseñas". 4.- Validar que se muestre Calificación del producto Reseña de los usuarios. Galería de fotos. *like*.	El sistema actualiza el contador de *like* y la interfaz de usuario para mostrar el nuevo número de *like*. Cada reseña muestra la calificación otorgada por el usuario y Se proporciona una galería de fotos	Conexión a la base de datos	El botón de "*like*" está habilitado.	Pruebas funcionales

Anexo H Matriz de Trazabilidad

TITULO DEL PROYECTO	K¡Vo!	FECHA	04/08/2023

Id	Requisito	Tipo	Prioridad	Estado	Objetivo	Entregable	Arquitectura	Prototipo	Parte de código
HU-HS-001	Pantalla de acceso	Funcional	Alta	Atendida	Contar con una pantalla de acceso a la página.	Cuando el usuario ingrese a la pantalla principal de la tienda encontrará: Menú de opciones: Colecciones Categorías Presentado. A cerca de: La marca. Sostenibilidad Reseñas. Carrito de compras. Carrusel de nueva colección. Atención al cliente. Información. Almacenista. Redes sociales.		Pantallas prototipo. K¡Vo!.	
HU-HS-002	Menú principal	Funcional	Alta	Atendida	Menú para visualizar producto clasificados	Cuando el usuario ingrese al sistema encontrara un menú principal (Comercio) con las opciones: COLECCIONES Nueva Colección de ánforas Oro rosa para repensar Colección Paraíso CATEGORÍAS Pendientes Collares Esposas Anillos		Pantallas prototipo. K¡Vo!.	
HU-HS-003	Colecciones	Funcional	Alta	Atendida	Productos clasificados por colecciones	Cuando el usuario ingrese a alguna de las colecciones visualizara: Barra de búsqueda por nombre (Campo de captura) Nombre de la colección (dato consulta). Breve descripción de la colección y producto. Imagen del producto, nombre del producto, calificación, costo unitario, cantidad de productos Vista rápida. Añadir a la cesta		Pantallas prototipo. K¡Vo!.	

Id	Requisito	Tipo	Prioridad	Estado	Objetivo	Entregable	Arquitectura	Prototipo	Parte de código
HU-HS-004	Categorías	Funcional	Alta	Atendida	Productos clasificados por categoría.	Cuando el usuario ingrese a alguna de las colecciones visualizara: Barra de búsqueda por nombre (Campo de captura) Nombre de la categoría (dato consulta). Breve descripción producto. Imagen del producto, nombre del producto, costo unitario, cantidad de productos Vista rápida. Añadir a la cesta		Pantallas prototipo. K¡Vo!.	
HU-HS-005	Pago electrónico	Funcional	Alta	Atendida	Administrar el material multimedia en la página web.	"Cuando el usuario realice el pago de sus productos, encontrará: Elegir método de pago (*select* de iconos) Ingresar los datos de cuenta o tarjeta (campos de captura) Correo electrónico (campo de captura) Nombre y apellido (Campo de captura) dirección (campo de captura)		Pantallas prototipo. K¡Vo!.	
HU-HS-006	Opiniones y calificaciones	Funcional	Alta	Atendida		"Cuando el usuario seleccione el botón de "Reseñas" encontrara: Calificación del producto Reseña de los usuarios. Galería de fotos. *Like*. Esta función debe ofrecer los elementos que permitan al cliente tener una idea general de los productos publicados.		Pantallas prototipo. K¡Vo!.	

Anexo I Formulario de reunión de retrospectivas

Formulario de reunión de retrospectiva

¿Qué salió bien en la iteración? (Aciertos)	¿Que no salió bien en la iteración? (Errores)	¿Qué mejoras vamos a implementar en la próxima iteración? (Recomendación de mejora continua)

Anexo J Reporte reunión de retrospectiva

Reporte de la reunión de retrospectiva

Información del proyecto.

Empresa/ Organización:	
Proyecto	

Información de la reunión

Lugar:	
Fecha:	
Número de iteraciones /Sprint	
Personas convocadas a la reunión:	
Personas que asistieron a la reunión	