Universidad Autónoma de Querétaro

Facultad de Informática

Diseño de un modelo simplificado de calidad para el desarrollo de software basado en ISO/IEC 29110

Tesis

Que como parte de los requisitos
para obtener el Grado de
Maestría en Sistemas Computacionales

Presenta

José Cruz Estefanía Almanza

Dirigido por:

Dr. Ricardo Chaparro Sánchez

Querétaro, Qro. a 20 de enero de 2026

Universidad Autónoma de Querétaro

Facultad de Informática

Maestría en Sistemas Computacionales

Diseño de un modelo simplificado de calidad para el desarrollo de software basado en ISO/IEC 29110

Tesis

Que como parte de los requisitos para obtener el Grado

Educativa Maestría en Sistemas Computacionales

Presenta

José Cruz Estefanía Almanza

Dirigido por:

Dr. Ricardo Chaparro Sánchez

Dr. Ricardo Chaparro Sánchez
Presidente

M.P.S. Flor G. Magalhy Zavala Ponce
Secretario

MSI. Fracisco Paulín Martínez
Vocal

Dra. Sandra Luz Canchola Magdaleno
Suplente

Dra. Ma. Teresa García Ramírez
Suplente

Centro Universitario, Querétaro, Qro.
20 de enero 2026
México

Dedicatorias

Esta tesis está dedicada a mi papá que me está viendo desde el cielo, por su apoyo, su amor, las enseñanzas y por pensar en la familia.

Agradecimientos

Quiero agradecer a mi familia por su amor y su cariño. A mi mamá por creer en mí y apoyarme. Agradezco a mis amigos que nunca me han dejado en los malos momentos, por brindarme la mano y escucharme siempre.

También agradezco a mi director de tesis Dr. Ricardo Chaparro Sánchez que ha sido un pilar fundamental y guía para la realización de la tesis, por mantenerme optimista y por toda la paciencia que me ha tenido.

Agradecimientos especiales al Consejo Nacional de Humanidades, Ciencia y Tecnología (CONAHCYT) por el apoyo económico brindado durante mis estudios de posgrado.

Resumen

Alcanzar la calidad de software es crucial para poder lograr competitividad en la región. Un mal desarrollo de software causa errores en el sistema y resulta en poca fiabilidad hacia las aplicaciones. En México, en 2018, el 76% del software fue desarrollado por pequeñas y medianas empresas. El desarrollo de tecnología ha tenido un crecimiento ascendente. Se puede asegurar la calidad del software siguiendo métricas y estándares internacionalmente reconocidos, la fragmentación de los procesos también ayuda con estos objetivos. Existen extensas y varias metodologías que pueden ser aplicadas, sin embargo, muchas de ellas resultan complejas de entender y aplicar debido al presupuesto reducido de los equipos de software. El principal objetivo es la creación de una metodología simplificada basada en ISO 29110. De esta manera se podrá mejorar la calidad de las empresas de software en la región. La metodología utilizada para elaborar el modelo es *Design Thinking*, una metodología que nos ayuda a la resolución de problemas con un enfoque empático e iterativo basado en cinco etapas, empatizar, definir, idear, prototipar y evaluar. Este modelo simplificado basado en ISO 29110 ayudó al equipo a estandarizar y documentar los procesos de software ya que incluyen los criterios necesarios para implementar el estándar, esto a su vez facilitó una mejor organización dentro del equipo y un mayor control en cada etapa del desarrollo de software.

Palabras clave: ISO/IEC 29110, Calidad, Software

Abstract

Achieving software quality is crucial to be competitive in the region. Bad software development causes system errors and results in unreliability towards applications. In Mexico, in 2018, 76% of the software was developed by small and medium-sized companies. Technology development has grown upward. Software quality can be ensured by following internationally recognized standards and metrics, process fragmentation also helps with these goals. There are extensive and various methodologies that can be applied, however, many of them are complex to understand and apply due to the reduced budget of the software teams. The main objective is the creation of a simplified methodology based on ISO 29110. In this way it will be possible to improve the quality of the software companies in the region. Part of the methodology used is a systematic review for the search and adaptation of the standard towards small and medium-sized entities. The methodology used to develop the model is Design Thinking, a methodology that helps us solve problems with an empathetic and iterative approach based on five stages: empathize, define, ideate, prototype, and test. This simplified model, based on ISO 29110, helped the team standardize and document software processes, as it includes the necessary criteria for implementing the standard. This, in turn, facilitated better organization within the team and greater control at each stage of software development.

Keywords: ISO/IEC 29110, Quality, Software

Índice

Índice de tablas

Índice de Figuras

Abreviaturas y siglas

ISO	***International Organization for Standarization***
IEEE	*Institute of Electrical and Electronics Engineers*
IEC	*International Electrotechnical Commission*
PYMES	Pequeñas y Medianas Empresas
INEGI	Instituto Nacional de Estadística y Geografía
TIC	Tecnologías de la Información y la Computación
ENOE	Encuesta Nacional de Ocupación y Empleo
AWS	*Amazon Web Services*
PIB	Producto Interno Bruto
CASE	*Computer-Aided Software Engineering*
CSLCP	*Cloud Software Life Cycle Process*
COSS	*Commercial Open Source Software*
GQM	*Goal/Question/Metric*
DevOps	*Development and operations*
PM	*Project Management*
UML	*Unified Modeling Language*
API	*Application Programming Interface*
SI	*Software Implementation*
ID	*Identification*
RAM	*Random Access Memory*
PDCA	*Plan, Do, Check, Act*
CST	*Central Standard Time*
AES	*Advanced Encryption Standard*
ER	*Entidad Relación*
LRS	*Locally Redundant Storage*

1. Introducción

Se desarrolló una metodología simplificada basada en el ISO 29110 que ayuda a las Pymes a tener una base sobre los requisitos que implican para esta certificación y lo puedan implementar en los equipos de desarrollo de software. de esta manera, lograr una adaptación gradual hacia la certificación para que en un futuro se pueda comenzar un proceso de evaluación ante un organismo certificador.

A pesar de que existen diferentes enfoques para establecer métricas y lograr la calidad, en la mayoría de los equipos no se sigue un estándar debido a la estructura organizacional de las empresas, al costo y a la preparación previa para la acreditación que implica implementar un estándar (Anacona et al., 2020). Esto indica un grave problema de competitividad, ya que en México alrededor del 76% del desarrollo de software es desarrollado por Pymes (Muñoz et al., 2018), esto nos da una oportunidad para mirar hacia este sector y generar más visibilidad a las empresas regionales.

Existen varias normas para el desarrollo de software, las cuales se pueden encontrar públicamente en Internet en varios sitios y blogs de empresas y programadores. Se basan en el conocimiento empírico de los actores durante este proceso. Las opiniones sobre las recomendaciones son muy diversas y puede resultar confuso para un equipo de desarrollo de software sobre cuál elegir para su proyecto. Esto puede ocasionar que se abstengan de elegir recomendaciones.

Los motivos son lograr que los equipos de software tengan otra herramienta que les ayude a lograr la calidad de software y así crear empresas competitivas en la región. Un mal desarrollo de software causa errores en el sistema y poca fiabilidad ante los usuarios externos.

1.1. Industria de software en México

Las Tecnologías de la información han cambiado la manera en que se hacen las actividades diarias. Los procesos de automatización derivados de su uso han llevado a la creación de nuevas ocupaciones (I.N.E.G.I., 2019) esta creciente demanda abre la oportunidad para buscar que las empresas de software en la región logren posicionarse sobre otras compañías.

En México 75.9% de trabajadores relacionados con las TIC son autoempleados, los cuales son empresas pequeñas que están comenzando y no tienen una planeación y presupuesto para buscar una certificación durante sus inicios.

Por cada 100 personas empleadas en las tecnologías de la información 83 están involucradas en el sector terciario y el resto se encuentra en el sector secundario. Conocer la ocupación nos ayuda a enfocar la metodología en procesos que son conocidos dentro de estas industrias.

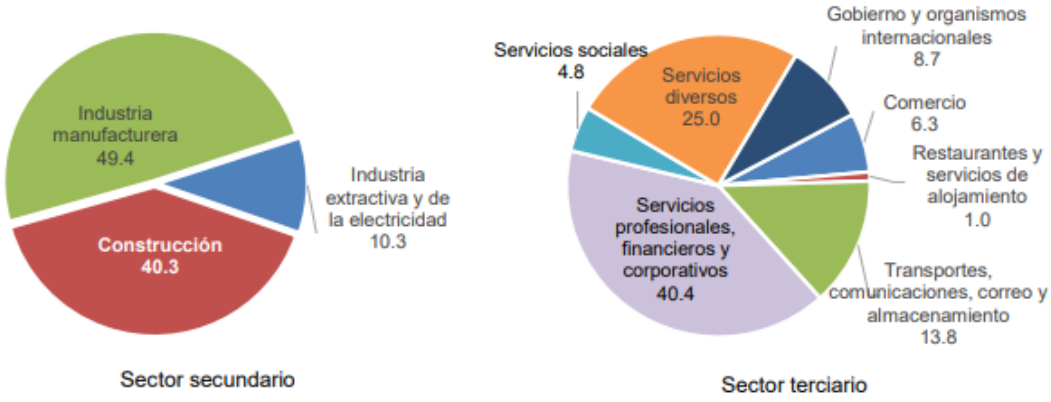
1.2. Distribución del empleo de las TIC por sector económico

Como se muestra en la Figura 1, las personas de las tecnologías de información se encuentran mayormente empleadas en la industria de la manufactura en el sector secundario con un 49.4%, en segundo lugar, con un 40.3% en el área de la construcción y el resto se encuentra empleado industria extractiva y electricidad con un 10.3%.

En cuanto a quienes trabajan en el sector terciario, se encuentran principalmente en servicios profesionales, financieros y corporativos (I.N.E.G.I., 2019).

Figura 1.

Encuesta Nacional de Ocupación y Empleo, ENOE.



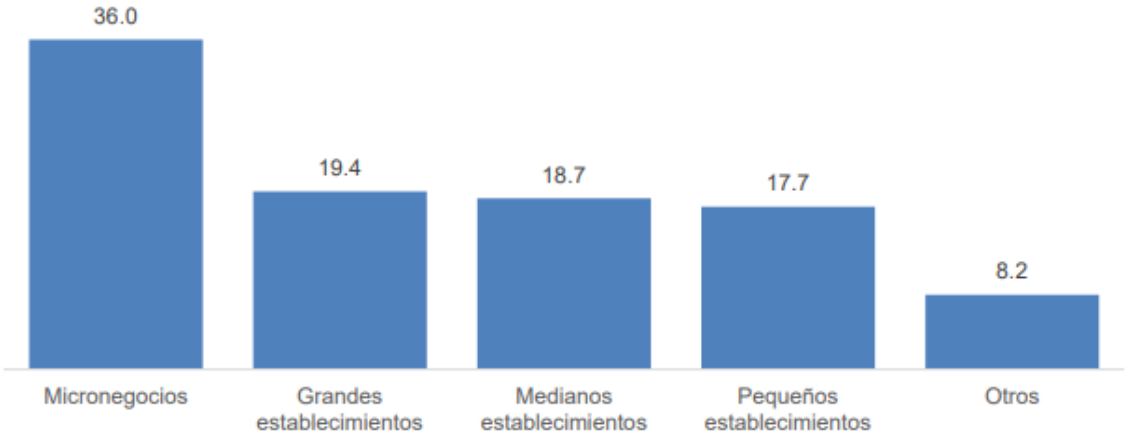
Nota: Datos obtenidos en el cuarto trimestre de 2018.

Fuente: (INEGI, 2019).

Según la Figura 2, de cada 100 personas ocupadas en las TIC 36 trabajan en micronegocios, 19 en empresas grandes, 19 en medianas, 18 en pequeñas, en tanto que ocho en otro tipo de establecimiento.

Figura 2.

Población ocupada en tecnologías de información



Nota: La población considerada es a partir de los quince años en adelante por tamaño del módulo económico en 2018.

Fuente: (INEGI, 2019).

Dentro de las Tecnologías de Información y Comunicación, la clasificación de desarrolladores y analistas de software es la ocupación que más puestos se tiene registrada (I.N.E.G.I., 2019), la importancia hacia el desarrollo de software es muy clara, lo cual nos obliga a mirar cómo podemos mejorar los procesos para tener más visibilidad entre los competidores.

1.3. Industria del software en Querétaro

Según los datos del INEGI la industria de software en Querétaro se encuentra en constante crecimiento y consolidación. Este sector forma parte de las actividades terciarias que incluyen los servicios relacionados con el conocimiento y tecnología.

Tabla 1.

Indicador Trimestral de la Actividad Económica Estatal.

Denominación	2023[1/]					2024			
	Trimestre				Anual	Trimestre			9 meses
	I	II	III	IV		I[1/]	II[1/]	III[2/]	
Total	**4.4**	**1.7**	**5.2**	**5.0**	**4.1**	**2.5**	**5.0**	**2.4**	**3.3**
Primarias	2.5	-4.8	5.8	2.5	1.5	-3.0	-0.4	1.4	-0.5
Secundarias	2.3	0.4	5.1	5.4	3.3	1.8	7.2	4.0	4.3
Terciarias	6.3	3.0	5.3	4.7	4.8	3.3	3.4	1.1	2.6

Nota: Con datos del Estado de Querétaro en los años 2023 y 2024. *Fuente:* (INEGI, 2025).

El Instituto Nacional de Estadística y Geografía (INEGI) reportó que Querétaro experimentó un incremento del 1.8% en su actividad industrial durante el primer semestre de 2024, posicionándose como una de las entidades con mayor crecimiento en el país.

Querétaro ha sido reconocido por su desarrollo en sectores de alta tecnología y servicios especializados, lo que sugiere un entorno favorable para la industria del software. Algunos ejemplos de ello es la inversión de varios centros de datos en Querétaro.

Google Cloud®, se estima que esta inversión generará más de 100,000 empleos y contribuirá con más de 11 mil millones de dólares al PIB mexicano para 2030 (Feijóo, 2024).

Amazon Web Services® informó una inversión equivalente a más de 5,000 millones de dólares para instituir otra región de datos en la región de Querétaro, compuesta por tres zonas de disponibilidad. Además, AWS planea capacitar a 200,000 personas en habilidades relacionadas con la nube, impulsando el desarrollo del talento local (Joaquin D, 2024).

Microsoft Azure®, proporciona acceso a servicios de nube de alta calidad. La iniciativa forma parte de una inversión de 1,100 millones de dólares destinada a impulsar la transformación digital en el país (Microsoft, 2024).

1.4. Pymes en Querétaro

Según el clúster Vórtice IT, especializado en tecnologías de la información, la industria de TI en Querétaro proyectó un crecimiento anual de entre 10% y 12% para 2023. Este incremento se atribuyó a la alta demanda de soluciones de tecnología por parte de empresas que buscan mejorar sus procesos y servicios. En particular, las pymes han mostrado un mayor interés en incorporar herramientas digitales para optimizar la producción, establecer contacto con clientes y gestionar inventarios (Estrella, 2023).

Las Pymes en Querétaro no solo se concentran en comercios y servicios, también tienen presencia en la manufactura, la tecnología y la industria automotriz. Se estima que la pasada pandemia COVID-19 apresuró la inserción de tecnologías en tendencia dentro de negocios desde micro hasta grandes empresas.

2. Marco teórico

2.1. Desarrollo impulsado por comportamiento

El desarrollo basado en comportamiento es una metodología que se basa en pruebas para reducir la brecha de comunicación entre los equipos de negocios y técnicos dentro del desarrollo de software. Nace para proyectos de pequeña escala, el desarrollo basado en comportamiento promueve la compresión compartida mediante casos de pruebas utilizados como especificaciones. Los miembros del equipo de software involucrados consideraron el enfoque útil y proporcionaron retroalimentación lo que condujo a revisiones y mejoras del proceso (Irshad et al., 2021).

2.2. Métricas de software

El aseguramiento de la calidad del software se encuentra estrechamente relacionado con la aplicación sistemática de métricas, las cuales permiten evaluar de manera objetiva tanto el desempeño de los procesos como las características del producto desarrollado. Desde el enfoque de la ingeniería de software, las métricas representan un elemento clave en la toma de decisiones, al ofrecer información cuantificable sobre atributos como la complejidad, la mantenibilidad, la confiabilidad y el grado de cumplimiento de los requisitos.

Asimismo, distintos modelos y normas de calidad coinciden en que la medición continua resulta indispensable para asegurar la mejora de los procesos y el control del producto durante todo el ciclo de vida del software. Bajo esta perspectiva, el uso adecuado de métricas favorece la identificación temprana de desviaciones, la evaluación del impacto de los cambios y la verificación del cumplimiento de los objetivos de calidad definidos, contribuyendo directamente a una gestión de la calidad más eficiente, controlada y predecible (Ming-Chang, 2014), por lo cual es importante mirar las métricas más utilizadas y confiables. Dentro del ámbito de la

ingeniería de software, las metodologías ágiles, como Scrum y Kanban, impulsan la descomposición del trabajo y de los procesos en unidades pequeñas y controlables. Esta forma de organización favorece la colaboración entre los integrantes del equipo, la retroalimentación continua y la capacidad de responder oportunamente a cambios en los requisitos.

Scrum organiza el desarrollo a través de iteraciones breves y roles claramente definidos, promoviendo la transparencia y el monitoreo permanente del avance del proyecto. Por su parte, Kanban pone énfasis en la visualización del flujo de trabajo y en la limitación del trabajo en proceso, con el objetivo de incrementar la eficiencia y minimizar los cuellos de botella. En conjunto, ambas metodologías favorecen la creación de modelos organizacionales flexibles y colaborativos, donde la comunicación y la responsabilidad compartida adquieren un papel fundamental para asegurar la calidad del software, particularmente en equipos de desarrollo de pequeño tamaño(Lizcano et al., 2021), así entonces, se podrá implementar la nueva metodología dentro de un equipo de una manera práctica y sostenible durante su ejecución.

La calidad del software es un factor crítico en el desarrollo de aplicaciones, ya que influye directamente en la confiabilidad, la mantenibilidad, la seguridad y la satisfacción del usuario final. De acuerdo con la literatura de ingeniería de software, la calidad no debe evaluarse únicamente en el producto final, sino que debe gestionarse de manera sistemática a lo largo de todo el ciclo de vida del desarrollo mediante procesos, prácticas y estándares bien definidos.

Diversos organismos de normalización, como la Organización Internacional de Normalización (ISO) y el *Institute of Electrical and Electronics Engineers* (IEEE), coinciden en que la incorporación temprana de prácticas de aseguramiento de la calidad permite reducir los defectos, costos de retrabajo y riesgos asociados a fallos en producción. La calidad del software se concibe como el resultado de un conjunto de actividades estructuradas que abarcan desde la definición de requisitos hasta las

fases de implementación, pruebas y mantenimiento, lo cual resulta especialmente relevante en equipos de desarrollo de tamaño reducido ya que los ayuda a ser más competitivos en la región.

La calidad del desarrollo de aplicaciones se puede medir cuantitativamente (Setiawan et al., 2018). Los criterios incluyen:

1. Corrección, fiabilidad y robustez
2. Rendimiento
3. Usabilidad
4. Verificabilidad
5. Mantenibilidad (que incluye la reparación y la capacidad de evolución)
6. Reutilizable
7. Portabilidad
8. Comprensibilidad
9. Interoperabilidad
10. Productividad
11. Oportunidad
12. Visibilidad

2.3. Metodologías ágiles

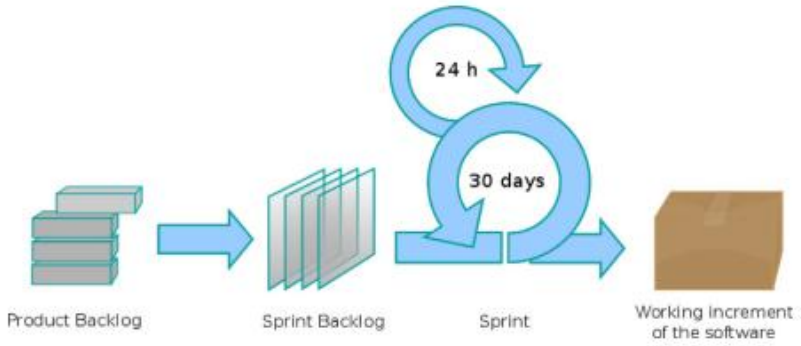
Existen diversas metodologías dentro del desarrollo de software, las más sonadas se encuentran dentro del espectro de las metodologías llamadas ágiles tenemos a *Scrum* y *Kanban* que se enfocan en la colaboración. Las metodologías ágiles tienen una orientación de gestión de proyectos de software que principalmente promueven un desarrolló flexible, la entrega continua y colaboración constante. Es un ciclo iterativo e incremental que se adapta a los cambios durante todo el desarrollo con los usuarios.

2.4. Metodología Scrum

Scrum es un marco que se utiliza para gestionar el desarrollo de productos complicados con un origen desde la década de 1990. Es un marco donde empleamos diversas técnicas y procesos. Scrum aclara la eficacia relativa de la gestión de productos y las técnicas de trabajo para que se pueda mejorar continuamente el producto, el equipo y el entorno laboral (Schwaber & Sutherland, 2017).

Figura 3.

Componentes de un evento de scrum.



Fuente: (Lei et al., 2017).

El contenido de scrum consiste en equipos de scrum, eventos, artefactos y reglas como se muestra en la Figura 3. Las reglas son esenciales para unir el contenido durante el desarrollo del proyecto.

Equipo Scrum

El equipo consiste en el dueño del producto, *Product Owner*, *Scrum Master* y el equipo de desarrollo. Los equipos se autoorganizan y son multifuncionales. Por lo tanto, controlan el proyecto y saben cómo lograr los objetivos sin depender de instrucciones externas (Lei et al., 2017).

Eventos Scrum

Maneja eventos con límite de tiempo en las etapas del desarrollo y organización del proyecto. Se diseñan con el propósito de inspeccionar los artefactos y adaptar nuevos métodos para la continua resolución del proyecto. El objetivo de estos eventos es facilitar la transparencia, la adaptación y la inspección en el proceso de desarrollo (Lei et al., 2017).

Artefactos Scrum

Los artefactos Scrum consisten en el *Backlog* en el Producto, también contiene un *Backlog* en del *Sprint* y la definición de un producto "Terminado" después de cada Incremento, que son todos los elementos completados a lo largo de todos los *Sprints*. El *Backlog* del Producto contiene la lista de requisitos, funciones, mejoras y correcciones necesarias en el producto (Lei et al., 2017).

2.5. Los principios de Kanban

Kanban es una metodología de gestión de proyectos que está basado en la entrega “justo a tiempo”. Su objetivo es definir con exactitud qué tiene que realizarse y cuándo. Se logra priorizando todas las tareas existentes, definiendo un flujo de trabajo y definiendo un plazo para entregar. Como muestra la Figura 4, el proceso Kanban presenta explícitamente las tareas más importantes que requieren mayor atención para reducir el riesgo de que no se completen y para aumentar la flexibilidad entre las demás tareas del proyecto (Lei et al., 2017).

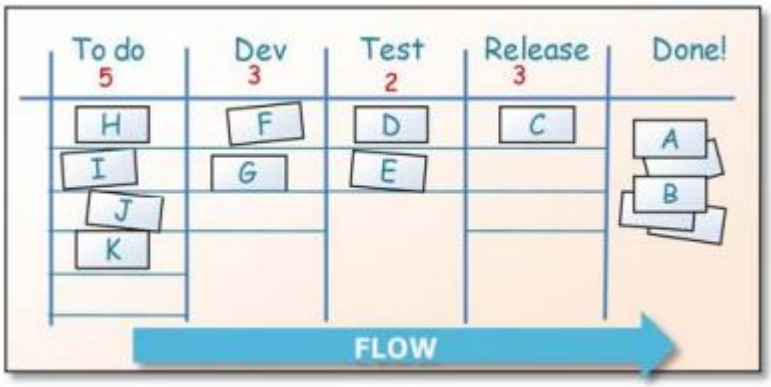
Los siguientes son los principios básicos de Kanban para el desarrollo de software (Lei et al., 2017):

1. Limita el trabajo en proceso.
2. Aporta valor al proceso de desarrollo.
3. Visibiliza el proceso de desarrollo.
4. Incrementa el rendimiento.

5. Usa un backlog fijo.
6. Incorpora calidad.

Figura 4.

Representación típica de un pizarrón Kanban.

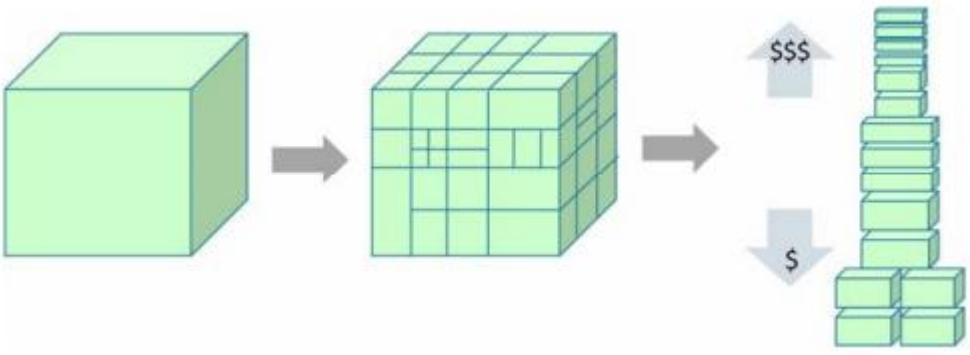


Fuente: (Kniberg, 2010).

Se divide el trabajo creando una lista de entregas pequeñas y concretas como se muestra en la Figura 5. Se ordena esta lista dándole prioridad y calculando el esfuerzo a realizar en cada elemento.

Figura 5.

Representación del trabajo.



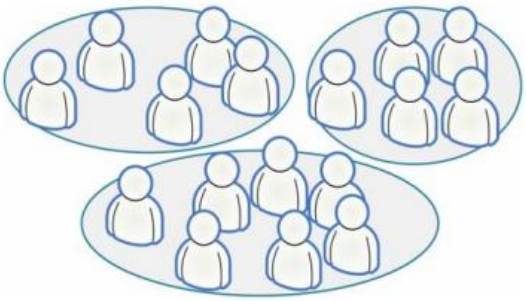
Fuente: (Kniberg, 2010).

Divide la organización en equipos pequeños, multifuncionales y autoorganizados como se muestra en la Figura 6. Es necesario dividir en pequeños equipos

dependiendo de las necesidades de la organización, por ejemplo, el equipo que hace software para contabilidad puede estar dividido del equipo que hace software para nóminas.

Figura 6.

Representación de varios equipos.



Fuente: (Kniberg, 2010).

2.6. Comparación Kanban y Scrum

Kanban es ideal para equipos de mantenimiento, soporte y flujo continuo de tarea, Scrum en contraste está desarrollado para equipos con tarea cíclicas o por fases definidas. Kanban es mejor para equipos con demandas cambiantes, Scrum es más estructurado y es útil en equipos que necesitan más organización. Es posible utilizar elementos de ambas metodologías para adaptarlas a las necesidades del equipo.

Tabla 2.

Comparación estándares de software Scrum y Kanban.

Scrum	Kanban
Contine sus iteraciones con un límite de tiempo anteriormente preestablecidas.	El tiempo en las iteraciones es opcional y pueden llegar más elementos de trabajo.
El equipo define una cantidad específica para cada iteración.	El compromiso de entrega es opcional y se trabajan bajo la demanda y disponibilidad del equipo.

Utiliza la velocidad como medición. (Es la suma de los puntos de historia)	Utiliza el plazo de entrega como métrica predeterminada para la planificación y la mejora de procesos.
Los elementos se acoplan para encajar en el *sprint*.	Es indiferente el número de elementos a trabajar por el equipo.
Diagrama de evolución prescrito (*Burndown chart*).	No se señala ningún tipo de diagrama en particular.
El progreso está limitado por el sprint.	Trabajo en progreso limitado directamente por el estado del flujo de trabajo.
Hay una estimación clara y establecida.	La estimación de entrega no es obligatoria y puede cambiar.
La iteración en curso se encuentra “congelada” y no se pueden agregar nuevos requerimientos.	Se pueden agregar nuevos elementos siempre que haya capacidad disponible.
El *backlog* le pertenece a un equipo en específico.	El tablero puede ser utilizado por varios equipos simultáneamente.
Define tres roles: *Product Owner*, *Scrum Master* y el equipo.	No define roles.
El tablero se reinicia cada que hay un *sprint*.	El tablero de Kanban no cambia.
Define una prioridad de tareas.	La priorización es opcional.

2.7. Mejora en procesos

Para lograr la calidad en proyectos de software, se han aplicado varios mecanismos como el desarrollo impulsado por el comportamiento que ha demostrado que se puede adaptar para lograr la interacción (Irshad et al., 2021), los procesos entonces deben tomar un enfoque que beneficie a la empresa y a los empleados, para que la implementación sea sostenible a corto y largo plazo.

2.8. Productividad en el software

Es necesario tener en cuenta las prácticas que ayudan a mejorar la productividad como el reutilizar el código y el uso de librerías de terceros (Arvanitou et al., 2021), es importante mencionar que la metodología no pretende reemplazar las buenas

prácticas que ya existen dentro del desarrollo de software, al contrario, busca mejorar, estandarizar y adaptarse a los procesos actuales.

3. Antecedentes

El auge de múltiples enfoques desarrollo y herramientas para la codificación de software ha abierto la posibilidad de generar diferentes soluciones. Pero el objetivo más desafiante continúa siendo encontrar mejores técnicas y métodos para desarrollar software resistente a un costo razonable.

3.1. Descripción del problema

Se ha observado en la mayor parte de las normas, como las ISO/IEEE no siempre se acoplan a las necesidades de las Pymes. Estas normas son complejas, requieren recursos, conocimientos y habilidades intensivas como un profundo conocimiento técnico dentro del equipo de software (Lohier & Rodriguez, 2018).

El desarrollo de tecnología ha tenido un crecimiento ascendente en México (I.N.E.G.I., 2019). En la Tabla 1, se muestra que el gasto en investigación y desarrollo tecnológico del sector productivo como proporción del PIB ha duplicado su porcentaje de acuerdo con datos de 2010 a 2016. Esta es una tendencia que sigue subiendo para temas relacionados a ciencia y tecnología.

Tabla 3.

Indicadores INEGI.

Denominación	2010-2011	2012-2013	2014-2015	2016
Empresas que realizaron proyectos de innovación/a	11.7	6.4	7.3	8.3
Empresas que introdujeron al mercado un producto nuevo o que implementaron un proceso novedoso	8.2	2.5	4.4	5
Empresas que desarrollaron al menos un proyecto de innovación en productos o en proceso	10.3	3.4	5.1	5.9
Ingresos de las empresas innovadoras derivados de nuevos productos	39.5	16.7	19.6	21.8

Ingresos de las empresas innovadoras derivados de productos significativamente mejorados	21.9	14.9	43.3	44.1
Ingresos de las empresas innovadoras derivados de productos sin cambios	38.6	68.3	37.1	34.1

Nota: Investigación y desarrollo de las tecnologías el respecto al Producto Interno Bruto.

Fuente: (INEGI, 2019).

La innovación siempre ha estado presente en las empresas mexicanas. Aunque podemos ver un repunte en algunos indicadores, la disminución en otros puede sugerir que las empresas tienden a enfocarse en mejoras de productos existentes en lugar de innovar completamente.

Los ingresos derivados de productos significativamente mejorados aumentaron considerablemente, lo que indica que las empresas están enfocándose en optimizar lo que ya tienen.

En parte “empresas que introdujeron al mercado un producto nuevo o implementaron un proceso novedoso” pasó del 8.2% a solo 2.5% lo cual representa una caída de casi 70%. Esto indica una disminución significativa en el dinamismo innovador durante este periodo, algunas posibles causas podrían ser una menor inversión por parte del sector público o privado, políticas públicas con menor impulso a la innovación empresarial y un contexto económico desfavorable.

Lo cual muestra un área de oportunidad de la política pública, la inversión de privados y un enfoque de competitividad a nivel empresarial.

3.2. Organizaciones que desarrollan los estándares

Las organizaciones que desarrollan estándares de software son entidades nacionales e internacionales dedicadas a establecer normas que aseguren la calidad del desarrollo. Algunas de las más relevantes son ISO, IEEE y IEC.

3.2.1. IEEE (Institute of Electrical and Electronics Engineers)

Se trata de la mayor organización profesional de carácter técnico a nivel mundial, integrada por más de 450,000 ingenieros, científicos, tecnólogos y especialistas distribuidos en más de 160 países, cuyo objetivo principal es promover la innovación tecnológica y la excelencia en beneficio de la sociedad.

Las publicaciones del IEEE constituyen una de las principales fuentes de información técnica para la comunidad científica y de ingeniería a nivel internacional, abarcando diversas áreas del conocimiento. En particular, estas publicaciones concentran alrededor del 30 % de la producción técnica en los ámbitos de la ingeniería eléctrica, la electrónica y las ciencias de la computación (IEEE España, 2026).IEC (*International Electrotechnical Commission*)

La Comisión Electrotécnica Internacional (IEC) integra a cerca de 170 países y ofrece una plataforma internacional de carácter neutral e independiente para la normalización y la evaluación de la conformidad, en la que participan aproximadamente 30,000 expertos a nivel global. Asimismo, gestiona cuatro sistemas de evaluación de la conformidad que garantizan que dispositivos, sistemas, instalaciones, servicios y competencias profesionales cumplan con los requisitos técnicos establecidos.

La IEC ha desarrollado alrededor de 10,000 normas internacionales que, junto con los mecanismos de evaluación de la conformidad, constituyen un marco técnico fundamental para que los gobiernos establezcan infraestructuras nacionales de calidad y para que las organizaciones, independientemente de su tamaño, comercialicen productos seguros y confiables a nivel internacional. Estas normas sirven como referencia para la gestión de la calidad y del riesgo, y son empleadas en procesos de prueba y certificación con el fin de verificar el cumplimiento de las especificaciones declaradas por los fabricantes (IEC, 2026).

3.2.2. Estándar ISO 29110

El propósito del ISO es tener una mejor calidad dentro de todo el proceso de desarrollo, optimizar el desempeño dentro de las organizaciones incluyendo y adaptándose a las metodologías actuales como las ágiles. Este estándar se divide en varias partes dependiendo de la audiencia.

Tabla 4.

Audiencia estándar ISO 29110.

ISO/IEC 29110	Título	Audiencia
Parte 1	Visión general	Empresas, evaluadores, desarrolladoras, consultores, etc.
Parte 2	Marco de referencia y taxonomía	Normalizadores, desabolladores, consultores, no es para empresas.
Parte 3	Guía de evaluación	Evaluadores y empresas.
Parte 4	Especificaciones de los perfiles	Normalizadores, desabolladores, consultores, no es para empresas.
Parte 5	Guía de gestión e ingeniería	Empresas.

Fuente: (Zúñiga, 2021).

Tabla 5.

Perfiles de la parte 5 del ISO 29110.

Perfil	Aplicación principal
Entrada	Pymes sin procesos definidos
Básico	Pymes que desarrollan aplicaciones software con cierta estructura
Intermedio	Pymes con proyectos más complejos o múltiples equipos
Avanzado	Pymes con procesos optimizados

Fuente: (Zúñiga, 2021).

3.3. Estándares existentes

Existen varias métricas para lograr la calidad de software, algunas de ellas apoyadas por empresas privadas o instituciones, algunas de las métricas más populares son:

1. MS-QuAAF, un marco de evaluación cuantitativa destinado a evaluar la arquitectura de software a través de un conjunto de métricas genéricas (Kadri & Aouag, 2021).

2. El uso de un modelo de calidad basado en la ISO/IEC 25010/2011 para el análisis de la calidad del software (Dartora et al., 2021).

3. El uso del desarrollo impulsado por el comportamiento (BDD), ya que facilita la colaboración entre las partes interesadas, y un proceso de BDD adaptado puede ayudar a mejorar la cooperación en un proyecto a gran escala (Irshad et al., 2021).

4. El desarrollo de la Ontología de Referencia de Scrum que es utilizado para construir artefactos de software en una integración semántica, ayudando a mejorar las estimaciones gestionar la productividad y resolver problemas (Santos et al., 2021).

5. El uso de Ingeniería de software Asistida por Computadoras (CASE) sirve para ayudar al equipo de desarrollo incluidos los analistas, codificadores, diseñadores y administradores de proyecto para la construcción de nuevo software (Shafiee et al., 2021).

6. La creación de una nueva metodología, Especificación y Estimación de Indicadores Estratégicos de Software (SESSI) para apoyar a las organizaciones intensivas en desarrollo de software con orientación y herramientas para explotar el desarrollo (Manzano et al., 2021).

7. El desarrollo de software basado en el Software de Código Abierto basado en Consorcios (COSS) como una estrategia alternativa a la creación de recursos de software (Liu et al., 2021).

8. La creación de un nuevo modelo para el Proceso del Ciclo de Vida del Software en la Nube (CSLCP) para producir software en la nube confiable y de calidad cuando se tienen recursos limitados (Alshazly et al., 2021).

9. MeTeaM, el diseño de un modelo describir las características de un equipo de métricas maduro, que diseña, desarrolla, mantiene y evoluciona de manera eficiente el programa de medición de su organización (Meding et al., 2021).

10. La utilización de la ingeniería de software basada en modelos, como el uso de UML, ya que contempla varios enfoques de desarrollo de software (Rosca & Domingues, 2021). Que también se combina con el uso de ciertas herramientas para el modelado como MagicDraw y Papyrus (Planas & Cabot, 2020).

11. Uso del enfoque Meta-Pregunta-Métrica (GQM) para el diseño de un indicador para medir las cualidades en del desarrollo de aplicaciones (Setiawan et al., 2018).

12. El uso de DevOps para aumentar la velocidad, la frecuencia y la calidad de la implementación de proyectos de software (Mishra & Otaiwi, 2020).

13. El uso de CrossRec para ayudar a los desarrolladores en la creación de nuevo software, incluyendo varias métricas de calidad (Nguyen et al., 2020) .

14. La implementación de métricas del ISO/IEC 29110 para los procesos durante el desarrollo Haga clic o pulse aquí para escribir texto.(Garzás et al., 2013). El cual también ha sido combinado con la metodología ágil y el ISO/IEC 12207 (Mas et al., 2020).

15. Implementación del ISO/IEC 10018, 2911 y, OMG Essence enfocado a la calidad de software en pequeñas empresas (Sanchez et al., 2017).

4. Metodología

4.1. Objetivo general

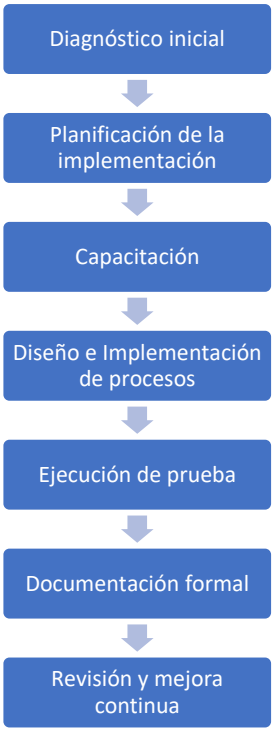
El objetivo general es desarrollar un modelo simplificado basado en ISO 29110 para mejorar la calidad en el desarrollo de software, así como la estandarización y formalización de procesos dentro y fuera del equipo.

4.2. Desarrollo

La norma ISO/IEC 29110-4-1:2018 está pensada fundamentalmente para pequeñas empresas con menos de 25 empleado como se resume en la Figura 7. Es una de las partes de la serie ISO/IEC 29110 y ofrece un enfoque resumido para implementar procesos de calidad.

Figura 7.

Diseño del modelo simplificado del ISO 29110.



El perfil básico del ISO contempla dos procesos clave: la gestión de proyectos "*Project Management*" y por último la implementación de software "*Software Implementation*"

4.3. Diagnóstico inicial

a. Evaluaremos el estado existente de la empresa con respecto a la gestión de proyecto y la implementación de software.

Es trascendental tener un acercamiento con el equipo para conocer la forma de trabajo actual, esto ayuda a evaluar cómo se encuentra el equipo y se conoce cuál es la manera de organización.

b. Identificar brechas y áreas de mejora frente a los requisitos de la norma.

Mediante las listas de conformidad mostradas en el siguiente punto podremos enfocar los esfuerzos de una manera más clara en el área de mejora pudiendo hacer hincapié en procesos que requieran mayor atención.

Estas áreas se documentan formalmente utilizando el formato del Anexo "Entregables" en la sección "Descripción" creando un nuevo documento "Lista_de_conformidad.docx"

c. Lista de conformidad de la implementación del software. Ver tabla 6.

d. Lista de conformidad del control de cambios. Ver tabla 7.

e. Lista de conformidad de la gestión de calidad. Ver tabla 8.

Tabla 6.

Lista conformidad de la implementación de software.

No.	Requisito	Cumple	Comentarios
1	¿Se documentaron los requisitos del software?		
2	¿Se realiza el diseño del software antes del desarrollo?		
3	¿Existen pruebas unitarias y de integración?		
4	¿El software fue revisado por el cliente?		
5	¿Se entrega el producto con documentación adecuada?		

6	¿Se entrega el proyecto en un repositorio conocido por el cliente?		

Tabla 7.

Lista de conformidad del control de cambios

No.	Requisito	Cumple	Comentarios
1	¿Se tiene un repositorio controlado para el código fuente?		
2	¿Se registra y aprueba todo cambio en los entregables?		

Tabla 8.

Lista de conformidad de la gestión de calidad

No.	Requisito	Cumple	Comentarios
1	¿Existe un contrato de aceptación de requerimientos?		
2	¿Se ha designado un gerente de proyecto?		
3	¿Existe un plan de proyecto documentado?		
4	¿Se han identificado y asignado responsabilidades del equipo?		
5	¿Se realiza seguimiento del cronograma y costos?		
6	¿Se gestionan los riesgos del proyecto?		
7	¿Se obtiene un acta de cierre del proyecto?		

4.4. Planificación de la implementación

a. Asignar roles

Se asignan los roles del equipo. Se agrega esta información utilizando el formato del Anexo “Entregables” en la sección “Descripción” creando un nuevo documento “Plan_de_proyecto.docx” descrito en el PM3 de la lista de orden de cumplimiento. Una persona puede tener varios roles. Las columnas sugeridas se pueden consultar en la Tabla 9.

b. Asignación de tareas

Se crea el cronograma. Se agrega esta información utilizando el formato del Anexo “Entregables” en la sección “Descripción” creando un nuevo documento “Plan_de_proyecto.docx” descrito en el PM3 de la lista de orden de cumplimiento.

Independiente de la herramienta que se utilice para la gestión de tiempos, que puede ser Jira, GanttProject, Excel, etc. Las columnas sugeridas se pueden consultar en la Tabla 10.

c. Documentación de metas

Se agrega esta información utilizando el formato del Anexo “Entregables” en la sección “Descripción” creando un nuevo documento “Plan_de_proyecto.docx” descrito en el PM3 de la lista de orden de cumplimiento. El Project Manager debe reconocer cuando una meta se cumpla para incentivar y reconocer el esfuerzo del equipo. Las columnas sugeridas se pueden consultar en la Tabla 11.

d. Elaborar un plan de implementación del sistema de gestión de procesos con un cronograma

Se agrega esta información utilizando el formato del Anexo “Entregables” en la sección “Descripción” creando un nuevo documento “Plan_de_proyecto.docx” descrito en el PM3 de la lista de orden de cumplimiento.

Se sugiere que el diagrama sea modificado a conveniencia y se revise periódicamente. Se recomienda usar diagramas de Gantt. Las columnas sugeridas se pueden consultar en la Tabla 12.

e. Herramientas de gestión a utilizar

Se agrega esta información utilizando el formato del Anexo “Entregables” en la sección “Descripción” creando un nuevo documento “Plan_de_proyecto.docx” descrito en el PM3 de la lista de orden de cumplimiento. Las columnas sugeridas se pueden consultar en la Tabla 13.

f. Riesgos

Se listan los riesgos. Se agrega esta información utilizando el formato del Anexo “Entregables” en la sección “Descripción” creando un nuevo documento “Riesgos.docx” descrito en el PM5 de la lista de orden de cumplimiento. El *Project Manager* tiene la responsabilidad de identificar los riesgos, pero es recomendable que el equipo brinde ayuda y sea una actividad colaborativa. Las columnas sugeridas se pueden consultar en la Tabla 14.

Tabla 9.

Formato roles del equipo de software.

Rol	Responsabilidad	Encargado
Project Manager	Planifica y monitorea el proyecto. Genera los entregables del proyecto. Crea el cronograma. Es un puente de comunicación con el cliente. Define las metas. Define los entregables al cliente. Define las herramientas de gestión a utilizar, por ejemplo: Jira, Trello, Wekan etc.	Nombre de los encargados.
Analista de requisitos	Reúne los requisitos funcionales y no funcionales del cliente. Asegura que los requisitos sean claros y entendibles por el equipo. Genera el documento de requerimientos.	Nombre de los encargados.
Arquitecto de software	Diseña la arquitectura a utilizar. Escoge el grupo de tecnologías a utilizar según las necesidades del cliente. Crea los diagramas UML.	Nombre de los encargados.
Líder del equipo	Verifica que el código siga los estándares de codificaciones. Verifica que el diseño del software se cumpla.	Nombre de los encargados.
Programador	Implementa y codifica el software. Entrega documentación técnica. Sigue estándares de codificación y buenas prácticas. Entrega el código fuente.	Nombre de los encargados.
Responsable de pruebas	Aprueba los entregables. Valida los cambios. Entrega las actas de validación. Entrega los casos de prueba.	Nombre de los encargados.

Tabla 10.

Formato asignación de tareas.

Campo	Descripción
Nombre de la actividad	Define el título del requerimiento
Id de la actividad	Para tener un seguimiento más fácil de los requisitos
Fecha de inicio estimada	Día en que se contempla el inicio de la tarea
Fecha de fin estimada	No tiene que ser exacta
Responsable	Persona asignada
Dependencias	Que tareas o equipos dependen de esta tarea
Estado	Estado de la tarea, por ejemplo, en progreso, en prueba, en desarrollo, terminado, etc.

Tabla 11.

Formato documentación de metas.

Campo	Descripción
Meta	Cuál es el objetivo específico para lograr
Indicador	Como se medirá el cumplimiento de la meta, puede ser, por ejemplo, que la tarea termine antes del tiempo estimada.
Fecha del objetivo	Cuando se espera que se cumpla
Responsable de la meta	Involucrados en el cumplimiento de esta meta

Tabla 12.

Formato descripción de actividades.

Campo	Descripción
Actividad	Breve descripción de la actividad
Duración estimada	Estimado de la duración
Responsable	Responsable de la actividad

Tabla 13.

Formato herramientas a utilizar.

Campo	Descripción
Nombre de la herramienta	Se define el nombre comercial de la herramienta
Propósito	La razón por la cual el equipo necesita hacer uso de están herramienta
Tipo de herramienta	Definir si es una herramienta de documentación, de gestión, control de versiones, etc.
Responsable de la herramienta	Involucrados en el uso de esta herramienta.

Licencia de la herramienta	Indicar si es gratuita, con licencia o hospeda por el equipo.

Tabla 14.

Formato riesgos.

Campo	Descripción
Id del riesgo	Para tener un seguimiento más fácil de los riesgos
Descripción	Una descripción del riesgo, en base a la experiencia del equipo se pueden tener unos riesgos predefinidos de acuerdo con la experiencia de proyectos pasados.
Impacto	Cómo afecta este riesgo al proyecto o involucrados, se puede seguir una métrica como: crítico, alto, medio, bajo
Plan de acción	Cómo se pretende resolver este riesgo
Responsable	Quién es responsable de solucionar el riesgo

4.5. Capacitación

a. Capacitar al equipo con conceptos de la norma ISO, cuáles son los entregables que se esperan y cómo se integran los procesos de gestión.

Se agrega esta información utilizando el formato del Anexo “Entregables” en la sección “Descripción” creando un nuevo documento “Plan_de_capacitacion.docx” descrito en el PM3 de la lista de orden de cumplimiento. Las columnas sugeridas se pueden consultar en la Tabla 15.

b. Crear un enfoque en prácticas de calidad

Se agrega esta información utilizando el formato del Anexo “Entregables” en la sección “Descripción” creando un nuevo documento “Plan_de_calidad.docx” descrito en el PM3 de la lista de orden de cumplimiento. Las columnas sugeridas se pueden consultar en la Tabla 16.

4.6. Diseño e Implementación de procesos

a. Documentar e implementar los procesos de gestión de proyectos

    a. Requerimientos iniciales

Se agrega esta información utilizando el formato del Anexo “Entregables” en la sección “Descripción” creando un nuevo documento “Requerimientos_iniciales.docx” descrito en el PM1 de la lista de orden de cumplimiento.

Son requerimientos no estructurales que se realizan por medio de llamadas, correos, conversaciones, etc. Las columnas sugeridas se pueden consultar en la Tabla 17.

Los requerimientos informales deben convertirse en requerimientos formales, este documento se convertirá posteriormente en la especificación de requisitos de software.

b. Aprobación del cliente

Se agrega esta información utilizando el formato del Anexo “Entregables” en la sección “Descripción” creando un nuevo documento “Aprobacion_del_cliente.docx” descrito en el PM2 de la lista de orden de cumplimiento. Aquí podemos tener tantas aprobaciones como sean necesarias, puede ser por una por cada iteración o una aprobación del proyecto final, según lo que sea más conveniente con cada cliente.

Las columnas sugeridas se pueden consultar en la Tabla 18.

c. Seguimiento y control de proyecto

Se agrega esta información utilizando el formato del Anexo “Entregables” en la sección “Descripción” creando un nuevo documento “Seguimiento_proyecto.docx” descrito en el PM4 de la lista de orden de cumplimiento. Este documento muestra el avance con respecto al cronograma. Las columnas sugeridas se pueden consultar en la Tabla 19.

d. Evaluación de desempeño del equipo

Se agrega esta información utilizando el formato del Anexo “Entregables” en la sección “Descripción” creando un nuevo documento

“Evaluacion_equipo.docx” descrito en el PM6 de la lista de orden de cumplimiento. Ayuda a evaluar las contribuciones del equipo, las competencias desarrolladas y el área de mejora. Las columnas sugeridas se pueden consultar en la Tabla 20.

e. Lecciones aprendidas

Se agrega esta información utilizando el formato del Anexo “Entregables” en la sección “Descripción” creando un nuevo documento “Lecciones_aprendidas.docx” descrito en el PM6 de la lista de orden de cumplimiento. Este documento muestra el avance con respecto al cronograma. Este proceso ayuda a la madurez de un proyecto y a transferir el conocimiento a futuros miembros del equipo. Las columnas sugeridas se pueden consultar en la Tabla 21.

f. Cierre del proyecto.

Se agrega esta información utilizando el formato del Anexo “Entregables” en la sección “Descripción” creando un nuevo documento “Acta_de_cierre.docx” descrito en el PM6 de la lista de orden de cumplimiento. Las columnas sugeridas se pueden consultar en la tabla 22.

b. Documentar e implementar los procesos de implementación de software:

a. Análisis de requisitos

Se agrega esta información utilizando el formato del Anexo “Entregables” en la sección “Descripción” creando un nuevo documento para los requerimientos funcionales “Reqs_funcionales.docx” y otro con el mismo formato para los requerimientos no funcionales “Reqs_no_funcionales.docx” descrito en el SI1 de la lista de orden de cumplimiento. Las columnas sugeridas se pueden consultar en la tabla 23.

b. Diseño técnico

Se agrega esta información utilizando el formato del Anexo “Entregables” en la sección “Descripción” creando un nuevo documento “Arqitectura.docx” descrito en el SI2 de la lista de orden de cumplimiento.

- Diagrama de clases

  Son usados para representar la estructura del sistema, así como mostrar atributos, dependencias y relaciones entre ellos. El diagrama debe alinearse completamente a los requisitos del cliente y debe ser entendido por todo el equipo de desarrollo y validado por el equipo. En este apartado se agregan los diagramas necesarios para la infraestructura tecnológica, por ejemplo, servidores, servicios de aplicaciones, componentes lógicos, así como diagrama que contengan reglas de negocio. Las columnas sugeridas se pueden consultar en la Tabla 24.

- Diseño de la base de datos

  Se recomienda crear un documento como el siguiente para especificar el modelado de la base de datos, es indispensable presentar un diseño lógico de la base de datos como el Modelo Entidad Relación, de ser posible agregar comentarios a las columnas donde se necesite especificar más información. Las columnas sugeridas se pueden consultar en la Tabla 25.

c. Desarrollo del software

Se agrega esta información utilizando el formato del Anexo “Entregables” en la sección “Descripción” creando un nuevo documento “Espeficacion_codigo.docx” descrito en el SI3 de la lista de orden de cumplimiento.

- Gestor de versiones

  Es necesario utilizar un gestor de versiones para mantener el código en un repositorio seguro, algunos ejemplos populares son Git® que se utiliza mediante plataformas como Bitbucket ® y Github ®.

  En este apartado también se especifica la estructura del repositorio como son las ramas, carpetas y convenciones. Las columnas sugeridas se pueden consultar en las Tabla 26 y Tabla 27.

d. Pruebas unitarias e integradas

Se agrega esta información utilizando el formato del Anexo “Entregables” en la sección “Descripción” creando un nuevo documento “Pruebas_unitarias.docx” descrito en el SI4 de la lista de orden de cumplimiento.

Este proceso busca verificar que cada módulo cumpla con los requisitos establecidos. También se debe documentar la herramienta utilizada en el entorno de pruebas. Se justifican los casos de prueba acordados con el cliente. Las columnas sugeridas se pueden consultar en la Tabla 28.

e. Documentación técnica

Se agrega esta información utilizando el formato del Anexo “Entregables” en la sección “Descripción” creando los documentos “Manual_tecnico.docx” y “Manual_Usuario.docx” descrito en el SI5 de la lista de orden de cumplimiento.

Este paso facilita el uso e instalación del software por parte de los miembros implicados, usuarios finales, técnicos, personal de soporte.

- Manual de usuario: Las columnas sugeridas se pueden consultar en la Tabla 29.
- Manual técnico: Las columnas sugeridas se pueden consultar en la Tabla 30.

f. Validación con el cliente

Se agrega esta información utilizando el formato del Anexo “Entregables” en la sección “Descripción” creando el documento “Aprobacion_cliente.docx” descrito en el SI6 de la lista de orden de cumplimiento.

Se busca la validación y acuerdo todos los requisitos. Es importante documentas las observaciones y cambios por menores que parezcan para lo cual es necesario generar un acta de validación. Las columnas sugeridas se pueden consultar en la Tabla 31.

Tabla 15.

Formato descripción de capacitaciones.

Campo	Descripción
Tópico	Tema de la capacitación
Participantes	Participantes en la reunión
Fecha de la capacitación	Fecha en que se realizó
Lugar de la capacitación	Lugar físico o especificar si fue en un canal digital
Responsable	Responsables de la capacitación, pueden ser tanto externos como internos
Justificación	Razón por la cuál es necesario la capacitación
Comentarios relevantes de la capacitación	

Tabla 16.

Formato enfoque de prácticas de calidad.

Criterio	Cumple	Autoriza	Fecha	Comentarios
Cumple con un formato de código específico como PascalCase.				
Se tienen todos los documentos requeridos				
Se autorizan todas las pruebas unitarias				
Se tienen los manuales de usuario				
Se registraron todos los documentos de despliegue de la aplicación				

Se registraron todas las no conformidades

Tabla 17.

Formato requerimientos iniciales.

Campo	Descripción
Fecha	Fecha en que se registra el requisito
Medio	Lugar donde se obtiene el requerimiento informal, puede ser una llamada, un correo, etc.
Solicitante	Persona o personas que solicitan el cambio
Requerimiento	Descripción del requerimiento
Observaciones	Comentarios adicionales

Tabla 18.

Formato aprobación del entregable.

Acta de aprobación del entregable

Nombre del proyecto:

Entregable aprobado: modulo/documento/API

Fecha de entrega: Entrega de la aprobación

Resultado de la revisión: Aprobado/Pendiente/Negado

Observaciones: Comentarios adicionales

Cliente:

Firma:

Fecha:

Tabla 19.

Formato seguimiento del proyecto.

Campo	Descripción
Fecha de seguimiento	Fecha del control
Actividad revisada	Nombre de la tarea o actividad revisada
Estado actual	Estado de la tarea
Avance de la tarea	Porcentaje de avance
Responsable	Persona encargada
Acciones correctivas	Plan de ejecución para los retrasos
Observaciones	Comentarios adicionales

Tabla 20.

Formato evaluación de desempeño.

Campo	Descripción
Nombre del proyecto	Proyecto al que pertenece esta evaluación
Periodo de evaluación	Fecha de evaluación
Evaluador	Responsable de evaluar
Criterios a evaluar	Competencias evaluadas
Evaluado	Persona a evaluar
Observaciones	Comentarios adicionales
Recomendaciones	Sugerencias para posteriores proyectos

Tabla 21.

Formato lecciones aprendidas.

Campo	Descripción
Nombre del proyecto	Nombre principal del proyecto
Inicio del proyecto	Fecha de inicio
Fin del proyecto	Fecha de fin
Aspecto del proyecto	Modulo, planificación, control, etc.
Que funcionó bien	Buenas prácticas y decisiones inteligentes y asertivas
Que no funcionó	Problemas o fallas
Responsable	Quien reporta
Recomendaciones	Acciones para proyectos posteriores
Comentarios	Información adicional

Tabla 22.

Formato cierre del proyecto.

Campo	Descripción
Nombre del proyecto	Nombre
Fecha de cierre	Fecha de cierra
Entregables completados	Lista de todos los entregables entregadores
Firma del cliente	Firma digital o física del cliente
Observaciones	Comentarios adicionales

Tabla 23.

Formato análisis de requisitos.

Campo	Descripción
Id	Id del requerimiento
Tipo de requerimiento	
Descripción	Descripción detallada, clara, competa y verificable del requisito.
Responsable	Responsable de darle seguimiento
Prioridad	Critico/Alta/Mediana/Baja
Criterio de aceptación	Prueba que tiene que pasar para poder cerrar el requerimiento
Dependencia	Dependencia de otros requerimientos
Diseños asociados	Para la trazabilidad del requerimiento: Diagramas Entidad Relación, UML, Casos de uso, etc.
Código fuente relacionado	Para la trazabilidad del requerimiento: Enlace a un repositorio

Tabla 24.

Formato diagrama de clase.

Diagrama de clases
Propósito: Describir el objetivo del diagrama **Descripción del diagrama:** Diseño y modelado del diagrama en UML, es indispensable agregar todas las relaciones internas o externas, también es necesario agregar los componentes de seguridad si el diagrama se trata de

infraestructura, por ejemplo, la autenticación con Entra ID en Oauth o las claves gestionadas con algún baúl de llaves.

Observaciones: Comentarios adicionales

Tabla 25.

Formato diseño de la base de datos.

Diseño base de datos

Propósito: Describir el objetivo de la base de datos

Modelado de datos: Diseño lógico, por ejemplo, modelo entidad relación

Requisitos de integridad: pueden ser reglas de integridad de referencias, restricciones, disparadores, funciones

Consideraciones de rendimiento: índices, particiones, cachés, vistas adicionales

Seguridad: consideración sobre datos sensibles y roles, algunos datos podrían estar encriptados

Observaciones: Comentarios adicionales

Tabla 26.

Formato tipos de ramas para gestionar versiones.

Rama	Propósito
Tipo de rama	Propósito de la rama en los diferentes entornos, puede ser por ejemplo la llama de desarrollo, pruebas y productiva

Tabla 27.

Formato control de versiones.

Control de versiones
Herramienta utilizada: Especificar la herramienta, por ejemplo, “Git”, “Github”, “Gitlab”, “Bitbucket”, etc.
Ruta del repositorio: Si el software se tiene que guardar en algún repositorio del cliente especificar la ruta.
Configuración adicional: Si se necesita algún tipo de certificado o permisos adicionales para subir el código en el repositorio.
Estructura del repositorio: Especificar el uso de cada rama
Observaciones: Comentarios adicionales

Tabla 28.

Formato pruebas unitarias e integradas.

Campo	Valor
Id	Identificador único
Tipo de prueba	Por ejemplo, pruebas unitarias o integradas
Nombre de la prueba	Nombre corto de la prueba a realizar
Descripción	Descripción de la prueba
Módulos implicados	El código o funcionalidad para probar
Entrada	Los datos iniciales para que se ejecute la prueba
Resultado esperado	El resultado que se espera obtener
Resultado del aplicativo	Resultado real de la ejecución
Estado	Estado de la prueba puede ser aprobado o rechazado
Evidencia	Captura de pantalla de la prueba
Fecha	Fecha en que se realizó la prueba
Responsable	Nombre de quién realizó la prueba

Tabla 29.

Formato manual de usuario.

Sección	Contenido
Introducción	Propósito y alcance del manual

Requisitos del sistema	Requisitos técnicos como el navegador, la RAM mínima, sistema operativo etc.
Guía de usuario	Como realizar las tareas comunes
Capturas de pantalla	Ilustraciones detallando el uso
Preguntas frecuentes	Solución para problemas comunes dentro del sistema
Glosario	Definición de términos técnicos para el usuario final

Tabla 30.

Formato manual técnico.

Sección	Contenido
Introducción	Propósito y alcance del manual
Requisitos del sistema	Requisitos técnicos de hardware y software
Configuración	Comandos, rutas, tareas programadas a configurar, parámetros, variables de entorno
Gestión de usuarios	Como gestionar las cuentas
Seguridad	Control de acceso, control de roles, plan de restauración de bases de datos
Mantenimiento	Como brindar actualizaciones de software
Logs	Ubicación de registros de logs, interpretación de los logs
Plan de recuperación a fallos	Procedimiento en caso de contingencia

Tabla 31.

Formato Acta de validación del cliente.

Acta de validación

Nombre del proyecto: Especificar el nombre del proyecto

[Nombre del cliente], declaro que he revisado el sistema en su versión [número de versión] y se han concluido las pruebas exitosamente en compañía del equipo de desarrollo.

Resultado: Producto aprobado/Aprobado con observaciones/Producto rechazado

Observaciones: Comentarios adicionales

Cliente: **Firma:** **Fecha:**

4.7. Ejecución de prueba

a. Aplicar los procesos definidos en uno o dos proyectos reales

En este punto se aborda la aplicación de cada uno de los segmentos que conforman los entregables siguiendo la plantilla de orden de cumplimiento para la implementación del software y la plantilla para la gestión de proyecto. Este documento sirve para hacer un *checklist* de cada uno de los entregables.

b. Monitorear la ejecución y documentar las evidencias: planes, entregables, revisiones y pruebas

Se recomienda utilizar un gestor de proyectos basado en metodologías ágiles como Jira® o similares ya que incluyen la implementación de Kanban y Scrum y es utilizado especialmente en equipos de software. Todos los requerimientos funcionales tienen que ir documentados en el gestor de proyectos. Estos gestores también permiten el seguimiento de control de versiones ayudando a la optimización de tereas e incluye integraciones para el seguimiento de pruebas.

4.8. Documentación formal

a. Elaborar los manuales de proceso, formatos, procedimientos, plantillas

La elaboración y mantenimiento de una estructura clara para la gestión del proyecto ayuda a preparar un entrono para una futura evaluación, así como tener procedimientos inentendibles por los desabolladores y el cliente. Se recomienda estructurar las carpetas y los archivos de la siguiente forma:

- ISO29110_[Nombre_Del_Proyecto]_Documentacion/
  - Diagnostico_Inicial/
    - Lista_de_conformidad.docx
  - Gestion_de_proyecto/
    - Planificacion_de_implementacion/
      - Plan_de_proyecto.docx
      - Riesgos.docx
    - Capacitacion/
      - Plan_de_capacitacion.docx
      - Plan_de_calidad.docx
    - Diseño_e_Implementacion/
      - Requerimientos_iniciales.docx
      - Aprobacion_del_cliente.docx
      - Plan_de_calidad.docx
      - Tablero_inicial.docx
      - Acta_de_cierre.docx
      - Lecciones_aprendidas.docx
      - Evaluacion_equipo.docx
  - Implementacion_del_Sofware /
    - Reqs_funcionales.docx
    - Reqs_no_funcionales.docx

Arquitectura.docx

Especificacion_codigo.docx

Pruebas_unitarias.docx

Manual_tecnico.docx

Manual_usuario.docx

Aprobacion_cliente.docx

4.9. Revisión y mejora continua

a. Establecer un ciclo de mejora continua como PDCA, (Planea, Ejecuta, Verifica, Actúa)

Este proceso ayuda a ajustar los procesos en base a la retroalimentación y las lecciones aprendidas. Se revisan en la plantilla de la Tabla 32 y Tabla 33.

El ciclo consiste en cuatro pasos que enfocado al ISO 29110 puede aplicarse de la siguiente forma:

- Planificar
    - Se define que se va a mejorar, con que recursos y a qué plazo, en este paso se revisa las lecciones aprendidas de otros proyectos si existen y se identifican las fortalezas y debilidades de la gestión de proyectos y de la implementación de software.
- Ejecutar
    - Se implementan los cambios y mejoras planeadas, se capacita al equipo en los nuevos procedimientos, así como a los nuevos miembros en los procesos actuales, se modifican las plantillas generadas y los formatos anteriormente realizados.
- Verificar

- Se evalúa los resultados antes y después para ver si hubo mejora con la implementación del estándar, aquí se pueden generar gráficas de indicadores de cumplimiento de entregas, también se pueden generar encuestas de satisfacción del equipo.

- Actuar
    - Estandarizar los procesos que funcionaron y corregir los que no. Algunas plantillas requerirán más campos por ejemplo dependiendo de la empresa.

4.10. Plantilla orden de cumplimiento para la Gestión del Proyecto

Tabla 32.

Formato de orden de cumplimiento de la gestión del proyecto.

NO.	Tarea	Productos	Plantilla sugerida	Cumple
PM1	Definir el alcance del proyecto	Plan de Proyecto con definición de requerimientos iniciales. Es un listado de requerimientos en lenguaje natural que refleja la necesidad del cliente.	Requerimientos_iniciales.docx	
PM2	Contrato/Acuerdo del proyecto	Aprobación del cliente	Aprobacion_del_cliente.docx	
PM3	Crear un plan de proyecto	Cronograma Entregables Metas En este punto debemos definir los responsables Herramientas de gestión a utilizar Plan de capacitación	Plan_de_calidad.docx Plan_de_proyecto.docx Plan_de_capacitacion.docx	
PM4	Ejecución del proyecto	Asignación de tareas	Tablero_inicial.docx	

PM5	Identificar y gestionar riesgos	Lista de Riesgos	Riesgos.xlsx
PM6	Cerrar formalmente el proyecto	Acta de Cierre Evaluación del desempeño del equipo Aprobación del cliente Lecciones aprendidas	Acta_de_Cierre.docx Lecciones_aprendidas.docx Evaluacion_equipo.docx

4.11. Plantilla orden de cumplimiento para la Implementación del Software

Tabla 33.

Formato de cumplimiento para la implementación de software.

NO.	Tarea	Producto	Plantilla sugerida	Cumple
SI1	Especificar requisitos del software	Especificación de Requisitos los cuales son funcionales y no funcionales con la validación oportuna del cliente	Reqs_funcionales.docx Reqs_no_funcionales.docx	
SI2	Diseñar la solución técnica	Diseño del Software, se puede hacer uso de diagramas UML Diseño de la arquitectura que se planea utilizar puede ser computación en la nube o servidores físicos del cliente Especificar las tecnologías a utilizar	Arquitectura.docx	
SI3	Construcción del software	Código fuente generado en un repositorio de control de versiones o en su defecto un ejecutable de acuerdo con lo acordado con el cliente	Especificacion_codigo.docx	
SI4	Verificar componentes individualmente	Resultados de Pruebas Unitarias para validar que las reglas de negocio cumplen con lo codificado	Pruebas_Unitarias.xlsx	

		Reporte de resultados con inputs y outputs del código.	
SI5	Preparar documentación de usuario y técnica	Manual Técnico con las versiones específicas de las tecnologías utilizadas Cualquier configuración adicional Manual de Usuario	Manual_Tecnico.docx Manual_Usuario.docx
SI6	Prueba de software	Demostración funcional con la aprobación del cliente	Aprobacion_cliente.docx

5. Resultados

Este es un ejemplo real de una empresa, por datos confidenciales se usarán nombres y referencias ficticias.

5.1. Diagnóstico Inicial

a. Lista de conformidad de la implementación del software. Ver tabla 34.

b. Lista de conformidad del control de cambios. Ver tabla 35.

c. Lista de conformidad de la gestión de calidad. Ver tabla 36.

Tabla 34.

Ejemplo lista de conformidad.

No.	Requisito	Cumple	Comentarios
1	¿Se documentaron los requisitos del software?	Si	Se registran los requisitos en Jira
2	¿Se realiza el diseño del software antes del desarrollo?	Si	Se generan diagramas UML
3	¿Existen pruebas unitarias y de integración?	No	Las pruebas son hechas por los usuarios, pero no hay pruebas automatizadas
4	¿El software fue revisado por el cliente?	Si	
5	¿Se entrega el producto con documentación adecuada?	No	En algunos proyectos faltan los documentos de despliegue
6	¿Se entrega el proyecto en un repositorio conocido por el cliente?	Si	

Tabla 35.

Ejemplo lista de conformidad de control de cambios.

No.	Requisito	Cumple	Comentarios
1	¿Se tiene un repositorio controlado para el código fuente?	Si	Se utiliza Bitbucket
2	¿Se registra y aprueba todo cambio en los entregables?	Si	Los cambios son autorizados antes de ir a entornos productivos.

Tabla 36.

Ejemplo lista de conformidad de la gestión de calidad.

No.	Requisito	Cumple	Comentarios
1	¿Existe un contrato de aceptación de requerimientos?	Si	
2	¿Se ha designado un gerente de proyecto?	Si	
3	¿Existe un plan de proyecto documentado?	Si	
4	¿Se han identificado y asignado responsabilidades del equipo?	Si	
5	¿Se realiza seguimiento del cronograma y costos?	Si	
6	¿Se gestionan los riesgos del proyecto?	No	No existen posibles riesgos
7	¿Se obtiene un acta de cierre del proyecto?	Si	

5.2. Planificación de la implementación

a. Asignar Roles. Ver Tabla 37.
b. Documentación de metas. Ver Tabla 38.
c. Elaborar un plan de implementación del sistema de gestión de procesos con un cronograma. Ver Tabla 39.
d. Herramientas de gestión a utilizar. Ver Tabla 40.
e. Riesgos. Ver Tabla 41.

Tabla 37.

Ejemplo asignación de roles.

Rol	Responsabilidad	Encargado
Dueño del producto	Dueño del producto	Vicent Owner
Manager del Producto	Planifica y monitorea el proyecto. Genera los entregables del proyecto. Crea el cronograma. Es un puente de comunicación con el cliente. Define las metas. Define los entregables al cliente. Define las herramientas de gestión a utilizar, por ejemplo: Jira ®, Trello ®, Wekan ® etc.	Juan Manager
Analista de requisitos	Reúne los requisitos funcionales y no funcionales del cliente.	Rafael Analista

	Asegura que los requisitos sean claros y entendibles por el equipo. Genera el documento de requerimientos.	
Arquitecto de software	Diseña la arquitectura a utilizar. Escoge el grupo de tecnologías a utilizar según las necesidades del cliente. Crea los diagramas UML.	Cruz Arquitecto
Líder del equipo	Verifica que el código siga los estándares de codificaciones. Verifica que el diseño del software se cumpla.	Cruz Arquitecto
Programador	Implementa y codifica el software. Entrega documentación técnica. Sigue estándares de codificación y buenas prácticas. Entrega el código fuente.	Carlos Programador
Responsable de pruebas	Aprueba los entregables. Valida los cambios. Entrega las actas de validación. Entrega los casos de prueba.	Elena Pruebas

Tabla 38.

Ejemplo documentación de metas.

Meta	Indicador	Fecha	Responsable
Definir herramientas	Documento finalizado con las tecnologías a utilizar	10 de enero	Cruz Arquitecto
Definir requisitos con cliente	Tareas creadas con los requerimientos funcionales y no funcionales	25 de enero	Rafael Analista
Creación de los diagramas de arquitectura	Documento con el diagrama de arquitectura terminado	10 de febrero	Cruz Arquitecto
Creación de los diagramas de clases	Documento con el diagrama de clases terminado	20 de febrero	Cruz Arquitecto
Asegurar los estándares de codificación	El código pasa las pruebas en el analizador de código SonarQube	30 de marzo	Carlos Programador Cruz Arquitecto
Entrega de código fuente	Validar que hasta el último *commit* se encuentre en estado de *push*	10 de abril	Carlos Programador
Entrega de documentación de despliegue	Entrega de documento entendible claro y preciso	10 de abril	Carlos Programador Cruz Arquitecto

Entrega de manual de usuario	Entrega de manual entendible por usuarios finales	10 de abril	Carlos Programador Cruz Arquitecto
Aprobación de cambios	Todos los cambios deben estar aprobados y probados	15 de abril	Elena Pruebas

Tabla 39.

Ejemplo plan de implementación del sistema de gestión de procesos.

Actividad	Duración Estimada	Responsable
Realizar el diagnóstico	1 semana	Juan Manager
Crear y planificar los roles de los involucrados	2 días	Juan Manager
Documentar metas	3 días	Juan Manager
Definir riesgos y el plan de mitigación	2 días	Juan Manager
Definir las herramientas a utilizar	2 días	Cruz Arquitecto, Rafael Analista
Capacitar al equipo con los conceptos ISO	3 días	Juan Manager
Crear un enfoque de prácticas de calidad	2 días	Juan Manager
Creas la especificación de los requisitos de software	2 semanas	Rafael Analista
Validar y aprobar los requisitos	3 días	Elena Pruebas
Analizar los requisitos y deseñar la solución	2 semana	Cruz Arquitecto
Preparar los diagramas de diseño	2 semana	Cruz Arquitecto
Codificar el software	16 semanas	Carlos Programador
Realizar pruebas unitarias	2 semanas	Elena Pruebas
Validar el software con el cliente	2 semanas	Elena Pruebas
Gestionar cambios o ajustes necesarios	2 semanas	Rafael Analista
Preparar toda la documentación necesaria para proyecto	1 semana	Cruz Arquitecto, Carlos programador
Instalar y entregar el software	1 semana	Cruz Arquitecto, Carlos programador
Cierre de proyectos y lecciones aprendidas	4 días	Juan Manager, Elena Pruebas

Tabla 40.

Ejemplo herramientas de gestión a utilizar.

Herramienta	Propósito	Tipo de Herramienta	Responsable	Licencia

Jira ®	Gestionar las tareas y seguimiento basado en kanban	Gestión de proyecto	Rafael Analista	Gratuito
Git ®		Control de versiones	Cruz Arquitecto	Gratuito
Draw IO ®	Crear diagramas de arquitectura	Diagramas	Cruz Arquitecto	Gratuito
Bitbucket ®	Guardar el código fuente	Gestión de versiones	Cruz Arquitecto	Gratuito
Jenkins ®	Gestionar el despliegue de integración continua	Gestión de software	Cruz Arquitecto	Licencia
Terraform ®	Controlar el despliegue de recursos en Azure	Gestión de software	Cruz Arquitecto	Gratuito
Azure ®	Crear los recursos de la arquitectura	Gestión de software	Cruz Arquitecto	Licencia

Tabla 41.

Ejemplo riesgos.

Id	Descripción	Impacto	Plan de Acción	Responsable
1	Cambios frecuentes durante	Medio	Asegurar que los requerimientos están claramente definidos	Rafael Analista
2	Estimación inexacta en la planeación	Alto	Priorizar claramente los nuevos requerimientos	Rafael Analista
3	Comunicación deficiente con el cliente	Medio	Crear reuniones de seguimiento	Rafael Analista
4	Falta de claridad en los roles	Medio		Juan Manager
5	Tecnología inadecuada	Medio	Revisar la compatibilidad de las herramientas a utilizar	Cruz Arquitecto
	Falta de conocimiento técnico por parte del equipo	Medio	Realizar un curso rápido con los miembros que desconozcan las herramientas	Cruz Arquitecto
6	Rotación de personal clave	Medio	Compartir el conocimiento con otros miembros del equipo	Juan Manager
7	Sobrecarga de trabajo	Medio	Reasignar las tareas conforme a las prioridades	Rafael Analista
8	Fallas en las herramientas	Baja	Buscar herramientas de reemplazo y guardar	Cruz Arquitecto

			toda la información posible	
9	Definición ambigua de los requisitos	Medio	Obtener más detalles en los requisitos necesarios	Rafael Analista
10	Presupuesto insuficiente	Alta	Ajustar los requerimientos al presupuesto	Juan Manager
11	Actividades sin dependencias claras	Medio	Validar nuevamente las dependencias de las tareas	Rafael Analista
12	Contratos pocos claros	Alta	Revisar las cláusulas del contrato	Juan Manager

5.3. Capacitación

a. Capacitar al equipo con conceptos de la norma ISO, cuáles son los entregables que se esperan y cómo se integran los procesos de gestión. Ver Tabla 42.

b. Crear un enfoque en prácticas de calidad. Ver Tabla 43.

Tabla 42.

Ejemplo cronograma de capacitación.

Tópico	Participantes	Lugar	Fecha	Responsable	Justificación
Introducción a ISO 29110	Juan Manager, Todo el equipo de desarrollo	Virtual	6 De enero	Juan Manager	Explicar la norma, su propósito, beneficios y estructura general.
Gestión de proyectos con un enfoque ISO 29110	Juan Manager, Rafael Analista	Virtual	6 de enero	Rafael Analista	Capacitar al equipo sobre la planificación del proyecto, así como el monitoreo y control.
Implementación de software con enfoque ISO 29110	Juan Manager, Todo el equipo de desarrollo	Virtual	7 de enero	Cruz Arquitecto	Enseñar las mejores prácticas para la codificación, el diseño y las

					pruebas de software.
Revisión de requisitos	Juan Manager, Rafael Analista	Virtual	8 de enero	Rafael Analista	Enseñar como obtener requisitos de manera eficaz.
Preparación de documentos	Juan Manager, Rafael Analista	Virtual	9 enero	Rafael Analista	Enseñar los documentos que deben generarse y la estructura de estos.
Gestión de cambios y control de incidentes	Juan Manager, Todo el equipo de desarrollo	Virtual	13 enero	Rafael Analista	Capacitar como deben manejarse los cambios al software, así como los problemas encontrados.
Uso de herramientas	Juan Manager, Todo el equipo de desarrollo	Virtual	14 enero	Cruz Arquitecto	Capacitar como utilizar las herramientas
Auditoría Interna	Juan Manager, Todo el equipo de desarrollo	Virtual	15 enero	Juan Manager	Enseñar el proceso para las auditorías internas y como preparase para las auditorías externas.
Cierre del proyecto	Juan Manager, Todo el equipo de desarrollo	Virtual	16 enero	Juan Manager	Enseñar como hacer las retrospectivas del equipo y que documentos se ocupan para el cierre.

Tabla 43.

Ejemplo cronograma enfoque de prácticas de calidad.

Criterio	Cumple	Autoriza	Fecha	Comentarios
Cumple con un formato de código específico como *PascalCase*.	Sí	Cruz Arquitecto	20 de abril	Pasó las pruebas de formato.

Se tienen todos los documentos requeridos	Sí	Rafael Analista	20 de enero
Se autorizan todas las pruebas unitarias	Sí	Rafael Analista	30 de abril
Se tienen los manuales de usuario	Sí	Cruz Arquitecto	15 mayo
Se registraron todos los documentos de despliegue de la aplicación	Sí	Cruz Arquitecto	15 mayo
Se registraron todas las no conformidades	Sí	Rafael Analista	30 mayo

5.4. Diseño e implementación de procesos

c. Documentar e implementar los procesos de gestión de proyectos

   a. Requerimientos iniciales. Ver Tabla 44.

   b. Aprobación del cliente. Ver Tabla 45.

   c. Seguimiento y control de proyecto. Ver Tabla 46, Tabla 47 y Tabla 48.

   d. Evaluación de desempeño del equipo. Ver Tabla 49 y Tabla 50.

   e. Lecciones aprendidas. Ver Tabla 51.

   f. Cierre del proyecto. Ver Tabla 52.

d. Documentar e implementar los procesos de implementación de software:

   a. Análisis de requisitos. Ver Tabla 53, Tabla 54, Tabla 55, Tabla 56, Tabla 57, Tabla 58, Tabla 59, Tabla 60, Tabla 61, Figura 8 y Figura 9.

   b. Diseño técnico

- Diagrama de clases. Ver Tabla 62.
- Diseño de la base de datos**.** Ver Tabla 63.

   c. Desarrollo del software

- Gestor de versiones. Ver Tabla 64 y Tabla 65.

   d. Pruebas unitarias e integradas. Ver Tabla 66 y Tabla 67.

e. Documentación técnica

- Manual de usuario. Ver Tabla 68.
- Manual técnico. Ver Tabla 69.

f. Validación con el cliente. Ver Tabla 70.

Tabla 44.

Ejemplo requerimientos iniciales.

Fecha	Medio	Solicitante	Requerimiento
15 de enero	Correo electrónico	Elena Pruebas	El sistema debe obtener el precio del tipo de cambio que tienen registrado en otro sistema.
15 de enero	Correo electrónico	Elena Pruebas	El sistema debe consultar el precio de una fuente confiable.
15 de enero	Correo electrónico	Elena Pruebas	El sistema debe actualizar los registros dos veces al día.
15 de enero	Llamada	Elena Pruebas	La información debe estar respaldada.
16 de enero	Llamada	Elena Pruebas	El cliente ocupa un reporte de la información que solicita.
16 de enero	Correo electrónico	Elena Pruebas	El cliente ocupa saber cuándo el sistema falla y no recibe el reporte para no estar esperando.
20 enero	Correo electrónico	Elena Pruebas	Se ocupa respaldar la información por un tiempo.
21 enero	Llamada	Elena Pruebas	El cliente requiere que su información esté segura.
21 enero	Llamada	Elena Pruebas	Se necesita tener pruebas si el proceso falla para que cliente pueda reportarlo a sus auditores.

Tabla 45.

Ejemplo acta aprobación de entregables.

Acta de aprobación del entregable
Nombre del proyecto: Integración de la cotización de monedas

Entregable aprobado: modulo con integración externa de API con datos de monedas

Fecha de entrega: 28 de abril

Resultado de la revisión: **Aprobado**

Observaciones: Se aprueba correctamente la conexión y obtención de datos

Cliente: Elena Pruebas

Firma: *Elena P*

Fecha: 30 de abril

Tabla 46.

Ejemplo seguimiento de tarea 1.

Fecha de seguimiento	15 de febrero
Actividad revisada	Obtención de los requerimientos para el sistema
Estado actual	Completada
Avance de la tarea	100%
Responsable	Rafael Analista
Acciones correctivas	No existieron retrasos
Observaciones	Los requisitos fueron documentados acordes al formato

Tabla 47.

Ejemplo seguimiento de tarea 2.

Fecha de seguimiento	30 de febrero
Actividad revisada	Definición de la arquitectura
Estado actual	Completada
Avance de la tarea	100%
Responsable	Cruz Arquitecto
Acciones correctivas	No existieron retrasos
Observaciones	Se crearon los documentos conforme a los requerimientos descritos

Tabla 48.

Ejemplo seguimiento de tarea 3.

Fecha de seguimiento	01 abril
Actividad revisada	Obtención de los requerimientos para el sistema
Estado actual	En progreso
Avance de la tarea	80%
Responsable	Carlos Programador
Acciones correctivas	Se está trabajando activamente en la tarea
Observaciones	Se continúa trabajando en tiempo y forma esperada

Tabla 49.

Ejemplo evaluación de desempeño 1.

Nombre del proyecto	Integración de la cotización de monedas
Periodo de evaluación	Abril
Evaluador	Juan Manager
Criterios para evaluar	• Entrega a tiempo de arquitectura • Diagramas con el formato correcto
Evaluado	Cruz Arquitecto
Observaciones	Buena presentación y orden en la entrega de documentos
Recomendaciones	

Tabla 50.

Ejemplo evaluación de desempeño 2.

Nombre del proyecto	Integración de la cotización de monedas
Periodo de evaluación	Abril
Evaluador	Cruz Arquitecto
Criterios para evaluar	• Cumplimiento de los estándares de codificación • Seguimiento de la arquitectura
Evaluado	Carlos Programador
Observaciones	Código cumple con todos los requisitos de seguridad y sigue los diagramas señalados
Recomendaciones	

Tabla 51.

Ejemplo lecciones aprendidas.

Nombre del proyecto	Integración de la cotización de monedas
Inicio del proyecto	1 enero
Fin del proyecto	1 mayo
Aspecto del proyecto	Implementación del ISO
Que funcionó bien	La integración de reuniones para explicar claramente de que trata el ISO. Después de una implementación adecuada se pudieron observar un mejor control de tiempo y recursos. Ayudó a clarificar roles dentro del equipo.
Que no funcionó	La implementación ocasionó cierta resistencia al cambio y procesos. Existía una falta de cultura documental ya que no todos los miembros estaban acostumbrados a crear documentos.
Responsable	Juan Manager
Recomendaciones	Comunicar la importancia de la implementación. Capacitar correctamente a los miembros del equipo. Apoyar en la curva de aprendizaje y designar a un miembro del equipo que apoye a la creación de documentos.
Comentarios	

Tabla 52.

Ejemplo cierre del proyecto.

Nombre del proyecto	Integración de la cotización de monedas
Fecha de cierre	1 mayo
Entregables completados	Lista de entregables: • Manual de instalación • Manual de usuario • Código fuente en repositorio del cliente
Firma del cliente	Elena P
Observaciones	Se entregan los documentos y software previamente autorizados por cliente

Tabla 53.

Ejemplo análisis de requisitos 1.

Id	1
Tipo de requerimiento	Requerimiento funcional
Descripción	El sistema debe obtener el precio de las siguientes monedas: USD, EUR, JPY, GBP, AUD, CAD, CHF, CNY, CNY, NZD, MXN, BRL, INR, ZAR, KRW, RUB, TRY, SGD, HKD, NOK y guardarlos en la base de datos
Responsable	Cruz Arquitecto
Prioridad	Media
Criterio de aceptación	Se tienen que visualizar todas las monedas mencionadas
Dependencia	No Aplica
Diseños asociados	Diagrama Arquitectura General, Diagrama General Entidad relación
Código fuente relacionado	No Aplica

Tabla 54.

Ejemplo análisis de requisitos 2.

Id	2
Tipo de requerimiento	Requerimiento funcional
Descripción	El sistema debe consultar el precio de la API CoinGecko ®
Responsable	Carlos Programador
Prioridad	Media
Criterio de aceptación	Se tiene que hacer una conexión exitosa a la API
Dependencia	No Aplica
Diseños asociados	Diagrama Arquitectura General
Código fuente relacionado	No Aplica

Tabla 55.

Ejemplo análisis de requisitos 3.

Id	3
Tipo de requerimiento	Requerimiento funcional
Descripción	El sistema debe actualizar los registros diario a las 8:00 am CST y 6:00 pm CST
Responsable	Carlos Programador
Prioridad	Alta
Criterio de aceptación	Se tiene que guardar un registro cuando empiece a correr el proceso
Dependencia	No Aplica
Diseños asociados	Diagrama Arquitectura General
Código fuente relacionado	No Aplica

Tabla 56.

Ejemplo análisis de requisitos 4.

Id	4
Tipo de requerimiento	Requerimiento funcional
Descripción	El sistema debe guardar los datos obtenidos en una carpeta de respaldo en la nube.
Responsable	Cruz Arquitecto
Prioridad	Alta
Criterio de aceptación	Se tiene que guardar correctamente el documento en una carpeta con la fecha del día de la ejecución
Dependencia	No Aplica
Diseños asociados	Diagrama Arquitectura General
Código fuente relacionado	No Aplica

Tabla 57.

Ejemplo análisis de requisitos 5.

Id	5
Tipo de requerimiento	Requerimiento funcional
Descripción	El sistema debe generar un reporte con las monedas obtenidas.
Responsable	Carlos Programador
Prioridad	Media
Criterio de aceptación	Se generará un reporte con la información de las monedas y el tipo de cambio
Dependencia	No Aplica
Diseños asociados	Diagrama Arquitectura General
Código fuente relacionado	No Aplica

Tabla 58.

Ejemplo análisis de requisitos 6.

Id	6
Tipo de requerimiento	Requerimiento funcional
Descripción	El sistema debe alertar a los usuarios cuando exista una falla mediante un correo electrónico proporcionado por el cliente, este correo debe estar configurado en las variables de entorno del proyecto
Responsable	Carlos Programador
Prioridad	Media

Criterio de aceptación	Tiene que llegar un correo al correo que esté registrado dentro de las variables de entorno cuando falle el programa
Dependencia	No Aplica
Diseños asociados	Diagrama Arquitectura General
Código fuente relacionado	No Aplica

Tabla 59.

Ejemplo análisis de requisitos 7.

Id	7
Tipo de requerimiento	Requerimiento funcional
Descripción	El sistema debe guardar la información por 5 años dentro del *blob storage* en la cuenta de Azure
Responsable	Cruz Arquitecto
Prioridad	Alta
Criterio de aceptación	El *blob storage* debe configurarse para que la información dure 5 años
Dependencia	No Aplica
Diseños asociados	Diagrama Arquitectura General
Código fuente relacionado	No Aplica

Tabla 60.

Ejemplo análisis de requisitos 8.

Id	8
Tipo de requerimiento	Requerimiento funcional
Descripción	La información descargada debe estar encriptada usando el método de encriptación AES.
Responsable	Cruz Arquitecto
Prioridad	Media
Criterio de aceptación	La información debe guardarse encriptada en el *blob storage* y la llave para cifrar debe ser guarda dentro de un baúl de llaves dentro del *resource group* de *Azure*.
Dependencia	No Aplica
Diseños asociados	Diagrama Arquitectura General
Código fuente relacionado	No Aplica

Tabla 61.

Ejemplo análisis de requisitos 9.

Id	9

Tipo de requerimiento	Requerimiento funcional
Descripción	El sistema debe contar con los logs de errores para temas de auditoría.
Responsable	Cruz Arquitecto
Prioridad	Alta
Criterio de aceptación	Se tienen que visualizar los registros usando el software Datadog ®
Dependencia	No Aplica
Diseños asociados	Diagrama Arquitectura General
Código fuente relacionado	No Aplica

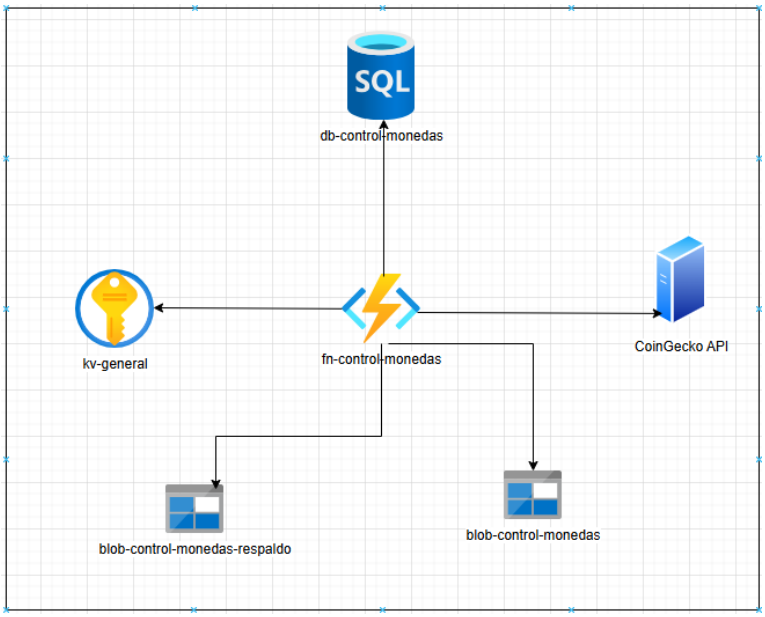
Figura 8.

Ejemplo diagrama de arquitectura.

Diagrama de arquitectura

Propósito: Arquitectura general

Descripción del diagrama:



Observaciones: Los recursos descritos de desplegarán en el *resource group* del cliente.

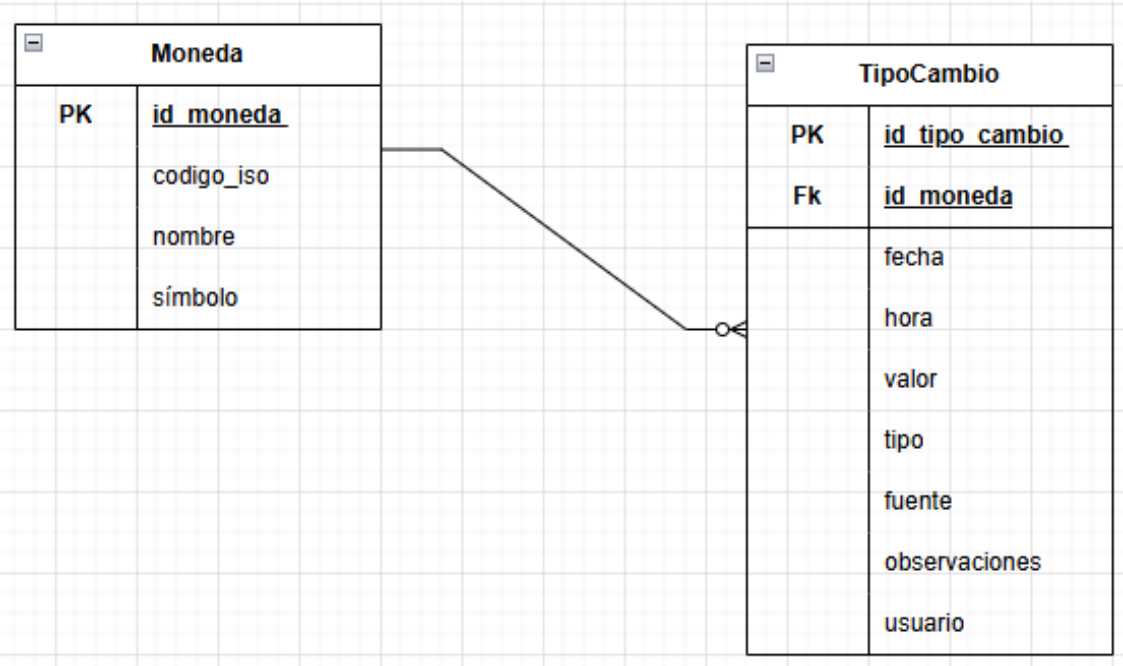
Figura 9.

Diseño base de datos.

Diseño base de datos

Propósito: Diagrama ER General

Modelado de datos:



Requisitos de integridad: Una moneda puede tener varios tipos de cambio por lo cual es necesario crear una llave foránea en el tipo de cambio

Consideraciones de rendimiento: Se creará un índice en las llaves primarias

Seguridad: Se creará una cuenta de lectura y escritura para poder guardar los archivos desde la aplicación, pero los usuarios finales tendrán cuentas de solo lectura.

Observaciones: Sin comentarios adicionales

Tabla 62.

Ejemplo identificación de ramas en el gestor de versiones.

Rama	Propósito
Main/master	Rama estable para producción
Develop	Rama para la integración y desarrollo de nuevas funcionalidades
Test	Rama de prueba para usuarios finales
*Feature/**	Ramas de nuevas características
*Hotfix/**	Ramas para el manejo de cambio de código, puede ser el cambio de una fórmula, confundir con ramas de errores.
Bugfix/*	Ramas para arreglo de errores en el código

Tabla 63.

Ejemplo formalización de control de versiones.

Control de versiones
Herramienta utilizada: Bitbucket
Ruta del repositorio: Se guardará en la ruta del repositorio entregada por el cliente
Configuración adicional: Durante el desarrollo se otorgarán permisos a los
Observaciones: Sin comentarios adicionales

Tabla 64.

Ejemplo pruebas unitarias 1.

Id	1
Tipo de prueba	Prueba Unitaria
Nombre de la prueba	Revisión de tipos de cambio
Descripción	Validar que el sistema genere un reporte del tipo de cambio y se mande al correo indicado.
Módulos implicados	Modulo tipo de cambio.
Entrada	Correo electrónico para mandar el reporte.
Resultado esperado	Un reporte con el valor de todas las monedas indicadas.
Resultado del aplicativo	Reporte generado exitosamente.
Estado	Aprobado
Evidencia	Captura de pantalla de la prueba enviada por correo.

Fecha	28 de abril
Responsable	Elena Pruebas

Tabla 65.

Ejemplo pruebas unitarias 2.

Id	2
Tipo de prueba	Prueba unitaria
Nombre de la prueba	Guardar datos históricos
Descripción	Se tiene que guardar los reportes generados en una carpeta de respaldo.
Módulos implicados	Modulo tipo de cambio
Entrada	Reporte enviado por email.
Resultado esperado	El email enviado por email se tiene que visualizar en las carpetas de respaldo.
Resultado del aplicativo	Reporte guardado exitosamente.
Estado	Probado
Evidencia	Captura de pantalla de la prueba enviada por correo.
Fecha	29 abril
Responsable	Elena Pruebas

Tabla 66.

Ejemplo documentación técnica.

Sección	Contenido
Introducción	Revisión de documentos del tipo de cambio
Requisitos del sistema	Cuenta de correo activa con espacio para recibir documentos.
Guía de usuario	Llegará un correo de lunes a viernes a las 8AM y las 6PM con el título “Reporte Automatizado - Tipo de Cambio”, ese archivo se deberá descargar para ver la información.
Capturas de pantalla	Ilustraciones entregadas al cliente
Preguntas frecuentes	Si el correo con los tipos de cambio no llega en la hora indicada se tendrá que correr el sistema manual, para este proceso es necesario mandar un correo al administrador y él avisará cuando esté lista la ejecución.
Glosario	No aplica

Tabla 67.

Ejemplo manual técnico.

Sección	Contenido
Introducción	Definir los recursos requeridos para el sistema

Requisitos del sistema	Nombre del recurso: kv-general Ubicación: *Central US* Tier: *Standard* (*Tier* 1) Tipo de identidad: *Managed Identity* (habilitado para acceso desde la función hacia los otros recursos) Nombre del recurso: fn-control-monedas Runtime: .NET 8 Ubicación: *Central US* Plan: *Consumption Plan* (Y1) Trigger: *Timer Trigger* ("0 8,18 * * *" a las 8AM y 6PM) Nombre del recurso: sqldb-control-monedas Servidor lógico: sql-control-monedas-prod.database.windows.net Ubicación: *Central US* SKU: *Standard S1* (*DTU-based*) Base de datos: tipocambio Autenticación: *SQL Auth* Nombre del recurso: blob-control-monedas Ubicación: *Central US* Tipo de cuenta: *Storage* V2 (*general purpose* v2) Nivel de redundancia: LRS (*Locally Redundant Storage*) Nombre: rg-tipocambio Ubicación: *Central US*
Configuración	Los nombres y valores de los recursos deberán ser definidos en los archivos de entornos de variables de Terraform.
Gestión de usuarios	Las cuentas deben ser entregadas por cliente.
Seguridad	Es necesario asignar permisos a un usuario entregado por el cliente, los siguientes recursos deben tener un *lock* dentro de Azure para evitar ser borrados: • *Blob storage* • *Sql server* • *Key vault* Azure Function con Managed Identity activada. Asignación de roles: • *Key Vault Reader* sobre kv-general • *Storage Blob Data Contributor* sobre blob-control-monedas

Mantenimiento	Las actualizaciones deben ser gestionadas mediante CD/CI utilizando un repositorio de Jenkins descrito por el cliente.
Logs	Se debe utilizar una cuente de Dadatog entregada previamente configurada por el cliente.
Plan de recuperación a fallos	Restaurar una versión funcional dentro de Jenkins

Tabla 68.

Ejemplo acta de validación de cliente.

Acta de validación

Nombre del proyecto: Integración de la cotización de monedas

Vicent Owner, declaro que he revisado el sistema en su versión v1 y se han concluido las pruebas exitosamente en compañía del equipo de desarrollo.

Resultado: Producto aprobado

Observaciones: Sin comentarios adicionales

Cliente: Elena Pruebas

Firma: *Vicent Owner*

Fecha: 02 de mayo

5.5. Encuestas de la implementación

Se realizó una encuesta para evaluar la percepción del equipo de desarrollo con la implementación del nuevo estándar como se muestra en la Figura 10. Esta encuesta está dividida en tres categorías hacia el equipo de software y una encuesta a los usuarios finales.

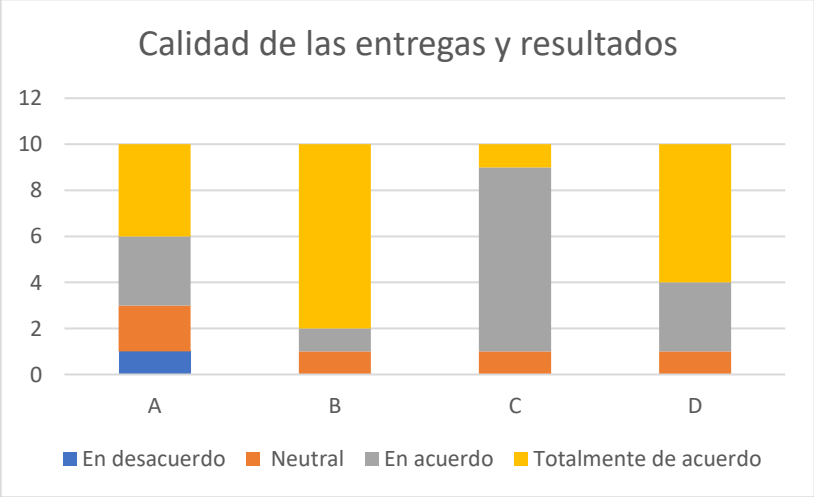
Categoría: Calidad de las entregas y resultados.

Evalúa la satisfacción con el software desarrollado, así como su calidad técnica. Considera si se han reducido errores y si el esfuerzo invertido ha sido justificado. Tiene el propósito de concluir si el trabajo realizado ha tenido un impacto positivo.

A. Estoy muy satisfecho con las entregas en las últimas semanas.
B. El número de errores detectados en etapas finales ha disminuido.
C. Los entregables tienen una mejor presentación y coherencia.
D. El tiempo invertido en la implementación ha valido la pena.

Figura 10.

Resultados encuesta calidad de entregas.



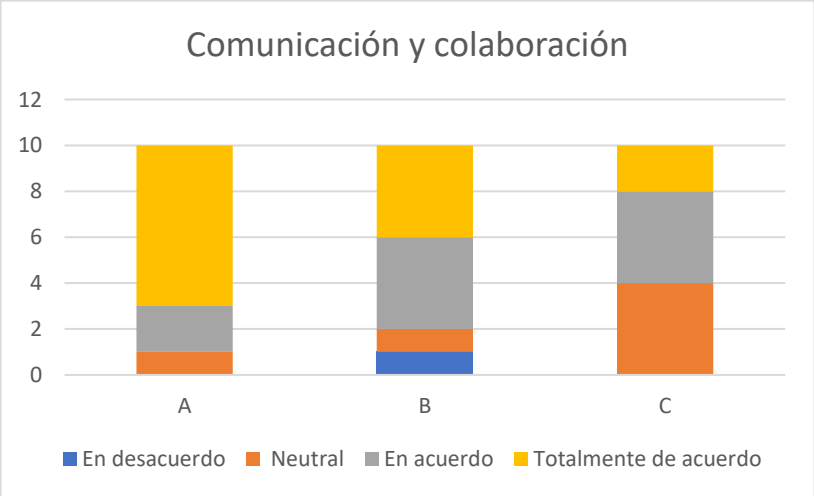
Categoría: Comunicación y colaboración.

Analiza la interacción entre los miembros del equipo de desarrollo, la efectividad de las reuniones y la alineación con las prioridades. Tiene el propósito de evaluar la dinámica colaborativa y la coordinación de esfuerzos representado en la Figura 11.

A. La comunicación con otros miembros del equipo ha mejorado.
B. En nuestras reuniones diarias los enfocamos en alinear en la importancia de las tareas.
C. Las reuniones de revisión y seguimiento son más efectivas.

Figura 11.

Resultados comunicación y colaboración.



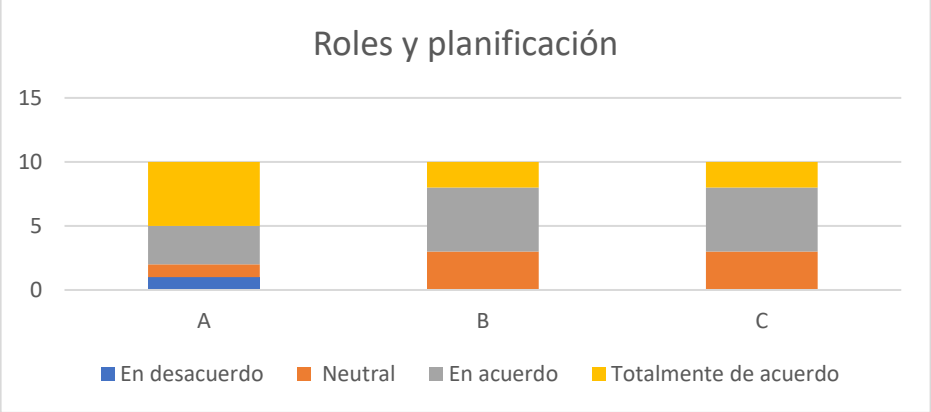
Categoría: Roles y planificación.

Se centra en la claridad de las responsabilidades, la definición de los procesos de trabajo y la eficiencia en la planificación de las actividades. El propósito es evaluar si el equipo tuvo un marco de trabajo claro para lograr la mejora en la organización y la productividad como se muestra en la Figura 12.

A. Comprendo claramente mis roles y responsabilidades en los proyectos
B. Hay una mejora en la planificación y seguimiento de actividades
C. Los procesos de desarrollo están ahora mejor definidos y documentados

Figura 12.

Resultados encuesta roles y planificación.



Encuesta para la evaluación por parte del cliente

Busca evaluar la satisfacción del cliente enfocado en el proceso de comunicación, documentación y control de errores durante el desarrollo del software.

A. El número errores ha disminuido. Respuesta: De acuerdo
B. La documentación es clara y precisa: Totalmente de acuerdo
C. Los plazos de entrega han sido cumplidos correctamente: Totalmente de acuerdo
D. Hubo comunicación entre el equipo: Totalmente de acuerdo
E. Volvería a contactar al equipo para futuros cambios: De acuerdo

6. Conclusión y Discusión

6.1. Discusión

Los resultados obtenidos durante el desarrollo e implementación del modelo basado en ISO 29110 permiten concluir que cumple la hipótesis de mejorar la calidad de los productos de software. Se demostró que es posible adaptar el modelo simplificado a las necesidades de un equipo de desarrollo pequeño teniendo en cuenta los principios de mejora continua que propone el estándar, la aplicación del estándar mejorar la calidad y la competitividad en las pequeñas empresas (O'Connor, 2014).

El modelo simplificado facilitó la comprensión, adaptación, entendimiento y aplicación de prácticas que son clave en la gestión de proyectos y desarrollo de software (Buchalcevova, 2020) reduciendo la complejidad de adaptar desde cero y sin conocimiento la implementación completa del estándar. Así mismo ayudó a tener más control sobre que actividades y roles le corresponden y como deberían ejecutarse para no olvidar pasos importantes y generar más confianza hacia el cliente (Laporte & Miranda, 2020). Las evaluaciones ayudaron a identificar las áreas de mejora, así como conocer cómo planificar la documentación y el seguimiento.

Así mismo, la implementación ayudó a validar que el estándar se adapta a la realidad de equipos pequeños usando metodologías ágiles (Muñoz et al., 2020) que tienen experiencia limitada en normas de calidad siendo efectivo para ayudar a la madurez de los procesos del equipo. Esto confirma que el modelo simplificado ISO 29110 contribuye a tener una mejor adaptación y adoptar buenas prácticas.

6.2. Conclusión

La consumación del modelo simplificado basado en ISO 29110 ayudó al equipo a estandarizar y documentar los procesos de software ya que incluyen los criterios necesarios para implementar el estándar, esto a su vez facilitó una mejor

organización dentro del equipo y un mayor control en cada etapa del desarrollo de software. Al aplicar correctamente las prácticas que se han recomendado en este documento logró una reducción significativa de retrasos en las entregas debido a que los requisitos estaban claramente definidos desde el principio, así mismo hubo una disminución de errores durante la validación.

La claridad de roles y responsabilidades fomentó una mejor comunicación dentro del equipo lo cual resultó en una ejecución más eficiente. Se pudo comprobar que es una norma adaptable en equipos pequeños ya que los procesos son ligeros y repetitivos, además de seguir formatos previamente establecidos. También se establecieron formatos y criterios que son base para el cumplimiento de la norma ISO 29110.

El cliente percibió una mejora en los tiempos y formalidad de los entregables, lo cual incrementó la confianza con el cliente. Se recomienda no saltar el paso de realizar una evaluación previa de los procesos existentes ya que ayuda a capacitar al equipo desde etapas más tempranas.

7. Referencias bibliográficas

Alshazly, A., ElNainay, M., El-Zoghabi, A., & Abougabal, M. (2021). *A cloud software life cycle process (CSLCP) model*. https://www.sciencedirect.com/science/article/pii/S2090447920302495

Anacona, A., Pino, F., Buitron, S., Rodriguez, M., & Piattini, M. (2020). Esquema de certificación por conformidad de requisitos del estándar ISO/IEC 29110 para la calidad de las empresas software. En *Iberian Conference on Information Systems and Technologies, CISTI* (Vols. 2020-June). https://doi.org/10.23919/CISTI49556.2020.9141029

Arvanitou, E., Ampatzoglou, A., Chatzigeorgiou, A., & Carver, J. (2021). *Software engineering practices for scientific software development: A systematic mapping study*. https://www.sciencedirect.com/science/article/abs/pii/S0164121220302387

Buchalcevova, A. (2020). Towards Higher Software Quality in Very Small Entities: ISO/IEC 29110 Software Basic Profile Mapping to Testing Standards. *International Journal of Information Technologies and Systems Approach*. https://doi.org/https://doi.org/10.4018/IJITSA.2021010105

Castillo, L., Sanchez, S., Villarroel, J., & Sánchez, M. (2020). *Evaluation of the implementation of a subset of ISO/IEC 29110 Software Implementation process in four teams of undergraduate students of Ecuador. An empirical software engineering experiment*. https://www.sciencedirect.com/science/article/abs/pii/S0920548918304203

Dartora, J., Herbert, J., & César, S. (2021). *Model for quality analysis of neonatal hearing screening software: theory applied*. https://www.sciencedirect.com/science/article/abs/pii/S1386505621000617

Estrella, V. (2023). *Industria de TI proyecta crecimiento de doble dígito en Querétaro*.

Feijóo, M. (2024). *Así es la nueva región nube de Google que abre en Querétaro: Los detalles*. https://www.adn40.mx/mexico/2024-12-05/la-nueva-region-nube-google-que-abre-en-queretaro-los-detalles?utm_source=chatgpt.com

Garzás, J., Pino, F., Piattini, M., & Fernández, C. (2013). *A maturity model for the Spanish software industry based on ISO standards*. https://www.sciencedirect.com/science/article/abs/pii/S092054891300024X

IEC. (2026). *What IEC does*. https://www.iec.ch/what-we-do

IEEE España. (2026). *IEEE Sección España*. https://ieeespain.org/quienes-somos/

I.N.E.G.I. (2019). Estadísticas a Propósito de las Ocupaciones Relacionadas con las Tecnologías de la Información y de la Comunicación Datos Nacionales. En *Instituto Nacional de Estadística y Geografía (INEGI)* (Vol. 29, p. 12). https://www.inegi.org.mx/contenidos/saladeprensa/aproposito/2019/OcupaTIC2019_Nal.pdf

Irshad, M., Britto, R., & Petersen, K. (2021). *Adapting Behavior Driven Development (BDD) for large-scale software systems*. https://www.sciencedirect.com/science/article/pii/S0164121221000418

Joaquin, D. (2024). *Amazon construirá centro de datos en Querétaro; todo lo que sabemos*. https://lasillarota.com/estados/2024/2/27/amazon-construira-centro-de-datos-en-queretaro-todo-lo-que-sabemos-471530.html?utm_source=chatgpt.com

Kadri, S., & Aouag, S. (2021). *MS-QuAAF: A generic evaluation framework for monitoring software architecture quality*. https://www.sciencedirect.com/science/article/abs/pii/S095058492100166X

Laporte, C., & Miranda, J. (2020). *Delivering Software- and Systems-Engineering Standards for Small Teams*. https://doi.org/https://doi.org/10.1109/MC.2020.2993331

Lei, H., Ganjeizadeh, F., Jayachandran, P. K., & Ozcan, P. (2017). A statistical analysis of the effects of Scrum and Kanban on software development projects. *Robotics and Computer-Integrated Manufacturing*, *43*, 59–67. https://doi.org/10.1016/j.rcim.2015.12.001

Liu, M., Hansen, S., & Tu, Q. (2021). *Sustaining collaborative software development through strategic consortium*. https://www.sciencedirect.com/science/article/abs/pii/S0963868721000184

Lizcano, D., Leonardo, A., López, G., & Smith, P. (2021). *Modelling web component quality using Delphi study*. https://www.sciencedirect.com/science/article/abs/pii/S0920548921000428

Lohier, P., & Rodriguez, P. (2018). Yes, very small organizations in the IT sector can benefit from being recognised internationally. *Proceedings - 2018 International Conference on the Quality of Information and Communications Technology, QUATIC 2018*, 8–14. https://doi.org/10.1109/QUATIC.2018.00012

Manzano, M., Ayala, C., Abherve, A., Franch, X., & Mendes, E. (2021). *A Method to Estimate Software Strategic Indicators in Software Development: An Industrial Application*. https://www.sciencedirect.com/science/article/pii/S0950584920301890

Mas, A., Mesquida, A., & Pacheco, M. (2020). *Supporting the deployment of ISO-based project management processes with agile metrics*. https://www.sciencedirect.com/science/article/abs/pii/S0920548919303575

Meding, W., Staron, M., & Söder, O. (2021). *MeTeaM, A method for characterizing mature software metrics teams*. https://www.sciencedirect.com/science/article/pii/S0164121221001035

Microsoft Latinoamérica. (2024). *Microsoft anuncia el inicio de operaciones de la primera región de centros de datos de nube a hiper escala en México*.

Ming-Chang, L. (2014). Software Quality Factors and Software Quality Metrics to Enhance Software Quality Assurance. *British Journal of Applied Science & Technology*, *4*(21), 3069–3095. https://doi.org/10.9734/bjast/2014/10548

Mishra, A., & Otaiwi, Z. (2020). *DevOps and software quality: A systematic mapping*. https://www.sciencedirect.com/science/article/pii/S1574013720304081

Muñoz, M., Laporte, C., & Mejia, J. (2020). Implementing ISO/IEC 29110 to reinforce four very small entities of Mexico under an agile approach. *IET Software*. https://doi.org/https://doi.org/10.1049/IET-SEN.2019.0040

Muñoz, M., Mejia, J., & Lagunas, A. (2018). Implementation of the ISO/IEC 29110 standard in agile environments: A systematic literature review. *Iberian Conference on Information Systems and Technologies, CISTI*, *2018-June*, 1–6. https://doi.org/10.23919/CISTI.2018.8399332

Nguyen, P., Rocco, J., Ruscio, D., & Penta, M. (2020). *CrossRec: Supporting software developers by recommending third-party libraries*. https://www.sciencedirect.com/science/article/abs/pii/S0164121219302341

O'Connor, R. (2014). *Deploying a software process lifecycle standard in very small companies*. IGI Global. https://doi.org/https://doi.org/10.4018/978-1-4666-5888-2.CH073

Planas, E., & Cabot, J. (2020). *How are UML class diagrams built in practice? A usability study of two UML tools: Magicdraw and Papyrus*. https://www.sciencedirect.com/science/article/abs/pii/S0920548918303659

Rosca, D., & Domingues, L. (2021). *A Systematic Comparison of Roundtrip Software Engineering Approaches applied to UML Class Diagram*. https://www.sciencedirect.com/science/article/pii/S1877050921002830

Sanchez, M., Amescua, A., O'Connor, R., & Larrucea, X. (2017). *A standard-based framework to integrate software work in small settings*. https://www.sciencedirect.com/science/article/abs/pii/S0920548916301891

Santos, P., Perini, M., Falbo, R., & Almeida, J. (2021). *From a Scrum Reference Ontology to the Integration of Applications for Data-Driven Software Development*. https://www.sciencedirect.com/science/article/abs/pii/S0950584921000537

Schwaber, K., & Sutherland, J. (2017). *The Scrum GuideTM The Definitive Guide to Scrum: The Rules of the Game*.

Setiawan, R., Erlisal, Z., & Effendi, A. (2018). *Design Metric Indicator to Improve Quality Software Development (Study Case: Student Desk Portal)*. https://www.sciencedirect.com/science/article/pii/S1877050918315217

Shafiee, S., Wautelet, Y., Callesen, S., Lis, L., Harlou, U., & Hvam, L. (2021). *Evaluating the benefits of a computer-aided software engineering tool to develop and document product configuration systems*. https://www.sciencedirect.com/science/article/abs/pii/S0166361521000397

8. Anexo Entregables

Tabla 69.

Anexo formato de entregables.

Nombre del entregable		
Versión:	**Fecha:**	**Autor:**
Objetivo:		
Descripción:		
Comentarios:		
Aprobaciones: Responsable técnico: Revisor: Fecha de aprobación:		