PAPUSCORP

Papu Software Corporation

Estándares de desarrollo y mantenimiento.

Integrantes:

Paola Alejandra Novelo Chi

Diego Alfonso Burgos Tzuc

Juan Osvaldo Salazar Puerto

Francisco Joaquin Ruiz Ayala

Jose Armando Aviles Lopez

control de la configuración.

Nombre	Versión	Descripción
Plan de aseguramiento de la calidad	1.0	Versión inicial del plan de aseguramiento

Link del GitHub:
https://github.com/ThuPapuha/PlanSQA_PapusCorp

Plan de aseguramiento de la calidad del software

1. Propósito

El presente documento describe, en base a el anexo plan de desarrollo de software un plan de aseguramiento de la calidad de software para todos los ítems de software que se planean entregar. En esta entrega, al mínimo en nivel 2 de Capability, mejorando el nivel en la versión final.
El objetivo de este documento es desarrollar un plan de aseguramiento de calidad (SQA) para el proyecto “Calculadora de matrices”. Este plan se centrará específicamente en la fase de diseño del software. Para lograr este objetivo, se seguirá el estándar IEEE 730, que establece las directrices para la planificación del SQA en todas las fases del ciclo de vida del software.
Se identificarán los objetivos específicos del proyecto y se definirán los criterios de calidad para la fase de diseño del software. A continuación, se identificarán los estándares y prácticas recomendadas para el diseño de software. Se definirá un proceso de revisión para asegurarse de que los documentos de diseño cumplan con los requisitos de calidad identificados.
Además, se establecerán métricas de calidad para evaluar el desempeño del proceso de diseño y se asignan responsabilidades claras para el aseguramiento de calidad. Estas responsabilidades incluirán la designación de un arquitecto de software o desarrollador experimentado como responsable de la revisión de diseño.
Este documento es consistente con el estándar de desarrollo y mantenimiento de software anexado, y es consistente con los estándares que se encuentran en ese documento.

2. Documentos de referencia

- 730-2002 Software Quality Assurance Plans.
- 730-2014 Software Quality Assurance Processes.
- 7301-1995 Guide for Software Quality Assurance Planning.
- ISO_IEC_15939-2007---roles.

3. Gestión

Se espera un enfoque sistemático y disciplinado para la identificación, control y seguimiento de los elementos de configuración del software. El plan se ha desarrollado teniendo en cuenta las características y necesidades específicas del proyecto y se ha diseñado para garantizar la integridad y la trazabilidad de los elementos de configuración a lo largo de todo el ciclo de vida del software. Además, se han establecido procedimientos y herramientas para la gestión de versiones, la gestión de cambios, la auditoría y la retroalimentación.

3.1 Organización

La organización para el proyecto "calculadora de matrices" tendrá suficiente nivel de libertad y objetividad para evaluar y monitorear la calidad del software, así como para verificar las resoluciones de problemas. Además, se deben definir los procedimientos necesarios para garantizar la calidad del diseño, a su vez, se deberá documentar la cantidad de libertad organizativa y objetividad en la evaluación de la calidad del diseño.
En cuanto a la organización encargada de preparar y mantener el plan SQA, se ha identificado al equipo de diseño y al gerente de proyecto, quienes serán responsables de asegurar que el Plan SQA se mantenga actualizado y de que se realicen las revisiones necesarias para garantizar la calidad del producto de software.

3.2 Tareas

- Definir y documentar los requisitos de diseño, asegurándose de que sean claros, precisos y coherentes con los requisitos de software definidos en la fase de análisis.
- Seleccionar las técnicas de diseño más apropiadas para el proyecto, considerando los requisitos de software, los recursos disponibles y los plazos establecidos.
- Realizar revisiones de diseño para detectar y corregir posibles errores y mejorar la calidad del diseño.
- Realizar pruebas de diseño para comprobar su funcionalidad y fiabilidad, así como para evaluar su capacidad de respuesta a diferentes escenarios.
- Documentar el diseño del software, proporcionando detalles suficientes para permitir su revisión y comprensión.
- Verificar que el diseño cumple con los estándares de codificación, arquitectura y diseño establecidos para el proyecto.
- Asegurar la coherencia y la integridad del diseño, mediante la revisión y la actualización constante de los artefactos de diseño.
- Evaluar el diseño a través de revisiones y pruebas de diseño para identificar posibles problemas y asegurar que el diseño sea adecuado para el uso previsto.
- Realizar análisis de riesgos del diseño, identificando los riesgos asociados y definiendo planes de contingencia para minimizar los riesgos potenciales.
- Asegurar la trazabilidad de los requisitos de diseño a los requisitos de software definidos en la fase de análisis, para garantizar que el diseño cumpla con los requisitos establecidos.

3.3 Roles y responsabilidades

Líder de diseño: responsable de liderar y supervisar el equipo de diseño, asegurando que se cumplan los plazos y los estándares de calidad del diseño.

Diseñadores de software: responsables de crear y revisar los documentos de diseño, asegurando que cumplan con las especificaciones del cliente y los estándares de calidad establecidos.

Revisor de calidad del diseño: responsable de verificar que los diseños cumplan con los criterios de calidad establecidos, identificar y rastrear los problemas de diseño y trabajar con el equipo de diseño para resolverlos de manera oportuna y efectiva.

Comité de SQA: responsable de revisar y aprobar los planes de diseño y de realizar revisiones técnicas de los documentos de diseño para asegurar que cumplan con los estándares de calidad y las especificaciones del cliente.

Gerente de proyecto: responsable de coordinar y supervisar el proyecto de software en su conjunto, asegurando que se cumplan los plazos, presupuestos y objetivos de calidad establecidos.

Equipo de pruebas: responsable de realizar pruebas de diseño para garantizar la calidad del software y asegurar que cumpla con las especificaciones del cliente.

3.4 Estimación de recursos para el aseguramiento de la calidad

4 Documentación

La documentación es una herramienta la cual beneficia de forma directa si usada correctamente, y todo plan de software necesita un mínimo de documentos para su correcto desarrollo, esto no solo para tener una relación de los documentos desarrollados durante el proceso de desarrollo, sino también para poder tener una retrospectiva de los cambios, decisiones, y crítica que se pueda desarrollar en base a el seguimiento o falta del mismo, creando una forma concreta para poder verificar los fallos o triunfos del equipo de desarrollo y el software desarrollado.

La documentación de este plan consiste en los siguientes documentos, los cuales serán descritos tanto como en uso esperado como propósito en el desarrollo y que rol forman en este, todos estos, se pueden encontrar en los documentos referenciados, en el apartado 4.2:

A. Software Requirements Specifications (SRS)

La especificación de los requerimientos debe tener los requerimientos tratados con el desarrollador y el cliente que lo requiere, este normalmente está desarrollado y deriva de varios documentos y contiene una lista de referencia que identifica a los modelos en los que se estén trabajando tanto prototipos como productos, este debe identificar a los documentos que van a ser usados y sean de mayor importancia para el desarrollo y debe ayudar a identificar cuando

estos documentos contengan requerimientos contradictorios o no plausibles para el desarrollo en el que se encuentra.

B. Software Design Description (SDD)

Esta es una descripcion tecnica de como el software cumplira los requerimientos a los que se comprometen en el SRS, su función más fundamental es la descomposición de todo el sistema en componentes completos, documentan también las razones por las cuales, estas decisiones son las más sensatas para facilitar el entendimiento de la estructura del sistema.

Este documento debe describir las funciones mayores dentro del sistema, como lo pueden ser las bases de datos, interfaces tanto externas como internas y la estructura general del diseño, implica describir el ambiente de operación, procesamiento, modelamiento, simulación, etc.

C. Software Verification and Validation Plan (SVVP)

El esfuerzo de V&V debe ser como el especificado en la siguiente tabla, la cual esta hecha con el nivel de integridad definida en este miso documento, por lo tal, se necesitan realizar ciertas tareas como las siguientes:

Actividad: Software Concept V&V (Software, 12207-Proceso de Análisis de Requisitos de Software)		
Tareas V&V	**Inputs Requeridos**	**Outputs Requeridos**
(1) Evaluación de la documentación conceptual a) Validar que la documentación conceptual satisface las necesidades del usuario y es coherente con las necesidades de adquisición. b) Validar las restricciones de los sistemas interconectados y las restricciones o limitaciones del enfoque propuesto. c) Analizar los requisitos del sistema y validar que lo siguiente satisface las necesidades del usuario: 1) Funciones del sistema. 2) Rendimiento del sistema de extremo a extremo. 3) Viabilidad y comprobabilidad de los requisitos funcionales. 4) Diseño de la arquitectura del sistema. 5) Requisitos y entornos de funcionamiento y mantenimiento. 6) Requisitos de migración desde un sistema existente, si procede.	Documentación conceptual Diseño arquitectónico del sistema Planes y calendarios de desarrollo de proveedores Necesidades de los usuarios Necesidades de adquisición	Eeport(es)—de taks Documentacion de concepto Evaluacion Report(es) de anomalias

(2) Análisis de la asignación de requisitos Verificar la corrección, exactitud e integridad de la asignación de requisitos del sistema al software en función de las necesidades del usuario. a) Corrección Verificar que los requisitos de rendimiento (por ejemplo, temporización, tiempo de respuesta y rendimiento) asignados al hardware, software e interfaces de usuario satisfacen las necesidades del usuario. b) Precisión Verificar que las interfaces internas y externas especifican los formatos de datos, los protocolos de interfaz, la frecuencia de intercambio de datos en cada interfaz y otros requisitos de rendimiento para demostrar la satisfacción de los requisitos del usuario. c) Exhaustividad 1) Comprobar que los requisitos específicos de la aplicación, como la diversidad funcional, la detección de fallos, el aislamiento de fallos y el diagnóstico y la recuperación de errores, satisfacen las necesidades del usuario. 2) Comprobar que los requisitos de mantenimiento del usuario para el sistema están completamente especificados. 3) Comprobar que la migración desde el sistema existente y la sustitución del sistema satisfacen las necesidades del usuario.	Necesidades de los usuarios Documentación conceptual Requisitos del sistema Arquitectura del sistema	Eeport(es)—de taks Documentacion de concepto Evaluacion Report(es) de anomalias
(3) Análisis de trazabilidad a) Identificar todos los requisitos del sistema. b) Verificar que estos requisitos del sistema son trazables a las necesidades de adquisición. c) Iniciar el análisis de trazabilidad de los requisitos con los requisitos del sistema.	Documentación conceptual	Eeport(es)—de taks Documentacion de concepto Evaluacion Report(es) de anomalias)
(4) Análisis de criticidad a) Determinar si se han establecido niveles de integridad para requisitos, funciones detalladas, módulos de software, elementos de hardware, subsistemas u otras particiones. b) Verificar que los niveles de integridad asignados son correctos. Si no se han asignado niveles de integridad, asigne niveles de integridad a los requisitos del sistema. c) Documentar el nivel de integridad asignado a los componentes individuales.	Documentación conceptual (requisitos del sistema) Asignaciones del nivel de integridad del desarrollador	Eeport(es)—de taks Documentacion de concepto Evaluacion Report(es) de anomalias

¿

Actividad: Software Concept V&V (Software, 12207-Proceso de Análisis de Requisitos de Software)		
V&V tasks	**Required inputs**	**Required outputs**
(por ejemplo, requisitos, funciones detalladas, módulos de software, elementos de hardware, subsistemas u otras particiones). A efectos de planificación de V&V, se asignará al sistema el mismo nivel de integridad que el nivel más alto asignado a cualquier elemento individual. d) Se comprobará si algún componente puede influir en los componentes individuales a los que se haya asignado un nivel de integridad del software superior y, si se dan estas condiciones, se asignará a dicho componente el mismo nivel de integridad superior.		
(5) Análisis de peligros Analizar los peligros potenciales hacia y desde el sistema conceptual. El análisis deberá realizar lo siguiente a) Identificar los peligros potenciales del sistema. b) Evaluar las consecuencias de cada peligro. c) Evaluar la probabilidad de cada peligro. d) Identificar estrategias de mitigación para cada peligro.	Documentacion conceptual	Eeport(es)—de taks Analisis de daños Report(es) de anomalias
(6) Análisis de seguridad a) Revisar la definición del propietario del sistema de un nivel aceptable de riesgo para la seguridad. b) Analizar el concepto del sistema desde el punto de vista de la seguridad y asegurarse de que se han identificado los riesgos potenciales para la seguridad con respecto a la confidencialidad (revelación de información/datos sensibles), la integridad (modificación de información/datos), la disponibilidad (retención de información o servicios) y la responsabilidad (atribución de acciones a un individuo/proceso). Incluya una evaluación de la sensibilidad de la información o los datos que se van a procesar. c) Analizar los riesgos de seguridad introducidos por el propio sistema, así como los asociados al entorno con el que interactúa el sistema.	Documentación Conceptual TRA Preliminario	Eeport(es)—de taks Analisis de seguridad Report(es) de anomalias

(7) Análisis de riesgos a) Identificar los riesgos técnicos y de gestión. b) Formular recomendaciones para eliminar, reducir o mitigar los riesgos	Documentación conceptual Planes y calendarios de desarrollo de proveedores Informe de análisis de riesgos Security analysis report V&V task results	Eeport(es)—de taks Analisis de daños Report(es) de anomalias

Quedando la Matriz de la siguiente manera.

V&V Activities	Activity: Software Concept V&V (see 9.1)				Activity: Software Requirements V&V (see 9.2)				Activity: Software Design V&V (see 9.3)				Activity: Software Construction V&V (see 9.4)				Activity: Software Integration V&V (see 9.5)				Activity: Software Qualification V&V (see 9.6)				Activity: Software Acceptance V&V (see 9.7)				Activity: Software Installation and Checkout V&V (see 9.9)				Activity: Software Operation V&V (see 9.11)				Activity: Software Maintenance V&V (see 9.12)				Activity: Software Disposal V&V (see 9.13)			
Integrity Levels	Levels				Levels				Levels				Levels				Levels				Levels				Levels				Levels				Levels				Levels				Levels			
	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1
Anomaly Evaluation																																					X	X	X					
Concept Documentation Evaluation	X	X	X																																									
Criticality Analysis	X	X	X	X	X	X	X	X	X	X	X		X	X	X																						X	X	X	X				
Design Evaluation									X	X	X	X																																
Evaluation of New Constraints																																	X	X	X									
Hazard Analysis	X	X			X	X			X	X			X	X			X	X			X	X			X	X			X	X			X	X			X	X						
Installation Checkout																													X	X														
Installation Configuration Audit																													X	X														
Interface Analysis					X	X	X		X	X	X		X	X	X																													
Migration Assessment																																					X	X						
Operation Procedures Evaluation																																	X	X										
Requirements Allocation Analysis	X																																											
Requirements Evaluation					X	X	X	X																																				
Retirement Assessment																																					X	X						
Risk Analysis	X	X			X	X			X	X			X	X			X	X			X	X			X	X			X	X			X	X			X	X						

Donde la X son los rangos de niveles esperados de cada uno de estos documentos.

D. Software Verification and Validation Report (SVVR)

Objetivos de la verificación y validación: Esta sección del plan debe describir los objetivos y las metas específicas que se quieren lograr mediante la verificación y validación del software. Esto puede incluir la identificación de los requisitos que se van a verificar y validar, así como los objetivos de calidad que se desean alcanzar.

Procedimientos de verificación y validación: El plan SVVP debe definir los procedimientos que se utilizarán para verificar y validar el software. Esto puede incluir una descripción detallada de los métodos de prueba que se utilizarán, los criterios de aceptación que se aplicarán y los procedimientos para la revisión de los resultados de las pruebas.

Recursos necesarios: El plan SVVP debe identificar los recursos necesarios para llevar a cabo la verificación y validación del software. Esto puede incluir los recursos humanos, financieros y tecnológicos necesarios para llevar a cabo las pruebas de verificación y validación.

Responsabilidades y roles: El plan SVVP debe definir las responsabilidades y los roles de cada uno de los miembros del equipo de verificación y validación. Esto puede incluir la definición de los roles de los miembros del equipo de prueba y las responsabilidades específicas que tendrán durante el proceso de verificación y validación.

Planificación y programación: El plan SVVP debe incluir un cronograma detallado para llevar a cabo la verificación y validación del software. Esto debe incluir fechas límite para las pruebas, las revisiones y las entregas del software.

E. User documentation

La sección de documentación de usuarios se concentra en el archivo de todos los medios por los cuales se le describe al usuario el uso operacional de la pieza de software, tanto los métodos como instrucciones como muestras de todo la metodología de uso para el usuario, esto con el conocimiento de que existen varios tipos de usuarios o anuales que necesitan un método diferente para que se les sea comunicado esta información de manera oportuna.

F. Software Configuration Management Plan (SCMP)

El plan del manejo de la configuración describe las tareas, metodologías y herramientas que aseguran el control adecuado de procedimientos y los controles que son documentados e implementados de manera correcta, si este no es un documento por sí mismo, deberá formar parte del SQAP, y si no,no es necesario que el elemento de la organización de SQA prepare el SCMP

5. Estándares, prácticas, convenciones y métricas (Sección 5 de SQAP)

5.1 Propósito

Los estándares que se serán utilizados son los siguientes:
Gestión de Configuración del Software:

- IEEE Std 828-2012 (Identificación de la configuración)
- ISO/IEC 12207 (Control de versiones)
- CMMI-DEV, V1.3

Plantillas:

- J-STD-016-1995

Corrección de los defectos encontrados:

- ISO/IEC 12207:2017 (Identificación y corrección de defectos encontrados)

Documentos de control:

- IEEE Std 1063 (Documentación del usuario)
- ISO 9001 (Documentos de Control)

Revisiones:

- IEEE 1028

Métricas:

- IEEE 1061
- ISO/IEC 9126
- ISO IEC 15939

Control del progreso del proyecto y costos de calidad:

- IPMBOK (Project Management Body of Knowledge)

Aseguramiento de la calidad de la contribución de participantes externos:

- IEEE 12207

Pruebas de software:

- IEEE 829

Para asegurar la conformidad con los estándares de Gestión de Configuración del Software, es importante implementar un sistema de monitoreo y control que permita verificar la identificación de la configuración del software, el control de versiones, la corrección de los defectos encontrados, la documentación del usuario, las revisiones, las métricas, el control del progreso del proyecto y costos de calidad, el aseguramiento de la calidad de la contribución de participantes externos y las pruebas de software.
En primer lugar, se debe establecer un proceso formal de Gestión de Configuración del Software que defina los elementos del software y su estado, los cambios que se han realizado, las versiones del software y las herramientas necesarias para llevar a cabo esta tarea. Este proceso debe ser documentado y seguido por todo el equipo de desarrollo.

Para el control de versiones, se debe establecer una herramienta que permita la identificación y seguimiento de las versiones del software y de los cambios realizados a lo largo del tiempo. Es importante asegurarse de que todas las versiones estén documentadas y disponibles para su revisión.
Para la corrección de los defectos encontrados, se debe implementar un proceso de seguimiento y resolución de los errores, que permita identificar los errores de forma temprana y resolverlos de manera oportuna.

En cuanto a la documentación del usuario, es necesario seguir las directrices establecidas en el estándar IEEE Std 1063 para garantizar que la documentación sea clara, precisa y completa.
Para las revisiones, se debe establecer un proceso formal que permita revisar regularmente el software para identificar posibles errores o mejoras. El estándar IEEE 1028 proporciona una guía detallada para la realización de revisiones.
Las métricas deben ser definidas y monitoreadas para evaluar la calidad del software y el progreso del proyecto. Los estándares IEEE 1061, ISO/IEC 9126 y ISO IEC 15939 proporcionan directrices para la definición y medición de métricas.
El control del progreso del proyecto y los costos de calidad deben ser monitoreados y controlados siguiendo las directrices establecidas en IPMBOK.
Para asegurar la calidad de la contribución de participantes externos, se debe seguir el estándar IEEE 12207 y se deben establecer mecanismos para garantizar que los participantes externos cumplan con los requisitos establecidos.
Finalmente, para las pruebas de software, se debe seguir el estándar IEEE 829 y establecer un proceso formal de pruebas que permita verificar que el software cumple con los requisitos establecidos y que funcione correctamente.

5.2 Contenido

- (IEEE Std 1063-2001)Las normas de documentación incluyen la definición de plantillas para los diferentes tipos de documentos que se deben crear durante el ciclo de vida del software, tales como especificaciones de requerimientos, diseños de software, manuales de usuario y manuales de mantenimiento.
- (ISO/IEC/IEEE 42010:2011)Las normas de diseño son igualmente importantes, ya que establecen la estructura y organización del software. Estas normas definen las mejores prácticas para la creación de diagramas de flujo, diagramas de clases, diagramas de secuencia y otros diagramas que permiten visualizar la estructura del software y las relaciones entre los diferentes componentes.
- (Estándar de codificación de Google)Las normas de codificación se centran en cómo se escriben y organizan las líneas de código. Estas normas establecen reglas para la nomenclatura de variables y módulos, indentación, estructuras de control y otros aspectos que afectan la legibilidad y mantenibilidad del código.

- (Estándar de documentación de código Javadoc de Oracle)Los estándares de comentarios son un componente importante de las normas de codificación, ya que ayudan a entender la intención detrás del código. Los comentarios también proporcionan información valiosa sobre el uso de funciones y variables, así como sobre cualquier limitación o precaución que se deba tener en cuenta al utilizar el software.
- (IEEE Std 829-2008)Las normas y prácticas de prueba son esenciales para garantizar que el software funcione correctamente y cumpla con los requerimientos del usuario. Estas normas definen cómo se deben realizar las pruebas, qué tipos de pruebas se deben llevar a cabo y cómo se deben documentar los resultados de las pruebas.
- (ISO/IEC 9126)Por último, las métricas seleccionadas para el aseguramiento de la calidad del software son críticas para evaluar el desempeño del equipo de desarrollo y la calidad del software. Las métricas seleccionadas deben permitir medir el desempeño y calidad de los productos y procesos del software de manera objetiva, para que se puedan tomar decisiones informadas y mejorar continuamente el proceso de desarrollo de software.

6. Revisiones de Software (Sección 6 de SQAP)

6.1 Propósito

El propósito de esta sección es definir los tipos de revisiones de software que se llevarán a cabo en el proyecto, incluyendo revisiones gerenciales, revisiones adquiriente-proveedor, revisiones técnicas, inspecciones, revisiones de seguimiento y auditorías. Además, se debe establecer el cronograma de revisiones de software y cómo se llevarán a cabo, así como las acciones adicionales requeridas para implementar y verificar los resultados de las revisiones.

6.2 Requerimientos mínimos

6.2.1 Revisión de especificaciones de software (SSR)

Se realizará para verificar que las especificaciones de software definidas en la SRD sean adecuadas y cumplan con los requisitos del usuario. Para realizar esta revisión, se utilizarán técnicas como revisión por pares, inspección de documentos, verificación de requisitos, pruebas unitarias y análisis estático.

1. Se convoca a un equipo de revisión formado por expertos en el dominio y en la tecnología del software.
2. Se proporciona al equipo de revisión una copia de la SRD y las especificaciones de software.

3. Se define un plan de revisión y se establece un plazo para la revisión.
4. Se utiliza una técnica de revisión por pares para identificar y evaluar los errores, omisiones y ambigüedades en las especificaciones de software.
5. Se lleva a cabo una inspección de documentos para detectar cualquier problema de formato o estilo en las especificaciones de software.
6. Se verifica que todos los requisitos del usuario se hayan incluido en las especificaciones de software.
7. Se realizan pruebas unitarias para verificar que los requisitos están bien definidos y son verificables.
8. Se lleva a cabo un análisis estático del código fuente para detectar cualquier posible error o problema en el diseño.

6.2.2 Revisión de diseño de arquitectura (ADR)

La revisión de diseño de arquitectura se realiza para evaluar la adecuación técnica del diseño preliminar del software, también conocido como diseño de nivel superior. Esto puede incluir la revisión de la estructura general del software, la interfaz del usuario, la funcionalidad y la compatibilidad con otros sistemas.

1. Se convoca a un equipo de revisión formado por expertos en el dominio y en la tecnología del software.
2. Se proporciona al equipo de revisión una copia del diseño preliminar del software.
3. Se define un plan de revisión y se establece un plazo para la revisión.
4. Se utiliza una técnica de revisión por pares para identificar y evaluar los errores, omisiones y ambigüedades en el diseño preliminar.
5. Se evalúa la estructura general del software, la interfaz del usuario, la funcionalidad y la compatibilidad con otros sistemas.
6. Se verifica que el diseño cumpla con los requisitos del usuario definidos en la SRD.
7. Se realizan pruebas unitarias para verificar que el diseño es viable y adecuado para su implementación.
8. Se lleva a cabo un análisis estático del código fuente para detectar cualquier posible error o problema en el diseño.

6.2.3 Revisión detallada de diseño (DDR)

La revisión detallada de diseño se realiza para evaluar los diseños detallados del software y garantizar que cumplan con los requisitos de la SRD. Esto puede incluir la revisión de diagramas de flujo, diagramas de clase, diagramas de secuencia y otros artefactos de diseño.

1. Se convoca a un equipo de revisión formado por expertos en el dominio y en la tecnología del software.
2. Se proporciona al equipo de revisión una copia de los diseños detallados del software.
3. Se define un plan de revisión y se establece un plazo para la revisión.

4. Se utiliza una técnica de revisión por pares para identificar y evaluar los errores, omisiones y ambigüedades en los diseños detallados.
5. Se evalúa la coherencia de los diagramas de flujo, diagramas de clase, diagramas de secuencia y otros artefactos de diseño.
6. Se verifica que los diseños cumplan con los requisitos de la SRD.
7. Se realizan pruebas unitarias para verificar que los diseños son adecuados para su implementación.
8. Se lleva a cabo un análisis estático del código fuente para detectar cualquier posible error o problema en los diseños.

6.2.4 Revisión de plan de verificación y validación

La revisión del plan de verificación y validación se realiza para evaluar la adecuación y la completitud de los métodos de verificación y validación definidos en los planes de verificación y validación. Esto puede incluir la revisión de la estrategia de pruebas, la metodología de pruebas, los casos de prueba y los criterios de aceptación.

1. Revisar el plan de verificación y validación para asegurarse de que cubre todos los aspectos relevantes del software.
2. Evaluar la adecuación de los métodos de verificación y validación definidos en el plan, asegurándose de que sean apropiados para el software y su contexto de uso.
3. Revisar la estrategia de pruebas, la metodología de pruebas, los casos de prueba y los criterios de aceptación, asegurándose de que sean completos y adecuados para la verificación y validación del software.

6.2.5 Auditoría funcional

La auditoría funcional se realiza antes de la entrega del software para verificar que se han cumplido todos los requisitos especificados en la SRD. Esto puede incluir la revisión de la funcionalidad del software, los casos de prueba, los resultados de las pruebas y la documentación asociada.

1. Revisar la SRD (documento de requerimientos de software) para asegurarse de que se han cumplido todos los requisitos especificados.
2. Verificar la funcionalidad del software y los casos de prueba asociados, comprobando que se han implementado correctamente y que proporcionan la cobertura adecuada.
3. Revisar los resultados de las pruebas y la documentación asociada para asegurarse de que sean precisos y completos.

6.2.6 Auditoría física

La auditoría física se realiza para verificar la consistencia interna del software y su documentación, y su preparación para su lanzamiento. Esto puede incluir la

revisión de la documentación del software, la calidad del código, el control de versiones, la gestión de configuración y la compatibilidad del software.

1. Revisar la documentación del software para asegurarse de que sea precisa y completa.
2. Evaluar la calidad del código, revisando los estándares de codificación y las prácticas de desarrollo para asegurarse de que se cumplan.
3. Verificar el control de versiones y la gestión de configuración, asegurándose de que se hayan implementado correctamente y que sean apropiados para el software.
4. Comprobar la compatibilidad del software, asegurándose de que el software sea compatible con los sistemas operativos y plataformas necesarios.

6.2.7 Auditorías en proceso

Estas revisiones se llevarían a cabo en muestras del diseño para verificar la consistencia del diseño en áreas como el código versus la documentación de diseño, las especificaciones de interfaz y las implementaciones de diseño versus los requisitos funcionales. Esto se podría lograr mediante la revisión del código fuente, la comprobación de la documentación y la realización de pruebas en el software.

1. Realizar una revisión del diseño para comprobar la consistencia del código con la documentación de diseño y los requisitos funcionales.
2. Verificar las especificaciones de interfaz y las implementaciones de diseño para asegurarse de que sean consistentes y adecuadas para el software.
3. Realizar pruebas en el software para asegurarse de que cumple con los requisitos funcionales y de diseño, y que proporciona la funcionalidad deseada.

6.2.8 Revisiones gerenciales

Las revisiones gerenciales se llevan a cabo para evaluar la adecuación y la completitud del proceso de desarrollo de software en general. Para realizar esta revisión, se seguirán los siguientes pasos:

1. Reunir a los miembros del equipo de desarrollo de software y a los gerentes de proyecto.
2. Proporcionar a los asistentes una lista de verificación de revisiones gerenciales que cubra los puntos clave a revisar.
3. Discutir el proceso de desarrollo de software en general y cualquier problema o desafío que haya surgido durante el proyecto.
4. Evaluar la eficacia del proceso de desarrollo de software para garantizar que se cumplan los objetivos del proyecto y que se sigan las mejores prácticas.

5. Discutir los próximos pasos para mejorar el proceso de desarrollo de software en el futuro.
6. Documentar los hallazgos de la revisión y cualquier recomendación para mejorar el proceso de desarrollo de software.

6.2.9 Revisión del plan de gestión de configuración de software (SCMPR)

Para llevar a cabo esta revisión, se evaluará la adecuación y la completitud de los métodos de gestión de configuración definidos en el SCMP. Esto incluiría la revisión del plan de gestión de configuración, la identificación de los elementos de configuración, el control de cambios y la gestión de versiones.

1. Reunir a los miembros del equipo de desarrollo de software y a los gerentes de proyecto.
2. Proporcionar a los asistentes una copia del SCMP y una lista de verificación de revisión del SCMP.
3. Revisar el SCMP y evaluar su efectividad para garantizar la integridad del software y su documentación, así como la gestión de cambios y versiones.
4. Identificar cualquier área del SCMP que necesite mejoras y hacer recomendaciones para mejorar el plan.
5. Documentar los hallazgos de la revisión y cualquier recomendación para mejorar el SCMP.

6.2.10 Revisión post-implementación

Esta revisión se llevaría a cabo para evaluar las actividades de desarrollo del proyecto y proporcionar recomendaciones para acciones apropiadas. Para ello, se podría llevar a cabo una evaluación de la implementación del software y su rendimiento, revisar los informes de progreso y la documentación relevante, y realizar entrevistas con el personal implicado en el desarrollo y la implementación del software.

1. Reunir a los miembros del equipo de desarrollo de software y a los gerentes de proyecto.
2. Proporcionar a los asistentes una lista de verificación de revisión post-implementación que cubra los puntos clave a revisar.
3. Evaluar la implementación del software y su rendimiento en el entorno en el que se ha desplegado.
4. Revisar los informes de progreso y la documentación relevante para evaluar el éxito del proyecto y los problemas que puedan haber surgido durante el proceso.
5. Realizar entrevistas con el personal implicado en el desarrollo y la implementación del software para obtener información adicional sobre el proceso de desarrollo.

6. Documentar los hallazgos de la revisión y cualquier recomendación para mejorar el proceso de desarrollo de software en el futuro.

6.3 Otras revisiones y auditorías

Pueden llegar a ser necesitadas otras revisiones y auditorías, por ejemplo, la revisión de documentación de usuario (UDR), el cual serviría para evaluar la adecuación de la documentación de usuario. Ahora, si hablamos de las normas específicas de documentación, diseño, codificación y pruebas, se establecerán de acuerdo a las necesidades del proyecto, tales como el IEEE 1063 para la documentación del usuario, IEEE 1061 para el diseño de software y el IEEE 829 para las pruebas de software. Así mismo, es posible utilizar métricas seleccionadas de productos y procesos de aseguramiento de calidad de software, por ejemplo, las métricas establecidas del IEEE 1061 y del ISO/IEC.

7. Pruebas (Sección 7 de SQAP)

Las siguentes pruebas serán implementadas en el plan SQA::

- **Pruebas de seguridad:** Estas pruebas evalúan la seguridad del software y su capacidad para proteger los datos y sistemas sensibles. Los pasos pueden incluir la identificación de posibles vulnerabilidades, la ejecución de pruebas de penetración, la evaluación de la resistencia a los ataques y la documentación de los resultados.
- **Pruebas de usabilidad:** Estas pruebas evalúan la facilidad de uso del software y su capacidad para satisfacer las necesidades de los usuarios. Los pasos pueden incluir la identificación de escenarios de uso, la selección de usuarios de prueba, la ejecución de las pruebas y la documentación de los resultados.

8. Reportes de problemas y acciones correctivas (Sección 8 de SQAP)

- La responsabilidad principal de la sección 4.8 del SQAP recae en el Responsable de Administración del Proyecto Específico (RAPE), quien debe establecer las prácticas y procedimientos para la identificación, reporte, seguimiento y resolución de problemas en el software y en el proceso de desarrollo y mantenimiento.
- El Responsable de Desarrollo y Mantenimiento de Software (RDM) es responsable de llevar a cabo las acciones correctivas para resolver los problemas identificados en el software y en el proceso de desarrollo y mantenimiento.

- El Analista (AN) y el Diseñador de Interfaz de Usuario (DU) son responsables de asegurar que los requisitos del usuario se haya especificado correctamente en los documentos de análisis y diseño, para evitar problemas en la implementación.
- El Diseñador (DI) y el Programador (PR) son responsables de implementar el software de acuerdo con los documentos de análisis y diseño, y de hacerlo de manera correcta y efectiva para evitar problemas de calidad y errores.
- El Responsable de Pruebas (RPU) es responsable de asegurarse de que las pruebas se realicen de manera adecuada y completa, y de reportar los problemas encontrados durante las pruebas.
- El Revisor (RE) es responsable de revisar y validar el software y los documentos relacionados con el software para asegurarse de que cumplen con los requisitos y de que se han seguido los procesos establecidos.
- El Responsable de Manuales (RM) es responsable de asegurarse de que los manuales de usuario y de operación sean precisos y estén actualizados, para evitar problemas de comprensión y uso incorrecto del software.
- El Equipo de Trabajo (ET) es responsable de reportar cualquier problema o error que encuentre durante el proceso de desarrollo y de colaborar en la resolución de los problemas.
- El Cliente (CL) y el Usuario (US) son responsables de reportar cualquier problema o error que encuentren durante el uso del software y de colaborar en la resolución de los problemas, proporcionando información adicional y feedback.

9. Herramientas, técnicas y metodologías.

El SQAP deberá identificar las herramientas, técnicas y metodologías que se utilizarán para respaldar el aseguramiento de la calidad del software. Debe enumerar o hacer referencia a las herramientas, técnicas y metodologías que están disponibles y aquellas que necesitan ser adquiridas o desarrolladas. También se debe identificar la(s) organización(es) responsable(s).

9.1 Herramientas

A lo largo de todas las fases del desarrollo se llevarán a cabo distintos procesos los cuales están detallados en el modelo de procesos de PAPUSCORP. Cada uno arrojando un documento como resultado, basado en las siguientes plantillas:

Anexo G (Pags 93-109)	Diseño de software y descripción de estos productos, esta plantilla se utilizará para definir nuestro proceso de diseño y análisis de nuestro producto de software.
Anexo K (Pags 139-144)	Los documentos usados deberán cumplir con las definiciones y plantillas definidas en el anexo K

Anexo L (Pags 145-151)	Para poder realizar algún cambio, se deberá evaluar cualquier producto de software bajo el checklist que se puede encontrar en el anexo L.
Anexo M (Pags 152-153)	Para los change requests, se deberá usar el sistema del Anexo M, para poder
Anexo N (Pags 154-156)	Se utilizara el apartado N3.6(Pag 155) para organizar meetings donde se resuelvan los problemas reportados conforme a nuestros Items de diseño y documentos de control

9.2 Tecnicas

A lo largo del proceso de desarrollo se seguirán los siguientes estándares para la gestion de la configuracion del software:

Estándar	Descripción
IEEE Std 828-2012 (Identificación de la configuración)	Define la configuración de software como una colección de elementos de trabajo relacionados que se gestionan y mantienen a lo largo del ciclo de vida del software.
ISO/IEC 12207 (Control de versiones)	Proporciona una guía para el ciclo de vida del software y define un marco para la gestión de la configuración de software en todas las etapas del ciclo de vida. Se enfoca en la identificación, control y seguimiento de los cambios en los elementos de configuración del software.
CMMI-DEV, V1.3	Ayuda a establecer y mantener la integridad de los productos de trabajo utilizando la identificación de la configuración, el control de la configuración, el informe del estado de la configuración y las auditorías de la configuración.

En estos se encuentran procesos como la corrección de defectos encontrados (ISO/IEC 12207:2017(Identificación y corrección de defectos encontrados)). Así como los documentos de control, revisiones , métricas y planes de pruebas.

Documentos de control

Estándar	Descripción
IEEE Std 1063 (Documentación del usuario)	En esta norma se proporcionan los requisitos mínimos para la estructura, el contenido de la información y el formato de la documentación del usuario, incluidos los documentos impresos y electrónicos utilizados en el entorno de trabajo por los usuarios de sistemas que contienen software.
ISO 9001 (Documentos de control)	Los documentos requeridos por el Sistema de Gestión de la Calidad deben controlarse, la norma ISO 9001 pide un procedimiento documentado que defina los controles para la aprobación, revisión y actualización de los documentos, los cambios deben identificarse así como el estado de revisión de los documentos.

Revisiones

Estándar	Descripción
IEEE 1028	Proporciona directrices para llevar a cabo revisiones formales de software. Define los procesos y procedimientos para realizar revisiones técnicas y auditorías de software, incluyendo revisiones de diseño, código, documentación y pruebas.

Métricas

Estándar	Descripción
IEEE 1061	Proporciona directrices para la selección, definición y aplicación de métricas de calidad del software. Define un enfoque sistemático para establecer y utilizar métricas de calidad del software, incluyendo la identificación de métricas apropiadas, la recopilación y análisis de datos, y la interpretación de los resultados obtenidos.
ISO/IEC 9126	Proporciona un marco de trabajo para la evaluación de la calidad del software, incluyendo aspectos relacionados con el diseño. Esta norma incluye un conjunto de características y subcaracterísticas de calidad, y proporciona indicadores (que pueden ser considerados como métricas) para evaluar cada una de ellas. Algunas de las subcaracterísticas de calidad, como la cohesión y el acoplamiento, están relacionadas con el diseño del software.

Pruebas de software

Estándar	Descripción
IEEE 829	Este es un estándar para la documentación de pruebas de software. Proporciona un conjunto de requisitos para los documentos de prueba, incluyendo el plan de pruebas, el caso de prueba y el informe de prueba.

9.3 Metodologías.

Toda metodología y procesos está documentado en el modelo de procesos Papus Corp:

10. Control de medios.

Para tener una óptima gestión en nuestros documentos emplearemos GIT para almacenar los documentos relacionados al diseño del proyecto. Dichos documentos seran nombradas de tal manera que podamos identificar las versiones y nombres de los documentos de forma más eficiente.

11. Control de proveedores.

Esta seccion no es aplicable a este plan

Nuestros proyectos no requerirán de proveedores externos, por lo que cada elemento de software será desarrollado de manera local utilizando este mismo documento SQAP manteniendo así un mejor control de la calidad de nuestros productos.

12. Recopilación, mantenimiento y retención de registros.

Los procedimientos de recopilación, mantenimiento y retención de registros SQA en el SQAP deben ajustarse a los métodos SCMP aprobados. Para evitar la redundancia, se podría anotar una referencia a los procedimientos SCMP en el SQAP lo que es este caso, por lo cual tenemos la siguiente referencia.

Gestión de la Configuración del Software.

Estándar	Descripción
IEEE Std 828-2012 (Identificación de la configuración)	Define la configuración de software como una colección de elementos de trabajo relacionados que se gestionan y mantienen a lo largo del ciclo de vida del software.
ISO/IEC 12207 (Control de versiones)	Proporciona una guía para el ciclo de vida del software y define un marco para la gestión de la configuración de software en todas las etapas del ciclo de vida. Se enfoca en la identificación, control y seguimiento de los cambios en los elementos de configuración del software.
CMMI-DEV, V1.3	Ayuda a establecer y mantener la integridad de los productos de trabajo utilizando la identificación de la configuración, el control de

	la configuración, el informe del estado de la configuración y las auditorías de la configuración.

13. Entrenamiento

Evaluación de necesidades de capacitación:

En esta sección del Plan de Aseguramiento de la Calidad, se llevará a cabo una evaluación exhaustiva de las necesidades de capacitación del personal designado para realizar las tareas y actividades definidas en el SQAP. La evaluación se realizará mediante los siguientes pasos:

Descripción de las tareas y actividades definidas en el SQAP:

Tarea	Descripción
Realización de revisiones y auditorías de calidad en el proceso de desarrollo del software.	Esta tarea implica llevar a cabo revisiones y auditorías periódicas para garantizar que se cumplan los estándares de calidad establecidos en el proyecto. Durante estas revisiones, se evaluarán la documentación, el código fuente y otros elementos relevantes para identificar posibles problemas, incumplimientos de estándares o áreas de mejora. Esta tarea es fundamental para asegurar que el producto final cumpla con los requisitos de calidad y satisfaga las necesidades de los usuarios.
Establecimiento y seguimiento de los estándares de calidad en el desarrollo del software.	Esta tarea consiste en definir y mantener los estándares de calidad que deben aplicarse en todas las fases del desarrollo del software. Esto incluye la definición de buenas prácticas, estándares de codificación, convenciones de nomenclatura, entre otros aspectos. Además, implica el seguimiento y control del cumplimiento de estos estándares durante todo el ciclo de vida del proyecto. El objetivo es garantizar la consistencia, la calidad y la eficiencia en el desarrollo del software.
Gestión de la documentación y registros del proyecto.	Esta tarea se encarga de la creación, mantenimiento y control de la documentación y registros del proyecto. Incluye la elaboración de documentos como el Plan de Aseguramiento de la Calidad, informes de auditoría, informes de pruebas y otros documentos relacionados. Asimismo, implica establecer procedimientos para la gestión de

	versiones, revisiones y aprobaciones de la documentación. Una correcta gestión de la documentación y registros es esencial para garantizar la trazabilidad, la transparencia y la conformidad con los estándares de calidad establecidos.
Identificación y resolución de problemas en el proceso de desarrollo del software.	Esta tarea se enfoca en identificar y resolver problemas que puedan surgir durante el desarrollo del software. Incluye la recolección y análisis de datos, la investigación de causas raíz y la implementación de acciones correctivas y preventivas. La tarea también implica el seguimiento de las acciones tomadas y la evaluación de su efectividad. Una identificación y resolución efectiva de problemas contribuye a mejorar la calidad del software, reducir riesgos y asegurar un proceso de desarrollo más eficiente.
Control y seguimiento del código, medios y proveedores externos.	Esta tarea se centra en el control y seguimiento del código fuente, los medios utilizados en el desarrollo del software y los proveedores externos involucrados en el proyecto. Esto implica establecer y aplicar prácticas de control de versiones del código, asegurar la integridad y confidencialidad de los medios utilizados, y supervisar el cumplimiento de los acuerdos y requisitos establecidos con los proveedores externos. El objetivo es garantizar la calidad de los componentes utilizados en el desarrollo del software y minimizar los riesgos asociados a factores externos.

Identificación de las habilidades y conocimientos necesarios para cada tarea:

Tareas	Conocimientos requeridos
Realización de análisis de requerimientos y definición de la visión del proyecto.	● Habilidades de análisis de requerimientos para comprender las necesidades del cliente y traducirlas en requisitos funcionales y no funcionales. ● Conocimiento de técnicas de entrevistas y encuestas para recopilar información relevante.

	● Capacidad para definir la visión del proyecto y establecer objetivos claros. ● Conocimiento de herramientas de modelado de procesos y diagramas para visualizar el flujo de trabajo y la estructura del proyecto.
Diseño de la arquitectura del proyecto y selección de tecnologías.	● Competencia en la definición de la arquitectura del proyecto, incluyendo la elección de patrones de diseño adecuados y la organización de los componentes del sistema. ● Conocimiento de diferentes tecnologías y frameworks utilizados en el diseño de proyectos, como bases de datos, lenguajes de programación, plataformas de desarrollo, entre otros. ● Capacidad para evaluar y seleccionar las tecnologías más adecuadas para el proyecto, considerando requisitos de rendimiento, escalabilidad y mantenibilidad. ● Habilidades en el diseño de interfaces de usuario intuitivas y atractivas, teniendo en cuenta los principios de usabilidad y experiencia de usuario.
Definición y gestión de la estrategia de prueba y validación del diseño.	● Competencia en la planificación y ejecución de pruebas de usabilidad para evaluar la eficacia y la experiencia del usuario con el diseño. ● Conocimiento de metodologías y técnicas de prueba, como pruebas de usabilidad, pruebas A/B y pruebas de accesibilidad. ● Capacidad para recopilar y analizar datos de pruebas y utilizarlos para tomar decisiones informadas sobre mejoras en el diseño. ● Habilidades de comunicación para colaborar con otros miembros del equipo de desarrollo y garantizar la

	implementación adecuada de los diseños.
Evaluación y mejora continua del diseño del proyecto.	● Habilidades de análisis y evaluación de métricas clave de diseño, como tasa de conversión, tiempos de respuesta y niveles de satisfacción del usuario. ● Conocimiento de técnicas de retroalimentación del usuario, como encuestas y entrevistas, para recopilar comentarios y opiniones sobre el diseño. ● Capacidad para identificar áreas de mejora en el diseño y proponer soluciones y cambios iterativos. ● Competencia en la realización de pruebas de concepto y pilotos para validar nuevas ideas y funcionalidades antes de su implementación completa.

Determinación de las herramientas, técnicas, metodologías y recursos informáticos requeridos:

Herramientas de gestión de calidad y seguimiento de incidencias:

- Herramientas de gestión de proyectos, como Jira, Trello o Asana, para el seguimiento y asignación de tareas relacionadas con el diseño del proyecto.
- Herramientas de seguimiento de incidencias y control de versiones, como Bugzilla, Redmine o GitLab, para registrar y hacer seguimiento de los problemas identificados durante el diseño.

Técnicas de revisión y auditoría de documentación y código:

- Técnicas de revisión de documentación, como revisión por pares o revisión formal, para garantizar la precisión, claridad y coherencia de la documentación del proyecto.
- Técnicas de revisión de código, como revisión por pares o inspección de código estática, para identificar problemas de calidad, cumplimiento de estándares y buenas prácticas de codificación.

Metodologías ágiles o en cascada, dependiendo del enfoque del proyecto:

- Metodologías ágiles, como Scrum o Kanban, que permiten una iteración rápida y flexibilidad en el diseño del proyecto, fomentando la colaboración y la retroalimentación continua.
- Metodología en cascada clásica, que sigue una secuencia lineal de fases, desde el análisis hasta la implementación, adecuada para proyectos con requisitos bien definidos y cambios mínimos en el diseño.

Recursos informáticos, como software de pruebas o entornos de desarrollo:

- Herramientas de diseño gráfico y prototipado, como Adobe XD, Sketch, Figma o InVision, para crear y validar los diseños del proyecto.
- Herramientas de pruebas de usabilidad, como UserTesting o UsabilityHub, que facilitan la realización de pruebas con usuarios para evaluar la experiencia y usabilidad del diseño.
- Entornos de desarrollo integrados (IDE, por sus siglas en inglés) específicos para el lenguaje de programación utilizado en el proyecto, que brindan herramientas y funciones para facilitar el desarrollo y la depuración del código.

Evaluación de las brechas de capacitación actuales:

La evaluación de las brechas de capacitación actuales en el contexto del diseño del proyecto implica comparar las habilidades y conocimientos del equipo designado con los requisitos identificados. A continuación, se presenta un ejemplo de cómo llevar a cabo esta evaluación:

Tarea	Habilidades y conocimientos	Evaluación del equipo	Resultado de evaluación
Análisis de requerimientos	● Capacidad para comprender y analizar los requisitos del proyecto. ● Conocimiento de técnicas de recopilación y documentación de requisitos. ● Competencia en la identificación de requisitos funcionales y no funcionales.	● Realización de entrevistas y revisiones de proyectos anteriores. ● Evaluación del equipo en base a su comprensión y análisis de los requerimientos de proyectos anteriores.	● Brecha de capacitación identificada: Algunos miembros del equipo tienen experiencia limitada en el análisis de requerimientos y podrían beneficiarse de un mayor desarrollo en esta área. ● Punto fuerte del equipo: Algunos miembros del equipo tienen una sólida experiencia en el análisis de requerimientos y pueden brindar orientación y apoyo al resto del equipo.

Diseño de interfaz de usuario	● Conocimiento de principios de diseño de interfaz de usuario y experiencia de usuario. ● Competencia en el uso de herramientas de diseño gráfico y prototipado. ● Capacidad para crear interfaces intuitivas y atractivas.	● Revisión de proyectos anteriores y evaluación del diseño de las interfaces de usuario. ● Evaluación del equipo en base a su conocimiento de principios de diseño de interfaz de usuario y experiencia en el uso de herramientas de diseño.	● Brecha de capacitación identificada: Algunos miembros del equipo tienen habilidades limitadas en el diseño de interfaces de usuario y pueden necesitar un desarrollo adicional para mejorar sus capacidades. ● Punto fuerte del equipo: Algunos miembros del equipo tienen experiencia sólida en el diseño de interfaces de usuario y pueden brindar orientación y liderazgo en esta área.
Arquitectura del proyecto	● Conocimiento de principios de arquitectura	● Revisión de proyectos anteriores y	● Brecha de capacitación identificada:

	de software y patrones de diseño. ● Capacidad para diseñar una arquitectura escalable y modular. ● Competencia en el uso de herramientas de modelado y documentación de arquitectura.	evaluación de la arquitectura de los sistemas diseñados. ● Evaluación del equipo en base a su conocimiento y experiencia en la arquitectura de proyectos anteriores.	Algunos miembros del equipo tienen un conocimiento limitado de principios de arquitectura y pueden necesitar fortalecer sus habilidades en esta área. ● Punto fuerte del equipo: Algunos miembros del equipo tienen una sólida experiencia en la arquitectura de proyectos y pueden brindar orientación y apoyo en la definición de la arquitectura del proyecto actual.

Determinación de las necesidades de formación:

Con base en la evaluación de brechas de capacitación, se determinan las necesidades de formación específicas para cada tarea y cada individuo. Establece qué áreas de conocimiento o habilidades deben fortalecerse a través de la capacitación.

Tarea	Brecha de capacitación identificada	Necesidades de formación especificas
Análisis de requerimientos	Algunos miembros del equipo tienen experiencia limitada en el análisis de requerimientos y podrían beneficiarse de un mayor desarrollo en esta área.	● Curso de análisis de requerimientos para proporcionar una comprensión más profunda de las técnicas y mejores prácticas en el análisis de requisitos. ● Talleres prácticos sobre la documentación efectiva de los requerimientos del proyecto. ● Sesiones de capacitación internas para compartir conocimientos y experiencias en el análisis de requerimientos.
Diseño de interfaz de usuario	Algunos miembros del equipo tienen habilidades limitadas en el diseño de interfaces de usuario y pueden necesitar un desarrollo adicional para mejorar sus capacidades.	● Curso de diseño de interfaz de usuario para adquirir conocimientos sólidos sobre principios de diseño y experiencia de usuario. ● Talleres prácticos sobre el uso de herramientas de diseño gráfico y prototipado para mejorar las habilidades técnicas.

		● Participación en conferencias o seminarios relacionados con el diseño de interfaces de usuario para estar al tanto de las últimas tendencias y prácticas.
Arquitectura del proyecto	Algunos miembros del equipo tienen un conocimiento limitado de principios de arquitectura y pueden necesitar fortalecer sus habilidades en esta área.	● Curso de arquitectura de software y patrones de diseño para mejorar la comprensión de los fundamentos teóricos y prácticos. ● Talleres prácticos de modelado y documentación de arquitectura para fortalecer las habilidades de diseño arquitectónico. ● Participación en proyectos internos o externos de arquitectura para ganar experiencia práctica y aplicar los conocimientos adquiridos.

<u>Establecimiento de prioridades:</u>

1. Impacto en el éxito del proyecto: Evalúa qué habilidades o conocimientos son críticos para el diseño del proyecto y tienen un impacto directo en su éxito. Estos aspectos deben abordarse con prioridad para garantizar la calidad y la satisfacción del cliente.
2. Cumplimiento de estándares de calidad: Identifica aquellas habilidades o conocimientos que son esenciales para cumplir con los estándares de calidad establecidos. Estos aspectos deben recibir

una atención prioritaria, ya que son fundamentales para garantizar que el proyecto cumpla con los requisitos y las expectativas de calidad.

3. Disponibilidad de recursos: Considera los recursos disponibles, como instructores capacitados, materiales de capacitación y presupuesto asignado. Prioriza aquellas necesidades de capacitación para las cuales tienes los recursos adecuados disponibles o puedes obtenerlos de manera oportuna.
4. Tiempo necesario para la capacitación: Evalúa el tiempo requerido para capacitar al equipo en cada habilidad o conocimiento identificado. Prioriza aquellas necesidades que puedan abordarse en un tiempo razonable sin comprometer las fechas límite del proyecto.
5. Fechas límite del proyecto: Considera las fechas límite establecidas para el proyecto. Prioriza las necesidades de capacitación que se pueden abordar antes de las fechas límite críticas, de modo que el equipo esté debidamente capacitado y preparado para enfrentar los desafíos del proyecto.
6. Importancia estratégica a largo plazo: Evalúa las habilidades o conocimientos que tienen un valor estratégico a largo plazo para la organización. Si alguna de estas necesidades de capacitación tiene un impacto significativo en el crecimiento y desarrollo de la empresa, considérala como una prioridad.

Recopilación de datos:

Tipos de datos recopilados:

1. Se llevarán a cabo pruebas o evaluaciones para medir el nivel de habilidades y competencias del personal en relación con las necesidades identificadas. Estos resultados proporcionarán una visión objetiva de las brechas existentes.
2. Se solicitará a los miembros del equipo que proporcionen retroalimentación sobre sus propias áreas de mejora y necesidades de capacitación. Esto se llevará a cabo a través de encuestas, entrevistas individuales o sesiones de retroalimentación.
3. Se revisarán las evaluaciones de desempeño existentes para identificar áreas específicas donde se requiere desarrollo adicional. Estas evaluaciones brindarán información sobre las fortalezas y debilidades individuales en relación con las tareas de diseño del proyecto.
4. Se recopilarán comentarios y observaciones de los supervisores o líderes de equipo que trabajan directamente con el personal. Ellos tendrán una visión más detallada de las habilidades y conocimientos necesarios para el diseño del proyecto.

Plantilla de matriz de capacitación

Matriz de capacitación que se utilizara la cual permitirá identificar rápidamente las brechas de capacitación para cada tarea y cada individuo.

Tarea	Requisito de habilidades	Habilidades del personal	Brechas de capacitación

Plan de capacitación.

Actividad de capacitación	Descripción	Objetivos de aprendizaje	Recursos necesarios	Responsable del curso	Cronograma
Diseño de arquitectura de software	En esta capacitación, los participantes adquirirán los conocimientos necesarios para diseñar una arquitectura de software eficiente y escalable para el proyecto. Se explorarán las mejores prácticas y principios de diseño que permitirán desarrollar un sistema robusto y modular.	Comprender los conceptos fundamentales de la arquitectura de software. Aprender a identificar los requisitos del proyecto y traducirlos en una arquitectura adecuada. Familiarizarse con patrones de diseño y principios SOLID para construir un sistema flexible y mantenible. Desarrollar habilidades para documentar y comunicar eficientement e el diseño de la arquitectura.	Herramientas de modelado de arquitectura (por ejemplo, UML). Ejemplos de arquitecturas de software exitosas. Materiales de lectura sobre principios y patrones de diseño de software.	Arquitecto de software senior.	Duración: 2 días Fecha: 10-11 de agosto de 2023 Horario: 9:00 AM - 5:00 PM
Diseño de bases de datos	En esta capacitación, los participantes aprenderán a diseñar bases de datos eficientes y bien estructuradas	Comprender los principios de diseño de bases de datos relacionales. Aprender a modelar y normalizar datos para evitar	Herramientas de modelado de bases de datos (por ejemplo, MySQL Workbench, Oracle SQL Developer).	Ingeniero de bases de datos.	Duración: 1 día Fecha: 5 de septiembre de 2023 Horario: 9:00 AM - 5:00 PM

	para el proyecto. Se abordarán los conceptos de modelado de datos, normalización y optimización de consultas, permitiendo crear una base de datos escalable y de alto rendimiento.	redundancias y garantizar la integridad. Familiarizarse con herramientas y técnicas para optimizar consultas y mejorar el rendimiento de la base de datos. Desarrollar habilidades para documentar y comunicar eficientement e el diseño de la base de datos.	Ejemplos de diseños de bases de datos eficientes. Materiales de lectura sobre modelado de bases de datos y optimización de consultas.		
Diseño de interfaz de usuario y experiencia de usuario	En esta capacitación, los participantes aprenderán a diseñar interfaces de usuario intuitivas y atractivas, centrándose en la experiencia del usuario. Se explorarán las mejores prácticas de diseño de interfaces, la usabilidad y la interacción, permitiendo crear una experiencia de	Comprender los principios fundamentales del diseño de interfaces de usuario. Aprender a analizar las necesidades y expectativas de los usuarios para diseñar interfaces efectivas. Familiarizarse con técnicas de diseño centrado en el usuario, como la creación de prototipos y pruebas de usabilidad. Desarrollar habilidades en	Software de diseño de interfaces de usuario (por ejemplo, Adobe XD, Sketch). Herramientas de prototipado (por ejemplo, InVision, Figma). Ejemplos de interfaces de usuario exitosas. Materiales de lectura sobre diseño de interfaces y experiencia de usuario.	Diseñador de interfaces de usuario.	Duración: 2 días Fecha: 20-21 de octubre de 2023 Horario: 9:00 AM - 5:00 PM

	usuario satisfactoria.	el uso de herramientas de diseño de interfaces y gráficos.			

Se revisarán los programas de capacitación existentes en la organización y se determinara si pueden adaptarse para cubrir las necesidades identificadas en el proyecto. Si no existen programas adecuados, se considerara desarrollar nuevos programas de capacitación o buscar recursos externos para cubrir las brechas de conocimiento.

A medida que se completen las actividades de capacitación, se actualizara la matriz de capacitación con los nombres de las personas capacitadas y los cursos o actividades realizados. Esto ayudará a mantener un registro actualizado de la capacitación y a evaluar el cumplimiento de los objetivos de capacitación.

14. Gestión de riesgos

Identificación de riesgos:

Tareas necesarias:

- Enumerar y describir los posibles riesgos que podrían afectar la calidad del software o el éxito del proyecto.
- Identificar los riesgos específicos asociados con el diseño del proyecto y las actividades de desarrollo relacionadas.
- Utilizar técnicas como el análisis de riesgos, revisión de documentos y experiencias previas para identificar los riesgos potenciales.

Evaluación de riesgos:

Evaluar la probabilidad de que ocurran los riesgos identificados y el impacto que tendrían en el proyecto y en la calidad del software.

Clasificación de riesgos:

1. Incompatibilidad de requisitos de diseño (Alta prioridad)
2. Cambios frecuentes en los requisitos de diseño (Alta prioridad)
3. Falta de experiencia en la tecnología seleccionada (Media prioridad)
4. Dependencia de terceros para componentes clave (Media prioridad)
5. Incumplimiento de estándares de diseño establecidos (Baja prioridad)

Monitoreo y control de riesgos:

Procedimientos y responsabilidades:

Se designará a un miembro del equipo de desarrollo como responsable de monitorear y controlar los riesgos identificados a lo largo del ciclo de vida del proyecto. Esta persona será responsable de llevar a cabo las siguientes actividades:

- Monitorear regularmente los riesgos identificados y su evolución a lo largo del proyecto.
- Registrar y mantener actualizada la información relacionada con los riesgos, incluyendo su probabilidad, impacto y clasificación de riesgo.
- Comunicar y compartir los informes de riesgos con el equipo de proyecto y los interesados relevantes.
- Coordinar la implementación de las estrategias de mitigación y supervisar su efectividad.
- Realizar revisiones periódicas de los riesgos y ajustar las estrategias de mitigación según sea necesario.

Métricas y criterios de evaluación: Se establecerán métricas y criterios claros para evaluar y medir el impacto de los riesgos identificados.

Métricas que considerar:

- Porcentaje de cumplimiento de las estrategias de mitigación establecidas.
- Tiempo promedio de respuesta y resolución de los riesgos.

- Costo adicional incurrido debido a los riesgos identificados.
- Nivel de satisfacción de los interesados en relación con la gestión de riesgos.

Estas métricas y criterios permitirán evaluar la efectividad de las estrategias de mitigación y tomar acciones correctivas si es necesario, para aplicar las métricas nos basaremos del estándar ISO/IEC 15939.

Estrategias y planes de mitigación

Riesgo: Retraso en la entrega del software debido a problemas de infraestructura.

Estrategias y planes de mitigación:

1. Desarrollo de planes de contingencia:
   - Establecer un plan alternativo de infraestructura que pueda utilizarse en caso de problemas con la infraestructura principal.
   - Definir procedimientos claros para la migración rápida a la infraestructura de respaldo en caso de falla.
2. Medidas preventivas:
   - Realizar una evaluación exhaustiva de la infraestructura antes de comenzar el proyecto para identificar posibles problemas y solucionarlos de antemano.
   - Establecer acuerdos de nivel de servicio (SLA) con los proveedores de infraestructura para garantizar un soporte rápido y eficiente en caso de fallas.
3. Asignación de recursos adicionales o ajuste de plazos:
   - Mantener un margen de tiempo adicional en el cronograma del proyecto para abordar posibles retrasos relacionados con la infraestructura.
   - Asignar recursos adicionales, como personal técnico especializado, para solucionar rápidamente cualquier problema que pueda surgir.
4. Realización de pruebas y análisis adicionales:
   - Realizar pruebas de carga y rendimiento exhaustivas en la infraestructura para identificar cuellos de botella y problemas de escalabilidad.
   - Realizar análisis de riesgos específicos de la infraestructura y establecer medidas correctivas para abordarlos.

Riesgo: Cambios frecuentes en los requisitos del cliente.

Estrategias y planes de mitigación:

Desarrollo de planes de contingencia:

- Establecer un proceso de gestión de cambios sólido para evaluar y controlar los cambios en los requisitos del cliente.
- Definir claramente los pasos y los criterios para la aprobación de cambios, de modo que los cambios no autorizados se minimicen.

5. Medidas preventivas:
   - Realizar reuniones regulares con el cliente para alinear expectativas y obtener retroalimentación temprana sobre posibles cambios en los requisitos.
   - Documentar de manera exhaustiva los requisitos iniciales y obtener la aprobación formal del cliente antes de comenzar el desarrollo.
6. Asignación de recursos adicionales o ajuste de plazos:

- Mantener un margen de tiempo adicional en el cronograma del proyecto para acomodar posibles cambios en los requisitos.
- Asignar recursos adicionales, como analistas de negocios o consultores, para ayudar en la gestión de los cambios y en la comunicación con el cliente.

7. Realización de pruebas y análisis adicionales:
   - Realizar pruebas de validación de requisitos a medida que se definen para identificar posibles ambigüedades o conflictos.
   - Realizar análisis de impacto de cambios para evaluar las implicaciones de los nuevos requisitos en el alcance, los plazos y los recursos del proyecto.

Comunicación y reporte de riesgos:

En el proyecto, se designará al Gerente de Proyecto como responsable de gestionar la comunicación de los riesgos identificados. Además, se establecerá un equipo de gestión de riesgos compuesto por representantes de diferentes áreas del proyecto.

Para asegurar una comunicación clara y accesible, se utilizarán los siguientes canales de comunicación:

- Reuniones periódicas: Se llevarán a cabo reuniones semanales donde se discutirán los riesgos identificados, se actualizará su estado y se tomarán decisiones sobre las acciones de mitigación. Estas reuniones permitirán la participación de todo el equipo y facilitarán la comunicación directa y la resolución de problemas.
- Correo electrónico: Se utilizará el correo electrónico como un canal de comunicación principal para compartir información detallada sobre los riesgos identificados, sus impactos y las medidas de mitigación propuestas. Además, se utilizará para solicitar actualizaciones y brindar instrucciones claras sobre las acciones a seguir.
- Sistema de gestión de proyectos: Se empleará un sistema de gestión de proyectos que permita documentar y rastrear los riesgos identificados. Este sistema contendrá una sección específica para la comunicación de riesgos, donde se registrarán los riesgos, su descripción, impacto y estado actual. Asimismo, permitirá asignar tareas y dar seguimiento al progreso de las acciones de mitigación.

Para asegurar una comunicación clara y consistente, se establecerá la siguiente estructura de reporte y formatos de documentación:

- Reporte semanal de riesgos: Se elaborará un informe semanal que resuma el estado de los riesgos identificados, las acciones tomadas y los próximos pasos. Este informe contendrá una descripción detallada de cada riesgo, su nivel de impacto y probabilidad, y las acciones de mitigación planificadas.
- Formato de documentación de riesgos: Se utilizará una plantilla estandarizada para documentar cada riesgo identificado. Esta plantilla incluirá campos como la descripción del riesgo, las causas y consecuencias potenciales, la evaluación de probabilidad e impacto, y las estrategias de mitigación propuestas.

Formato y canales de comunicación

1. Formato de reporte de riesgos: El formato de reporte de riesgos estará compuesto por los siguientes elementos:
    - Descripción del riesgo: Se proporcionará una descripción clara y concisa del riesgo identificado, incluyendo las posibles causas y consecuencias asociadas.
    - Impacto potencial: Se evaluará el impacto que el riesgo podría tener en el proyecto y en la calidad del software. Esto puede incluir aspectos como retrasos en el cronograma, aumento de costos, disminución de la satisfacción del cliente, etc.
    - Probabilidad de ocurrencia: Se asignará una estimación de la probabilidad de que el riesgo se materialice. Esto puede ser expresado en términos de baja, media o alta probabilidad, o mediante una escala numérica.
    - Acciones de mitigación: Se detallarán las acciones específicas que se tomarán para mitigar el riesgo identificado. Esto puede incluir medidas preventivas, planes de contingencia, asignación de recursos adicionales, entre otros.
2. Canales de comunicación adecuados: Para reportar los riesgos identificados, se establecerán los siguientes canales de comunicación:
    - Informes escritos: Se elaborarán informes escritos que contengan la información detallada sobre los riesgos identificados. Estos informes podrán ser compartidos a través de correo electrónico o mediante un sistema de gestión de documentos.
    - Presentaciones en reuniones: Se programarán reuniones periódicas donde se presentarán los riesgos identificados y se discutirán las acciones de mitigación propuestas. Estas reuniones brindarán la oportunidad de intercambiar ideas, recibir retroalimentación y tomar decisiones conjuntas.
    - Sistemas de seguimiento de riesgos: Se utilizará un sistema de seguimiento de riesgos, como una herramienta de gestión de proyectos o un software específico, que permita registrar, rastrear y monitorear los riesgos identificados. Este sistema garantizará la visibilidad y el acceso a la información actualizada sobre los riesgos en todo momento.

Actualización y revisión de la gestión de riesgos

1. Periodicidad de revisión y actualización: Se establecerá una periodicidad de revisión y actualización de la gestión de riesgos a lo largo del proyecto. Por ejemplo, se realizará una revisión trimestral para evaluar y actualizar la situación de los riesgos identificados. Esta periodicidad permitirá mantener la gestión de riesgos actualizada y relevante a medida que el proyecto avance.
2. Consideración de cambios en el proyecto, el entorno o las circunstancias: Se considerará la posibilidad de revisar y actualizar la identificación, evaluación y control de riesgos en función de los cambios en el proyecto, el entorno o las circunstancias. Algunos ejemplos de situaciones que pueden requerir una revisión y actualización de la gestión de riesgos son:
    - Cambios en los objetivos del proyecto: Si se producen cambios significativos en los objetivos del proyecto, es necesario evaluar cómo estos cambios pueden influir en los riesgos previamente identificados y si surgen nuevos riesgos.
    - Cambios en el alcance o los requisitos: Si se realizan modificaciones en el alcance o los requisitos del proyecto, es importante evaluar cómo estos cambios pueden afectar los riesgos existentes y si se presentan nuevos riesgos como resultado de los cambios.

- Cambios en el equipo o los recursos: Si hay cambios en el equipo de trabajo o en los recursos asignados al proyecto, es necesario evaluar si estos cambios pueden tener un impacto en la gestión de riesgos y si es necesario ajustar las estrategias de mitigación.
- Cambios en el entorno externo: Si se producen cambios en el entorno externo, como cambios en las regulaciones o avances tecnológicos, es necesario evaluar si estos cambios introducen nuevos riesgos o modifican la evaluación de los riesgos existentes.

15. Glosario

1. SQAP (Software Quality Assurance Plan): Plan de Aseguramiento de la Calidad del Software. Documento que establece las actividades y los procesos para garantizar la calidad del software desarrollado.
2. Ciclo de vida del software: Conjunto de fases o etapas por las que atraviesa el desarrollo de un software, desde su concepción hasta su retirada. Incluye etapas como análisis de requisitos, diseño, implementación, pruebas y mantenimiento.
3. Control de calidad: Conjunto de actividades y procesos que se llevan a cabo para asegurar que el software cumple con los estándares de calidad establecidos.
4. Métricas de calidad: Indicadores cuantitativos utilizados para medir y evaluar la calidad del software. Ejemplos de métricas de calidad incluyen la tasa de defectos, la cobertura de pruebas y el tiempo promedio de respuesta.
5. Revisión de código: Proceso de examinar y evaluar el código fuente del software para identificar posibles problemas, errores o violaciones de estándares de codificación.
6. Prueba de aceptación: Pruebas realizadas para verificar si el software cumple con los requisitos del cliente y si está listo para ser entregado y utilizado en el entorno de producción.
7. No conformidad: Situación en la que el software o los procesos no cumplen con los estándares o requisitos establecidos en el SQAP. Requiere acciones correctivas para solucionar la no conformidad.
8. Auditoría de calidad: Proceso de evaluación independiente para determinar si los procesos y productos del software cumplen con los estándares de calidad establecidos.
9. Acción preventiva: Medida tomada para evitar la aparición de problemas o desviaciones en el desarrollo del software, con el objetivo de mejorar la calidad y evitar riesgos potenciales.
10. Plan de pruebas: Documento que describe las estrategias, técnicas y recursos utilizados para realizar pruebas en el software y garantizar su funcionamiento correcto y la detección de posibles fallos.

15.1 Acronimos

Los acrónimos encontrados en este documento son los siguientes:

ADR	architecture design review
DDR	detailed design review
SCM	software configuration management
SCMP	software configuration management plan
SCMPR	software configuration management plan review
SDD	software design description
SQA	software quality assurance
SQAP	software quality assurance plan
SRD	software requirements description
SSR	software specifications review
UDR	user documentation review

16. Procedimiento e historial de cambio de SQAP

Procedimiento para Modificar el SQAP:

1. Cualquier cambio propuesto al SQAP deberá ser presentado por escrito al Comité de Control de Cambios (CCB, por sus siglas en inglés) designado.
2. El CCB revisará los cambios propuestos y evaluará su impacto en el SQAP y los objetivos del proyecto.
3. Basándose en la evaluación, el CCB aprobará o rechazará los cambios propuestos.
4. Si se aprueba, el CCB documentará los cambios aprobados y actualizará el SQAP en consecuencia.
5. El SQAP actualizado deberá ser comunicado a todos los miembros del equipo y partes interesadas relevantes.

Historial de Modificaciones:

El SQAP mantendrá un historial de todas las modificaciones realizadas, que incluirá lo siguiente:

- Fecha de la modificación.
- Descripción detallada de los cambios realizados.
- Justificación o razón detrás de la modificación.
- Autor o responsable de la modificación.

- Estado actualizado del SQAP después de la modificación.