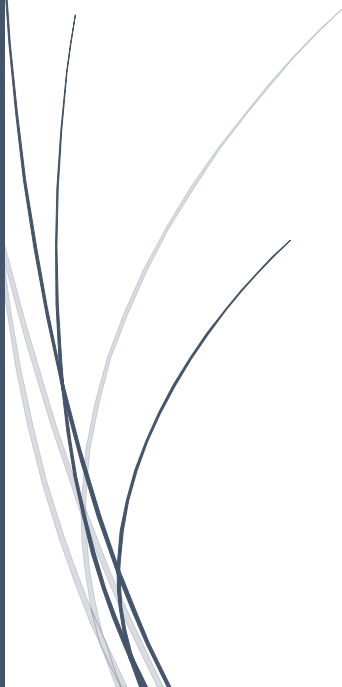
A dark blue vertical bar on the left side of the page. A blue arrow points to the right from this bar, containing the date.

11/26/2024

MODELO DE ANÁLISIS

Ingeniería de Software II

Several thin, curved lines in shades of blue and grey that originate from the bottom left and curve upwards and to the right.

Saúl Cervantes Candia – 177927, Ana Karen
Cuenca Esquivel – 177932, Carlos Mauricio
Rosales Rodríguez - 177691

UNIVERSIDAD POLITÉCNICA DE SAN LUIS POTOSÍ

PLAN DE SQA Y ESTÁNDAR IEEE – 730

PLAN DE ASEGURAMIENTO DE LA CALIDAD

Fecha: 26/11/2024

Versión: 1

Responsables: Saúl Cervantes Candia – 177927, Ana Karen Cuenca Esquivel – 177932, Carlos Mauricio Rosales Rodríguez- 177691

ÍNDICE

PROPÓSITO:	3
ACRÓNIMOS Y ABREVIATURAS:	5
REFERENCIAS:	5
GESTIÓN:	6
Organización:	7
Tareas:	9
Elaboración del Plan de SQA:	10
Identificar las propiedades de calidad:	10
EVALUAR LA CALIDAD DE LOS PRODUCTOS	11
REVISAR EL AJUSTE AL PROCESO	11
REALIZAR REVISIÓN TÉCNICA FORMAL	12
EVALUAR Y AJUSTAR EL PLAN DE SQA	12
REALIZAR LA EVALUACIÓN FINAL DE SQA	12
Responsabilidades:	13
DOCUMENTACIÓN:	15

Documentación mínima requerida:	15
Especificación de Requerimientos IEEE-830	15
Diseño del Sistema IEEE-1471	16
Plan de Verificación y Validación IEEE-1012	16
Reportes de Verificación	16
Documentación de Usuario	17
Plan de Configuración IEEE-828	17
Plan de Gestión del Proyecto IEEE-1058.1	18
OTRA DOCUMENTACIÓN	18
ESTÁNDARES Y MÉTRICAS: 19	
Estándares para documentación:	19
Métricas	20
REVISIONES: 21	
Descripción:	21
Evaluación de la calidad de los productos:	21
Revisar el ajuste al proceso:	22
Revisión Técnica Formal (RTF):	22
Requerimientos Mínimos:	23
Especificación de Requerimientos	23
Modelo de Diseño y Descripción de la Arquitectura	23
Plan de Verificación y Validación (v&v)	23
Plan de Gestión del Proyecto	24
Plan de Gestión de Configuración	24
Diseño vs. Especificación de Requerimientos	24
Implementación vs. Diseño	24
Verificación vs. Especificación de Requerimientos	25
Casos de Prueba y Resultados de Pruebas	25

Código Fuente25

Pruebas de Integración y Aceptación.....25

Documentación del Usuario.....26

Seguimiento de Defectos y Desviaciones.....26

Agenda: 26

Fase I – Inicial.....26

Fase II – Elaboración27

Fase III – Construcción27

Fase IV – Transición.....28

TESTEO: 28

REPORTE DE PROBLEMAS Y ACCIONES CORRECTIVAS: 30

HERRAMIENTAS, TÉCNICAS Y METODOLOGÍAS: 31

GESTIÓN DE CONFIGURACIÓN (CONTROL DE CÓDIGO Y CONTROL DE MEDIOS) 34

GESTIÓN DE RIESGOS: 35

PROPÓSITO:

El propósito de este plan es especificar las actividades que se realizarán para garantizar la calidad del software a construir en el proyecto [nombre del proyecto, e.g., Sistema de Administración de Waffles]. Este documento detalla los productos que se someterán a revisión, los estándares, normas y métodos que se aplicarán, así como los procedimientos para verificar que el desarrollo del software cumple con lo establecido en el modelo de ciclo de vida del proyecto. Además, describe cómo se identificarán, reportarán y gestionarán los defectos encontrados durante el proceso.

DESCRIPCIÓN DEL PRODUCTO

El producto desarrollado es un sistema para [gestionar un negocio de waffles], diseñado para cubrir las siguientes funcionalidades:

- Gestión de ingredientes.
- Registro de productos y personalización de waffles.

- Administración de pedidos y ventas.
- Reportes y análisis de datos, incluyendo el seguimiento de ventas.

El sistema está construido utilizando React en el frontend, con un backend desarrollado en Node.js y Sequelize como ORM para la base de datos. El objetivo es ofrecer una herramienta eficiente, intuitiva y escalable para apoyar la operación del negocio.

OBJETIVOS Y CRITERIOS DE CALIDAD

Los principales objetivos de calidad incluyen:

- **Confiable:** Asegurar que el sistema sea estable y funcione sin interrupciones.
- **Usabilidad:** Garantizar una experiencia de usuario intuitiva y accesible para todo tipo de usuarios.
- **Escalabilidad:** Permitir la incorporación de nuevas funcionalidades sin afectar el rendimiento actual.
- **Mantenibilidad:** Diseñar el sistema de forma modular para facilitar actualizaciones y correcciones.
- **Seguridad:** Proteger los datos del negocio y los usuarios ante accesos no autorizados.

USO DEL PLAN DE SQA POR PARTE DEL EQUIPO Y EL DIRECTOR DEL PROYECTO

Integrantes del equipo: Utilizarán este plan como referencia para garantizar que cada actividad del desarrollo cumpla con los estándares de calidad definidos. Se encargan de identificar y documentar defectos en los productos asignados y participar en revisiones técnicas.

Director del Proyecto: Supervisará la implementación del plan de SQA y revisará los informes de calidad para tomar decisiones informadas. Además, se asegurará de que los recursos necesarios para cumplir con el plan estén disponibles.

MODELO DE PROCESO Y CICLO DE VIDA

El proyecto utiliza el modelo de proceso scrum. Las etapas del ciclo de vida cubiertas por este plan incluyen:

- **Requisitos:** Validación de especificaciones y criterios funcionales.
- **Diseño:** Revisión del diseño de la arquitectura y componentes.
- **Implementación:** Verificación de calidad del código fuente y pruebas unitarias.
- **Pruebas:** Pruebas de integración, funcionalidad y aceptación.
- **Mantenimiento:** Gestión de defectos y actualizaciones post-despliegue.

LÍNEAS DE TRABAJO CONTEMPLADAS EN EL PLAN DE SQA

- Establecimiento de métricas de calidad.
- Revisión de artefactos como requisitos, diseño y código.
- Ejecución de pruebas automatizadas y manuales.

- Gestión y seguimiento de defectos.
- Auditorías de cumplimiento de estándares.

ACRÓNIMOS Y ABREVIATURAS:

A continuación, se definen los acrónimos y abreviaturas utilizados en este documento para facilitar su comprensión:

- SQA: Aseguramiento de la Calidad del Software (Software Quality Assurance).
- SCM: Gestión de Configuración del Software (Software Configuration Management).
- GP: Gestión del Proyecto.
- IEEE: Instituto de Ingenieros Eléctricos y Electrónicos (Institute of Electrical and Electronics Engineers).
- API: Interfaz de Programación de Aplicaciones (Application Programming Interface).
- UI: Interfaz de Usuario (User Interface).
- DB: Base de Datos.
- CRUD: Crear, Leer, Actualizar y Eliminar (Create, Read, Update, Delete).
- UX: Experiencia de Usuario (User Experience).
- React: Biblioteca de JavaScript para la creación de interfaces de usuario.
- React Native: Framework de desarrollo para aplicaciones móviles multiplataforma basado en React.
- Git: Sistema de control de versiones distribuido.
- Scrum: Metodología ágil para la gestión y desarrollo de proyectos de software.
- JSON: Notación de Objetos de JavaScript (JavaScript Object Notation), utilizada para intercambio de datos.
- CI/CD: Integración y Entrega Continua (Continuous Integration/Continuous Delivery).
- ORM: Mapeo Objeto-Relacional (Object-Relational Mapping), herramienta para interactuar con bases de datos desde lenguajes orientados a objetos.

REFERENCIAS:

A continuación, se enumeran los documentos, estándares y recursos utilizados para la elaboración de este plan de SQA. Estas referencias sirven como base para las actividades descritas y para guiar el desarrollo y aseguramiento de la calidad del software:

- IEEE Std 730-1998: IEEE Standard for Software Quality Assurance Plans.
- IEEE Std 730.1-1995: IEEE Guide for Software Quality Assurance Planning.
- Modelo de proceso: [Dirección web del modelo de proceso, si aplica].
- IS-1 (2001): Proyecto de Ingeniería de Software.
- IS-2 (2001): Modelo de Calidad.
- Guía de Scrum: Guía oficial de Scrum. Disponible en: <https://scrumguides.org>.
- Documentación de React: Guía oficial de React. Disponible en: <https://reactjs.org>.
- Documentación de React Native: Guía oficial de React Native. Disponible en: <https://reactnative.dev>.
- Git Documentation: Sistema de control de versiones. Disponible en: <https://git-scm.com>.
- ISO/IEC 9126: Software engineering – Product quality model.

GESTIÓN:

En esta sección se especifican los elementos organizativos que impactan la calidad del software, incluyendo la estructura de gestión de calidad, las tareas cubiertas por este plan y las responsabilidades asociadas a cada tarea.

4.1 AUTORIDAD Y RESPONSABILIDAD DE LA CALIDAD DEL SOFTWARE

La gestión de la calidad del software estará bajo la supervisión de los siguientes roles:

Director del Proyecto:

- Responsable de garantizar que el equipo cumpla con los estándares de calidad establecidos.
- Autoridad final para aprobar cambios en el plan de SQA.
- Supervisa las revisiones de calidad y recibe los reportes de defectos.

Equipo de Desarrollo:

- Implementa las actividades descritas en el plan de SQA.
- Identifica, documenta y corrige los defectos en el software.
- Realiza pruebas unitarias y de integración.

Coordinador de SQA:

- Monitorea las actividades de calidad en todas las fases del proyecto.
- Asegura que los entregables cumplan con los criterios de calidad.
- Informa al director del Proyecto sobre los resultados de las auditorías de calidad.

Equipo de Pruebas:

- Diseña, ejecuta y documenta los casos de prueba.
- Realiza pruebas funcionales, de integración y de aceptación.
- Reporta los defectos encontrados al equipo de desarrollo.

4.2 TAREAS CUBIERTAS POR ESTE PLAN

El plan de SQA cubre las siguientes tareas:

- 1. Establecimiento de estándares y métricas de calidad.
- 2. Revisión y validación de requisitos.
- 3. Revisión del diseño del sistema.
- 4. Inspección de código fuente.
- 5. Ejecución de pruebas automatizadas y manuales.
- 6. Revisión de documentación técnica y de usuario.
- 7. Gestión de defectos y auditorías de calidad.
- 8. Monitoreo del cumplimiento del modelo de proceso seleccionado.

4.3 RESPONSABILIDADES POR TAREA

A continuación, se asignan las responsabilidades específicas para cada tarea cubierta en el plan:

Tarea	Responsable(s)	Descripción
Establecimiento de estándares	Director del Proyecto	Define los estándares de calidad y métricas a seguir durante el desarrollo del proyecto.
Validación de requisitos	Equipo de Desarrollo	Realiza revisiones para asegurar que los requisitos sean completos, claros y verificables.
Revisión del diseño	Coordinador de SQA	Evalúa la arquitectura y el diseño técnico para asegurar que cumplen con los objetivos de calidad.
Inspección de código fuente	Equipo de Desarrollo	Realiza revisiones para identificar defectos y asegurar el cumplimiento de estándares de codificación.
Ejecución de pruebas	Equipo de Pruebas	Diseña y ejecuta casos de prueba para verificar la funcionalidad y estabilidad del sistema.
Gestión de defectos	Equipo de Desarrollo y SQA	Documenta, clasifica y resuelve defectos detectados durante el ciclo de vida del proyecto.
Auditorías de calidad	Coordinador de SQA	Monitorea y documenta el cumplimiento de las actividades del plan de SQA y asegura la adherencia al modelo de proceso.
Revisión de documentación	Equipo de Desarrollo	Garantiza que la documentación sea clara, precisa y cumpla con los estándares definidos.

ORGANIZACIÓN:

La gestión de calidad en este proyecto es responsabilidad del encargado del área de calidad, quien asegura que los procesos establecidos sean implementados correctamente y que los productos del proyecto cumplan con los criterios de calidad definidos en este plan. La gestión de calidad opera en paralelo a las disciplinas básicas del ciclo de vida del desarrollo de software, brindando soporte a través de la Gestión del Proyecto (GP) y la Gestión de la Configuración del Software (SCM).

5.1 DEPENDENCIAS DEL ÁREA DE SQA

La siguiente tabla describe las dependencias del área de SQA con respecto a las distintas líneas de trabajo del proyecto:

Línea de trabajo	Objetivo de Calidad	Rol Responsable	Persona Responsable
Requerimientos	Validar que los requisitos sean claros y completos	Analista de Requerimientos	Ana Cuenca
Análisis	Asegurar que las especificaciones sean viables	Coordinador de SQA	Saul Cervantes
Diseño	Garantizar la adherencia a estándares técnicos	Arquitecto de Software	Mauricio Rosales
Implementación	Cumplir con las prácticas de codificación segura	Desarrollador Líder	Ana Cuenca
Verificación	Confirmar que el software cumple requisitos	Equipo de Pruebas	Saul Cervantes
Implantación	Asegurar una transición exitosa al entorno real	Responsable de Implantación	Mauricio Rosales

5.2 AUTORIDADES PARA LA APROBACIÓN DEL PLAN DE SQA

La aprobación del Plan de SQA será realizada por:

Director del Proyecto: Ana Cuenca

Coordinador de SQA: Saul Cervantes

5.3 AUTORIDAD PARA LA LIBERACIÓN DEL PRODUCTO

La liberación final del producto es responsabilidad de:

- Director del Proyecto, quien validará que el producto cumpla con los criterios de calidad establecidos.
- Responsable de Implantación, quien asegura que el producto esté listo para su uso en el entorno de destino.

5.4 LÍNEAS DE COMUNICACIÓN

Para garantizar una gestión de calidad eficiente, se establecen las siguientes líneas de comunicación:

Con el Director de Proyecto:

- Reportes semanales sobre el estado de las actividades de calidad.
- Comunicación inmediata en caso de encontrar defectos críticos.

Con el Equipo de Desarrollo:

- Revisiones periódicas de código y diseño técnico.
- Feedback sobre defectos detectados durante las auditorías.

Con el Equipo de Pruebas:

- Coordinación para la planificación y ejecución de casos de prueba.
- Discusión de resultados de pruebas y defectos encontrados.

Con el Cliente:

- Informes periódicos sobre el cumplimiento de los estándares de calidad acordados.

TAREAS:

En esta sección se describen las tareas de calidad a realizar, indicando el entregable asociado y la influencia de cada tarea en la calidad del producto.

Commented [AC1]: pendiente

Actividad	Entregable Asociado
Realizar el Plan de SQA	Plan de SQA
Identificar las propiedades de Calidad	Propiedades de Calidad
Evaluar la calidad de los productos	Informe de revisión de SQA
Revisar el ajuste al proceso	Informe de revisión de SQA
Realizar Revisión Técnica Formal	Informe de Revisión Técnica Formal
Evaluar y ajustar el Plan de SQA	Documento de Evaluación y Ajustes al Plan de SQA
Revisar la entrega semanal	Entrega semanal de SQA
Realizar evaluación final de SQA	Evaluación final de SQA
Reuniones de Apoyo a la calidad	No Aplica

ELABORACIÓN DEL PLAN DE SQA:

Descripción de la Actividad:

La elaboración del Plan de SQA consiste en la planificación detallada de todas las actividades relacionadas con el aseguramiento de la calidad durante el ciclo de vida del proyecto. Esta actividad tiene como objetivo garantizar que todas las fases del desarrollo del software cumplan con los estándares de calidad definidos previamente. El Plan de SQA proporciona una guía clara sobre cómo se gestionarán los aspectos relacionados con la calidad, estableciendo los métodos, procedimientos y métricas necesarios para el control de calidad.

Entregable Asociado:

Plan de SQA

Influencia de la Actividad en la Calidad del Producto:

El Plan de SQA es fundamental para la calidad del producto, ya que establece las bases para la implementación de controles de calidad a lo largo de todo el proyecto. Al definir los criterios y procedimientos para la evaluación y control de calidad, se aseguran procesos consistentes y se minimiza el riesgo de defectos en el producto final.

IDENTIFICAR LAS PROPIEDADES DE CALIDAD:

Descripción de la Actividad:

En este apartado, se identifican y describen las métricas y propiedades de calidad que serán evaluadas y utilizadas para asegurar la calidad del producto. Basado en el estándar ISO 9126, se definen seis propiedades fundamentales de calidad del software: funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad. Además, se deben considerar propiedades adicionales específicas al proyecto, como la seguridad, el rendimiento o la escalabilidad, dependiendo de los requisitos y características del software.

Entregable Asociado:

Propiedades de Calidad Definidas

Influencia de la Actividad en la Calidad del Producto:

La identificación y evaluación de las propiedades de calidad garantiza que el software cumpla con las expectativas de los usuarios y los requisitos del cliente. A través de estas métricas, es posible medir el

rendimiento y la estabilidad del software en diferentes aspectos clave, asegurando que el producto final sea confiable, fácil de usar y eficiente. Esto facilita la detección temprana de posibles problemas y permite tomar acciones correctivas a tiempo.

EVALUAR LA CALIDAD DE LOS PRODUCTOS

Descripción de la Actividad:

Consiste en analizar y revisar los productos de trabajo generados durante el desarrollo del software, incluyendo código, documentación, diseños y entregables intermedios, para garantizar que cumplen con los estándares de calidad definidos. Este proceso incluye la ejecución de pruebas, revisiones de diseño y auditorías que permitan identificar defectos o desviaciones.

Entregable Asociado:

Informe de Revisión de SQA

Influencia de la Actividad en la Calidad del Producto:

Permite identificar tempranamente errores y defectos, reduciendo el costo de corrección en fases posteriores. Al garantizar que los productos cumplen con los estándares, se asegura que el producto final sea de alta calidad y cumpla con los requisitos del cliente.

REVISAR EL AJUSTE AL PROCESO

Descripción de la Actividad:

Se analiza el cumplimiento del proceso de desarrollo con las políticas, estándares y procedimientos establecidos en el Plan de SQA. Esta actividad incluye auditorías y evaluaciones periódicas para detectar desviaciones y proponer mejoras al proceso.

Entregable Asociado:

Informe de Revisión de SQA

Influencia de la Actividad en la Calidad del Producto:

El monitoreo constante del proceso asegura que las actividades se realicen de manera eficiente y conforme a los estándares, minimizando errores derivados de prácticas inadecuadas y garantizando la calidad del producto final.

REALIZAR REVISIÓN TÉCNICA FORMAL

Descripción de la Actividad:

Consiste en realizar reuniones estructuradas con los responsables de las tareas para evaluar artefactos técnicos clave, como diseños, arquitecturas o código fuente, identificando defectos y áreas de mejora.

Entregable Asociado:

Informe de Revisión Técnica Formal

Influencia de la Actividad en la Calidad del Producto:

Ayuda a identificar problemas críticos en etapas tempranas del desarrollo, reduciendo costos de corrección y mejorando la calidad técnica del producto. Además, fomenta la colaboración entre los equipos y mejora la comprensión del sistema.

EVALUAR Y AJUSTAR EL PLAN DE SQA

Descripción de la Actividad:

Se realiza una evaluación periódica del Plan de SQA para determinar su efectividad y relevancia durante el ciclo de vida del proyecto. Si se detectan áreas de mejora, se ajustan los procedimientos y métricas según las necesidades del proyecto.

Entregable Asociado:

Documento de Evaluación y Ajustes al Plan de SQA

Influencia de la Actividad en la Calidad del Producto:

Mantener actualizado el Plan de SQA asegura que las actividades de calidad se alineen con los objetivos del proyecto y los requisitos del cliente. Esto contribuye a un enfoque más eficiente y efectivo para garantizar la calidad del producto.

REALIZAR LA EVALUACIÓN FINAL DE SQA

Descripción de la Actividad:

En esta actividad se realiza un análisis global del cumplimiento de los objetivos de calidad establecidos en el Plan de SQA al término del proyecto. Esto incluye la revisión de métricas, resultados de pruebas y auditorías realizadas.

Entregable Asociado:

Evaluación Final de SQA

Influencia de la Actividad en la Calidad del Producto:

La evaluación final valida que los entregables cumplen con los estándares de calidad definidos y proporciona una visión clara del éxito del proceso de SQA, asegurando la satisfacción del cliente y facilitando la entrega de un producto confiable.

MÉTRICAS CONSIDERADAS EN EL PROYECTO:

1. **Funcionalidad:**
 - **Métrica:** Porcentaje de funcionalidades completadas según los requisitos.
 - **Objetivo:** Asegurar que todas las funcionalidades necesarias están implementadas y operativas.
2. **Fiabilidad:**
 - **Métrica:** Número de fallos por cada mil horas de uso.
 - **Objetivo:** Evaluar la estabilidad y resistencia del software bajo condiciones normales de uso.
3. **Usabilidad:**
 - **Métrica:** Tiempo promedio para que un usuario realice una tarea específica.
 - **Objetivo:** Garantizar que la interfaz sea intuitiva y fácil de usar.
4. **Eficiencia:**
 - **Métrica:** Tiempo de respuesta promedio de la aplicación.
 - **Objetivo:** Medir la rapidez con la que el sistema responde a las solicitudes del usuario.
5. **Mantenibilidad:**
 - **Métrica:** Tiempo necesario para modificar el código para resolver un error.
 - **Objetivo:** Asegurar que el sistema sea fácil de mantener y actualizar.
6. **Portabilidad:**
 - **Métrica:** Número de plataformas en las que el software puede ser ejecutado sin modificaciones.
 - **Objetivo:** Garantizar que el software pueda ser utilizado en diferentes entornos de hardware y software.
7. **Seguridad** (adicional para el proyecto):
 - **Métrica:** Número de vulnerabilidades críticas detectadas.
 - **Objetivo:** Asegurar que el software sea seguro y resistente a ataques.

RESPONSABILIDADES:

El responsable de SQA debe realizar las actividades y entregables mencionados en la sección anterior. Además, se encargará de revisar los productos relevantes para la calidad del software y del proceso, y de implementar acciones correctivas cuando se detecten defectos. A continuación, se identifican los productos a revisar, junto con el rol responsable y la persona encargada de las acciones correctivas:

Producto	Rol responsable	Responsable
Documento de Requerimientos IEEE-830	Analista de Requerimientos	Ana Cuenca
Modelo de Casos de Uso	Analista de Sistemas	Saul Cervantes
Alcance del Sistema	Gerente de Proyecto	Mauricio Rosales
Descripción de la Arquitectura IEEE-1471	Arquitecto de Software	Ana Cuenca
Modelo de Diseño	Diseñador de Software	Saul Cervantes
Modelo de Datos	DBA (Administrador de Base de Datos)	Mauricio Rosales
Estándar de Implementación	Líder Técnico	Ana Cuenca
Estándar de documentación técnica	Documentador Técnico	Saul Cervantes
Documento de Estimaciones	Gerente de Proyecto	Ana Cuenca
Documento de Riesgos	Gerente de Proyecto	Ana Cuenca
Plan de Gestión del Proyecto IEEE-1058.1	Gerente de Proyecto	Ana Cuenca
Plan de Verificación y Validación IEEE-1012 V&V	Líder de Calidad	Mauricio Rosales
Reporte de pruebas unitarias, de integración y del Sistema IEEE-829	Ingeniero de Pruebas	Ana Cuenca
Plan de Implantación	Líder de Implantación	Saul Cervantes

Producto	Rol responsable	Responsable
Estándar de Documentación de Usuario	Documentador de Usuario	Saul Cervantes
Documentación de Usuario	Documentador de Usuario	Saul Cervantes
Plan de Gestión de Configuración IEEE-828	Líder de Gestión de Configuración	Mauricio Rosales

DOCUMENTACIÓN:

El objetivo de esta sección es especificar los documentos que dirigen el desarrollo del proyecto y que deberán ser revisados como parte de las actividades de aseguramiento de la calidad. Para cada documento se indica el objetivo del documento, la plantilla, norma y/o estándar que se usa para elaborar el documento y el contenido mínimo que debe tener dicho documento.

DOCUMENTACIÓN MÍNIMA REQUERIDA:

ESPECIFICACIÓN DE REQUERIMIENTOS IEEE-830

- Objetivo del documento: Definir de manera detallada los requisitos funcionales y no funcionales del sistema, los cuales sirven como base para el diseño, la implementación y la validación del sistema.
- Plantilla, norma y/o estándar: IEEE 830 - "IEEE Recommended Practice for Software Requirements Specifications"
- Contenido mínimo:
 - Introducción: Propósito, alcance, definiciones y acrónimos
 - Descripción general del sistema
 - Requisitos específicos (funcionales y no funcionales)
 - Restricciones del sistema
 - Interfaces del sistema (hardware, software, comunicación)
 - Aceptación de los requisitos

DISEÑO DEL SISTEMA IEEE-1471

- Objetivo del documento: Describir la arquitectura del sistema, sus componentes y cómo estos interactúan entre sí para cumplir con los requisitos establecidos.
- Plantilla, norma y/o estándar: IEEE 1471 - "IEEE Recommended Practice for Architectural Description of Software-Intensive Systems"
- Contenido mínimo:
 - Descripción de los componentes del sistema
 - Interfaces y relaciones entre los componentes
 - Diagrama de la arquitectura del sistema
 - Consideraciones sobre seguridad, escalabilidad y rendimiento
 - Estándares y herramientas utilizados para la arquitectura

PLAN DE VERIFICACIÓN Y VALIDACIÓN IEEE-1012

- Objetivo del documento: Establecer el enfoque y los procedimientos para asegurar que el software cumple con los requisitos especificados, y validar que el software funciona de acuerdo con las expectativas del cliente.
- Plantilla, norma y/o estándar: IEEE 1012 - "IEEE Standard for Software Verification and Validation"
- Contenido mínimo:
 - Estrategia y enfoque de verificación y validación
 - Plan de pruebas (unitarias, integración, sistema, aceptación)
 - Criterios de aceptación
 - Herramientas y recursos necesarios para las pruebas
 - Cronograma y responsables

REPORTES DE VERIFICACIÓN

- Objetivo del documento: Documentar los resultados obtenidos durante las actividades de verificación, destacando los problemas encontrados y las acciones correctivas tomadas.
- Plantilla, norma y/o estándar: Basado en IEEE 829 - "IEEE Standard for Software Test Documentation"
- Contenido mínimo:
 - Informe de pruebas ejecutadas
 - Resultados de las pruebas
 - Análisis de errores encontrados
 - Acciones correctivas y acciones pendientes

DOCUMENTACIÓN DE USUARIO

- Objetivo del documento: Proporcionar a los usuarios finales la información necesaria para utilizar el sistema, incluyendo instrucciones y guías de uso.
- Plantilla, norma y/o estándar: ISO/IEC 26514 - "Software and Systems Engineering — Documentation for Product Usage"
- Contenido mínimo:
 - Guía de instalación y configuración
 - Manual de usuario con ejemplos de uso
 - Solución de problemas comunes
 - Detalles sobre la interfaz de usuario y la interacción con el sistema

PLAN DE CONFIGURACIÓN IEEE-828

- Objetivo del documento: Establecer las directrices y procedimientos para la gestión de la configuración del software, asegurando que las versiones y configuraciones sean controladas y trazables.
- Plantilla, norma y/o estándar: IEEE 828 - "IEEE Standard for Software Configuration Management Plans"
- Contenido mínimo:
 - Procedimientos de control de versiones

- Herramientas y métodos utilizados para la gestión de configuraciones
- Control de cambios y gestión de defectos
- Planificación y auditoría de la configuración

PLAN DE GESTIÓN DEL PROYECTO IEEE-1058.1

- **Objetivo del documento:** Definir cómo se gestionará el proyecto de software, incluyendo los recursos, cronograma, riesgos, y los métodos de control y seguimiento del progreso.
- **Plantilla, norma y/o estándar:** IEEE 1058.1 - "IEEE Standard for Software Project Management Plans"
- **Contenido mínimo:**
 - Introducción y objetivos del proyecto
 - Estructura del proyecto y asignación de roles
 - Cronograma y estimaciones de esfuerzo
 - Estrategias de gestión de riesgos
 - Plan de control de calidad
 - Plan de comunicación y coordinación

OTRA DOCUMENTACIÓN

1. **Informe de Auditoría de Calidad**
- **Objetivo del documento:** Realizar auditorías periódicas para verificar que los procedimientos y estándares de calidad se están cumpliendo, asegurando que el proceso de desarrollo esté alineado con los objetivos del proyecto.
 - **Plantilla, norma y/o estándar:** Basado en estándares de auditoría interna y calidad de software
 - **Contenido mínimo:**
 - Resumen de la auditoría
 - Hallazgos y observaciones
 - Acciones correctivas recomendadas

2. Informe de Riesgos

- **Objetivo del documento:** Identificar, evaluar y mitigar los riesgos potenciales que podrían afectar el éxito del proyecto.
- **Plantilla, norma y/o estándar:** ISO 31000 - "Risk Management"
 - **Contenido mínimo:**
 - Identificación de riesgos
 - Evaluación de la probabilidad e impacto
 - Estrategias de mitigación

ESTÁNDARES Y MÉTRICAS:

ESTÁNDARES PARA DOCUMENTACIÓN:

A continuación, se describen los estándares utilizados para los diferentes entregables dentro del proyecto. Para cada entregable, se especifica el estándar o plantilla que se empleará para su creación y revisión.

Entregable	Estándar
Especificación de Requerimientos	IEEE 830 - "IEEE Recommended Practice for Software Requirements Specifications"
Diseño del Sistema	IEEE 1471 - "IEEE Recommended Practice for Architectural Description of Software-Intensive Systems"
Plan de Verificación y Validación	IEEE 1012 - "IEEE Standard for Software Verification and Validation"
Reportes de Verificación	IEEE 829 - "IEEE Standard for Software Test Documentation"
Documentación de Usuario	ISO/IEC 26514 - "Software and Systems Engineering — Documentation for Product Usage"
Plan de Configuración	IEEE 828 - "IEEE Standard for Software Configuration Management Plans"

Plan de Gestión del Proyecto	IEEE 1058.1 - "IEEE Standard for Software Project Management Plans"
Informe de Auditoría de Calidad	Basado en estándares de auditoría interna y calidad de software
Informe de Riesgos	ISO 31000 - "Risk Management"

MÉTRICAS

Las métricas de calidad son esenciales para evaluar tanto el producto como el proceso a lo largo del ciclo de vida del proyecto. A continuación, se describen las propiedades de calidad identificadas para el proyecto y las métricas que se utilizarán para medir dichas propiedades, basadas en el estándar ISO/IEC 9126.

Propiedades de Calidad Identificadas:

- 1. **Funcionalidad**
 - Métricas:
 - COBERTURA DE REQUISITOS (porcentaje de requisitos cubiertos en el diseño y la implementación).
 - PRECISIÓN (porcentaje de respuestas correctas en comparación con las expectativas del cliente).
 - ADECUACIÓN (evaluación de si las funciones del sistema cumplen con los requisitos del cliente).
- 2. **Fiabilidad**
 - Métricas:
 - TASA DE DEFECTOS (número de defectos encontrados por unidad de tiempo o línea de código).
 - TIEMPO ENTRE FALLOS (promedio de tiempo entre la aparición de defectos).
 - DISPONIBILIDAD (porcentaje de tiempo en que el sistema está funcionando según lo esperado).
- 3. **Usabilidad**
 - Métricas:
 - FACILIDAD DE APRENDIZAJE (tiempo promedio necesario para que los nuevos usuarios dominen las funcionalidades clave del sistema).
 - SATISFACCIÓN DEL USUARIO (medida mediante encuestas o análisis de comentarios de usuarios).
- 4. **Eficiencia**
 - Métricas:
 - TIEMPO DE RESPUESTA (tiempo promedio que tarda el sistema en responder a las solicitudes del usuario).
 - USO DE RECURSOS (cantidad de CPU, memoria, etc., utilizada por el sistema en comparación con el rendimiento esperado).
- 5. **Mantenibilidad**
 - Métricas:

- **COMPLEJIDAD CICLOMÁTICA** (número de caminos independientes en el código fuente, indicador de la complejidad del código).
- **TIEMPO DE RESOLUCIÓN DE PROBLEMAS** (tiempo necesario para identificar, corregir y validar los defectos).

6. Portabilidad

- **Métricas:**

- **NÚMERO DE PLATAFORMAS SOPORTADAS** (cantidad de sistemas operativos o dispositivos en los que el software puede ejecutarse sin modificaciones).
- **ESFUERZO DE ADAPTACIÓN** (tiempo y recursos necesarios para adaptar el sistema a nuevas plataformas).

7. Seguridad

- **Métricas:**

- **NÚMERO DE VULNERABILIDADES** (cantidad de vulnerabilidades de seguridad detectadas durante las pruebas de seguridad).
- **TIEMPO DE EXPOSICIÓN A AMENAZAS** (duración durante la cual una vulnerabilidad no fue corregida, poniendo en riesgo el sistema).

MÉTRICAS DEL PROCESO:

Estas métricas ayudan a evaluar la calidad del proceso de desarrollo y la efectividad de las actividades de aseguramiento de calidad:

1. Tasa de Defectos en el Proceso

- **Métrica:** Porcentaje de defectos encontrados durante las fases de desarrollo y las fases de prueba respecto al total de defectos detectados.

2. Eficiencia del Proceso de Desarrollo

- **Métrica:** Relación entre el esfuerzo de desarrollo (en horas) y el número de funcionalidades implementadas y entregadas.

3. Cumplimiento de los Tiempos de Entrega

- **Métrica:** Porcentaje de entregas realizadas a tiempo según el cronograma definido en el plan de gestión del proyecto.

4. Eficiencia en la Gestión de Cambios

- **Métrica:** Porcentaje de cambios solicitados que fueron gestionados dentro del tiempo y presupuesto establecidos.

REVISIONES:

DESCRIPCIÓN:

Esta sección define los tres tipos de revisiones utilizados para garantizar la calidad en el desarrollo del software, sus objetivos, mecanismos y las salidas esperadas.

EVALUACIÓN DE LA CALIDAD DE LOS PRODUCTOS:

Commented [AC2]: Hasta aquí me quede

Commented [CR3R2]: @Cuenca Esquivel Ana Karen Ya quedó esto

Objetivo: Revisar los productos clave para garantizar que cumplan con los estándares y objetivos de calidad establecidos. Identificar y reportar desviaciones para su corrección.

Mecanismo:

1. Revisar los productos utilizando listas de verificación específicas (checklists) que aseguren el cumplimiento de los estándares definidos en la sección 6 y los objetivos de calidad.
2. Se debe confirmar que no existan correcciones pendientes de revisiones previas. Si se encuentran, deberán incluirse en esta revisión.
3. Documentar todas las desviaciones detectadas, hacer un seguimiento de su resolución y verificar su corrección en revisiones posteriores.
4. Generar un Informe de Revisión de SQA, donde se detallan desviaciones o defectos encontrados y se especifiquen las acciones correctivas requeridas. Este informe será entregado a los responsables del producto para asegurar su comprensión y cumplimiento.

REVISAR EL AJUSTE AL PROCESO:

Objetivo: Garantizar que los productos clave hayan sido generados cumpliendo con las actividades y pasos definidos en el modelo de proceso adoptado para el proyecto.

Mecanismo:

1. Revisar los productos clave para evaluar si se han desarrollado siguiendo las actividades descritas en el proceso durante todo el ciclo de vida del software.
2. Consultar documentos relevantes como:
 - a. Plan del proyecto
 - b. Plan de la iteración
 - c. Plan de verificación
3. Verificar la existencia y consistencia de los productos previos que sirvieron como entrada para el producto revisado.
4. Se debe confirmar que las desviaciones de revisiones anteriores hayan sido resueltas; de lo contrario, incluirlas en la revisión actual.
5. Generar un Informe de Revisión de SQA enfocado en el ajuste al proceso, con las desviaciones detectadas y las acciones correctivas necesarias.

REVISIÓN TÉCNICA FORMAL (RTF):

Objetivo: Detectar errores en función, lógica o implementación de los productos del software, asegurando que cumplan con las especificaciones y estándares establecidos, y documentar cualquier desviación encontrada.

Mecanismo:

1. Realizar una revisión rigurosa y sistemática de productos críticos, diseñada para detectar defectos o desviaciones tempranamente.
2. Convocar formalmente a los participantes, quienes deberán preparar el material requerido con anticipación y revisar listas de preguntas relacionadas con el producto.
3. Realizar la reunión de revisión con el responsable de SQA y los miembros del equipo de desarrollo.
4. Documentar los resultados en un Informe de RTF, incluyendo las desviaciones detectadas y las acciones correctivas propuestas.

REQUERIMIENTOS MÍNIMOS:

En esta sección se describen los elementos mínimos que deben ser revisados para garantizar la calidad del software desarrollado. Estos elementos son seleccionados con base en el estándar IEEE 730 y otros aspectos críticos del proyecto, asegurando que se aborden las áreas clave en cada etapa del desarrollo. Además, se indica el tipo de revisión aplicable para cada elemento:

ESPECIFICACIÓN DE REQUERIMIENTOS

Tipo de revisión aplicable:

- Evaluación de la calidad de los productos.
- Revisar el ajuste a precio.

Justificación:

Este documento define las funcionalidades, restricciones y objetivos del sistema. Su revisión asegura que las necesidades del cliente se hayan capturado correctamente, sin ambigüedades, y que sean consistentes con los objetivos de calidad establecidos.

MODELO DE DISEÑO Y DESCRIPCIÓN DE LA ARQUITECTURA

Tipo de revisión aplicable:

- Evaluación de la calidad de los productos.
- Revisión Técnica Formal.

Justificación:

Este modelo define la estructura del sistema, incluyendo módulos, componentes, relaciones y tecnologías empleadas. Su revisión verifica que el diseño sea eficiente, escalable y conforme con los estándares de calidad.

PLAN DE VERIFICACIÓN Y VALIDACIÓN (V&V)

Tipo de revisión aplicable:

- Revisar el ajuste al proceso.

Justificación:

Este plan describe los métodos y herramientas que se usarán para verificar y validar los productos. Su revisión asegura que las actividades estén alineadas con los estándares establecidos y que cubran todos los aspectos críticos del sistema.

PLAN DE GESTIÓN DEL PROYECTO**Tipo de revisión aplicable:**

- Evaluación de la calidad de los productos.
- Revisar el ajuste al proceso.

Justificación:

Este plan define los cronogramas, recursos, roles y responsabilidades. Su revisión garantiza que el proyecto esté adecuadamente organizado y que los plazos sean realistas y alcanzables.

PLAN DE GESTIÓN DE CONFIGURACIÓN**Tipo de revisión aplicable:**

- Revisar el ajuste al proceso.

Justificación:

Este plan describe los procedimientos para controlar cambios en los productos del software. Su revisión asegura que las prácticas de gestión de configuración sean consistentes y que eviten problemas derivados de cambios descontrolados.

DISEÑO VS. ESPECIFICACIÓN DE REQUERIMIENTOS**Tipo de revisión aplicable:**

- Evaluación de la calidad de los productos.
- Revisión Técnica Formal.

Justificación:

La revisión asegura que el diseño cumpla con los requisitos establecidos y que no haya desviaciones importantes entre ambos documentos.

IMPLEMENTACIÓN VS. DISEÑO**Tipo de revisión aplicable:**

- Evaluación de la calidad de los productos.
- Revisión Técnica Formal.

Justificación:

Verificar que el código desarrollado sea una implementación fiel del diseño aprobado y que respete las buenas prácticas de programación.

VERIFICACIÓN VS. ESPECIFICACIÓN DE REQUERIMIENTOS**Tipo de revisión aplicable:**

- Revisar el ajuste al proceso.
- Evaluación de la calidad de los productos.

Justificación:

Se debe asegurar que las pruebas realizadas confirmen que el producto cumple con los requisitos funcionales y no funcionales establecidos.

CASOS DE PRUEBA Y RESULTADOS DE PRUEBAS**Tipo de revisión aplicable:**

- Evaluación de la calidad de los productos.
- Revisar el ajuste al proceso.

Justificación:

Los casos de prueba deben ser revisados para garantizar su alineación con los requisitos y los resultados deben analizarse para identificar posibles defectos o desviaciones.

CÓDIGO FUENTE**Tipo de revisión aplicable:**

- Revisión Técnica Formal.

Justificación:

Revisar el código fuente asegura que cumpla con los estándares de codificación, sea mantenible y esté libre de defectos críticos.

PRUEBAS DE INTEGRACIÓN Y ACEPTACIÓN**Tipo de revisión aplicable:**

- Evaluación de la calidad de los productos.

- Revisar el ajuste al proceso.

Justificación:

Estas pruebas validan que el sistema funcione correctamente como un todo y que satisfaga los requerimientos del cliente.

DOCUMENTACIÓN DEL USUARIO

Tipo de revisión aplicable:

- Evaluación de la calidad de los productos.

Justificación:

Verificar que los manuales y guías sean claros, precisos y útiles para los usuarios finales.

SEGUIMIENTO DE DEFECTOS Y DESVIACIONES

Tipo de revisión aplicable:

- Revisar el ajuste al proceso.

Justificación:

Se debe asegurar que todos los defectos identificados hayan sido documentados, evaluados y corregidos.

AGENDA:

FASE I – INICIAL

ITERACIÓN I

Entregable	Realizado	Revisión	Tipo de revisión
Especificación de Requerimientos	Semana 2	Semana 3	Evaluación de la calidad de los productos

ITERACIÓN II

Entregable	Realizado	Revisión	Tipo de revisión
Plan del proyecto	Semana 2	Semana 3	Revisar el ajuste al proceso

FASE II – ELABORACIÓN

ITERACIÓN I

Entregable	Realizado	Revisión	Tipo de revisión
Modelo de Diseño y Arquitectura	Semana 4	Semana 5	Revisión Técnica Formal

ITERACIÓN II

Entregable	Realizado	Revisión	Tipo de revisión
Plan de Verificación y Validación	Semana 5	Semana 6	Revisar el ajuste al proceso

FASE III – CONSTRUCCIÓN

ITERACIÓN I

Entregable	Realizado	Revisión	Tipo de revisión
Diseño detallado	Semana 7	Semana 8	Evaluación de la calidad de los productos

ITERACIÓN II

Entregable	Realizado	Revisión	Tipo de revisión
Implementación inicial	Semana 9	Semana 10	Revisión Técnica Formal

ITERACIÓN III

Entregable	Realizado	Revisión	Tipo de revisión
Pruebas Unitarias	Semana 10	Semana 11	Revisar el ajuste al proceso

FASE IV – TRANSICIÓN

ITERACIÓN I

Entregable	Realizado	Revisión	Tipo de revisión
Pruebas de Integración	Semana 12	Semana 13	Revisión Técnica Formal

ITERACIÓN II

Entregable	Realizado	Revisión	Tipo de revisión
Pruebas de Aceptación	Semana 13	Semana 14	Evaluación de la calidad de los productos

ITERACIÓN III

Entregable	Realizado	Revisión	Tipo de revisión
Manual de Usuario	Semana 14	Semana 15	Revisar el ajuste al proceso

TESTEO:

Identificación de propiedades de calidad no verificadas en el VVP

1. **Definición** **inicial:**
Revisar exhaustivamente el Documento de Verificación y Validación del Proyecto (VVP) para identificar cualquier propiedad de calidad no cubierta, como:
 - **Portabilidad:** ¿El sistema puede trasladarse fácilmente a diferentes entornos operativos?
 - **Rendimiento:** ¿Cumple con tiempos de respuesta aceptables bajo diferentes cargas?
 - **Seguridad:** ¿Están los datos protegidos contra accesos no autorizados y vulnerabilidades?
2. **Clasificación** **y** **priorización:**
Una vez identificadas, se priorizarán según su impacto en el usuario final y el sistema.
Ejemplo:
 - Crítico: Seguridad y rendimiento.
 - Medio: Usabilidad.
 - Bajo: Portabilidad.

Definición y detalle de pruebas

Para cada propiedad identificada:

1. **Pruebas de seguridad:**
 - **Prueba de penetración:** Simular ataques para evaluar vulnerabilidades.
 - Herramientas: OWASP ZAP, Burp Suite.
2. **Pruebas de rendimiento:**
 - **Pruebas de carga:** Medir tiempos de respuesta con 1000 usuarios concurrentes.
 - Herramientas: JMeter, Gatling.
3. **Pruebas de usabilidad:**
 - **Pruebas con usuarios:** Realizar evaluaciones con usuarios reales y encuestas.
 - Herramientas: Maze, Lookback.
4. **Pruebas de portabilidad:**
 - Ejecutar el software en entornos con sistemas operativos distintos (Linux, Windows, macOS).

Resultados esperados

- Registro detallado de las pruebas con métricas específicas (ej., latencia máxima 2ms, porcentaje de errores de acceso <0.1%).
- Asegurar que cada propiedad cumple con las expectativas definidas por los stakeholders.

IDENTIFICACIÓN DE PROPIEDADES DE CALIDAD NO VERIFICADAS EN EL VVP

1. **Definición** **inicial:**
Revisar exhaustivamente el Documento de Verificación y Validación del Proyecto (VVP) para identificar cualquier propiedad de calidad no cubierta, como:
 - **Portabilidad:** ¿El sistema puede trasladarse fácilmente a diferentes entornos operativos?
 - **Rendimiento:** ¿Cumple con tiempos de respuesta aceptables bajo diferentes cargas?
 - **Seguridad:** ¿Están los datos protegidos contra accesos no autorizados y vulnerabilidades?
2. **Clasificación** **y** **priorización:**
Una vez identificadas, se priorizarán según su impacto en el usuario final y el sistema.
Ejemplo:
 - Crítico: Seguridad y rendimiento.
 - Medio: Usabilidad.
 - Bajo: Portabilidad.

DEFINICIÓN Y DETALLE DE PRUEBAS

Para cada propiedad identificada:

1. **Pruebas de seguridad:**
 - **Prueba de penetración:** Simular ataques para evaluar vulnerabilidades.
 - Herramientas: OWASP ZAP, Burp Suite.
2. **Pruebas de rendimiento:**
 - **Pruebas de carga:** Medir tiempos de respuesta con 1000 usuarios concurrentes.
 - Herramientas: JMeter, Gatling.
3. **Pruebas de usabilidad:**
 - **Pruebas con usuarios:** Realizar evaluaciones con usuarios reales y encuestas.
 - Herramientas: Maze, Lookback.
4. **Pruebas de portabilidad:**
 - Ejecutar el software en entornos con sistemas operativos distintos (Linux, Windows, macOS).

RESULTADOS ESPERADOS

- Registro detallado de las pruebas con métricas específicas (ej., latencia máxima 2ms, porcentaje de errores de acceso <0.1%).
- Asegurar que cada propiedad cumple con las expectativas definidas por los stakeholders.

REPORTE DE PROBLEMAS Y ACCIONES CORRECTIVAS:

REPORTE DE PROBLEMAS

1. Métodos de reporte:

- Uso de sistemas de seguimiento como **Jira** o **Redmine**, con plantillas predefinidas.
- Campos requeridos:
 - Identificador del problema.
 - Descripción detallada.
 - Producto afectado.
 - Severidad: Baja, Media, Alta, Crítica.

2. A quién se reportarán los problemas:

- Responsable del área de calidad.
- Líder técnico del módulo afectado.
- Equipo de desarrollo.
- Stakeholders si afecta directamente objetivos del proyecto.

SEGUIMIENTO DE PROBLEMAS

1. Etapas:

- **Asignación:** El problema se asigna al área técnica adecuada.
- **Revisión:** Análisis de la causa raíz con herramientas como **Ishikawa** o **5 Porqués**.
- **Resolución:** Implementación de la solución y revalidación.
- **Cierre:** Confirmación de resolución y registro en el historial.

2. Herramientas sugeridas:

- Jira para gestión ágil.
- Slack o Microsoft Teams para comunicación.

RESPONSABILIDAD DE ACCIONES CORRECTIVAS

- **Líder técnico:** Resolver problemas técnicos.
- **Responsable de calidad:** Supervisar las correcciones.
- **Equipo de desarrollo:** Implementar soluciones.
- **Stakeholders:** Aprobar cambios de alto impacto.

HERRAMIENTAS, TÉCNICAS Y METODOLOGÍAS:

El uso de herramientas, técnicas y metodologías adecuadas asegura que las actividades de aseguramiento de calidad (SQA) se realicen de manera efectiva, proporcionando un marco estructurado para la revisión y verificación de los productos de software.

1. HERRAMIENTAS

Se seleccionarán herramientas específicas que apoyen las actividades de aseguramiento de calidad, alineadas con las necesidades del proyecto y los estándares establecidos.

1. Herramientas para gestión de calidad y revisiones:

- **SonarQube:** Para análisis estático del código, identificación de vulnerabilidades, duplicados y problemas de calidad.
 - **Jira:** Para rastrear problemas, gestionar tareas y registrar actividades relacionadas con defectos encontrados en las revisiones.
 - **TestLink:** Para la gestión de casos de prueba, registro de resultados y análisis de cobertura de pruebas.
2. **Herramientas para revisión de documentos:**
 - **Google Docs/Microsoft Word:** Uso compartido y revisiones colaborativas con control de cambios habilitado.
 - **Grammarly/LanguageTool:** Para revisar la claridad y precisión de la documentación.
 3. **Herramientas de pruebas:**
 - **Postman:** Para pruebas de APIs RESTful y validación de servicios web.
 - **Selenium:** Para automatización de pruebas funcionales en aplicaciones web.
 - **JMeter:** Para pruebas de carga y rendimiento.
 4. **Herramientas para checklist y control de calidad:**
 - **Excel/Google Sheets:** Para elaborar listas de verificación (checklists) con elementos específicos para cada producto a revisar.
 - **Trello:** Organización visual de listas de tareas relacionadas con revisiones específicas de producto.

2. TÉCNICAS

1. **Revisión por pares (Peer Reviews):**
 - Los productos de software, como documentos y código, serán revisados por un equipo de pares técnicos.
 - Se utilizará una plantilla estándar para registrar observaciones, comentarios y acciones a realizar.
2. **Inspecciones formales:**
 - Metodología estructurada para identificar defectos en etapas tempranas.
 - Incluye reuniones de revisión con la participación de un moderador, un registrador, y revisores asignados.
3. **Pruebas estáticas:**
 - Análisis estático del código fuente y documentación sin ejecutar el programa.
 - Uso de herramientas como SonarQube y Linters para detectar errores comunes y adherencia a estándares de codificación.
4. **Pruebas dinámicas:**
 - Pruebas funcionales y no funcionales que se ejecutan directamente sobre el software.
 - Se incluirán pruebas unitarias, de integración, de sistema y de aceptación.

3. METODOLOGÍAS

1. **Metodologías para revisiones:**
 - **Revisión incremental:**
 - Cada producto será revisado en etapas a medida que se desarrolla.
 - Las revisiones se enfocarán en módulos pequeños antes de proceder a una integración mayor.
 - **Revisión en múltiples instancias:**
 - Se elaborarán **checklists específicas** para las distintas etapas de revisión de un mismo producto (e.g., diseño, implementación, pruebas).
2. **Metodologías para pruebas:**

- **TDD (Test-Driven Development):**
 - Se escriben las pruebas antes de desarrollar el código funcional. Esto asegura que el desarrollo está guiado por los requisitos y casos de prueba definidos.
- **BDD (Behavior-Driven Development):**
 - Enfoque en el comportamiento del software según criterios definidos.
 - Las pruebas se redactan en lenguaje natural para facilitar la comprensión por parte de todos los stakeholders.
- **Metodología V-Model:**
 - Vincula cada fase de desarrollo con su correspondiente fase de prueba. Por ejemplo:
 - Requisitos ↔ Pruebas de aceptación.
 - Diseño ↔ Pruebas de integración.

4. ELABORACIÓN DE CHECKLISTS

1. Creación y mantenimiento:

- Cada producto tendrá un conjunto de checklists específicas adaptadas a su propósito y etapa del ciclo de vida del software.
- Ejemplos de productos y áreas de revisión:
 - **Documentos de requisitos:** Verificar completitud, claridad y ausencia de ambigüedades.
 - **Código fuente:** Revisar adherencia a estándares de codificación, modularidad y manejo de excepciones.
 - **Casos de prueba:** Asegurar cobertura suficiente y consistencia en las descripciones.

2. Formato y estructura:

- Cada checklist incluirá los siguientes campos:
 - **Elemento a verificar:** Descripción clara de la característica o requisito que debe cumplirse.
 - **Criterio de aceptación:** Condición específica que determina si el elemento pasa la revisión.
 - **Responsable:** Persona encargada de verificar ese elemento.
 - **Estado:** Marcar si está **aprobado**, **rechazado** o **pendiente**.

3. Revisiones periódicas:

- Los checklists serán revisados y actualizados regularmente para reflejar cambios en los requisitos o estándares.

5. REFERENCIAS A DOCUMENTOS EXTERNOS

1. Documento maestro de checklists:

- Las checklists completas estarán recopiladas en un documento aparte, denominado **CHECKLISTS DE REVISIÓN Y ASEGURAMIENTO DE CALIDAD**.
- Este documento estará vinculado a la sección correspondiente de este plan y será actualizado conforme se agreguen nuevos productos o etapas al proyecto.

2. Integración con otros documentos:

- El plan de pruebas (IEEE 829) será referenciado para detallar las pruebas dinámicas planificadas.
- Los estándares de configuración establecidos en el SCMP (IEEE 828) también serán referenciados para mantener consistencia en las revisiones relacionadas con versiones y control de cambios.

6. SUPERVISIÓN Y ASEGURAMIENTO

1. Actividades de supervisión:

- Validar que las herramientas seleccionadas sean utilizadas adecuadamente por los equipos.
 - Monitorear la correcta implementación de técnicas como TDD y revisiones por pares.
2. **Auditorías internas:**
- Realizar auditorías periódicas para verificar que los checklists y metodologías sean aplicados de forma consistente.

GESTIÓN DE CONFIGURACIÓN (CONTROL DE CÓDIGO Y CONTROL DE MEDIOS)

El objetivo principal de la gestión de configuración es garantizar que los productos del software sean identificados, controlados y monitoreados de manera adecuada durante todo el ciclo de vida del proyecto, reduciendo riesgos asociados con cambios no autorizados o pérdidas de información.

1. ALMACENAMIENTO, MANTENIMIENTO Y DOCUMENTACIÓN DE VERSIONES

1. **Herramientas de control de versiones:**
- **Git:** Proporciona un sistema distribuido para gestionar versiones con características como ramas (branches), fusiones (merges) y registros de auditoría.
 - **SVN:** Sistema centralizado que asegura el control de versiones y permite recuperar versiones anteriores fácilmente.
 - **Repositorios:**
 - **Privados (GitHub, GitLab):** Aseguran la confidencialidad del código mediante permisos de acceso.
 - **Copia de seguridad:** Respallos automáticos semanales en servidores externos (e.g., AWS S3 o Google Cloud).
2. **Identificación de versiones:**
- Se utiliza un esquema de versionado **semántico**: MAJOR.MINOR.PATCH.
 - **MAJOR:** Cambios incompatibles con versiones anteriores.
 - **MINOR:** Nuevas funcionalidades sin afectar versiones previas.
 - **PATCH:** Correcciones de errores o pequeños ajustes.
3. **Métodos de documentación:**
- **Archivos README:** Incluir descripción detallada de los cambios en cada versión.
 - **Changelogs:** Documentos estructurados que detallan las modificaciones realizadas en cada iteración.

2. PROTECCIÓN DE ALMACENAMIENTO Y CONTROL DE ACCESO

1. **Controles de acceso:**
- **Roles y permisos:**
 - **Desarrolladores:** Acceso de lectura/escritura en ramas específicas.
 - **Revisores:** Permisos de solo lectura y comentarios.
 - **Autenticación:** Uso de autenticación multifactor (MFA) para evitar accesos no autorizados.
2. **Prevención de alteraciones:**
- Implementación de **hooks** en Git para prevenir commits en ramas protegidas sin aprobación previa.
 - Auditorías trimestrales para revisar la integridad del repositorio.
3. **Almacenamiento seguro:**
- Infraestructura de repositorios almacenada en servidores con protocolos de cifrado (e.g., SSL/TLS).

- Backup incremental diario y backup completo semanal.

3. ACTIVIDADES DE ASEGURAMIENTO DE CALIDAD EN CONFIGURACIÓN

1. Aseguramiento de la línea base:

- **Creación de la línea base:** Se establece al final de cada sprint o ciclo principal, aprobada por el Comité de Control de Cambios (CCB).
- **Revisión periódica:** Validación de que los elementos de la línea base están completos y son correctos.

2. Monitoreo continuo:

- **Verificación de trazabilidad:** Cada cambio debe asociarse a una solicitud aprobada.
- **Análisis de logs:** Uso de herramientas como GitLens para rastrear el historial de cambios.

3. Supervisión del proceso SCM:

- Monitoreo del Comité de Control de Cambios para garantizar que las aprobaciones de cambios sean ejecutadas conforme a los estándares.
- Realización de auditorías internas trimestrales.

GESTIÓN DE RIESGOS:

La gestión de riesgos busca identificar, analizar, priorizar y mitigar los riesgos que puedan afectar el cumplimiento de los objetivos del proyecto, asegurando que se desarrollen planes de contingencia para responder ante posibles amenazas.

1. Métodos para identificar riesgos

1. Técnicas estructuradas:

- **Análisis FODA:** Identificar amenazas que puedan convertirse en riesgos.
- **Revisión de historial:** Evaluar riesgos ocurridos en proyectos previos similares.

2. Técnicas participativas:

- **Talleres de riesgos:** Sesiones de brainstorming con el equipo de proyecto para identificar y clasificar riesgos.
- **Cuestionarios:** Enviar encuestas al equipo técnico y stakeholders para identificar posibles problemas.

3. Fuentes principales de riesgos:

- **Técnicos:** Cambios en requisitos, fallos de integración, baja calidad del código.
- **Organizacionales:** Rotación de personal, falta de recursos.
- **Externos:** Cambios en regulaciones, proveedores no confiables.

2. Monitoreo y control de riesgos

1. Matriz de riesgos:

- Crear una matriz con dos ejes:
 - **Probabilidad:** Baja, Media, Alta.
 - **Impacto:** Insignificante, Moderado, Crítico.
- Clasificar riesgos según su nivel de prioridad (bajo, medio, alto, crítico).

2. Reuniones periódicas:

- Establecer reuniones mensuales para revisar el estado de los riesgos identificados y buscar nuevos.

3. Documentación:

- Uso de un Registro de Riesgos con campos como:
 - Identificación del riesgo.
 - Categoría.
 - Causa raíz.
 - Plan de respuesta (contingencia/mitigación).
 - Responsable.

3. Evaluación y seguimiento de riesgos**1. Evaluación inicial y recurrente:**

- Cada miembro del equipo realiza una evaluación individual de riesgos.
- En una reunión conjunta, se consolidan y priorizan los riesgos detectados.

2. Planes de acción para riesgos críticos:

- **Eliminación:** Evitar la causa del riesgo directamente (e.g., cambiar tecnología).
- **Mitigación:** Reducir la probabilidad o el impacto mediante acciones preventivas.
- **Contingencia:** Preparar un plan de respuesta si el riesgo ocurre.

3. Seguimiento continuo:

- Incorporar el monitoreo de riesgos como un punto regular en las reuniones de progreso del proyecto.
- Actualizar el Registro de Riesgos según nuevos hallazgos o cambios en el proyecto.

4. Aseguramiento de calidad en la gestión de riesgos

1. Revisión de la calidad de evaluaciones de riesgos:

- Validar que los riesgos sean evaluados por diferentes personas para obtener perspectivas diversas.
- Asegurar que los riesgos estén priorizados con base en datos objetivos.

2. Monitoreo de planes de respuesta:

- Verificar que los planes de mitigación y contingencia se implementen de acuerdo con las fechas previstas.
- Realizar auditorías para asegurar que los riesgos estén correctamente gestionados.

3. Incorporación en el plan del proyecto:

- Definir claramente en el plan los responsables y cronogramas relacionados con la gestión de riesgos.
- Documentar el impacto y seguimiento en las fases clave del proyecto.

La gestión de riesgos busca identificar, analizar, priorizar y mitigar los riesgos que puedan afectar el cumplimiento de los objetivos del proyecto, asegurando que se desarrollen planes de contingencia para responder ante posibles amenazas.

1. MÉTODOS PARA IDENTIFICAR RIESGOS**1. Técnicas estructuradas:**

- **Análisis FODA:** Identificar amenazas que puedan convertirse en riesgos.
- **Revisión de historial:** Evaluar riesgos ocurridos en proyectos previos similares.

2. Técnicas participativas:

- **Talleres de riesgos:** Sesiones de brainstorming con el equipo de proyecto para identificar y clasificar riesgos.
- **Cuestionarios:** Enviar encuestas al equipo técnico y stakeholders para identificar posibles problemas.

3. Fuentes principales de riesgos:

- **Técnicos:** Cambios en requisitos, fallos de integración, baja calidad del código.
- **Organizacionales:** Rotación de personal, falta de recursos.
- **Externos:** Cambios en regulaciones, proveedores no confiables.

2. MONITOREO Y CONTROL DE RIESGOS**1. Matriz de riesgos:**

- Crear una matriz con dos ejes:
 - **Probabilidad:** Baja, Media, Alta.
 - **Impacto:** Insignificante, Moderado, Crítico.
- Clasificar riesgos según su nivel de prioridad (bajo, medio, alto, crítico).

2. Reuniones periódicas:

- Establecer reuniones mensuales para revisar el estado de los riesgos identificados y buscar nuevos.

3. Documentación:

- Uso de un Registro de Riesgos con campos como:
 - Identificación del riesgo.
 - Categoría.
 - Causa raíz.
 - Plan de respuesta (contingencia/mitigación).
 - Responsable.

3. EVALUACIÓN Y SEGUIMIENTO DE RIESGOS

1. Evaluación inicial y recurrente:

- Cada miembro del equipo realiza una evaluación individual de riesgos.
- En una reunión conjunta, se consolidan y priorizan los riesgos detectados.

2. Planes de acción para riesgos críticos:

- **Eliminación:** Evitar la causa del riesgo directamente (e.g., cambiar tecnología).
- **Mitigación:** Reducir la probabilidad o el impacto mediante acciones preventivas.
- **Contingencia:** Preparar un plan de respuesta si el riesgo ocurre.

3. Seguimiento continuo:

- Incorporar el monitoreo de riesgos como un punto regular en las reuniones de progreso del proyecto.
- Actualizar el Registro de Riesgos según nuevos hallazgos o cambios en el proyecto.

4. ASEGURAMIENTO DE CALIDAD EN LA GESTIÓN DE RIESGOS

1. Revisión de la calidad de evaluaciones de riesgos:

- Validar que los riesgos sean evaluados por diferentes personas para obtener perspectivas diversas.
- Asegurar que los riesgos estén priorizados con base en datos objetivos.

2. Monitoreo de planes de respuesta:

- Verificar que los planes de mitigación y contingencia se implementen de acuerdo con las fechas previstas.
- Realizar auditorías para asegurar que los riesgos estén correctamente gestionados.

3. Incorporación en el plan del proyecto:

- Definir claramente en el plan los responsables y cronogramas relacionados con la gestión de riesgos.
- Documentar el impacto y seguimiento en las fases clave del proyecto.

CONCLUSIÓN

El Plan de Aseguramiento de la Calidad del Software (SQA) establece una estructura sólida para garantizar que los productos de software cumplan con los estándares de calidad definidos, satisfagan las necesidades de los usuarios y se desarrollen de manera eficiente. A través de un enfoque sistemático, que incluye la gestión de riesgos, la revisión de procesos, la documentación y la evaluación continua, este plan proporciona un marco para mitigar posibles fallos y garantizar el éxito del proyecto.

La implementación rigurosa de las estrategias y prácticas descritas en este documento no solo contribuye a minimizar los riesgos asociados al desarrollo de software, sino que también asegura la mejora continua de los procesos organizacionales. Esto resulta en productos de mayor calidad, mayor satisfacción del cliente y una optimización en el uso de los recursos del proyecto.

El compromiso del equipo con las actividades de SQA descritas en este plan será determinante para el logro de los objetivos del proyecto, permitiendo una entrega exitosa dentro de los plazos y presupuestos establecidos.