Plan de SQA
Testify

OSLO

Ojeda Valeria – Sly Eduardo
Levipichun Emilio – Oyarzo Malena

La Calidad del Software tiene como objetivo brindar la confianza de que el producto final logrará satisfacer los requisitos del cliente.

En el Plan de SQA se reflejan las evaluaciones a realizar, los estándares a aplicar, los productos a realizar, los procedimientos a seguir en la elaboración de los distintos productos y los procedimientos para informar de los defectos detectados a sus responsables y realizar el seguimiento de los mismos hasta su corrección.

Tabla de contenido

Plan de SQA

Propósito

El propósito de este plan es especificar las actividades que se realizarán para asegurar la calidad del software a construir. En él se detallan los productos que se van a revisar y los estándares, normas o reglas a aplicar, los métodos y procedimientos que se utilizarán para revisar que la elaboración de los productos se realice como lo establece el modelo de ciclo de vida del proyecto; y procedimientos para informar a los responsables de los productos los defectos encontrados y realizar un seguimiento de dichos defectos hasta su corrección.

Este software consiste en una aplicación web que no solo permite registrar sino también administrar y controlar los casos de pruebas creados por los alumnos de la cátedra de laboratorio de desarrollo de software para proyectos de desarrollo de software con el objetivo de que realmente se realicen las pruebas necesarias y que son requeridas para garantizar la calidad del producto final.

A continuación, se detallan los objetivos específicos que se establecen en el Plan de SQA

Obedecer a los estándares, normas y convenciones establecidas y aceptadas por los integrantes del grupo de desarrollo.

Producir el compromiso individual de los integrantes del grupo OSLO, para garantizar la calidad del producto.

Controlar la confección de la documentación asociada y la configuración correcta del software.

Asegurar que se cumplan correctamente los requerimientos solicitados por los docentes de la cátedra de laboratorio de desarrollo de software correspondiente a la Unidad Académica Rio Gallegos.

Referencias

A continuación, se listarán las fuentes y documentos que serán utilizados para desarrollar y respaldar al actual Plan SQA.

Nuestros documentos:

Documento de revisión de SQA.
Documento Informe Final de SQA.
Documento plan gestión de configuraciones.
Documento plan de gestión de riesgos.
Documento plantilla de Revisión técnica formal.

Normas, estándares y bibliografía:

ANSI/IEEE Std 730.1-1989, IEEE Standard for Software Quality Assurance.

SQuaRE, ISO 25000:2005,Quality management systems – Requirements ISO 9001:2008

Piattini, M. Calidad de Sistemas de Información. Madrid, 2011. Ra-Ma. ISBN 9788499640709.

Gestión

Organización

Rol	Función	Responsables
Líder del Proyecto.	Es el responsable de la planificación, ejecución y cierre de las actividades del proyecto coordinando y asignando los recursos necesarios. Da soporte a las tareas de estimación y definición de las actividades contenidas en los planes y realiza la revisión y aprobación de los mismos.	Valeria Ojeda.
Administrador de Configuración.	Es el responsable del control de versiones, control de cambios y coordinación del desarrollo colaborativo asegurando que cada producto de trabajo del proyecto estén disponibles en un repositorio que incluyen el código, la documentación generada y las herramientas necesarias, garantizando la integridad y trazabilidad del proyecto.	Levipichun Emilio.
Administrador SQA.	Su principal responsabilidad es garantizar que los compromisos asumidos en el proyecto referidos a la calidad sean cumplidos acorde a los procesos establecidos.	Levipichun Emilio.
Analista	Se encarga de recopilar información relevante para entender completamente la necesidad del cliente, mediante la documentación y priorización de requerimientos para que sean claros, completos y viables para el equipo de desarrollo.	Oyarzo Malena – Valeria Ojeda
Diseño y Código	Es el responsable de resolver y dar seguimiento de calidad al diseño, arquitectura y desarrollo del sistema como así también a la verificación de la calidad del código y su alineación con los	Eduardo Sly – Levipichun Emilio.

	estándares establecidos, además debe identificar, implementar y evaluar los factores de calidad que se aplicaran al sistema asegurándose de que se cumplan los requisitos técnicos y de negocio.	
Documentador	Es responsable de organizar, mantener y garantizar la calidad de toda la información generada durante el proceso de desarrollo, asegurando su almacenamiento adecuado para una fácil recuperación, actualización y coherencia en el formato y estructura.	Malena Oyarzo – Valeria Ojeda.
Administrador de Riesgos	Es el responsable de dar seguimiento continuo, evaluar y proponer medidas de mitigación y contingencia a los riesgos identificados para minimizar su impacto, notificando al líder del proyecto cuando un riesgo identificado se convierta en un problema concreto para coordinar las acciones necesarias de gestión de riesgos, asegurando que el equipo este informado y preparado para afrontarlo.	Malena Oyarzo
Pruebas	Es responsable de garantizar que el software cumpla con los requerimientos funcionales establecidos libre de fallas mediante una planeación, ejecución y evaluación rigurosa de pruebas certificando que el software está listo para un ambiente productivo, mediante la validación de los resultados de las pruebas y verificación de defectos corregidos antes del despliegue final.	Malena Oyarzo – Valeria Ojeda.
Arquitectura del sistema	Es el responsable de transformar los requisitos proporcionados por el Analista en una solución técnica coherente y efectiva, asegurando que el desarrollo se ajuste al diseño arquitectónico original revisándolo y adaptándolo para mantener la integridad y alineación con los requerimientos del sistema y objetivo del proyecto.	Eduardo Sly – Levipichun Emilio.

Todas estas actividades que se realizan durante el desarrollo del proyecto impactan en la calidad del producto final. Las líneas de trabajo que tienen un impacto directo son las siguientes:

- Requerimientos.
- Análisis.
- Diseño.
- Implementación.
- Pruebas.

Actividades

Ciclo de vida del software cubierto por el Plan

Las actividades que se tienen en cuenta son:

- **Captura de requerimientos** (Reuniones iníciales con el cliente, revisiones periódicas, requerimientos prioritarios, aprobación del cliente, deberán ser revisados mediante solicitudes y control de cambios).
- **Análisis y diseño** (Se debe seguir el patrón MVC, escalabilidad, usabilidad y accesibilidad, compatibilidad y responsabilidad, deberán ser revisados mediante RTF, pruebas de usabilidad, auditorias de código y arquitectura).
- **Implementación** (Se deben aplicar estándares y realizar revisiones de código y pruebas).
- **Pruebas** (Se deberán realizar: pruebas unitarias, pruebas de integración, pruebas funcionales, pruebas de aceptación del usuario, pruebas de rendimiento, pruebas de seguridad, aplicando criterios de calidad como: cobertura de pruebas, tasa de errores aceptable, estándares visuales y de usabilidad, documentación de pruebas).

Los productos **claves** a considerar son:

- **Especificaciones de Requerimientos** (Descripción funcional, requerimientos no funcionales, criterios de aceptación, prioridad).
- **Modelo de Caso de Uso** (Se debe anexar para facilitar la comprensión de la interacción de los actores y el sistema, Consistencia de CU/RF, Ambigüedad, Actor/CU identificables).
- **Modelo de Diseño** (Se deben realizar la revisión técnica formal (RTF), debe seguir el modelo MVC (principios de encapsulamiento, separación y responsabilidades), no deben existir inconsistencias en los diagramas UML, debe garantizar la seguridad, se debe poder verificar que el código es exactamente el mismo que el del diseño, se debe garantizar que las decisiones de diseño no afecten negativamente el control de acceso y manejo seguro de datos).
- **Arquitectura del Sistema** (Se deben realizar la revisión técnica formal (RTF), se debe verificar que no exista inconsistencia en la modularidad, es decir que esté bien estructurado en módulos o componentes independientes con una interacción clara entre ellos asegurando la escalabilidad y mantenibilidad permitiendo además que se realicen desarrollos/modificaciones sin afectar a todo el sistema, Se debe cumplir con

RNF básico como tolerancia a fallos o seguridad, se debe garantizar que la arquitectura siga estándares tecnológicos en el proyecto para la comunicación entre equipos, que sea flexible para prever posibles ajustes futuros reduciendo costo y complejidad).

- **Plan de Pruebas** (Se debe dar cobertura completa a cada caso de uso y estar alineados con los requerimientos, priorizar escenarios críticos, evaluación de la eficacia de las pruebas, documentación de los resultados).
- **Plan de Riesgos** (Identificación de riesgos, análisis de impacto y probabilidad, estrategias de mitigación definidas, revisión continua, cobertura completa y validación revisando lecciones aprendidas por experiencia).
- **Manual de Usuario** (Se deberán hacer pruebas con usuarios reales (UAT) para obtener retroalimentación directa, RTF de manual para verificar que el manual cubra todos los flujos de trabajo y escenarios, estandarización del lenguaje ya que debe ser sencillo y directo con un formato uniforme y consistente, se debe validar la estructura para que sea intuitiva y lógica para facilitar la navegación y búsqueda de información).

Actividades de calidad a realizarse

Revisar cada producto.

Aplicando las métricas proporcionadas por SQuaRE (ISO 25000) al producto:

1. Funcionalidad:
   - Cobertura de requisitos funcionales midiendo el porcentaje de requisitos funcionales implementados y aprobados.
   - Precisión de resultados: Evalúa la precisión cumple con los resultados esperados.
2. Fiabilidad:
   - Tasa de fallos: Midiendo el número de fallos ocurridos durante la operación normal en un periodo de tiempo.
3. Usabilidad:
   - Tiempo de aprendizaje: Evaluando cuanto tiempo les toma a los usuarios aprender a usar la aplicación web.
   - Satisfacción del usuario: Basado en encuestas a los usuarios finales con la interfaz y experiencia de uso.
4. Mantenibilidad:
   - Tasa de modificación: Cuanto tiempo toma en modificar el código en respuesta a cambios en los requerimientos.
   - Densidad de defectos: Numero de defectos por unidad de código (defectos cada 1000 líneas de código).

Aplicando métricas centradas en normas de gestión de calidad proporcionadas por ISO 9001 a las actividades en el proceso de desarrollo de software:

1. Índice de cumplimiento de proceso: Mide cuan alineados están los procesos de desarrollo con los estándares definidos.
2. Tiempo de resolución de no conformidades: Tiempo que lleva resolver problemas o desviaciones que no cumplen con los requisitos.

3. Eficacia del proceso de desarrollo: Basado en la relación entre el número de productos y el número de problemas identificados y resueltos.

Aplicando métricas centradas en normas de garantía de calidad IEEE/ANSI 730.1 a las actividades en el proceso de desarrollo de software:

1. Tasa de defectos durante las RTF: Número de defectos detectados durante las RTF.
2. Cobertura de pruebas: Midiendo el porcentaje de código o requisitos cubiertos por las pruebas.
3. Tiempo de corrección de defectos: Tiempo promedio que lleva corregir defectos reportados durante las pruebas o revisiones.

Revisar cada producto

Cada producto definido como **clave** será revisado siguiendo estos pasos:

1. Se verificará que no queden correcciones sin resolver en los informes de revisión. Si existe alguno sin resolver se incluye en la revisión
2. Se revisará que los productos cumplan con los estándares establecidos.
3. Se debe identificar, documentar y hacer seguimiento de posibles desviaciones encontradas y verificar que se hayan realizado las correcciones.

Para ello se adjuntará una plantilla Excel a modo de checklist como anexo para llevar el control de las revisiones durante las iteraciones llamado: ChecklistProductoClave, que contendrá distintas hojas correspondientes a los chequeos antes de la entrega correspondiente, si bien la Revisión SQA contemplara los productos **clave** también se podrán añadir otros productos bajo el sufijo **“apoyo”**.

Como salida se obtiene el Informe de revisión de SQA, este informe debe ser distribuido a los responsables del producto y se debe asegurar de que son conscientes de desviaciones o discrepancias encontradas.

Para cada elemento **clave** se define un documento

- Revisión de SQA – Especificación de Requerimientos de Software.
- Revisión de SQA – Modelo de Casos de Uso.
- Revisión de SQA – Modelo de Diseño.
- Revisión de SQA – Arquitectura del Sistema.
- Revisión de SQA – Plan de pruebas.
- Revisión de SQA – Plan de Riesgos.
- Revisión de SQA – Manual de usuario.

Cada Revisión de SQA se almacenará en el repositorio de Gestión de Calidad correspondiente a su fase e iteración.

Revisar el ajuste al proceso

En esta actividad, se revisarán los productos definidos como **clave** para verificar sus cumplimientos con las actividades y procesos establecidos en el plan del proyecto. El objetivo es asegurar la calidad en el producto final mediante el uso de estándares y especificaciones requeridas a lo largo del ciclo de vida del software.

Se recogerá la información necesaria de cada producto, incluyendo una revisión retrospectiva de los productos que deberían haberse generado permitiendo establecer los criterios de revisión y evaluar si el producto cumple con las especificaciones definidas.

Esta información necesaria se obtiene de los siguientes documentos:

1. Plan del Proyecto (Alcance, objetivos y proceso del proyecto). (ISO 9001).
2. Plan de la iteración (Actividades, entregables específicos pertenecientes a la iteración). (ISO 9001).
3. Plan de Pruebas (Métodos y criterios para la validación del producto). (IEEE 730.1).

Antes de comenzar, se debe verificar en los informes de revisión previos que todas las desviaciones fueron corregidas, si no fuese así, las faltantes se incluyen para ser evaluadas.

Como salida se obtiene el **Informe de revisión de ajuste al Proceso**, este informe debe ser distribuido a los responsables de las actividades y se debe asegurar de que son conscientes de desviaciones o discrepancias encontradas guiándose por normas ISO 9001 e IEEE 730.1.

Realizar Revisión Técnica Formal (RTF)

La Revisión Técnica Formal (RTF) tiene como objetivo principal identificar errores en la función lógica o implementación del producto software, verificando que satisfacen sus especificaciones definidas y se ajusta a los estándares establecidos, señalando las posibles desviaciones detectadas.

Este proceso de revisión riguroso está enfocado en detectar lo antes posible los defectos o desviaciones en los productos que se van generando a lo largo del desarrollo, en especial aquellos de alta importancia.

Proceso de revisión:

En la reunión de RTF se debe considerar:

1. El responsable de SQA e integrantes del equipo de desarrollo estén presentes.

2. La convocatoria de reunión debe ser formal e incluir a los involucrados yel material necesario por adelantado.
3. Los participantes deben preparar preguntas y dudas en un formato tipo “lista” surgidas del estudio del producto a ser revisado.
4. La reunión no debe ser mayor a dos horas.

Documentación/entregables:

La Revisión Técnica Formal (RTF) debe asegurar y dar salida a los siguientes documentos/entregables:

1. Informe de RTF: es el documento principal que detalla los hallazgos y desviaciones detectadas durante la revisión.
2. Documentos a revisar:
   a) Especificación de Requerimientos de Software.
   b) Modelo de Casos de Uso.
   c) Modelo de diseño.
   d) Arquitectura del Sistema.
   e) Plan de pruebas.
   f) Plan de riesgos.
   g) Manual de Usuario.

Relaciones entre las actividades de SQA y la planificación

Al finalizar cada iteración en cada fase se realizarán los controles correspondientes a las actividades de SQA durante todo el proyecto, las métricas que se establezcan en cada control deberán ser documentadas.

Responsables

Administrador del Proyecto y el Administrador de SQA son los encargados de realizar los controles de calidad durante la ejecución del proyecto, sus responsabilidades acordes al ciclo de prevención son:

1. Ejecutar una tarea:
   a) Se debe realizar la tarea según los requisitos y estándares establecidos en el plan de proyecto.
   b) Se debe asignar un responsable, ya sea el equipo de desarrollo en general o un designado en particular.
2. Realizar un control de revisiones:

a) Se debe llevar a cabo una revisión de la tarea para evaluar su conformidad con los requisitos y estándares.
b) Determinar si esta tarea se acepta tal como está o si requiere correcciones.
c) Se debe asignar un responsable, ya sea el Administrador de SQA y/o el Administrador del proyecto, dependiendo de la etapa de revisión

3. Análisis causal de errores:
   a) Se debe realizar un análisis causal para identificar la causa raíz del problema esto incluye:
      i. Análisis del error: (entender cómo y porque ocurrió).
      ii. Hipótesis de causa: (formular una hipótesis sobre la posible causa).
      iii. Investigación: (determinar en qué momento y porque se produjo el error).
      iv. Corrección: (implementar una solución para corregir el error).
      v. Acción correctiva: (desarrollar medidas para prevenir la ocurrencia del error).

4. Registro y métricas:
   a) Se debe documentar el resultado del análisis causal para obtener métricas de calidad y causa de los problemas.
   b) Se debe asignar un responsable: Administrador de SQA.

5. Reinicio del ciclo:
   a) Iniciar nuevamente el ciclo de prevención ejecutando la tarea revisada y corrigiendo cualquier defecto identificado.
   b) El responsable es el equipo de desarrollo supervisado por el Administrador del proyecto y el Administrador SQA.

Documentación

Propósito

El objetivo del presente apartado es definir la gestión de la documentación clave para la garantía de calidad de software cubriendo aspectos como desarrollo, pruebas, uso y mantenimiento de forma estructurada.

Documentación mínima requerida

Esta documentación mínima requerida garantiza que la implementación satisface los requerimientos.

1. Especificación de Requerimientos de Software.
2. Modelo de Casos de Uso.
3. Modelo de diseño.
4. Arquitectura del Sistema.
5. Plan de pruebas.
6. Plan de riesgos.
7. Manual de Usuario.

Especificación de requerimientos del software

El documento de especificación de requerimientos deberá describir, de forma clara y precisa, cada uno de los requerimientos esenciales del software, así como su trazabilidad a las funcionalidades desarrolladas.

Los integrantes de la cátedra de Laboratorio de desarrollo de software (nuestro cliente) deberá obtener como resultado del proyecto una especificación adecuada a sus necesidades en el área de alcance del proyecto, de acuerdo al compromiso inicial del trabajo y a los cambios que este haya sufrido a lo largo del proyecto, que cubra aquellos aspectos que se haya acordado detallar con él.

La especificación debe ser:

1. Completa.
2. Internamente, no deben existir elementos sin especificar.
3. Ser consistente, no puede haber elementos contradictorios.
4. Ser no ambigua, todo término referido al área de aplicación debe estar definido en un glosario.
5. Ser verificable, debe ser posible verificar siguiendo un método definido, si el producto final cumple o no con cada requerimiento.
6. Estar acompañada de un detalle de los procedimientos adecuados para verificar si el producto cumple o no con los requerimientos.
7. Incluir requerimientos de calidad del producto a construir.

Modelos de Casos de Uso

El modelo de caso de uso es una representación de las funciones previstas del sistema y su entorno y sirve como un contrato o acuerdo entre el cliente y los desarrolladores.

Los casos de uso sirven como hebra de unión a lo largo del desarrollo del sistema. El mismo modelo de caso de uso es el resultado de la disciplina de Requerimientos y se utiliza como entrada para disciplinas de Prueba, Diseño y Análisis.

Este documento tendrá las siguientes características:

1. Identificación y descripción de actores (se debe verificar que cada actor este identificado correctamente como así también los usuarios).

2. Identificación y descripción de casos de uso (se debe verificar que los casos de uso son completos, no ambiguos y verificables, verificando también que los diagramas estén alineados con descripciones textuales).
    a) Establecer precondiciones.
    b) Establecer post condiciones.
    c) Flujo de sucesos principal.
    d) Flujos alternativos.
3. Diagramas asociados: Diagramas de casos de uso, actividad, interacción (se debe verificar que los diagrama reflejen con precisión los flujos descritos en el texto y que son consistentes entre sí).

El Modelo de Casos de Uso debe ser redactado en lenguaje natural para la comprensión de los clientes. Se elabora en las primeras iteraciones y luego se extiende hasta lograr el documento preciso y claro.

Modelo de Diseño

El documento de diseño especifica como el software será construido para satisfacer los requerimientos.

Deberá describir los componentes y subcomponentes del diseño del software, incluyendo interfaces internas. Este documento deberá ser elaborado primero como Preliminar y luego será gradualmente extendido hasta llegar a obtener el Detallado.

El cliente deberá obtener como resultado del proyecto el diseño de un producto de software que cubra aquellos aspectos que se haya acordado con el cliente incorporar al diseño, en función de la importancia que estos presenten y de sus conexiones lógicas.

El diseño debe:

1. Corresponder a los requerimientos a incorporar (se debe detallar los módulos, componentes y subcomponentes especificando las relaciones entre ellos).
2. Todo elemento del diseño debe contribuir a algún requerimiento (se debe asegurar que las interfaces están bien definidas, verificables y consistentes respecto a los requerimientos).
3. La implementación de todo requerimiento a incorporar debe estar contemplada en por lo menos un elemento del diseño (Debe ser detallado y trazable alineado con los nuevos requerimientos incorporados o modificados).
4. Ser consistente con la calidad del producto (Se debe verificar que no existan elementos que no estén relacionados con un requerimiento y que no existan requerimientos sin soporte de diseño).

Plan de Pruebas

El Plan de Pruebas deberá identificar y describir los métodos a ser utilizados en:

1. La verificación de que:
   a) Los requerimientos descritos en el documento de requerimientos han sido aprobados por una autoridad apropiada ya sea el cliente o líder de proyecto y administrador de calidad (Se debe comprobar que los requerimientos fueron aprobados, claros, completos y consistentes).
   b) Los requerimientos descritos en el documento de requerimientos son implementados en el diseño expresado en el documento de diseño (Se debe poder realizar una traza entre requerimientos y diseño).
   c) El diseño expresado en el documento de diseño esta implementado en código.

2. Validar que el código, cuando es ejecutado, se adecua a los requerimientos expresados en el documento de requerimientos (Se deben aplicar usos de métricas como el

porcentaje de cobertura de pruebas, tasa de fallos críticos y cantidad de incidentes corregidos en comparación a los incidentes descubiertos).

Documentación de usuario

La documentación de usuario es esencial para asegurar que los usuarios finales puedan entender cómo funciona y opera la aplicación web Testify y que hacer en caso de errores.

Debe especificar y describir:

Los datos y entradas de control requeridos, así como la secuencia de entradas, opciones, limitaciones de programa y otros ítems necesarios para la ejecución exitosa del software (Para ello se emplearán estándares como ISO 25000 de funcionalidad, usabilidad y mantenibilidad).

Errores y las acciones correctivas para que los usuarios puedan tomar acciones de mitigación (Para ello se emplearan estándares como ISO 25000 de funcionalidad, usabilidad).

Como resultado del proyecto el cliente obtendrá una documentación para el usuario de acuerdo a los requerimientos específicos del proyecto.

Plan de Gestión de configuración

Propósito

El Plan de gestión de configuración debe contener métodos para identificar componentes de software, control e implementación de cambios, y registro y reporte del estado de los cambios implementados.

La Gestión de Configuraciones permite controlar el sistema como producto global a lo largo de su creación, obtener informes sobre el estado de desarrollo en que se encuentra y reducir el número de errores durante el mismo, lo que se traduce en un aumento de calidad del proceso de desarrollo y de mejora de la productividad en la organización.

La gestión de configuración facilita además el mantenimiento del sistema, aportando información precisa para valorar el impacto de los cambios solicitados y reduciendo el tiempo de implementación de un cambio, tanto evolutivo como correctivo.

Todas las especificaciones en relación al Plan de Gestión de Configuración están detalladas en el documento correspondiente.

Organización, Responsabilidades

Se designará a un integrante del grupo para la administración de gestión de versiones, el cual se encargará de administrar y dar los permisos en el gestor. Pudiendo cualquier integrante solicitarle al grupo algún cambio para que el mismo estudie su necesidad.

Herramientas, Entorno, e Infraestructura

Se utilizará la herramienta de Gestión de Configuraciones (CGS) GitHub y GitHub Desktop. Este maneja repositorio y directorios a lo largo del ciclo de vida del proyecto. Los directorios se almacenan en un repositorio central de forma online, recordando todos los cambios que se hayan realizado, permitiendo a los integrantes del grupo poder recuperar versiones anteriormente guardadas, examinar la historia de cuándo y cómo fueron modificados los datos, quien hizo los mismos y así poder coordinar el trabajo.

Siendo la misma especialmente útil para los documentos revisados frecuentemente, como el código fuente, la documentación, etc., como así también llevar un balance histórico de las diferentes versiones del sistema.

Forma de trabajo

Durante el proceso de gestión de configuración se utilizará la herramienta GitHub para el control de versiones del producto. Cuando algún miembro haga una modificación en el proyecto, deberá acceder a su repositorio local donde está alojada la documentación y aplicación, teniendo el resto del equipo de desarrollo la última versión actualizada en su repositorio local. La gestión para la actualización del repositorio online se hará mediante la herramienta GitHub Desktop para los documentos y también asociado al plugin integrado de Visual Estudio Code (VSC) para el código fuente.

Estándares, prácticas, convenciones y métricas

Esta sección deberá cumplir con las siguientes funciones:
Identificar los estándares, prácticas, convenciones y métricas que serán aplicadas.
Indicar como será monitoreado y asegurado el cumplimiento con estos ítems.

Objetivos

Existen dos objetivos importantes que se persiguen dentro del programa de métricas:

1. Documentar las metas a la hora de establecer un programa de métricas. Esto tiene sentido a la hora de decidir exactamente qué debe lograrse antes de gastar recursos estableciendo un programa de este tipo.

2. Identificar la información (la métrica) necesaria para lograr estas metas y establecer el marco de referencia de donde puede ser obtenida.

Métricas de proceso

Se recopilan de todos los proyectos y durante un largo periodo de tiempo
Caracterizados por:

1. Control y ejecución del proyecto.
2. Medición de tiempos de las fases.

Para este proyecto se trabajará con las siguientes métricas del proceso:

1. Distribución de esfuerzo por fase.

Métricas de proyecto

ISO 9001:

1. Índice de cumplimiento de procesos
2. Tiempo de resolución de no conformidades
3. Eficacia del proceso de desarrollo

IEEE/ANSI 730.1-1989:

1. Tasa de defectos detectados durante la revisión
2. Tiempo de corrección de defectos

Métricas de producto

Funcionalidad:

1. Cobertura de requisitos funcionales midiendo el porcentaje de requisitos funcionales implementados y aprobados.
2. Precisión de resultados: Evalúa la precisión cumple con los resultados esperados.

Fiabilidad:

1. Tasa de fallos: Midiendo el número de fallos ocurridos durante la operación normal en un periodo de tiempo.
2. Usabilidad y tiempo de aprendizaje: Evaluando cuanto tiempo les toma a los usuarios aprender a usar la aplicación web.
3. Satisfacción del usuario: Basado en encuestas a los usuarios finales con la interfaz y experiencia de uso.

Mantenibilidad:

1. Tasa de modificación: Cuanto tiempo toma en modificar el código en respuesta a cambios en los requerimientos.
2. Densidad de defectos: Numero de defectos por unidad de código (defectos cada 1000 líneas de código).

Estándar de documentación

Para la escritura de documentos se han definido plantillas para ser utilizadas en la elaboración de entregables.

En estas plantillas se definen:

1. Encabezado y pie de página.
2. Fuente y tamaño de fuente para estilo normal.
3. Fuente y tamaño de fuente para los títulos a utilizar.
4. Datos mínimos que se deben incluir: fecha, versión y responsables.

Estándar de programación

Además del documento denominado “Estándar de Programación” donde se definen las características principales que debe contener el código generado durante la implementación, contamos también con los siguientes estándares relacionados a codificación y desarrollo:

Estándares de codificación y desarrollo:

- Java Code Conventions.
- Guía de Estilo Oficial de Angular.

Estándar de Base de datos MySQL:

- Se definirá una estructura consistente de la base de datos, para ello se emplearán nombres en ingles de tablas y columnas en formato snake_case (ejemplo: test_case, project_id), estos deben ser descriptivos y evitar abreviaturas confusas.
- Las claves primarias (PK) siempre llevarán el sufijo _id (ejemplo: user_id).
- El diseño de las relaciones debe seguir un esquema bien definido que asegure que las llaves foráneas (FK) mantengan la integridad referencial.
- Para las convenciones de claves y relaciones se deben usar claves primarias autoincrementales para las tablas principales.
- Se debe mantener un modelo E-R actualizado dentro de MySQL Workbench para documentar las relaciones entre las tablas.
- Las credenciales de conexión a la base de datos no deben estar hardcodeadas en el código, sino que se deben gestionar mediante variables de entorno o sistemas seguros.

Seguridad y escalabilidad:

- Uso de Spring Boot y Spring Security conforme a las guías oficiales.

Arquitectura y Diseño:

- Principios de diseño RESTful API Best Practices.

Documentación y Versionado:

- Swagger para la documentación de APIs.
- Git/GitHub.

Herramientas, técnicas y metodologías

Las siguientes herramientas de software, técnicas, y metodologías darán soporte para las actividades de aseguramiento de calidad.

Herramientas de apoyo

- Notepad++ [Editor de texto y código fuente].
- IntelliJ Idea, Visual Studio Code[IDE con complemento para JAVA y Angular].
- Adobe Photoshop [Edición de imágenes].
- Visual Paradigm[Modelado UML].
- GIT/GitHub[Repositorio del proyecto y Roadmap].
- Trello [Seguimiento de tareas].
- Google Meet[Reuniones virtuales del grupo].
- Office 365.

- MySQL Workbench[Sistema de gestión de bases de datos].

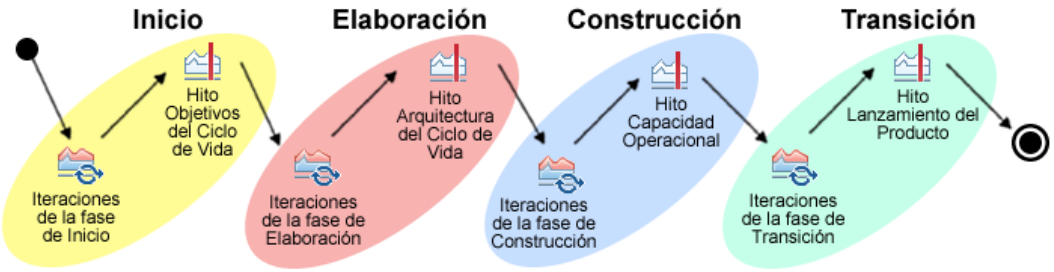
Técnicas

Se incluirá la revisión en el uso de estándares, inspecciones de software, rastreabilidad de requerimientos y casos de uso, verificación y validación de requerimientos y diseño, mediciones y análisis de la lógica de negocio.

- Estándares:
  - Estándar de Documentación.
  - Estándar de Programación JAVA.

Metodologías

- Metodología PSI (https://www.uarg.unpa.edu.ar/psi/).
  - *"La obtención de un software con calidad implica la utilización de metodologías o procedimientos estándares para el análisis, diseño, programación y prueba del software que permitan uniformar la filosofía de trabajo, con el fin de lograr una mayor confiabilidad, mantenimiento y facilidad de prueba, a la vez que eleven la productividad, tanto para la labor de desarrollo como para el control de la calidad del software."*



Flujo de trabajo de la metodología PSI

- Paradigma de programación: Orientado a objetos.